Received February 21, 2021, accepted March 21, 2021, date of publication April 5, 2021, date of current version April 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3070868

Energy Reduction Through Memory Aware Real-Time Scheduling on Virtual Machine in Multi-Cores Server

MOHAMMAD A. ALQUDAH[1], IQRA AHMED[2], FAHAD AHMAD[3], SHAHID NASEEM[4], AND KOTTAKKARAN SOOPPY NISAR[5]

[1]Department of Basic Sciences, SBSH, German Jordanian University, Amman 11180, Jordan
[2]Department of Computer Science, Institute of Management Sciences, Lahore 54700, Pakistan
[3]Department of Basic Sciences, Deanship of Common First Year, Jouf University, Sakaka 72341, Saudi Arabia
[4]Department of Information Sciences, University of Education, Lahore 54700, Pakistan
[5]Department of Mathematics, College Arts and Science, Prince Sattam Bin Abdulaziz University, Wadi Aldawaser 11991, Saudi Arabia

Corresponding author: Kottakkaran Sooppy Nisar (n.sooppy@psau.edu.sa)

ABSTRACT Not only weighty energy usage pose issues for the environment, but it also raises server maintenance costs in data centers. The massive task with the various power control functions in computer components was made to minimize energy consumption. Increasing consumption of energy in data server environments means that data centers will have high maintenance costs. Various geo-distributed data centers are starting to grow in an age of data proliferation and information growth. Energy management for servers is now demanded for technological, environmental, and economic reasons. In this environment, the main memory is a major energy consumer, not less than the processor. At the same time, an energy-efficient task scheduling strategy is a viable way to meet these goals. Unfortunately, mapping Virtual Machine (VM) resources to the Main Memory (MM) demands to achieve good performance by minimizing the energy consumption within a certain limit is a huge challenge. This paper simulates energy-efficient task scheduling algorithms in a heterogeneous virtualized environment using real-time virtual machine scheduling to resolve the issue of energy consumption. Using a simulator Real-Time system SIMulator (RTSIM), several hardware-based scheduling algorithms are implemented to observe VM memory scheduling efficiency to save memory energy. The simulation results show that, compared to current energy-efficient scheduling methods Rate Monotonic (RM), Earliest-Deadline-First (EDF), and Least-Laxity-First (LLF), helps to reduce energy consumption and improve performance. It is also observed that memory-aware energy management architecture reduces energy and memory consumption efficiently by using EDF scheduling algorithms. In particular, EDF saves approximately 58.3 percent of memory energy than conventional systems that cannot benefit from memory-aware energy management algorithms. The energy efficiency of the algorithms continues to improve as the level of server consolidation rises. We also implemented the EDF scheduling algorithm in Xen's Credit scheduler to see if the simulation outcomes can be simulated on physical systems. Results of simulation and deployment are equated, and comparable outcomes are achieved. We also identified that shared memory between virtual machines deliberately affects memory's energy consumption based on the implementation.

INDEX TERMS Energy reduction, energy consumption, virtual machine, memory aware, multi-core, real-time scheduling.

I. INTRODUCTION

Because of the growing size of data centers, the increasing energy consumption of computer systems is essential. The high energy consumption is causing environmental problems and increasing the maintenance costs of the servers. To reduce energy consumption, significant effort has been made with different energy management features in computer components. In the era of data deluge and explosion of information, a large number of geo-distributed data centers are beginning

The associate editor coordinating the review of this manuscript and approving it for publication was Xinyue Xu.

to emerge. Future data centers' infrastructure must be sustainable and energy-efficient to meet the increasing demand for massive data processing [1], [2]. For technical, financial, and environmental reasons, servers' energy management is now necessary [3].

An unintended effect of the internet's growth is the increased use of energy by servers and the infrastructure that supports them. Internet data centers are often referred to as an important source of consumption of energy increases, particularly in the United States. ''Lawrence Berkeley'' US Laboratory scientists recently discovered that up to 7 terawatt-hours (TWh) of electricity was used by websites and other data servers in the US in 1999, more than Internet infrastructure itself [4].

The SoCs today are ''heterogeneous architectures'' that integrate ''HWAs and CPUs.'' Special purpose HWAs and general-purpose CPU cores are widely used in SoCs due to their ability to quickly and energy efficiently perform specific operations. CPU cores and GPUs are often integrated into smartphones. Hardwired HWAs are implemented in a wide range of SoCs. Many of the application planning in CPU are multi-core systems in previous research [5], [6].

Recently, by virtualizing most of the VM's created in a single computer for a maintenance facility and efficient use of computer resources. It is well known that due to over-provisioning, most data center servers are often underused. Through Virtualization, these underused servers will be consolidated into physical servers. The technology changed in the processors, increased the support for VM virtualization and advanced the use of more underused systems and servers [7].

The requirement of memory has increased by a large amount to offer enough memory for each VM. This gives the memory system a high consumption of energy. With a large amount of memory, consumption of energy can exceed that in processors in server systems. For example, in recent studies, 128 gigabytes (GB) main memory and 16 processors in commercial servers, Processors only account for 28 percent of the total energy demand and 41 percent of memory. Therefore, the main factor in creating energy-efficient servers is to reduce memory energy consumption [7].

