Journal of Theoretical and Applied Information Technology
15th September 2021. Vol.99. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DVFS ENABLED GREEDY MECHANISM FOR ENERGY AWARE TASK SCHEDULING IN HIGH PERFORMANCE COMPUTING SYSTEMS

SIDDESHA K[1], JAYARAMAIAH G V[2]

[1]Research Scholar, Department of Electronics and Communication Engineering, Dr. Ambedkar Institute of Technology, Bengaluru, India

[2]Professor, Department of Electrical and Electronics Engineering, Dr. Ambedkar Institute of Technology, Bengaluru, India

E-mail: [1]siddesha.ec@drait.edu.in, [2]gvjayaram.ee@drait.edu.in

ABSTRACT

Nowadays, the high performance computing (HPC) systems are widely adopted in data centers to process and accomplish the tasks which consume more resources. These tasks increase load on the data centers however, HPC systems utilize multicore processors which facilitate faster computing. However, inappropriate use of these multicore systems leads to excessive energy consumption and wastage of computing resource. Dynamic voltage and frequency scaling is one of the promising solution to overcome energy consumption related issue in most of the processor based systems. In this paper, we present a combined energy aware scheduling model to schedule multiple types of heterogeneous tasks along with DVFS for high performance computing platform. We introduce a schedule architecture model which acts as a central scheduler along with DVFS. First of all, the incoming tasks are analyzed based on their deadline constraints and resource requirement. DVFS scheme is applied to adjust the voltage and frequency based on the core's configuration. Later, the proposed greedy energy aware scheduling approach assigns the tasks to the specific processor, based on the response time and probability value of the task. This problem is formulated as an optimization process. Moreover, the proposed technique minimizes the queuing delay, which helps to minimize the delay in scheduling. The comparative analysis shows that, the average waiting time and average energy saving ratio of the proposed approach is improved.

Keywords: *DVFS, Task scheduling, Multicore processors, Energy aware, Greedy mechanism.*

1. INTRODUCTION

Nowadays, the high-performance computing systems equipped with multicore chips have emerged as a promising solution for computing systems. In recent years, the demand for computing systems has increased drastically due to their use in various applications. Power consumption is a foremost concern in these systems which has a significant impact on the system performance. The surplus power dissipation leads to increased cooling costs, reduces battery life, and degrades the system reliability [1]. Recent advancement in modern computing technologies have enabled the multiple power mode which enable the performance and power trade-off. Currently, the research is ongoing in this field and researchers have identified that Dynamic power management (DPM) is a potentially powerful technique to minimize the power consumption [2]. This approach manages shutting off and slowing down the system components which are underutilized or idle. The power controller component makes the decision to switch the device power mode according to the system power consumption status [3]. Similarly, Dynamic voltage and frequency scaling (DVFS) is another prominent technique that has been adopted widely in modern processors to minimize the energy consumption. This approach dynamically varies the frequency and voltage of the processor. Thus, these two schemes are the important for runtime power management in high performance computing systems. Generally, the DPM is applied for peripheral devices such as hard disk, and network interfaces and DVFS is used in CMOS digital ICs such as microprocessors and microcontrollers.

In this work, we focus on DVFS technique for computing systems. Numerous techniques have been developed to further improve the performance of DVFS algorithm. According to the location of DVFS scheduling, the DVFS algorithms are classified as (a) intra-task and (b) inter-task. In intra-task scheduling model, the DVFS analyzes the execution flow of tasks and then adjusts the frequency and voltage of processor in the single task boundary. On the contrary the inter-task scheduling, the DVFS scheduling regulates the frequency and supply voltage based on the tasks [4]. Modern computing technology is leaning towards the heterogeneous computing systems. In [5] authors reported that existing techniques fail to efficiently manage the power consumption in dynamic scenarios where application is unknown. Moreover, these schemes require substantial offline data analysis.

Currently, combined task scheduling, mapping, distribution and load balancing along with DVFS enabled systems have gained huge attraction due to their fascinating performance to minimize the power consumption. These systems are embraced in diverse real-time applications where procuring the expected Quality of service (QoS) evolved as a challenging task. Zhang et al. [12] reported that hardware-based energy-efficient scheduling schemes do not consider the real-time constraints and decreased frequency levels, which increases the execution time resulting in violation of timing constraints.

Moreover, the existing schemes face three challenging issues:

- The existing scheduling schemes focused on the non-optimal global scheduling and partition-based scheduling which doesn't guarantee for set of real-time tasks.
- The existing energy aware scheduling approaches turn-off multiple cores to reduce the energy consumption, however, this model is applicable for low workloads but it fails to accomplish all constraints for high-work load scenarios.
- The existing scheduling schemes are based on the periodic task model and doesn't cover the scenario of sporadic task set which is more general in real-time applications.

In sporadic tasks, the incoming task and task deadline constraints are uncertain. Due to these issues, the DVFS scheduling techniques are further enhanced to consider the real-time and dynamic incoming tasks. In [4] authors suggested that combining DVFS with task allocation and adjustment can mitigate the QoS related issues in multicore systems. However, this amalgamation is nonlinear and combinatorial in nature, raise the complexity to solve the problem. To overcome this issue, authors presented a mixed-integer nonlinear programming problem that solves task-to-processor allocation, frequency-to-task assignment and optional task adjustment. Digalwar et al. [11] introduced energy aware task scheduling approach for multicore systems, called as Leakage Aware Multi-Core Scheduler (LAMCS). This approach mainly schedules mixed task set while considering deadline constraints and response time of tasks. Machine learning based schemes are also used in this field of multicore computing systems such as reinforcement learning for task allocation in real-time systems [7], linear regression [8], deep Q-learning [9] and many more [10].

