ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/346658105

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000. A migration aware scheduling technique for real-time aperiodic tasks over multiprocessor systems Correspon...

Research · December 2020

DOI: 10.13140/RG.2.2.14755.27681

CITATIONS
0

READS
50

4 authors:

Abid Ali
Abdul Wali Khan University Mardan
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Muhammad Zakarya
AWK University
86 PUBLICATIONS **672** CITATIONS
SEE PROFILE

Rahim Khan
Abdul Wali Khan University Mardan
50 PUBLICATIONS **398** CITATIONS
SEE PROFILE

Mukhtaj Khan
University of Haripur
48 PUBLICATIONS **502** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Improving Scheduling Technique for Microprocessor System View project

An efficient load balancing routing protocol for WSNs and IoT to prolong network lifespan View project

All content following this page was uploaded by Abid Ali on 06 December 2020.

The user has requested enhancement of the downloaded file.

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A migration aware scheduling technique for real-time aperiodic tasks over multiprocessor systems

AYAZ ALI KHAN[1], ABID ALI[1], MUHAMMAD ZAKARYA[1], RAHIM KHAN[1], MUKHTAJ KHAN[1], IZAZ UR RAHMAN[1], MOHD AMIRUDDIN ABD RAHMAN[2]*

[1]Abdul Wali Khan University, Mardan, Pakistan (e-mail(s): {ayazali, mohd.zakarya, rahimkhan, mukhtaj.khan, izaz}@awkum.edu.pk, abidalikhan6564@gmail.com)

[2]Faculty of Science, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia (e-mail(s): mohdamir@upm.edu.my)

Corresponding author(s): M. A. A. Rahman (e-mail(s): mohdamir@upm.edu.my)

"This work is financially supported by the Ministry of Higher Education, Malaysia under Fundamental Research Grant Scheme (FRGS) bearing number FRGS/1/2017/TK04/UPM/02/5."

ABSTRACT Multi-processor systems consist of more than one processor and are mostly used for computationally intensive applications. Real-time systems are those systems that require completing execution of their tasks within a pre-defined deadline. Traditionally, multiprocessor systems are given attention in periodic models, where tasks are executed at regular intervals of time. Gradually, as maturity in multiprocessor design had increased; their use has become very common for real-time systems, in the real world situations, for both periodic and aperiodic tasks. As, the priority of aperiodic task is greater than the priority of periodic task, but not essentially, therefore, they must be completed within the deadline. There is a lot of research work on multiprocessor systems with scheduling of periodic tasks. However, the scheduling is relatively remained unexplored with mixed workload (both periodic and aperiodic tasks). Moreover, higher energy consumption is the main issue of multiprocessor systems, that could be reduced through using energy aware scheduling technique; but, as a consequence the response time of aperiodic tasks is increased. In the literature, various techniques are suggested to decrease the energy consumption of these systems, however, reducing the response time of aperiodic tasks is ignored. In this paper, we propose a scheduling technique that: (i) executes aperiodic tasks at full speed and migrates periodic tasks to other processors if their deadline is earlier than aperiodic tasks – reduces the response time; and (ii) executes aperiodic tasks with lower speed by identifying appropriate processor speed without affecting the response time – reduces energy consumption. Through simulations, we demonstrate the efficiency of the proposed algorithm in terms of low energy consumption and response time. Furthermore, we show that our algorithm also outperforms the well-known Total Bandwidth Server (TBS) algorithm.

INDEX TERMS real-time systems, multi-processor scheduling, energy consumption, performance, dynamic voltage and frequency scaling

I. INTRODUCTION

A multi-processor system consists of two or more processors that shares the peripherals and memory to perform simultaneous execution of various tasks [1]. Moreover, tasks could be either periodic or aperiodic. A Real Time System (RTS) mostly deals with periodic tasks that regularly appear at certain time interval. The periodic model is, largely, used to schedule tasks on these systems using a particular scheduling technique. The Earliest Deadline First (EDF) is one of the optimal scheduling algorithms for scheduling periodic tasks on a uniprocessor system. However, most modern real-time systems are more complex in nature and may consist of both periodic and aperiodic tasks. The RTS may involve human activities, such as emergency task, that come to the system randomly and it must be completed within their allotted time period or deadline [2].

The aperiodic tasks must be completed as soon as possible contrary to the execution of periodic tasks. For example, imagine a car is driven by a computer or robot and reaches near the pedestrian crossing point on the road. It collects data

from deployed embedded sensors within the car and crosses the point only if the sensor do not detect any human being or other species. However, if sensor data is neglected or aborted at the pedestrian crossing point then either the car crosses the point safely or an accident occurs. Therefore, it is mandatory for a robot driven car to collect required data continuously and at a high sampling rate if possible. Additionally, if the car is going to cross that point and detects any human being or other species then appropriate actuators must be activated to apply the break operation on time and to avoid accidents. In order to avoid the car's accidents, a specific deadline, maximum time needed to detect an entity and apply break, must be considered.

The unused time slice of the periodic tasks can be used for the execution of aperiodic tasks. It is a simple technique, however, the response time of the aperiodic tasks would, probably, be increased. The aperiodic server can be used to reduce the response time of aperiodic tasks. The aperiodic server is just like the periodic tasks and it serves the aperiodic task immediately. The aperiodic server has its time period and execution time that is called server capacity. The algorithm that is used to schedule the periodic task will also schedule an aperiodic server. The aperiodic server will schedule the aperiodic task in their allotted time period or deadline. There are a number of server algorithms that have been used for single processor systems such as Deferrable Server [3], Sporadic Server [4] and Extended Priority Exchange Server [5] etc. The simplified model and best response time have been achieved by the Total Bandwidth Server (TBS) technique. However, to the best of our knowledge, fewer similar server techniques have been developed for multiprocessor systems so for. This paper presents a real-time system scheduling technique that reduces response time and energy consumption of the multiprocessor systems. The TBS and EDF has been chosen due to their simplicity and reduction in response time of aperiodic tasks.

High energy consumption is another major issue associated with these systems, more precisely battery powered systems, that arises due to their scale. According to the Gordon Moore law, the quantity of transistors in microprocessor systems would continue to be doubled after each half and an year [6]. Yet, the advancement in battery power is much less than that of the microprocessor systems [7], [8]. To overcome the high energy consumption problem in RTSs, energy and performance efficient scheduling algorithms are desperately needed to be implemented, while keeping the advancement ratio of battery power in view [9]. Dynamic Voltage and Frequency Scaling (DVFS) is commonly used as energy saving technique at system level, but, it could degrade the workload performance. To tackle both issues of energy consumption and performance (response time), we propose a scheduling algorithm that executes aperiodic tasks at full speed and migrates periodic tasks to other processors if their deadline is earlier than aperiodic tasks; and executes aperiodic tasks with lower speed (DVFS) by identifying appropriate processor frequency without affecting the response time. Performance of the proposed algorithm has been evaluated through simulations that suggests improvements in energy efficiency and performance.

The rest of the paper is organized as follows. Major contributions of this work are described in Section II. Section III gives a brief introduction to the scheduling. In Section IV, description of the existing problem along with problem of energy efficient scheduling is given. In Section V, the proposed scheduling technique is described in detail. Performance evaluation, experimental set-up and results of the proposed algorithms are discussed in Section VI. Related work and state-of-the-art techniques are given in Section VII. Finally, Section VIII concludes the paper with several directions for the future work.

II. MAJOR CONTRIBUTIONS

In this paper, we focus on the response time of aperiodic tasks (real-time), which have higher priorities than periodic tasks, while maintaining deadline restrictions when both periodic and aperiodic tasks are running over a multiprocessor system. The major contributions of the research presented in this paper are:

1) an energy and performance efficient scheduling algorithm is proposed for real-times systems;
2) a migration technique to move tasks among various processors;
3) the proposed algorithm has the capability to efficiently schedule both periodic and aperiodic tasks on a multiprocessor system; and
4) we evaluate, through assumptions and simulations, that the proposed algorithm outperforms the well-know TBS algorithm in terms of response time and energy consumption.

III. REAL-TIME SCHEDULING

Multi-processor systems are mostly used for computationally intensive applications whereas real-time systems are well-suited in scenarios where tasks completion deadline is needed to be followed. Multiprocessor systems are most commonly used in periodic model, where different tasks are executed after a pre-defined interval of time. As the maturity level of multi-processor design is improving with the passage of time; these systems are used for both periodic and aperiodic tasks, particularly, for real-time systems. A deadline-oriented real-time system needs to complete aperiodic tasks within the deadline as its priority is higher than periodic tasks. However, this may not be true, as every system may not, essentially, set the priority of aperiodic tasks higher than that of periodic tasks. For hard real-time systems, the most important requirement is to ensure that all periodic real-time tasks meet their timing constraints. Moreover, aperiodic server can be seen as a periodic task with pre-defined periodic and computation time (serviced for aperiodic tasks). Some systems set aperiodic server with a higher priority but some not. In the literature, real-time scheduling has been divided

into two different categories: (i) global scheduling; and (ii) partitioning scheduling [10].

A. GLOBAL SCHEDULING ALGORITHM

The global scheduling algorithm consists of one queue to hold different tasks, as shown in Figure 1 [left]; and selected tasks are needed to be executed among m number of processors [11]. The tasks having the highest priority among m processors in the global queue will be selected for execution. Several migration techniques are used to select the task [12], [13], [14]. In the global scheduling algorithm, every processor maintains a table that shows the number of tasks, currently, in execution. Apart from this, another table is maintained to show extra computational capacity among m processors. The total time is divided into smaller time slots so that each processor gets equal opportunity to execute their tasks. Processors regularly send messages to each other by sharing information about their computational capacity and the task is executed on the processor with additional computational capacity [15].