The basic demand for saving energy of memory is to understand physical memory properties first. The whole physical memory is usually made up of a small number of blocks of memory; every unit controls the energy. This unit is commonly referred to as a Synchronous Dynamic Random-Access Memory (SDRAM) technology and is called a node in this study. Every whole memory node can be used in one of the energy-saving modes. The power-saving mode empowers memory nodes to disperse less energy without any damage to data compared to the normal operating mode. The energy mode should be switched back to actual operating mode if data is retrieved in a memory node. This change in energy mode causes a significant delay in most cases [8].

The essential way to deal with memory control dissemination decrease is to put memory nodes that are relied upon to have generally long inactive occasions in one of the energy-saving modes by using the physical memory properties. A touch of the complete physical memory is assigned to each VM in the complete physical memory is assigned to each VM in the server consolidation environment. When a VM is running, just memory hubs containing the dispensed memory pages are retrieved. Not other memory nodes are then retrieved and can be used to diminish memory control dissipation in one of the energy-saving modes. Only one VM runs at a specific time in traditional single-processor systems. Therefore, the memory nodes used by a VM are the actual origin of energy utilization for the memory. However, numerous processor cores approach the memory simultaneously in multi-core systems because they simultaneously run a number of VMs. Depending on the VM that runs together, the energy consumption of the memory varies dynamically. A sophisticated scheduling policy must reduce the memory nodes' number accessed by these VMs to save memory energy [9], [10].

This study focuses mainly on the problem of multi-core computer systems' energy planning. It is quite challenging to map VM resources to the MM demands to achieve good performance by minimizing the energy consumption within a certain limit. This research presents a number of essential features for energy consumption on the memory and simulates a memory energy management architecture to provide an energy-efficient plan for VM's in multi-core systems. This paper goes through several algorithms for VM scheduling based on this architecture to achieve the goal of energy reduction from the memory system.

The paper's organization is as follows: section 2 describes the background and related work for the random-access memories, virtual machines, and scheduling algorithms. In section 3 explains material and methods. The experiments and results are analyzed and discussed in section 4. Finally, the conclusion and future work are presented in section 5.

II. RELATED WORK

In this section, a literature study is carried out to throw light on different researchers' attempts to manage energy consumption by researching precise scheduling algorithms for memories with virtual machines in multi-core processors. Several related researches experimentally highlighted the applications, which have proved to be a great source of guidance to come up with the proposed idea.

The law of MOORE implies that computer systems become more complex by providing supplementary functionality to the same chip. This trend is strongly correlated, and to maintain scalability, hardware engineers put a great deal of effort into designing these systems' architecture [11].

Julius, *et al.* introduced battery-powered autonomous robots called ''Unmanned Aerial Vehicles'' as a software stack for energy computation in high-performance embedded systems. They proposed these systems to overcome the issue of energy consumption and timing constraints in most heterogeneous systems. According to them, their proposed system is a multi-version system with equivalent functional behavior in the case of input and output response and some non-functional behavior, including time and energy consumption. To Increase the complexity of reducing the overall energy consumption, they used different compiler flags and different functionally equivalent algorithms. As the frequency cannot be continuous and dynamic consumption, therefore, In their proposed system, they measure the impact of frequency on static energy consumption required to perform an activity [12].

Kai and Dong, presented the Forward Listing Scheduling heuristic system for scheduling the tasks without backtracking. In the proposed system, four schedules have been selected based on the sorting strategy for the lowest-energy consumption as a heuristic solution. The concept of depth-first search and breadth-first search as a tie-breaking rule is used for scheduling because one is not outperforming the other. They use one of them for time reduction and the second for energy reduction [12].

Saad *et al.* proposed real-time embedded system for energy-aware scheduling of independent tasks and extensive surveys. Their proposed system introduced Dynamic Voltage and Frequency Scaling (DVFS) for energy saving because they slow the processing speed due to the energy consumption function. When the CPU sets the voltage on multi-core processors for real-time task scheduling, the DVFS mechanism suggests numerous strategies to save extra energy, as, in these processors, each core has a different processing energy and energy consumption [13].

Mohanamuraly and Stafferlbach, proposed Blocking-Aware-Based Partitioning (BABP) algorithm that guarantees the parallel tasks to access the same shared resources for assigned tasks in the same core with multi-core systems. This algorithm also assures to share parallel tasks with no shared resources to different cores for parallel execution [14].

Mascitti *et al.* proposed a novel-energy-aware scheduling strategy for running runtime activities in the embedded systems. They used their proposed technique for on-line segregating heuristics using EDF scheduling that is used only for task creation and suspension and for performing a single task placement to ensure scalability of runtime activities uploaded in the system. They also used the proposed strategy to achieve the minimum anticipated energy consumption for the execution of these activities to avoid the unexpected transformation that vitiates the recital of the task [15].