The increasing demand of high performance computing systems is still increasing hence development of a novel approach which can handle dynamic sporadic tasks is a challenging issue. The existing schemes fail to achieve the expected performance in high workload scenarios. In this work, we focus on developing DVFS enabled dynamic work load scheduling approach for sporadic incoming tasks to minimize the energy consumption. We focus on minimizing the overall power consumption of the system by selecting the optimal processing unit according to the given task set. In order to deal with this issue, we formulate a linear programming problem and presented a greedy selection based model which uses energy efficiency and task handling capacity of processing cores to select the best core.

The rest of the article is organized as follows: section 2 presents the literature review based on DVFS enabled task scheduling, migration and load balancing techniques, section 3 presents the proposed task scheduling solution to improve the energy performance of DVFS enabled computing systems for dynamic heterogeneous task set, section 4 presents the experimental analysis and comparative study to illustrate the robustness of proposed approach for dynamic and high workload scenario. Finally, section 5 presents the concluding remarks about this approach.

2. LITERATURE REVIEW

This section describes the existing schemes of, task scheduling, task allocation, task migration for DVFS enabled systems. During the years, several

schemes have been presented for DVFS enabled multicore systems.

As discussed before, dynamic power management is a well-known technique which is adopted in high performance computing systems to improve the productivity. Kumbhare et al. [2] focused on the productivity of high performance computing systems and exposed that resource underutilization is a challenging task in these systems which remains unaddressed by existing techniques. To overcome this issue, authors introduced a dynamic power management scheme which maximizes productivity of system, resource utilization and job completion rate. This scheme reallocates the power from running job to newly arrived jobs based on the power consumption status. Similar to DPM, in [4] discussed about DVFS and classified it as intra-task and inter-task. Minimizing power consumption for intra-task is challenging. According to this approach, first of all, path based analysis is conducted and the entire profile information about task is extracted which is used for integer linear programming (ILP) problem formulation.

However, managing the heterogeneous tasks require multicore system but according to a study presented in [13] discussed that multicore systems suffer from heat dissipation problem. In order to solve this problem green-hardware and energy aware software solution are highly demanded. Currently, job scheduling is considered as a promising solution for these aspects where incoming tasks are allocated to different cores according to their energy requirements. This scheme considers three characteristics for energy consumption optimization: first of all, it applies DVFS at instruction level, it classifies the CPU and memory intensive workloads for better thermal management, and differentiates heterogeneous cores for efficient scheduling. Moreover, incoming heterogeneous dynamic tasks cannot be handled by these schemes thus Basireddy et al. [5] reported that existing techniques are not considered as a feasible solution for dynamic tasks hence authors introduced an adaptive mapping and dynamic voltage and frequency scaling (DVFS) approach to minimize the energy consumption in multicore systems. On the contrary, this approach mitigates the offline analysis dependencies to utilize this scheduling scheme for dynamic tasks. Digalwar et al. [11] focused on mixed tasks scheduling for multicore processor that considers deadline constraints and response time constraints for real time. In this approach, the static energy consumption is minimized by shutting down the cores based on the predefined threshold. The dynamic energy is saved by applying DVFS according to the requirement. The DVFS prolongs the execution interval which results in reducing the chances of shutdown. Thus, the overall energy consumption of system is increased. The Leakage Aware Multi-Core Scheduler (LAMCS) scheme schedules the tasks in such a way that it attains minimum response time for all tasks and meet the deadline constraints. Similar to this, Zand et al. [14] proposed genetic algorithm (GA) based approach for static and dynamic scheduling schemes. GA based static scheduling model is used when task is ready before runtime. On the contrary, a new GA based heuristic approach is presented if the tasks are incoming after execution of task.

Hajiamini et al. [15] reported that per-core DVFS scheduling is a potential solution to achieve the energy efficiency in multicore systems. The current heuristic DVFS is known to provide fast and suboptimal solution whereas dynamic programming methods solve the smaller sub-problems iteratively. However, these schemes increase the overhead and delay. To overcome this issue, authors presented a dynamic programming framework using Viterbi algorithm by using Energy-Delay Product (EDP) as an objective function to predict the best voltage and frequency levels according to the tasks and energy consumption status. Choi et al. [16] developed a novel framework which combines DPM and DVFS for energy optimization in multicore systems. These schemes are combined adaptively where it considers device features and workload characteristics. Liu et al. [17] presented that cost saving is a crucial challenge in modern computing systems. Task scheduling is known as a promising technique to obtain the desired outcome for this issue. Hence, authors presented task scheduling approach for multicore embedding systems by considering time and resource constrains to minimize the cost. Bao et al. [18] developed a lightweight runtime scheme to exploit the properties of power profile corresponding to the processor. This approach is based on the one-time profiling of target CPU and also, it categorizes loop-based segments according to compile-time categorization. The combined model helps to determine the frequency and number of cores required to execute the tasks for given region.