B. PARTITIONING SCHEDULING ALGORITHM

The partitioning scheduling algorithm consists of many queues created by splitting the tasks among m number of processors, as shown in Figure 1 [right]. When the task set is divided among m processors, then these tasks are executed on each processor by using the uniprocessor scheduling algorithm. The optimal task splitting among m processor is very difficult and several bin-packing algorithms are suggested in state-of-the-art to tackle this issue [6], [7].

IV. EXISTING PROBLEM

The real-time system must give its output within a specified time period. If the task is not completed during that time period then it is considered useless. Generally, tasks are divided into two types: (i) periodic; and (ii) aperiodic tasks. For certain real-time systems, the priority of the aperiodic tasks is higher than the periodic tasks [10]; however, this is not essential, as described in Section III. Currently, researchers and scientists are focused on minimizing the energy consumption of multi-processor systems, and more specifically the real-time systems. However, minimizing the energy consumption of multi-processor systems results in increasing the overall response time that has drastic effects on the performance of the system. Existing system and techniques give more importance to the process of reducing the energy consumption while minimizing response time is mostly neglected, in the state-of-the-art, with the notable exception of [9]. In this paper, an efficient technique, based on the mathematical concepts, is presented in order to minimize the response time of aperiodic tasks while the energy consumption of the multi-processor system is kept minimum or, at least, within the bound.

V. PROPOSED SOLUTION

Traditionally, multi-processor system were focused on the periodic model where tasks were executed after pre-defined interval of time. As the maturity level of multiprocessing system, particularly in the design, is increasing with the passage of time therefore the working paradigm of the real-time systems is shifted towards real world problems where tasks must be executed on basis of their priority levels. The RTS is able to execute aperiodic and periodic tasks. The dead line (finish time) must be considered for aperiodic tasks; and extensive studies in terms of multi-processor real-time systems, are available in the literature, particularly, for periodic tasks.

The area of mixed workload in multi-processing systems (periodic and aperiodic tasks) remained relatively unexplored. The major issue associated with these systems is the higher energy consumption that is controlled by utilizing energy aware scheduling algorithms but the response time is increasing simultaneously. Due to these factors, the overall performance of the multi-processor system is degraded that is an alarming situation. The existing algorithms stressed on methodologies to reduce the consumption of energy in multi-processor systems while do not noticed the problem associated with aperiodic tasks that is reduced response times. In this paper, an efficient algorithm is presented to overcome these issues and reduce the energy consumption of aperiodic tasks in multi-processor systems while preserving overall performance of the systems. Additionally, the response time of multi-processor systems, aperiodic tasks, is kept in bound and is able to handle periodic tasks as well. The load balancing algorithm equally distributes the load among available processors while preserving the idea of maximum utilization of the processors and avoids underutilization. The response time of an aperiodic task is reduced by executing it before the periodic tasks.

As aperiodic tasks must be executed before periodic tasks, especially when priorities of aperiodic tasks are higher than those of periodic tasks; through transferring or migrating the later tasks to other available processors whose deadlines (times) are greater than the former tasks. This mechanism reduces the response time of aperiodic tasks whereas energy consumption of the system is reduced through applying the Dynamic Voltage and Frequency Scaling (DVFS) [7] technique; and executing the aperiodic tasks on slower speed, if possible.

The proposed algorithm has three portions. In first portion, dispatching techniques are addressed in detail and on the arrival of an aperiodic task in multi-processor system utilization of different processor is computed. The aperiodic task is assigned to the lowest utilized processor that results in reducing its response time. If a particular processor utilization is the least, then, that particular aperiodic task is executed quickly.

Second portion of the proposed algorithm mainly focused on the migration technique. When an aperiodic task arrived to the system and its deadline time is less than the periodic

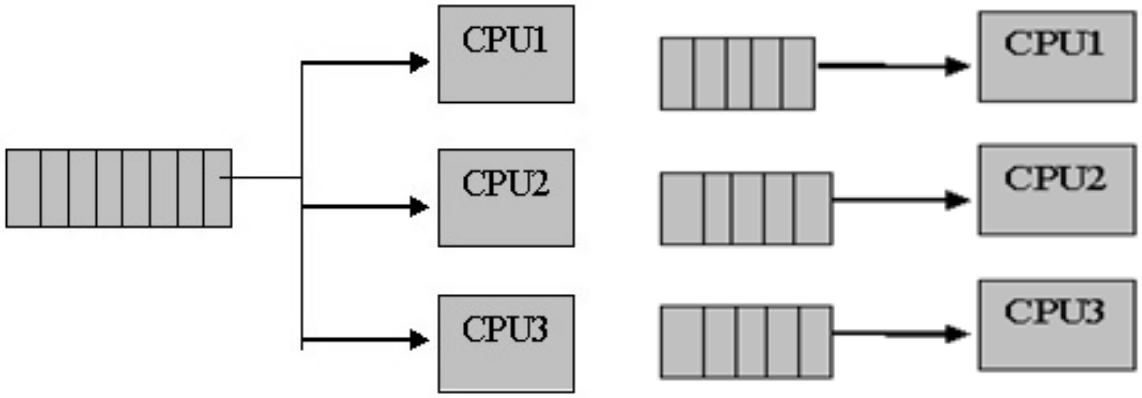

FIGURE 1: Global scheduling [left] and partitioning scheduling [right]

tasks, then the response time of aperiodic task is reduced by migrating the periodic tasks to other processors. The third portion of the proposed algorithm is associated with problem of finding an efficient way to reduce the system speed through DVFS and to decrease its energy consumption as well. These steps are:

1) aperiodic task arrives on the multi-processor system and is assigned to a processor having minimum response time, particularly, for aperiodic tasks. Furthermore, if there are more than one processor with the least response time, then the total utilization of various available processors are calculated and the aperiodic task is assigned to the lowest utilized processor.
2) after the aperiodic task is assigned to a specific processor, the deadline of all periodic tasks are checked simultaneously. If the deadline of periodic tasks, on assigned processors, are smaller than aperiodic tasks then it is migrated to other processors. Hence, the response time of aperiodic task is reduced.
3) the energy consumption of a particular system is reduced if aperiodic tasks are executed with lower speed by identifying the appropriate speed of the processor. By using the proposed algorithm the response time of aperiodic tasks is also kept within bound.

Algorithm 1 describes these steps, at the end of this section. In the remaining part of this section, we describe the TBS algorithm [Section V-A], the task migration technique [Section V-B], and DVFS approach [Section V-C].

A. TOTAL BANDWIDTH SERVER (TBS)

The Total Bandwidth Server (TBS) is a special type of server that assigns the earliest deadlines to aperiodic tasks, as shown in Figure 2 [16], [17]. The virtual deadline (α_c) for the aperiodic tasks is calculated through the formula as specified in Equation 1:

$$\alpha_c = \max(r_n, d_{n-1})\frac{e_n}{u_s} \tag{1}$$

where r_n, e_n represent the arrival and execution time of the n^{th} aperiodic task, respectively; d_{n-1} is the absolute deadline of the previous aperiodic task, and u denotes the total bandwidth of server utilisation. Assume that a task can be represented as $\tau_1(x, y)$ where x and y denote the task's τ execution time and deadline, respectively. There are two tasks $\tau_1(3, 6)$ and $\tau_2(3, 12)$ on processor P_a and similarly $\tau_3(2, 8)$ and $\tau_4(3, 6)$ on processor P_b. According to the TBS algorithm, the total bandwidth of processor P_a and P_b is 0.25. The first aperiodic task $\alpha_{a,1}(2, 3)$ arrives on processor P_a and its virtual deadline, according to TBS algorithm, is 14; which is calculated using Equation 2:

$$v_{a,1} = a_{a,1} + \frac{E_{a,1}}{U_b^{srv}} = 2 + \frac{3}{0.25} = 14 \tag{2}$$

Clearly, the deadline of $v_{a,1}$ is later than the deadline of τ_1 and τ_2 tasks. Therefore, the aperiodic task will be completed after the completion of periodic tasks and results in increasing the execution time of the aperiodic tasks.
The second aperiodic task $\alpha_{a,2}(10, 3)$ arrives on processor P_a and its virtual deadline, according to the Total Bandwidth Server (TBS) algorithm, is 22 as calculated below using Equation 3:

$$v_{a,2} = max(a_{a,2}, v_{a,1}) + \frac{E_{a,2}}{U_b^{srv}} = 14 + \frac{3}{0.25} = 26 \tag{3}$$

Since, the deadline of the aperiodic task is later than the deadline of the periodic task; and is, therefore, not executed immediately. The third aperiodic task $\alpha_{a,3}(26, 3)$ arrives on processor P_a and its virtual deadline is 38 given by the following Equation 4:

$$v_{a,3} = max(a_{a,3}, v_{a,2}) + \frac{E_{a,3}}{U_b^{srv}} = max(26, 26) + \frac{3}{0.25} = 38 \tag{4}$$

This whole scenario is shown visually, using the Gantt chart, in Figure 2.

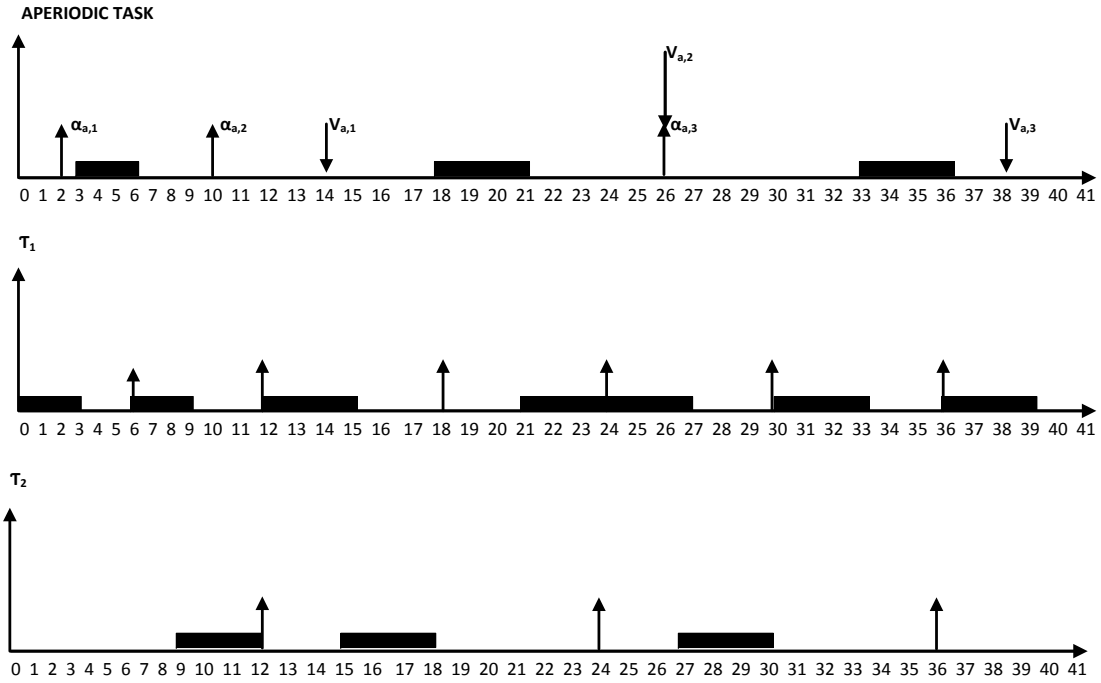

FIGURE 2: An example of total bandwidth server (TBS) [Gantt chart]

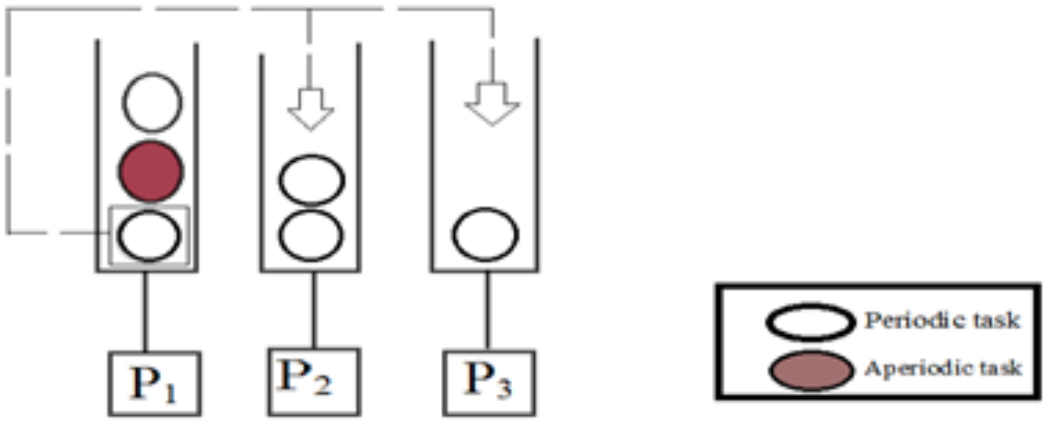

FIGURE 3: The migration technique [Scenario 1]

B. MIGRATION TECHNIQUE

A real-time system consists of periodic and aperiodic tasks. In certain RTS, the priority of an aperiodic task is greater than periodic task, but not essentially; and, therefore, must be executed immediately. An aperiodic task that has lower deadline time than a periodic task on processor P_1 is shown in Figure 3. In this scenario, aperiodic task completes its execution after the periodic task that is violation of the priority rules. Therefore, periodic tasks are being migrated or transferred to other processors where it is scheduled; and the aperiodic task is immediately executed on that particular processor [14], [18].

The migration technique, as shown in Figure 4, describes that the aperiodic task should be migrated to other processor such that the response time of aperiodic task is reduced [19]. This job (execution) is accomplished through migrating the task to those processors which: (a) currently, did not have any aperiodic task; (b) its utilization is the lowest; and (c) the periodic task is still schedulable. The aperiodic task is migrated to that processor which has the lowest level of utilization.

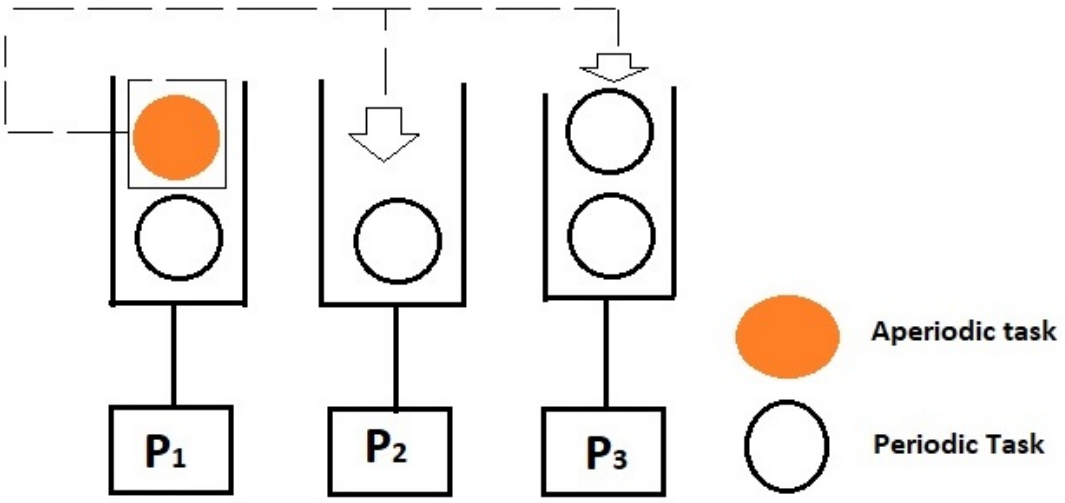

FIGURE 4: The migration technique [Scenario 2]

According to the TBS algorithm, lower the utilization level of a server, lower will be the deadline of aperiodic task and vice versa. Moreover, the deadline of aperiodic task is reduced by increasing the deadline of periodic tasks.

The $k_{t,h}$ aperiodic task arrives at the processor P_a and is represented by $\alpha_{a,k}(a_{a,k}, e_{a,k})$, where $\alpha_{a,k}$ is the k_{th} aperiodic task on a_{th} processor, $a_{a,k}$ is the arrival time and $e_{a,k}$ is the execution time of an aperiodic task. The response time of the aperiodic task can be calculated using the following Equation 5:

$$R_{a,k} = f_{a,k} - a_{a,k} \tag{5}$$

where $f_{a,k}$ represents finishing time and $a_{a,k}$ is the arrival time of an aperiodic task.

Consider an aperiodic task $\alpha_{a,k}$ arrives at processor P_a, its virtual deadline is $v_{a,k}$ and server bandwidth is U_a^{srv} which are calculated by Total Bandwidth Server Equation (1). The periodic tasks are migrated to other available processors if the following conditions hold. Find a periodic task $t_{i,j}$ that has the earliest deadline than the aperiodic task and it is not already migrated to any other processor during its execution time period. If the task does not exist then it is not migrated to any other processor.

Find the processor P_b from the multi-processor system that satisfies the following Equation 6:

$$d_{i,j} \geq \max(t, v_{b,l}) + \frac{c'_{i,j}(t)}{U_b^{srv}} \tag{6}$$

where $v_{b,l}$ is the virtual deadline of last aperiodic task $\alpha_{b,l}$ on the processor P_o and $c_{i,t}$ is the remaining execution time of the periodic task on that processor. The periodic task $t_{i,j}$ is migrated to the processor P_b where it is considered a l + 1^{th} aperiodic task and its virtual deadline is $v_{b,l+1}$. This mechanism is described by Equation 7:

$$v'_{b,l+1} = max(t, v_{b,l}) + \frac{c'_{i,j}(t)}{U_b^{srv}} \tag{7}$$

The aperiodic task $\alpha_{a,k}$ on processor P_a is given an improved virtual deadline $v_{a,k}$ that is calculated using the following Equation 8:

$$v'_{a,k} = max(t, v_{a,k-1}) + \frac{e_{a,k}}{U_a^{srv} + \frac{c'_{i,j}(t)}{t_i}} \tag{8}$$

The periodic job t_i is migrated to processor P_a where it is considered the next job to be executed $t_{i,j+1}$. Now applying second portion of the proposed algorithm, the response time of an aperiodic task is reduced. An example scenario of temporal migration, how it is actually performed, is depicted in Figure 5. To simplify the situation, it is assumed that the availability of aperiodic task on processor P_b is zero. An aperiodic task $\alpha_{a,1}(2, 3)$ arrives on processor P_a and without applying migration approach this task receives deadline of 14 seconds. However, periodic tasks τ_1(3, 6) and τ_2(3,12) deadlines are earlier than the aperiodic task $\alpha_{a,l}$(2,3) which is 14 on processor P_a. Therefore, its migration to processor P_b is necessary because it fulfils the Equation 6. The migrated periodic task τ_1 to processor P_b is considered an aperiodic task on processor P_b. The virtual deadline of this migrated task is computing by using Equation 9 and receives a virtual deadline of 6 as calculated below.