Reddy et al. [19] discussed that heterogeneous multicore system generate the mixed loads due to resource sharing. The excessive backlog of generated resource lead towards the increase in the energy consumption hence, runtime resource

management is considered as one of the important aspect to achieve the energy efficiency. The existing schemes do not exploit the characteristics of inter-cluster cores hence, in this work authors introduced run-time management model which performs thread-to-core mapping according to the performance requirement and unoccupied resources. Some techniques used hybrid schemes to deal with scheduling related issues. Based on this assumption, Jeevitha et al. [20] focused on energy consumption minimization in cloud data service centers where high performance computing systems performs several tasks for diverse applications. In this work authors presented the hybrid approach for scheduling, where shortest job first and round robin algorithms with Vibrant Quantum are combined and proposed shortest round vibrant queue (SRVQ) algorithm.

3. PROPOSED MODEL

Previous section reported that existing schemes are not suitable to deal with the dynamic incoming tasks. In this section, we present a DVFS enabled task scheduling scheme for dynamic incoming tasks. The DVFS algorithm plays a significant role for minimizing the energy consumption for data centers where it scales the voltage and frequency dynamically. Moreover, we formulate the problem in such a way that an efficient scheduling can be obtained by using optimization technique. To accomplish this objective, we adopt greedy scheme which is having better optimization rate according to the designed optimization problem.

3.1 Task Model

In this work, we assume a high performance computing system having multi-core processors with heterogeneous configuration. The cores are denoted as m and a task set is denoted as $\mathcal{T}$, which is divided into multiple tasks as $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_m$ for each core using best fit heuristic approach. We assume that incoming tasks are sporadic and doesn't have interdependency among simultaneous tasks. A specific task set $\mathcal{T}_k$ is formulated using n mixed tasks which can be expressed as:

$$\mathcal{T}_k = \{t_{1,k}(p_{1,k}, w_{1,k}), \ldots, t_{n,k}(p_{n,k}, w_{n,k})\} \quad (1)$$

where $t_{i,k}$ represents the i^{th} individual task in the given task set $\mathcal{T}_k$. This task has its own execution period and a worst case of execution, where it requires maximum frequency of processor to operate. Thus, the execution time varies inversely with the clock frequency. In each period of simulation, the task generates an instance which is called as job. This job is expressed as $t_{i,k}^{j}$ which is the j^{th} job of task $t_{i,k}$. Each job has some timing configurations such as task arrival time, execution time and deadline. The actual execution time of any specific task is expressed as $A_{i,k}^{j}$, and utilization of task set $\mathcal{T}_k$ is computed by using worst case execution time (w) and execution period (p), and it can be expressed as:

$$U_k = \sum_{i=1}^{n} \frac{w_{i,k}}{p_{i,k}} \quad (2)$$

Here, we compute the hyperperiod which is a smallest of time after this period all the periodic patterns of all tasks are repeated. We multiply the positive integer h with this hyperperiod to get the h^{th} hyperperiod. This can be computed as:

$$hyp(\mathcal{T}_k) = LCM(p_{1,k}, p_{2,k}, \ldots, p_{n,k}) \quad (3)$$

3.2 Energy Model

In this section, we present the energy consumption model of the processor. We adopt the processor energy model from existing works [1], [8]. It is assummed that in a multi-core processor, cores may operate with different power settings. For example, NVIDIA Tegra 3 processor uses two types of transistors in fabrication of cores, which has different dynamic and leakage power ratios [9]. The dynamic power of a CMOS processor can be computed as:

$$P_{dp} = c.f.V^2 \quad (4)$$

where c denotes the capacitance value, f is the clock frequency and V is the operating voltage. Generally, the main cause of leakage power is subthreshold leakage current I_{subn} which is expressed as:

$$P_{lp} = V.I_{subn} \quad (5)$$

Let us consider a task which has execution parameter τ_n at voltage V_n and frequency f_n (Frequency is normalized according to exectution time and voltage/frequency setting, where maximum voltage is V_{max} and maximum frequency is f_{max} respectively). The energy consumption for this task can be expressed as:

$$E_n = \rho\, V_n\, \tau_n + \lambda\, f_n\, \tau_n\, V_n^2 \quad (6)$$

Where λ and ρ denotes the total contribution in power consumption by dynamic and leakage power at the maximum voltage/frequency setting. In these multicore processors, both dynamic and leakage power is consumed during the execution of instructions and idle state of the processor respectively. By adding the power consumption of each processor, we can obtain the overall power consumption of the system.

3.3 Proposed Scheduling

This section presents proposed energy aware task scheduling for high performance computing systems like servers of a datacenter. Here, we assume each server of a datacenter has heterogeneous core's for incoming tasks processing. We consider a scenario where M number of processors are present and each processor is denoted as S_j, where $1 \leq j \leq M$. These processors are suitable to process the V types of tasks. We consider a task type i that is processed at j^{th} server, which has deadline constraint denoted as $B_{i,j}$, where $1 \leq i \leq V$, $1 \leq j \leq M$. The processing time for task type i at processor S_j is expresed as μ_{ij}. Number of tasks which are assigned to the processor are denoted as N. The task structure is expressed as:

$$N = \sum_{i=1}^{V} n_i \qquad 1 \leq i \leq V \tag{7}$$

The system has a central scheduler which performs the task assignment to the differnet cores. Figure 1 depicts a task scheduler architecture. Initially, the DVFS scheme is applied to adjust the voltage and frequency based on the incoming task and core configuration. The tasks are processed based on first-come-first serve concept. In case, if the core is busy then tasks are queued up. Here, we set the queuing of S_j for i type of tasks equal to the task deadline B_{ij}. The total number of i types of tasks assinged to processor S_j are denoted as x_{ij}. The central scheduler schedules the incoming tasks in a scheduling vector represented as X_j. The scheduling vector X_j is denoted as:

$$X_j = V \times \sum_{i=1}^{V} x_{i,j} \tag{8}$$

where $\sum_{i=1}^{V} x_{i,j}$ represents total number tasks assigned to S_j. This vector is in a matrix form which has elements either 0 or 1, which represents the task sequence. The row of the matrix denotes the task type and column denotes the time slot in which the tasks are scheduled in the processor. Let us consider an example where three types of four tasks are scheduled for processor S_j.