$$v'_{b,1} = max(t, v_{b,l}) + \frac{c'_{i,j}(t)}{U_b^{srv}} = 2 + \frac{1}{0.25} = 6 \tag{9}$$

After the deadline 6, the task is being migrated back to processor P_a from processor P_b. So the virtual deadline of aperiodic task $\alpha_{a,1}$ is improved to 9.2 from 14 according to Equation 10:

$$v'_{a,1} = 2 + \frac{3}{0.25 + \frac{1}{6}} = 9.2 \tag{10}$$

Similarly, the second aperiodic task $\alpha_{a,2}$ arrives at time t = 10 on processor P_a and receives a virtual deadline of 26 according to the TBS algorithm, that is without a migration

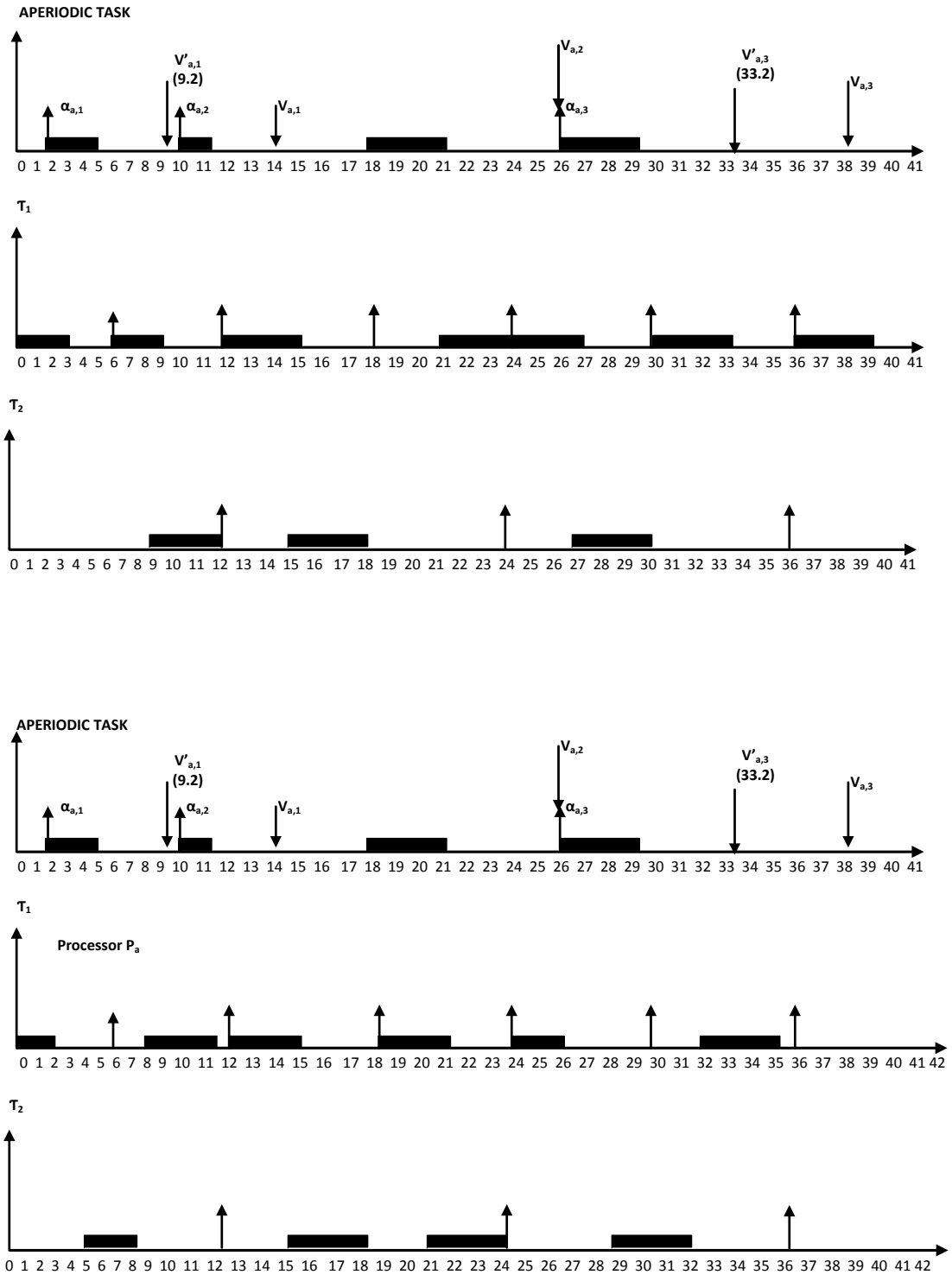

FIGURE 5: An example of temporal migration [Gantt chart] – The aperiodic task τ_1 is being migrated from processor P_a to processor P_b where its virtual deadlines $v_{b,1}$ is reduced from 14 to 6 according to Equations 9; note that after the deadline 6, the task τ_1 is migrated back to processor P_a from processor P_b. [processor P_a - up and processor P_b - down]

approach.

$$v_{a,2} = 10 + \frac{3}{0.25} = 26 \tag{11}$$

Similarly, periodic tasks τ_1 and τ_2 can not be migrated to processor P_b because they do not satisfy Equation 6. The deadline of both periodic tasks are increased when they are migrated to processor P_b and they are not schedulable on processor P_b.

The third aperiodic task arrives on processor P_a at time 26. Its deadline without migration by using TBS technique is 38. As, its deadline is greater than both periodic tasks which are 30 and 36, respectively. Therefore, these periodic tasks will be completed before the aperiodic task.

$$v_{a,3} = 26 + \frac{3}{0.25} = 38 \tag{12}$$

The task τ_1 satisfy Equation 6. Therefore, task τ_1 is being migrated to processor P_b which receives a virtual deadline of 30, according to equation 13.

$$v'_{b,1} = max(t, v_{b,l}) + \frac{c'_{i,j}(t)}{U_b^{srv}} = 26 + \frac{1}{0.25} = 30 \tag{13}$$

The virtual deadline of aperiodic task on processor P_a becomes 33.2 that is less than the deadline of the TBS technique which is 38.

$$v'_{a,3} = 26 + \frac{3}{0.25 + \frac{1}{6}} = 33.2 \tag{14}$$

Note that before migrating an aperiodic task from a particular processor, we check its schedulability on the target processor using Equation 16, as described in Algorithm 2. If the schedulibility test fails, we do not migrate the task; otherwise migration is completed. Furthermore, each migrated aperiodic task is considered a periodic task on the target processor, in order to ensure that the task finishes its execution with the time constraints. Note that Algorithm 2 is also used for initial schedule of all periodic and aperiodic task on each processor.

C. DYNAMIC VOLTAGE AND FREQUENCY SCALING (DVFS)

The third portion of the proposed algorithm is the Dynamic Voltage and Frequency Scaling (DVFS) technique that is applied to the multi-processor systems to minimize their energy consumption [9]. Initially, periodic and aperiodic tasks are assigned priorities based on the Total Bandwidth Server (TBS) algorithm where later tasks have higher priorities than former tasks but keeping in view that periodic tasks are still schedulable [6]. The standard TBS algorithm is tuned in proposed algorithm that allows multi-processor to run at slower speed so that the aperiodic task is scheduled and its response time is also kept within the bound. The Earliest Deadline First (EDF) schedules the periodic and aperiodic task if and only if:

$$U_t = \sum_{i=1}^{m} \frac{e_i}{p_i} \leq 1 \tag{15}$$

where U_t is the total utilization of processor, e_i is the execution time and p_i is the time period of the task. In case, if the multi-processor system has both periodic and aperiodic tasks then its utilization must be less than 1 that is according to the Equation 16:

$$u_p + u_s \leq 1 \tag{16}$$

The utilization of the periodic aperiodic task is less than or equal to 1 because $u_p + u_s \leq 1$ means that the system is running at maximum speed of $f = f_{max} = 1$. Therefore, in this case DVFS algorithms is not applied. As the system is running with its full speed and is idle most of the time. Hence, DVFS technique is applied to the system that restrict it to run in lower possible speed and results in improving the performance of the system by reducing the idle time interval of the processor. Running the system at lower possible speed results in increased response time. The system is tuned to run with an appropriate speed to create the possibility of scheduling mixed tasks and equally kept its response time within bound. Then Equation 16 becomes:

$$u_p + u_s \leq \frac{f_i}{f_{max}} \tag{17}$$

Equation 17 describes the way DVFS approach is applied to run both periodic and aperiodic task with an appropriate speed of the system by keeping the response time within bound. The $\frac{f_i}{f_max}$ is denoted by α_a. The Total Bandwidth Server (TBS) algorithm is modified and the system is run at lower speed α_a. The TBS deadline becomes as given by Equation 18:

$$d_n(\alpha_a) = max(r_n, d_{n-1}) + \frac{e_n}{u_s.\alpha_a} \tag{18}$$

The TBS deadline is delayed according to Equation 19:

$$d_n(\alpha_a) - d_n = \max(r_n, d_{n-1}) + \frac{e_n}{u_s.\alpha_a} - max(r_n, d_{n-1}) + \frac{e_n}{u_s} = \frac{e_n}{u_s}\left(\frac{1}{\alpha_a} - 1\right) \tag{19}$$

The Equation 19 shows that the deadline of aperiodic tasks are increased and the DVFS technique is used to reduce the energy consumption of the multi-processor system. Furthermore, Equation 19 is also used to find suitable frequency for the processor. The deadline of aperiodic task $d_n(\alpha_a) - d_n$ is less than P_s to avoid the performance degradation of system particularly aperiodic tasks. In a multi-processor system, the individual frequency for a single processor system is given by:

$$\alpha_{1a} = \left(1 + \frac{p_s.u_s}{e_n}\right) \tag{20}$$

The speed of the second processor is given by:

$$\alpha_{2a} = \left(1 + \frac{p_s.u_s}{e_n}\right) \tag{21}$$

The speed of m processors is given by:

$$\alpha_{2m} = \left(1 + \frac{p_s.u_s}{e_n}\right) \tag{22}$$

The proposed approach finds out the optimal speed of each processor and then schedule both aperiodic and periodic tasks accordingly. In multi-processor system, the frequency of the aperiodic tasks are marginally less than that of periodic tasks. Therefore, two alternative exists to execute these tasks in multi-processor environment. One approach is to run the aperiodic tasks at full speed of the processor and secondly running them at appropriate speed. In scenario where it is necessary to execute aperiodic tasks immediately then the best approach is to apply our first alternative i.e. executing tasks at full speed. In other cases, the alternative approach is applied, running the system at slower speed, while preserving the energy consumption of multi-processor systems because energy consumption is one the core issues associated with these systems. Additionally, the overall response time of the aperiodic tasks is improved by applying the proposed approach because their deadline is less than or equal to P_s. Similarly, if the deadline of a particular aperiodic task t_1 on a particular processor $p' \in P$ is greater than its deadline on any other processor(s) than p' $(P - p')$; then we migrate aperiodic tasks to other processor(s). Moreover, if possible, energy could be saved through reducing the processor speed using DVFS.

Based on the concepts, presented in Section V-B and Section V-C, we suggest a scheduling algorithm [as described in Algorithm 1] that: (i) executes aperiodic tasks at full speed and migrates periodic tasks to other processors if their deadline is earlier than aperiodic tasks i.e. reduces the response time; and (ii) executes aperiodic tasks with lower speed by identifying appropriate processor frequency without affecting the response time i.e. reduction in energy consumption. The following assumptions were made for the proposed algorithm and its implementation:

1) as uni-processor scheduling techniques cannot be used for global scheduling (with one global queue), therefore, the proposed algorithm works as a partitioning scheduling algorithm.
2) to ensure that the deadline of a task is not missed after it is being migrated to another processor, we estimate each task's deadline, in advance (before it is being migrated), on the target processor.

VI. PERFORMANCE EVALUATION

This section describes the experimental set-up, various evaluation metrics and the major findings of our proposed scheduling technique i.e. "Partitioning Scheduling Algorithm with DVS".

A. EXPERIMENTAL SET-UP

The proposed algorithm and others were implemented in Matlab programming environment. Since, Matlab provides an easy interface to simulate multiprocessor systems as these systems are difficult to implement, practically, due to the lack of such multiprocessor environment for experimentation [20]. Moreover, we tested various techniques using synthetic datasets. First, we produce periodic tasks and calculate their average utilization values. Similarly, the generated task set is then divided and mapped onto multiprocessors. Utilization of all tasks are considered. Moreover, aperiodic tasks are assigned to those processors which are under utilized – lesser periodic load is assigned. As a final outcome, response time of aperiodic task, frequency of the system/processor and corresponding energy consumption is shown in various figures. Priorities of tasks are assigned according to the well-known TBS algorithm.

Furthermore, we used an exponential distribution to generate the set of aperiodic tasks with parameters λ and μ, that denote the tasks inter-arrival and service times, given by $(\frac{1}{\lambda})$ and $(\frac{1}{\mu})$, respectively. Note that in our set-up, μ is static (fixed) and λ is dynamic which varies in order to control the workload p of aperiodic tasks given by $(p = \frac{\lambda}{\mu})$ under a fixed utilization U_p of periodic tasks. Moreover, a normal distribution function was used to generate the set of periodic tasks and each task's actual execution time. The execution times of periodic tasks were assumed in the range of $[b, w]$, where b and w are the best and worst case execution times, respectively. The mean (μ) and the standard deviation (σ) were set to $\frac{w+b}{2}$ and $\frac{w-b}{6}$, respectively. Moreover, in our experiments, we assume that the time delay for switching or migration of tasks and energy consumption of the voltage scaling (overhead) is negligible due to their lower impacts on overall system energy efficiency [21].

B. METRICS

The task parameters, C_i and P_i, were selected, as discussed in Section VI-A, and the proposed scheduling algorithm's performance was evaluated using two well-known performance metrics: (i) response time (seconds); and (ii) energy consumption (KWh). Moreover, the results were compared with several well-known and field proven scheduling techniques including: (a) No DVS (Dynamic Voltage Scaling) technique or simple scheduling algorithm without migration; (b) Total Bandwidth Server (TBS) algorithm [17]; (c) Energy-aware Stochastic Task Scheduling (ESTS) approach [22]; and (d) DVS enabled, Earliest Deadline First (EDF) algorithm [23].

C. RESULTS AND DISCUSSION

The proposed scheduling algorithm's performance results, response time, and their energy consumption are shown in Figure 6 and Figure 7, respectively. It is to be noted that the experimental results are shown by applying the mean approach to ten different iterations.

In Figure 6, the x-axis shows the aperiodic tasks load while the y-axis values represent the normalized response time and energy consumption of the system. The experimental results show that the proposed algorithm's response time is

Algorithm 1: PARTITIONING SCHEDULING ALGORITHM WITH DVFS

Input: task set τ, set of processors P
Output: energy and performance efficient schedule decision

for *tasks* $t, t_1 \in \tau$ *[where* t *is periodic task and* t_1 *is aperiodic task]* **do**
  for *processor* $p \in P$ **do**
    calculate utilization u of p;
  end for
  assign t to the lowest utilized p;
  if $u_t + u_{t_1} \leq 1$ *[*$\forall$ *tasks on* p *– task is schedulable on the target host – using Algorithm 2]* **then**
    if *deadline of aperiodic task* $t_1 >$ *other periodic tasks on* p **then**
      migrate periodic tasks to other processors [Section V-B]
       [the virtual deadline of aperiodic task is delayed using our approach that is less than the period of the server P_s so that the energy requirement is reduced];
      reduce speed of p through DVFS technique if possible [Section V-C];
    else
      if *deadline of* t_1 *on* $p' \in P >$ *deadline of* t_1 *on* $P - p'$ **then**
        migrate aperiodic tasks to other processors
         [the virtual deadline of aperiodic task is delayed using our approach that is less than the period of the server P_s so that the energy requirement is reduced];
        reduce speed of t_1 through DVFS technique if possible;
      end if
    end if
  else
    task is not schedulable on the target processor;
  end if
end for
return *schedule decision*

Algorithm 2: SCEDULABILITY TEST

Input: task set τ, set of processors P, migratable task t
Output: return true if τ or $t \in \tau$ is schedulable

Sort τ with respect to task deadlines ;
for *each processor* $p \in P$ **do**
  calculate utilization U of p with respect to both periodic and apeiodic tasks;
  for *all tasks in* τ **do**
    $U_p = \sum_{i=1}^{m} \frac{e_i}{p_i}$ [$\forall$ periodic tasks];
    $U_s = \sum_{i=1}^{m} \frac{e_i}{p_i}$ [$\forall$ aperiodic tasks];
    $U_t = U_p + U_s$;
    if $U_t \leq 1$ **then**
      return *true*;
    end if
  end for
end for
return *false*

significantly decreased with increase in aperiodic load while reduction ratio of other algorithms are low. Similarly, Figure 7 shows that when the aperiodic load is increased, then the energy consumption of the proposed scheduling algorithm is lower than other field proven algorithms. We believe that our proposed algorithm gives better performance and schedulability of tasks (both periodic and aperiodic tasks) along with the least energy consumption, for multi-processor real-time systems.

Furthermore, our evaluation suggests that the response time and energy consumption could be decreased further, if multiprocessor systems are less utilised. The utilization levels of the processors have a great impact on the energy and performance efficiencies of the multiprocessor systems. In our case, the response time of aperiodic tasks is reduced through transferring/migrating the periodic tasks to less uti-

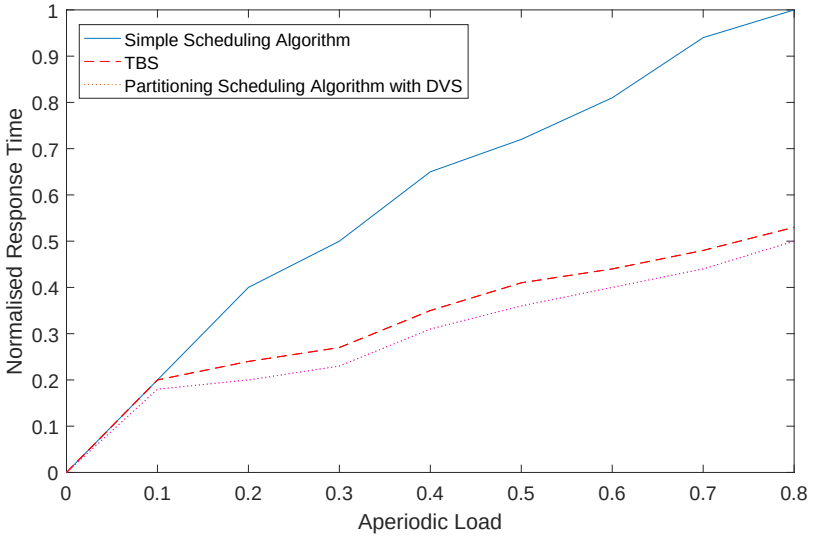

FIGURE 6: Response time [normalised] – lower values are “better"