Thus the scheduled vector can be represneted as:

$$X_j = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{9}$$

In this matrix, rows-1, 2 and 3 shows the task types-1, 2 and 3. The columns indicates that, two tasks of type-1 are sent to the processor in the first time slot, type-2 task is allocated in next time slot and task type-3 is allocated to the processor in the fourth time slot. In this work, the average queuing delay for i type of tasks at S_j is denoted as $\tau_{i,j}$, the average response time is denoted as T_w for a considered total makespan of T as $T = \{T_{w1}, T_{w2}, \ldots, T_{wj}\}$ and queued tasks are denoted as $w_{i,j}$.

Now, we formulate the enegy consumption problem for the high performance computing system. The optimizaiton problem is designed as an integer programming problem where the main objective of this problem is to minimize the energy consumption. The power consumed by processor S_j to accmplish i type of task is dentoed as P_{ij}. Similarly, the overall energy consumed by all the processors to finish the assinged tasks is denoted by E and can be computed as:

$$E = \sum_{i=1}^{V} \sum_{j=1}^{M} \mu_{i,j} * P_{i,j} * x_{i,j} \tag{10}$$

Here, our aim is to minimize the energy consumption. Hence, we define the objective function as:

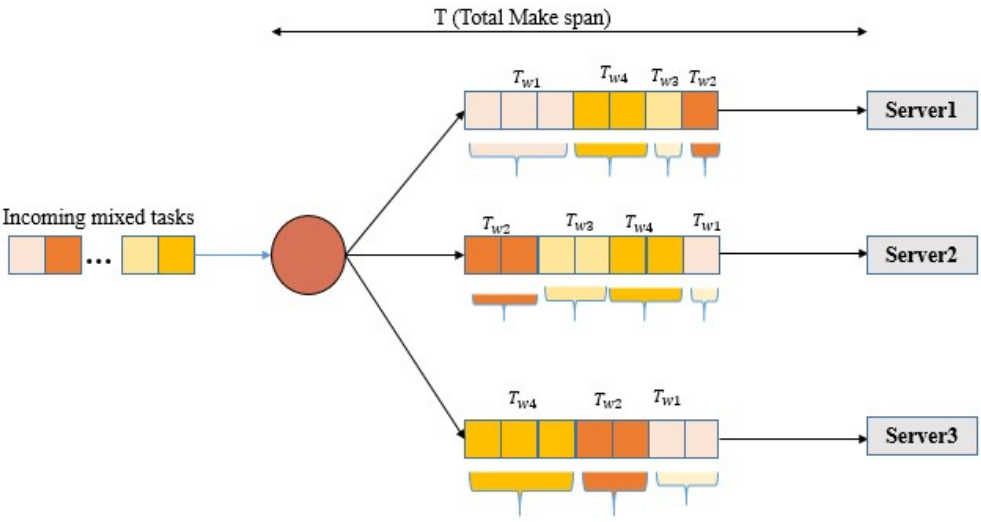

Figure 1: Scheduling Architecture

$$\min_{x} \quad E = \sum_{i=1}^{V}\sum_{j=1}^{M} \mu_{i,j} * P_{i,j} * x_{i,j} \tag{11}$$

$$\text{Subject to} \quad \sum_{j=1}^{M} x_{i,j} = n_i,\ x_{i,j} \le B_{i,j} - w_{i,j}$$

Each server has specific set of tasks and server ensures that all tasks need to be accomplished in the given deadline constraints. As mentioned before, the average response time for each task is given as T_w, which is defined by summing the avearage queuing delays and average processing delays, which can be expressed as:

$$T_w = \frac{\sum_{j=1}^{M}\left[\sum_{i=1}^{V} T_j X_j \omega + x_{i,j}\tau_{i,j}\right]}{N} \tag{12}$$

where T_j denotes the processing time vector as, $T_j = \left(\mu_{1,j}, \mu_{2,j}, .. \mu_{V,j}\right)$, and ω denotes the weight vector which is denoted as:

$$\omega = \left(\sum_{i=1}^{V} x_{i,j} - 1, \sum_{i=1}^{V} x_{i,j} - 2, \ldots 1, 0\right)^{T} \tag{13}$$