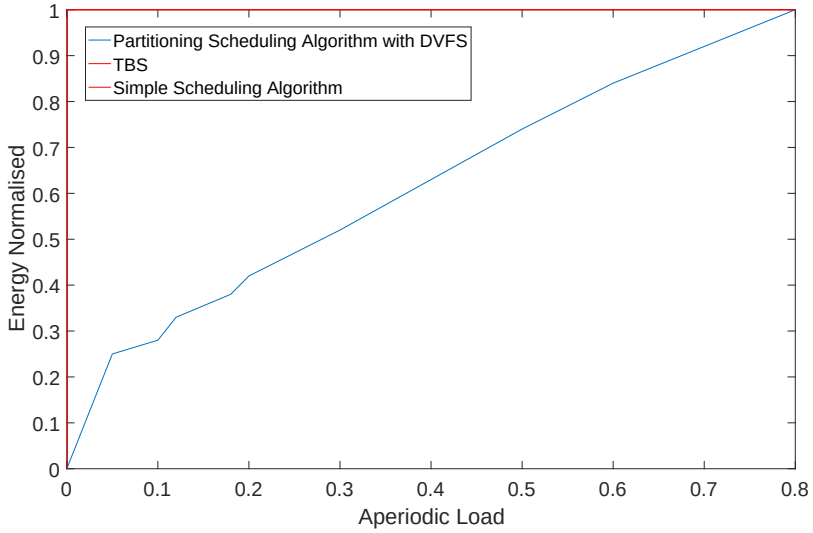

FIGURE 7: Energy consumption [normalised] – lower values are “better"; the energy consumption is adaptive to aperiodic load with our proposed algorithm

lized processors. Consequently, the deadline of the periodic tasks could be increased in the less utilised processors; as less utilised processors are running at a lower speed using the Dynamic Voltage and Frequency Scaling (DVFS) technique, in order to decrease the energy consumption. This shows an existing trade-off between the energy consumption and response time of multiprocessor systems and workload.

As shown in Figure 8 and Figure 9, the mean response time of tasks is minimum as compared to the Total Bandwidth Server (TBS) and No DVFS techniques, where the average service rate (μ) is 0.1 and the number of processors (M) is 2. When the overall utilization level of the system is low then the TBS and our algorithm have the best performance. Moreover, the energy efficiency of our technique is better than both, the TBS and No DVFS, techniques. Conversely, when the system overall utilization level is high then the improvement in mean response time does not exist very much. Therefore, our migration technique is effective, particularly, when the system load is moderate. However, if the number of processors (M) are increased e.g. from 2 to 8; then the improvement in mean response time is remarkable, even if the system load is low [as shown in Figure 9]. This demonstrates that our proposed migration technique is more effective, particularly, for those multiprocessor systems that support large number of processors.

The periodic tasks will have better mean response time in

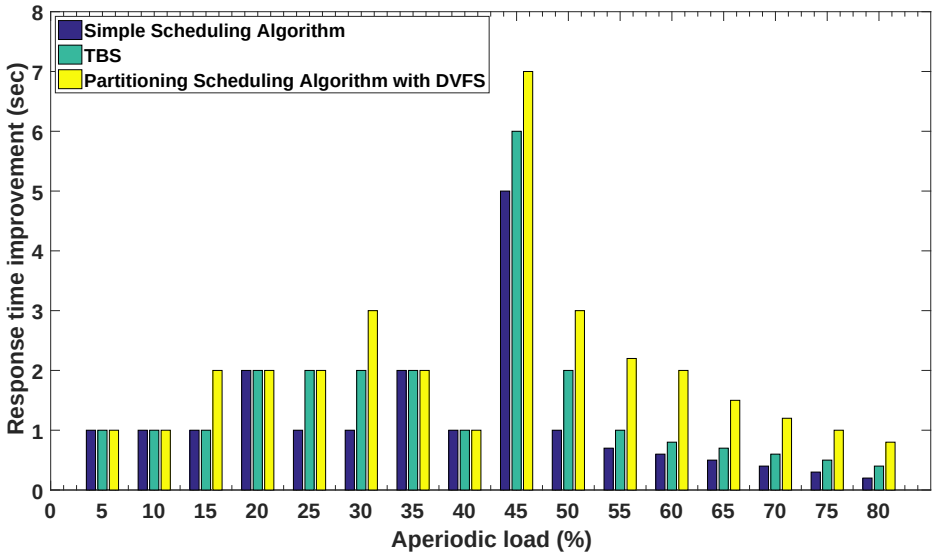

FIGURE 8: Performance improvement of the proposed migration technique and other algorithms when $M = 2$ and $\mu = 0.9$ – higher values are “better"

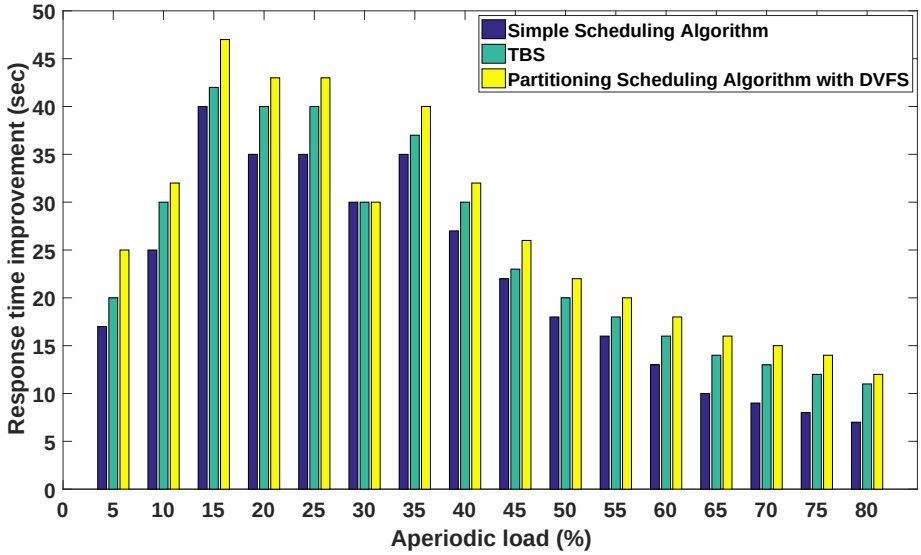

FIGURE 9: Performance improvement of the proposed migration technique and other algorithms when $M = 8$ and $\mu = 0.1$ – higher values are “better"

Energy-Efficient Stochastic Task Scheduling (ESTS) algorithm [22] as compared to our proposed algorithm. The ESTS algorithm chooses tasks based on their schedule length and the available energy budget; and then suitable tasks are assigned to a particular processor. In our proposed algorithm, as periodic tasks are assigned to the least utilized processor, therefore, the mean response time of periodic tasks will be higher than the ESTS algorithm, due to a number of reasons. For example:

1) the ESTS algorithm does not assign higher priorities to aperiodic tasks in the task set. Therefore, the response time of aperiodic tasks will be increased in the ESTS algorithm as compared to our approach that assigns higher priorities to aperiodic tasks;
2) during the execution of aperiodic tasks, there is no migration mechanism in the ESTS algorithm which could migrate periodic tasks with the lowest deadlines. In our proposed approach, an efficient migration mechanism is suggested which is responsible to migrate periodic tasks with the lowest deadlines (than the deadlines of aperiodic tasks) to other processors during the execution of aperiodic tasks; and

3) as we assume homogeneous processors for the evaluation of our proposed algorithm and, moreover, the load is balanced among various processors; therefore, there is no need to estimate task execution times and the energy budget on various processors – that will also take significant time in prediction/estimation. When processors are homogeneous and the load is balanced among various processors; it will take almost similar execution times and, therefore, energy on every processor; thus there is no need to predict tasks energy consumption and execution times. We assume that tasks are sorted based on their execution times and then assigned to least utilised processors, in an order. In our algorithm, the deadline of periodic task is compared with aperiodic task, only; if it is less than the deadline of an aperiodic task then the periodic task is migrated to another processor only if it is schedulable there.

The mean response time of aperiodic tasks, in our approach, is less than the ESTS algorithm. However, the energy consumption of both algorithms is almost similar. In our algorithm, we only calculate the response time of aperiodic tasks while the periodic tasks are not considered. The EDF-DVS algorithm [23] chooses the task which has the earliest deadline, first; and assigns it to a particular processor at the lowest frequency (voltage) in such a way that the task is schedulable. However, periodic, aperiodic and priority of tasks are not considered in EDF-DVS. To be consistent in our experiments, we also assume the priority of periodic and aperiodic tasks the same in our algorithm. Therefore, the mean response time of our algorithm is better than the EDF-DVS algorithm. However, the energy consumption of both algorithms is almost similar.

When the aperiodic load is lighter, then, there is a situation where the least number of tasks are migrated to other processors. In this particular situation, the mean response time of aperiodic tasks of the ESTS algorithm is smaller than other algorithms i.e. the EDF-DVS algorithm and our algorithm. As the aperiodic load increases, then, there are greater opportunities of tasks migration; and, as a consequence, the normalised response time of aperiodic tasks of our algorithm is lesser than other algorithms i.e. the ESTS algorithm and the EDF-DVS algorithm, as shown in Figure 10. Note that Figure 10 describes improvements in response time of various algorithms with respect to the baseline simple scheduling approach.

Similarly, when the aperiodic load is lighter, then, there is a situation where the least number of tasks are being migrated to other processors. Therefore, the normalised energy consumption of the ESTS algorithm is lesser than the EDF-DVS and our proposed algorithm. However, as the aperiodic load increases, then, there are greater opportunities for tasks migrations; and, as a consequence, the normalised energy consumption of aperiodic tasks of our approach is lesser than the ESTS and the EDF-DVS algorithms [23], as shown in Figure 11 below.