In this model, when there is very less load on the processor and no backlogged tasks are present in the queues then the upcoming tasks are assigned to the available server without causing any queuing delay. In this scenario, the optimization problem is given as:

$$\min_{x} \quad E(x) = \sum_{i=1}^{V}\sum_{j=1}^{M} \mu_{i,j} * P_{i,j} * x_{i,j} \tag{14}$$

$$\text{Subject to} \quad \sum_{j=1}^{M} x_{i,j} = n_i,\ x_{i,j} \le B_{i,j}$$

Here, T_w is limited to average processing time and task allocation sequence, denoted as:

$$T_w = T_j X_j \omega \tag{15}$$

Further, The average response time given in equation 12 can be rewritten, for single type tasks $(V = 1)$ as:

$$T_w = \sum_{j=1}^{M} \frac{\mu_j}{2} x_j\left(x_j - 1\right) \tag{16}$$

where $\sum_{j=1}^{M} x_j = n$

At this stage, we have obtained the task to be assigned to specific processor based on response time of task. Furthermore, we focus on assigning a specific task from set of jobs to specific processor by introducing a probability factor computation which considers waiting time, running time and

job-size. First of all, we consider waiting time probability, where wait time of job in queue and average waiting time is cosnidered as:

$$P_{waiting\ time} = \frac{t_q}{T_{avg}} \quad (17)$$

Where t_q denotes the queuing time of the job and T_{avg} is the average waiting time in the system. Similarly, we consider run-time based probability computation which is given as:

$$P_{runtime} = \frac{r}{R_{avg}} \quad (18)$$

Where r denotes the average run time of the task, and R_{avg} is the average run time of all jobs in the machine. Finally, we measure the probability based on the job size which is expressed as:

$$P_{Jobsize} = \frac{n}{NumTask_{avg}} \quad (19)$$

Where n represents the remaining tasks in the job set and $NumTask_{avg}$ denotes the average remaining tasks in each job set in the entire queue. Based on these three parameters, we calculate a probability value for the task as:

$$P = P_{waiting\ time} * P_{runtime} * P_{Jobsize} \quad (20)$$

Here, the task which has the higher probability value in the tasks sorted by greedy mechanism, can be assigned first to the most suitable processor. The computing machine consist of several heterogeneous cores. Heterogeneous cores will possess different computing capabilities in processing the tasks. Even these cores may operate with different power settings. In this set of multiple cores, the computation core which has the highest computation capacity, handles the maximum number of tasks is considered as the most suitable core for assigning the tasks. This computation unit even consumes less energy for processing the task. So, selection of proper processor core to process the incoming task is an important aspect. In this case of scheduling the optimization module follows a greedy mechanism where it sorts the processing unit based on the energy efficiency and the task is assigned to that unit, until the cores of processing unit are fully occupied. The steps of complete process of proposed approach are illustrated in algorithm 1.

Algorithm 1: Proposed Scheduling Scheme

Input: Processor and Task configurations
Output: Energy consumption
Steps:
(1) Define a mixed task model.
(2) Computing the task utilization rate by using WCET and execution period.
(3) Generate periodic pattern of incoming task for hyperperiod.
(4) Define the DVFS energy consumption model.
(5) Consider a task structure for multicore processors scenario with M processors, V types of mixed tasks.
(6) Initiate with first-come – first serve to allocate tasks, once tasks are queued, construct a scheduling vector based on total number of tasks and processors.
(7) Compute the energy requirement to accomplish the assigned tasks.
(8) Construct an optimization problem to minimize the energy consumption.
(9) Compute the average response time based on queuing delay, and processing delay.
(10) Reduce the response time for each task to reduce the energy requirement.
(11) Also compute the task probabilities based on their waiting time, run time and job size.
(12) Measure the final probability of each task to assign to the specific processor.
(13) Assign high probability task to the most suitable processor.
End

4. RESULTS AND DISCUSSION

In this section, we present the discussion about experimental results using proposed apporach and compared the performance with existing schemes.

Table 1: Physical Machine Configuration Parameters

Physical Machine	CPU	Min. Power	Max. Power
1	16	210 w	300 w
2	52	420 w	600 w
3	40	350 w	500 w

The proposed approach is tested on cloud data center where hetereogenous clusters are present. The virtual machines are considered as the multiple cores which are having different configurations.

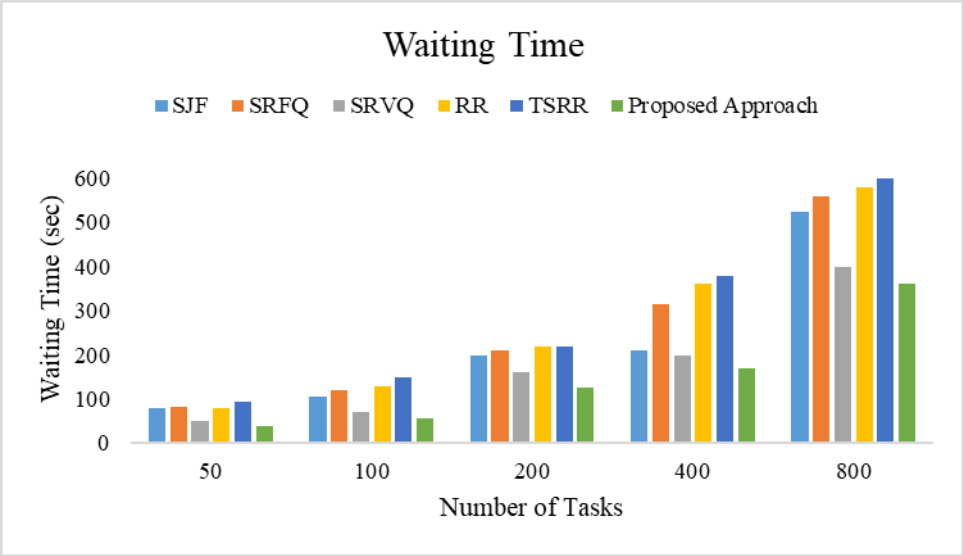

Figure 2: Waiting Time Comparison

Table 2: Waiting Time Comparison Parameters

Tasks	SJF	SRFQ	SRVQ	RR	TSRR	Proposed Approach
50	80	82	50	80	95	40
100	105	120	70	130	150	55
200	200	210	160	220	220	125
400	210	315	200	360	380	170
800	525	560	400	580	600	360

The main operating system is considered as physical machine in which all virtual machines are allocated. The configuration of physical machine is presented in Table 1. In order to measure the performance of proposed scheduling approach, we consider Waiting time and Energy consumption, as the performance measurement metrices.

- **Waiting time:** It is a time duration where the job has to wait in the queue because of ongoing tasks on the all available cores.
- **Energy consumption:** It is a measurement of total energy consumption for specific job to accomplish the task. The overall energy consumption of the machine is obtained by summing up the energy consumption by each job.

We have considered a task set which have the total number of tasks as 50, 100, 200, 400 and 800. These tasks are assigned to processors (VMs). First of all, we measure the performance in terms of waiting time for the task before scheduling it to any core. Figure 2 shows the performance of proposed approach in terms of waiting time and Table 2 presents the corresponding parameters. According to this experiment, we obtained that, the existing schemes require more time to schedule, hence the waiting time increases. The performance of proposed approach is compared with existing techniques discussed in [20]. The average waiting time for these techniques is obtained as 224s, 257.4s, 176s, 274s, 289s and 150s using SJF, SRFQ, SRVQ, RR, TSRR and Proposed Approach. The conventional scheduling schemes such as shortest-job-first, and round robin doesn't consider the resource requirement, deadline constraints and energy consumption status of the cores. Due to these issues, the average waiting time of these techniques increases. Similarly, the improved schemes such as SRFQ, SRVQ and TSRR improves the performance of these techniques but for high workload scenarios these techniques do not provide the satisfactory results. Whereas the proposed approach is able to handle the dynamic and high workload scenarios. The increased waiting time has the significant impact on the energy consumption. Thus, we measure the energy consumption and compare the obtained performance with existing techniques as depicted

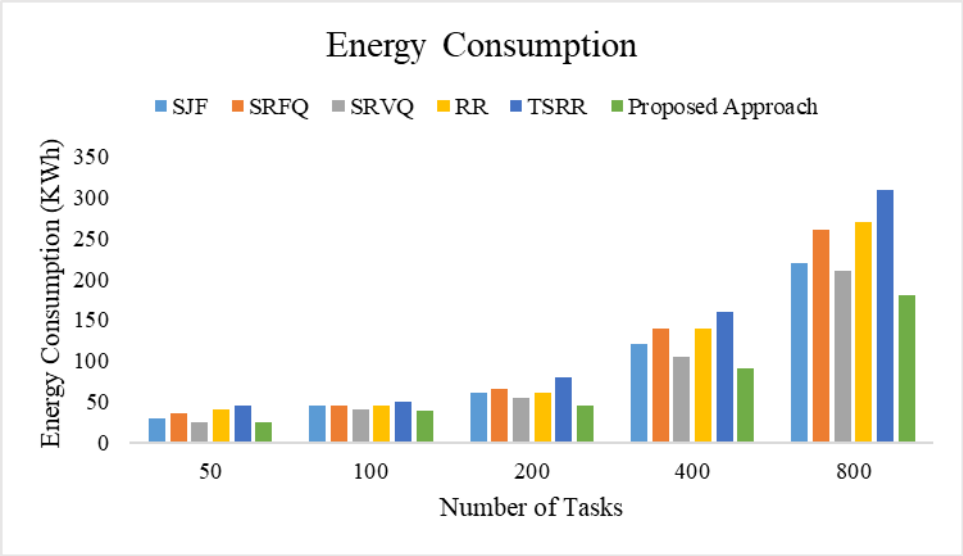

Figure 3: Energy Consumption Performance

Table 3: Energy Consumption Performance Parameters

Tasks	SJF	SRFQ	SRVQ	RR	TSRR	Proposed Approach
50	30	35	25	40	45	25
100	45	45	40	45	50	38
200	60	65	55	60	80	45
400	120	140	105	140	160	90
800	220	260	210	270	310	180

in Figure 3. Table 3 presents the corresponding energy consumption performance parameters. According to this experiment, we obtained the average energy consumption as 95kwh, 109kwh, 87kwh, 111kwh, 129kwh and 75.6kwh using SJF, SRFQ, SRVQ, RR, TSRR and Proposed Approach, respectively. The increasing number of tasks leads to increase in energy consumption. Moreover, the increased waiting time increases the switch on time of the core which consumes additional energy. This experiment shows that proposed approach achieves better performance by scheduling the jobs efficiently.

Further, we extend the experimental analysis and measured the energy consumption performance for varied number of tasks and processors. The obtained normalized energy consumption performance is compared with the existing techniques discussed in [21]. In this experiment, number of tasks are considered as 10, 20, 40, 80, 100 and 200 for varied number of processors as 3, 6, 9 and 12. The obtained performance for these experiments is depicted in Figure 4 and Figure 5 for varied tasks and processors respectively. The Figure 4 shows the comparative analysis in terms of total energy saving percentage for varied number of tasks. As the number of tasks are increasing, the energy consumption rate also increases. This scenario shows the significant performance improvement for more number of tasks. The average energy saving percentage for the given task set is obtained as 11.33, 12.18 and 13.22 using NCM, Improved NCM and Proposed Approach, respectively. The performance parameters of this experiment are presented in Table 4.