Complexity analysis is one of the significant measures to determine the applicability and robustness of a particular algorithm. The worst case complexity of the proposed algorithm for m number of tasks (both periodic and aperiodic) and n number of processors is $\mathcal{O}(mn)$. The best case scenario will occur when all the processors (P) are idle. Moreover, the algorithm complexity will increase when either total number of tasks, processors or both will increase.

VII. RELATED WORK

The literature on scheduling algorithms for multi-processor systems is quite bulky [24], [25], [26], [27], [28]. These systems are able to perform periodic and aperiodic tasks where former jobs are executed first due to their higher priorities. Similarly, in real-time system, the priority of an aperiodic job is higher than other jobs, periodic jobs, and is executed immediately after its arrival. Some of the existing scheduling algorithms, for both aperiodic and periodic tasks, are briefly described below.

The multiprocessor scheduling has been divided into partitioning and global scheduling algorithms as described in Section III. There are a number of scheduling algorithms that deal with global scheduling. Whereas, in partitioning scheduling algorithm, tasks are divided into multiple queues and each queue is assigned to a particular processor. The optimal uniprocessor scheduling algorithm can be used for partitioning scheduling in multiprocessor systems. Moreover, scheduling of aperiodic tasks have been discussed in the literature for multiprocessor systems [2]. However, it reduces the response time of aperiodic tasks on both partitioning and global scheduling algorithms. The authors in [2] developed a scheduling algorithm that is based on EDF and TBS techniques. The proposed approach guarantees the time constraint of aperiodic tasks and minimizes the response time of aperiodic tasks, as well. Moreover, [2] guarantees the time constraint of periodic tasks. However, the response time of periodic tasks are not much reduced.

In the absence of a periodic task, server manages the execution of aperiodic tasks in background using First Come First Serve (FCFS) based mechanism. The Background server uses separate queues for periodic and aperiodic tasks respectively without considering their priorities [29]. On the other hand, polling server executes available jobs based on their priority level. Polling server executes an aperiodic job immediately, due to its highest priority level, if it arrived during server's running cycle otherwise it waits for the next cycle. The response time of periodic jobs is increased if no aperiodic job enters during the cycle of a polling server.

The Priority Exchange and Deferrable Server (PEDS) [30] has a better response time for aperiodic tasks due to its waiting mechanism for the request of these jobs [8]. This approach gives the highest priority to the aperiodic tasks while keeping a schedule for the periodic tasks, simultaneously. Aperiodic tasks are executed immediately after their arrival, in PEDS mechanism, by using priority exchange approach. If aperiodic tasks do not exist in the system then server applies

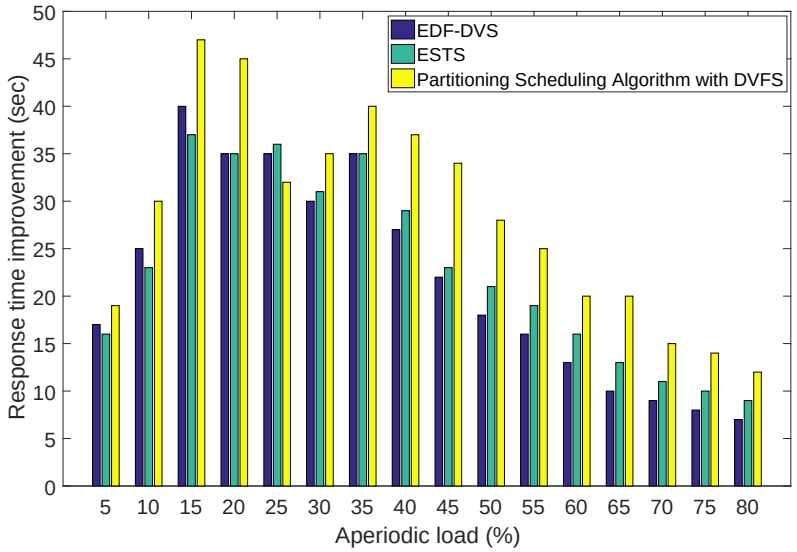

FIGURE 10: Performance improvement of the proposed migration technique and other algorithms – higher values are “better" [improvements are shown with respect to simple scheduling policy]

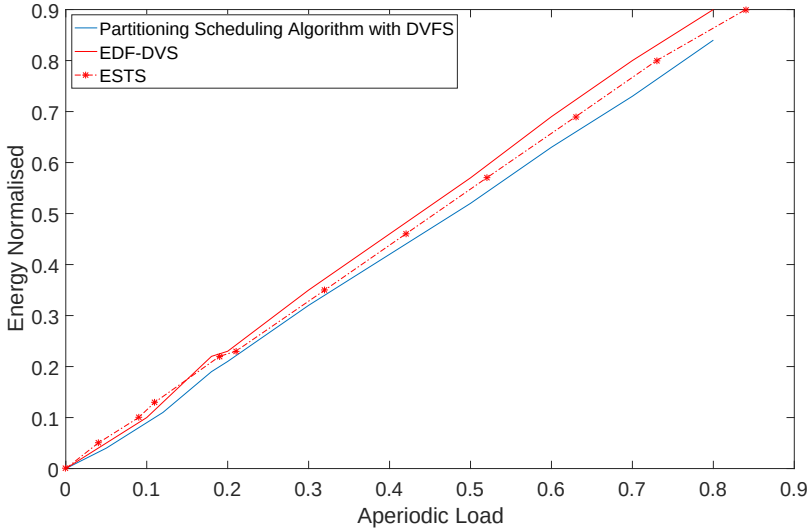

FIGURE 11: Energy consumption of the proposed migration technique and other algorithms – lower values are “better"

the priority exchange mechanism to the pool of available periodic tasks. In this scenario, the server does not wait for the arrival of an aperiodic task, wasting of resources, rather it concentrates on the execution of periodic tasks. These tasks are organized according to the server’s defined schedule usually based on their priority levels. Although, the performance of priority exchange server is better than other approaches but it is very difficult to implement in real environment due to its continuously changing criteria for the tasks’ priority levels. The Earliest Deadline First (EDF) is a mechanism proposed by Dertouzos et al. [31] in 1974. This approach schedules available tasks according to their priority levels that is assigned to a job based on its deadline (time). Among pre-emptive scheduling algorithms available in the literature, the earliest deadline first is the most optimum scheduling technique and it is due to the utilization of the deadline time. However, this algorithm, particularly, fails in scenarios where two or more jobs/tasks have similar deadlines.

The Earliest Deadline Late Server (EDLS) [32] approach describes the process of finding empty or free time slots and elaborates the mechanism to utilize them in the execution of aperiodic tasks. Additionally, the priority of the aperiodic is higher than periodic task. Therefore, aperiodic tasks are executed as soon as possible depending on the availability of empty time slots and periodic tasks are scheduled to be executed next. EDLS is an optimal algorithm that improves the

repose time of the aperiodic tasks but its complexity is very high as compared to other available benchmark algorithms. Improved Priority Exchange (IPE) Server is an approach to enhance the performance of the earliest deadline late server algorithm particularly off-line mode. It finds the unused time slots with EDLS mechanism and then use it to speed-up the execution of aperiodic tasks. If the system has no aperiodic task available for execution then periodic tasks are executed, to conserve its capacity, according to their schedule. During this stage, if aperiodic tasks appeared in the system then execution of the periodic tasks are pre-empted and the aperiodic tasks are executed first, as aperiodic tasks have the highest priority. Like EDLS approach, the complexity of this algorithm is high and is difficult to configure.

A special type server called Total Bandwidth Server (TBS) assigns the earliest deadlines to the aperiodic tasks [17], [26]. Like other approaches, aperiodic tasks have higher priority than periodic tasks and pre-emption mechanism, situation where aperiodic approaches the system when execution of periodic tasks is in progress, is used to abort execution of the periodic tasks. The scheduling of periodic and aperiodic tasks is performed through earliest deadline first algorithm that is based on their required server bandwidth. The virtual deadline of an aperiodic task is calculated by using the formula given in Equation 23:

$$v_{c,l} = \max(a_{c,l}, v_{c,l-1}) \frac{e_{c,l}}{u_c^s} \tag{23}$$

where $v_{c,l}$ is the virtual deadline of l^{th} aperiodic task on processor C, $a_{c,l}$ is the task's arrival time, $e_{c,l}$ is the execution time of l^{th} aperiodic task and $u_{c,s}$ is the total bandwidth of the server.

However, the above techniques cannot guarantee minimum schedule length and energy consumption, concurrently. Moreover, majority of scheduling algorithms schedule tasks based on their average-case runtimes; and do not consider the probability distributions of task runtimes. Kenli et al. [22] proposed a heuristic Energy-aware Stochastic Task Scheduling algorithm called ESTS to solve this issue. The authors have formulated the task scheduling problem as a linear programming (LP) problem; with the objective to maximize the weighted probability of both energy consumption and schedule length under energy consumption budget and deadline constraints. They demonstrated through extensive simulations that ESTS outperforms the well-known algorithms such as EDF-DVS [23], Min-Min, Max-Min [33] by approximately 19.8%, 63.3%, and 36.4%, respectively. However, ESTS is only designed for a single processor system and there is no evidence that task runtimes, essentially, follows a normal distribution. Moreover, tasks may belong to various applications; and there is no discussion on application heterogeneities.