Similarly, we extend further, the experiment for varied number of processors for 200 tasks. Figure 5 depicts the performance of energy saving of this experiment and corresponding performance parameters, presented in Table 5. According to this experiment, we varied the number of processors and measured the energy saving performance for 200 tasks. The experiment shows that, the average performance of NCM, Improved NCM, and Proposed Approach as 11.62, 14.05, and 14.46, respectively. The obtained results indicate that, increased number of cores helps to improve the energy saving ratio for dynamic workload.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

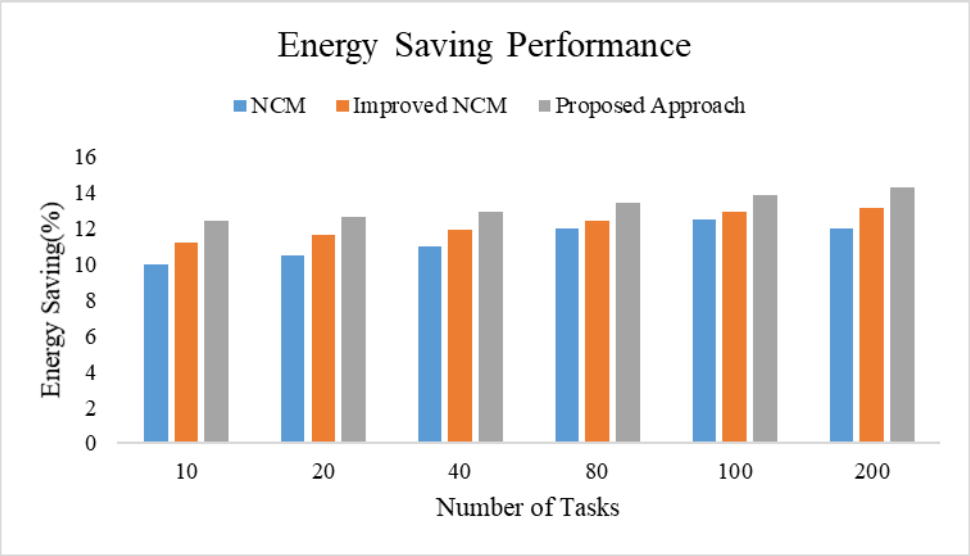

Figure 4: Energy Saving Performance for Varied Number of Tasks (12 Processors)

Table 4: Energy Saving Performance Parameters for Varied Number of Tasks

Tasks	NCM	Improved NCM	Proposed Approach
10	10	11.2	12.4
20	10.5	11.6	12.6
40	11	11.9	12.9
80	12	12.4	13.4
100	12.5	12.9	13.8
200	12	13.1	14.25

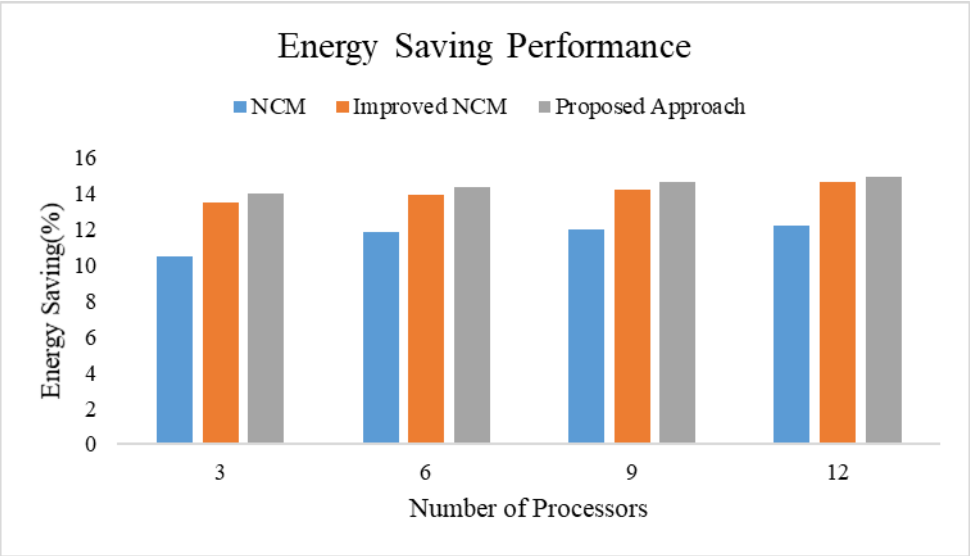

Figure 5: Energy Saving Performance for Varied Number of Processors (200 Tasks)

Table 5: Energy Saving Performance Parameters for Varied Number of Processors

Processors	NCM	Improved NCM	Proposed Approach
3	10.5	13.5	14
6	11.8	13.9	14.35
9	12	14.2	14.6
12	12.2	14.6	14.92

5. CONCLUSION

Currently, the demand of high performance computing systems has increased drastically, where energy consumption becomes a serious issue which degrades the system performance. Several schemes have been implemented to overcome this issue of excessive energy consumption. In this work, we focused on minimizing the energy consumption for heterogeneous DVFS enabled multiple cores used in high performance computing systems. We presented an optimization function to select the best core based on their energy efficiency and applied a greedy selection mechanism, which considers the deadline constraints, highest computation capacity and energy efficiency to select the best solution for task processing. An extensive simulation is carried out, which shows a significant improvement in the performance from proposed scheduling approach for DVFS enabled high performance computing systems like data centers. This scheme doesn't consider the migration of VMs. Our future study will focus on development of a robust approach for dynamic VMs migration in high performance computing systems.

ACKNOWLEDGMENT

This work is supported by the Research center, Department of electronics and communication engineering, Dr. Ambedkar Institute of Technology, Bengaluru, India.

REFERENCES

[1] Hajiaminia, S., & Shirazib, B. A., "A study of DVFS methodologies for multicore systems with islanding feature," *Advances in Computers*, 119, 35-71, 2020.

[2] Kumbhare, N., Akoglu, A., Marathe, A., Hariri, S., & Abdulla, G., "Dynamic power management for value-oriented schedulers in power-constrained HPC system," *Parallel Computing*, 99, 102686, 2020.

[3] Attia, K. M., El-Hosseini, M. A., & Ali, H. A., "Dynamic power management techniques in multi-core architectures: A survey study," *Ain Shams Engineering Journal*, 8(3), 445-456, 2017.

[4] Qin, Y., Zeng, G., Kurachi, R., Li, Y., Matsubara, Y., & Takada, H., "Energy-efficient intra-task DVFS scheduling using linear programming formulation," *IEEE Access*, 7, 30536-30547, 2019.

[5] Basireddy, K. R., Singh, A. K., Al-Hashimi, B. M., & Merrett, G. V., "AdaMD: Adaptive mapping and DVFS for energy-efficient heterogeneous multi-cores," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10), 2206-2217, 2019.

[6] Mo, L., Kritikakou, A., & Sentieys, "Energy-quality-time optimized task mapping on DVFS-enabled multicores," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11), 2428-2439, 2018.

[7] ul Islam, F. M. M., Lin, M., Yang, L. T., & Choo, K. K. R., "Task aware hybrid DVFS for multi-core real-time systems using machine learning," *Information Sciences*, 433, 315-332, 2018.

[8] Gupta, M., Bhargava, L., & Indu, S, "Dynamic workload-aware DVFS for multicore systems using machine learning," *Computing*, 1-23, 2020.

[9] Zhang, Q., Lin, M., Yang, L. T., Chen, Z., Khan, S. U., & Li, P., "A double deep Q-learning model for energy-efficient edge scheduling," *IEEE Transactions on Services Computing*, 12(5), 739-749, 2018.

[10] Pagani, S., PD, S. M., Jantsch, A., & Henkel, J., "Machine learning for power, energy, and thermal management on multi-core processors: A survey," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(1), 101-116, 2018.

[11] Digalwar, M., Raveendran, B. K., & Mohan, S., "LAMCS: A leakage aware DVFS based mixed task set scheduler for multi-core processors," *Sustainable Computing: Informatics and Systems*, 15, 63-81, 2017.

[12] Zhang, D., Guo, D., Chen, F., Wu, F., Wu, T., Cao, T., & Jin, S., "TL-plane-based multi-core energy-efficient real- time scheduling algorithm for sporadic tasks," *ACM Transactions on Architecture and Code Optimization (TACO)*, 8(4), 1-20, 2012.

[13] Kumar, N., & Vidyarthi, D. P., "A novel energy-efficient scheduling model for multi-core systems," *Cluster Computing*, 1-24, 2020.

[14] Zand, H. V., Raji, M., Pedram, H., & SharifAbadi, H. H., "A genetic algorithm-based tasks scheduling in multicore processors considering energy consumption," *International Journal of Embedded Systems*, 13(3), 264-273, 2020.

[15] Hajiamini, S., Shirazi, B., Crandall, A., & Ghasemzadeh, H., "A dynamic programming framework for DVFS-based energy-efficiency in multicore systems," *IEEE Transactions on Sustainable Computing*, 5(1), 1-12, 2019.

[16] Choi, J., Jung, B., Choi, Y., & Son, S., "An adaptive and integrated low-power framework for multicore mobile computing," *Mobile Information Systems*, 1-11, 2017.

[17] Liu, J., Li, K., Zhu, D., Han, J., & Li, K., "Minimizing cost of scheduling tasks on heterogeneous multicore embedded systems," *ACM Transactions on Embedded Computing Systems (TECS)*, 16(2), 1-25, 2016.

[18] Bao, W., Hong, C., Chunduri, S., Krishnamoorthy, S., Pouchet, L. N., Rastello, F., & Sadayappan, P., "Static and dynamic frequency scaling on multicore CPUs," *ACM Transactions on Architecture and Code Optimization (TACO)*, 13(4), 1-26, 2016.

[19] Reddy, B. K., Singh, A. K., Biswas, D., Merrett, G. V., & Al-Hashimi, B. M., "Inter-cluster thread-to-core mapping and DVFS on heterogeneous multi-cores," *IEEE Transactions on Multi-Scale Computing Systems*, 4(3), 369-382, 2017.

[20] Jeevitha, J. K., & Athisha, G., "A novel scheduling approach to improve the energy efficiency in cloud computing data centers," *Journal of Ambient Intelligence and Humanized Computing*, 12(7), 1-11, 2020.

[21] Maurya, A. K., Modi, K., Kumar, V., Naik, N. S., & Tripathi, A. K., "Energy-aware scheduling using slack reclamation for cluster systems," *Cluster Computing*, 23(2), 911-923, 2020.