The EDF-DVS algorithm [23] first sorts the tasks set based on the EDF priority scheme i.e. task with the nearest deadline took the highest priority. Secondly, if the deadline of a task can be extended, it is executed on a lower frequency/voltage to minimise energy consumption. Other heuristic based scheduling algorithms such as Min-Min and Max-Min are described in [33]. The Min-Min heuristic sorts the tasks list based on their completion times; and selects the task having the lowest completion time for execution first. The Max-Min algorithm is very similar to the Min-Min technique, however, Max-Min selects the task having the highest completion time for execution first.

VIII. CONCLUSION

The higher energy consumption is the main issue of multiprocessor systems. Several energy aware scheduling techniques have been suggested and applied to decrease their energy consumption. For example, the energy consumption of multi-processor system can be reduced through using the energy aware scheduling techniques; however, as a consequence, the response time of aperiodic tasks could be increased. As a result, the system performance is affected negatively. Rich literature has suggested various scheduling techniques to decrease the energy consumption of these systems, however, reducing the response time of aperiodic tasks is relatively ignored. In this paper, we proposed a scheduling algorithm that decreases the energy consumption of a multi-processor system through using DVFS and migration techniques. Moreover, the response time of aperiodic tasks are also kept with in the bound. The proposed algorithm has the capability to execute tasks at full speed to reduce the response time of aperiodic tasks in time critical situations. Our evaluation suggests improvement to the well-known TBS mechanism in terms of improved response time and the lowest energy consumption.

The contribution of this paper is limited because it only focuses on the response time of aperiodic tasks, which have higher priorities than periodic tasks, while maintaining deadline restrictions when both periodic and aperiodic tasks are running over a multiprocessor system. Furthermore, this work is limited to homogeneous processors; and, as a future work, we will evaluate and extend the proposed scheduling algorithm for heterogeneous systems. Moreover, this will also enable us to evaluate its performance algorithm against other scheduling algorithms such as ESTS [22] and EDF-DVS [23].

ACKNOWLEDGEMENT

This work is financially supported by the Ministry of Higher Education, Malaysia under Fundamental Research Grant Scheme (FRGS) bearing number FRGS/1/2017/TK04/UPM/02/5. Moreover, this work was conducted, as part of the PhD program, under the supervision of Muhammad Zakarya, Rahim Khan and Mukhtaj Khan, at the Department of Computer Science, Abdul Wali Khan University Mardan (AWKUM), Pakistan.

REFERENCES

[1] Anantha P Chandrakasan, Samuel Sheng, and Robert W Brodersen. Low-power cmos digital design. IEICE Transactions on Electronics, 75(4):371–382, 1992.

[2] Björn Andersson, Tarek Abdelzaher, and Jan Jonsson. Global priority-driven aperiodic scheduling on multiprocessors. In Parallel and Distributed Processing Symposium, 2003. Proceedings. International, pages 8–pp. IEEE, 2003.

[3] Jay K. Strosnider, John P. Lehoczky, and Lui Sha. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. IEEE Transactions on Computers, 44(1):73–91, 1995.

[4] Matthias Beckert and Rolf Ernst. Response time analysis for sporadic server based budget scheduling in real time virtualization environments. ACM Transactions on Embedded Computing Systems (TECS), 16(5s):161, 2017.

[5] Brinkley Sprunt, John Lehoczky, and Lui Sha. Exploiting unused periodic time for aperiodic service using the extended priority exchange algorithm. In Proceedings. Real-Time Systems Symposium, pages 251–258. IEEE, 1988.

[6] Thomas D Burd, Trevor A Pering, Anthony J Stratakos, and Robert W Brodersen. A dynamic voltage scaled microprocessor system. IEEE Journal of solid-state circuits, 35(11):1571–1580, 2000.

[7] Dongkun Shin and Jihong Kim. Dynamic voltage scaling of mixed task sets in priority-driven systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(3):438–453, 2006.

[8] Marco Spuri and Giorgio Buttazzo. Scheduling aperiodic tasks in dynamic priority systems. Real-Time Systems, 10(2):179–210, 1996.

[9] Nasro Min-Allah, Samee U. Khan, Xiuli Wang, and Albert Y. Zomaya. Lowest priority first based feasibility analysis of real-time systems. Journal of Parallel and Distributed Computing, 73(8):1066–1075, 2013.

[10] Robert I Davis and Alan Burns. A survey of hard real-time scheduling for multiprocessor systems. ACM computing surveys (CSUR), 43(4):35, 2011.

[11] Jing Li, Jian Jia Chen, Kunal Agrawal, Chenyang Lu, Chris Gill, and Abusayeed Saifullah. Analysis of federated and global scheduling for parallel real-time tasks. In Real-Time Systems (ECRTS), 2014 26th Euromicro Conference on, pages 85–96. IEEE, 2014.

[12] Shelby Funk, Joel Goossens, and Sanjoy Baruah. On-line scheduling on uniform multiprocessors. In Real-Time Systems Symposium, 2001.(RTSS 2001). Proceedings. 22nd IEEE, pages 183–192. IEEE, 2001.

[13] K Subramani. A specification framework for real-time scheduling. In SOFSEM, pages 195–207. Springer, 2002.

[14] Gang Zeng, Yutaka Matsubara, Hiroyuki Tomiyama, and Hiroaki Takada. Energy-aware task migration for multiprocessor real-time systems. Future Generation Computer Systems, 56:220–228, 2016.

[15] Björn B Brandenburg and Mahircan Gül. Global scheduling not required: Simple, near-optimal multiprocessor real-time scheduling with semi-partitioned reservations. In Real-Time Systems Symposium (RTSS), 2016 IEEE, pages 99–110. IEEE, 2016.

[16] Kiyofumi Tanaka. Adaptive total bandwidth server: Using predictive execution time. In International Embedded Systems Symposium, pages 250–261. Springer, 2013.

[17] Shinpei Kato and Nobuyuki Yamasaki. Scheduling aperiodic tasks using total bandwidth server on multiprocessors. In Embedded and Ubiquitous Computing, 2008. EUC'08. IEEE/IFIP International Conference on, volume 1, pages 82–89. IEEE, 2008.

[18] Houssam-Eddine Zahaf, Giuseppe Lipari, and Luca Abeni. Migrate when necessary: toward partitioned reclaiming for soft real-time tasks. In Proceedings of the 25th International Conference on Real-Time Networks and Systems, pages 138–147. ACM, 2017.

[19] Muhammad Zakarya. Energy, performance and cost efficient datacenters: A survey. Renewable and Sustainable Energy Reviews, 94:363–385, 2018.

[20] Nasro Min-Allah, Samee U Khan, Xiuli Wang, and Albert Y Zomaya. Lowest priority first based feasibility analysis of real-time systems. Journal of Parallel and Distributed Computing, 73(8):1066–1075, 2013.

[21] Jheng-Ming Chen, Kuochen Wang, and Ming-Ham Lin. Energy efficient scheduling for real-time systems with mixed workload. In International Conference on Embedded and Ubiquitous Computing, pages 33–44. Springer, 2007.

[22] Kenli Li, Xiaoyong Tang, and Keqin Li. Energy-efficient stochastic task scheduling on heterogeneous computing systems. IEEE Transactions on Parallel and Distributed Systems, 25(11):2867–2876, 2014.

[23] Kyong Hoon Kim, Rajkumar Buyya, and Jong Kim. Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters. In CCGrid, volume 7, pages 541–548, 2007.

[24] Sanjoy Baruah, Marko Bertogna, and Giorgio Buttazzo. Multiprocessor Scheduling for Real-Time Systems. Springer, 2015.

[25] Mayuri Digalwar, Praveen Gahukar, Biju K Raveendran, and Sudeept Mohan. Energy efficient real-time scheduling algorithm for mixed task set on multi-core processors. International Journal of Embedded Systems, 9(6):523–534, 2017.

[26] Doan Duy and Kiyofumi Tanaka. An effective approach for improving responsiveness of total bandwidth server. In Information and Communication Technology for Embedded Systems (IC-ICTES), 2017 8th International Conference of, pages 1–6. IEEE, 2017.

[27] Ayaz Ali Khan and Muhammad Zakarya. Performance sensitive power aware multiprocessor scheduling in real-time systems. Technical Journal UET Taxila (Pakistan), 2010.

[28] Kyong Hoon Kim, Anton Beloglazov, and Rajkumar Buyya. Power-aware provisioning of cloud resources for real-time services. In Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science, page 1. ACM, 2009.

[29] Teguh M Ghazalie and Theodore P. Baker. Aperiodic servers in a deadline scheduling environment. Real-Time Systems, 9(1):31–67, 1995.

[30] Rehan Ahmed, Ayoosh Bansal, Bhuvana Kakunoori, Parameswaran Ramanathan, and Kewal K Saluja. Thermal extension of the total bandwidth server. In VLSI Design (VLSID), 2015 28th International Conference on, pages 47–52. IEEE, 2015.

[31] Vivek Tiwari, Deo Singh, Suresh Rajgopal, Gaurav Mehta, Rakesh Patel, and Franklin Baez. Reducing power in high-performance microprocessors. In Proceedings of the 35th annual Design Automation Conference, pages 732–737. ACM, 1998.

[32] Daniel Casini, Luca Abeni, Alessandro Biondi, Tommaso Cucinotta, and Giorgio Buttazzo. Constant bandwidth servers with constrained deadlines. In Proceedings of the 25th International Conference on Real-Time Networks and Systems, pages 68–77. ACM, 2017.

[33] Tracy D Braun, Howard Jay Siegel, Noah Beck, Ladislau L Bölöni, Muthucumaru Maheswaran, Albert I Reuther, James P Robertson, Mitchell D Theys, Bin Yao, Debra Hensgen, et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed computing, 61(6):810–837, 2001.

View publication stats