III. BASIC STRUCTURE

A. MAGNETO-RESISTIVE RANDOM-ACCESS MEMORY

"Magneto-Resistive Random-Access Memory (MRAM)" has been developed with several difficulties since the 1980s. The ultimate goal is to achieve non-volatile working memories to save energy consumption from conventional volatile working memories such as SRAM and DRAM. However, all non-volatile memories, including MRAM, were confronted with a dilemma of non-volatility and high energy consumption in their active mode because non-volatility led to high consumption of writing energy. Therefore, they have been used as data storage, and none of them overcame the historical dilemma for busy applications. This is one of the crucial reasons why MRAM has not had big markets so far [16], [17].

B. DYNAMIC RANDOM-ACCESS MEMORY

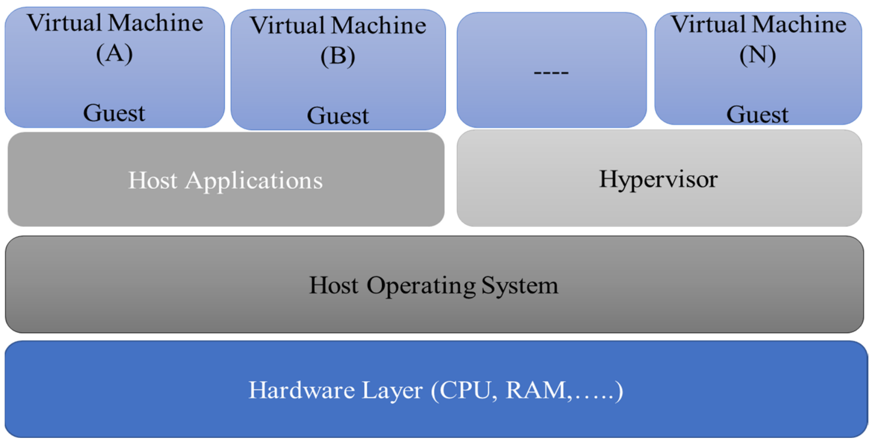
"Dynamic random-access memory (DRAM)" is a type of random-access semiconductor memory that stores every bit of data within an integrated circuit in a separate, small capacitor. The capacitor can go ON or OFF. These two states represent the two values of a bit, commonly referred to as 0 and 1. The electrical load leaks slowly on the condenser so that the data on the chip is lost without intervention. An external refresh circuit is required by DRAM to prevent this, which writes back the data in the capacitor to load its actual state. Thus, it is directly opposite to the SRAM, in which refreshing of data is not required. DRAM is volatile memory (vs) as opposed to flash memory. Non- volatile memory because it loses its data quickly when energy is removed. However, DRAM shows limited data reminisce. In DDR3 SDRAM, numerous functioning and energy modes, such as energy down and self-fresh modes, are deactivated for certain subcomponents. If one of these low-energy modes has a memory node, less energy is dissipated during data storage in any case; the memory nodes ought to be changed to standby mode, where all subcomponents are empowered, read/write to work. This energy change involves an extensive progress deferral and builds memory dormancy, prompting a noteworthy disintegration in framework execution [18], [19].

C. VIRTUAL MACHINE

The virtual machine-based reduced architecture is using Xen VM monitor (VMM) for energy memory management. Xen's fundamental function is to produce multiple safe and detached runtime settings on a single computer. It produces virtual devices such as processors, disks, and networks when creating a new VM. It is possible to run more than two virtual processors and an SMP operating system on a single VM. VMware uses co-planning techniques to plan this kind of virtual processor. Now, Xen does not limit the schedule of more than two VM-owned virtual processors. It is assumed that all VM except the isolated driver domain (IDD) use a single virtual processor [20]. In Fig. 1, the VM leading architecture has been shown [21], [22].

D. MEMORY SCHEDULER

The high latency of access to high-capacity off-chip memory and the limited memory bandwidth in modern systems have made main memory a key performance limit. Main memory is typically shared by applications running in a multi-core system in the different core (or hardware contexts). Requests for these applications contain the off-chip memory bandwidth that leads to interference. Commonly used memory controllers use the First Ready First Come Served memory planning policy. This makes use of the row buffer by giving


Virtual Machine (A) Guest
Virtual Machine (B) Guest

Virtual Machine (N) Guest
Host Applications
Hypervisor
Host Operating System
Hardware Layer (CPU, RAM,…..)

FIGURE 1. **Virtual machine architecture.**

priority to row hits over row misses/conflict. Older applications are then prioritized over newer applications [23].

E. MEMORY-AWARE VIRTUAL MACHINE SCHEDULING FORMULATION

This section formally defines the problem of VM memory scheduling in this section. The system depends on MN Memory-nodes and J is a processor, and J should be $\geq$ 2. Let's say VMT is for all VM and VMT_i for all the VM runs on the ith Processor. For the VM that executes on the ith processor at time t a new equation can be established for VM is VMT_i^t. So $J(t)$ can be explained as in equation (1)

$$\mathrm{J(t)} = \bigcup\nolimits_{\mathrm{i}=1}^{\mathrm{J}} \mathrm{A}\left(\mathrm{VMT_i^t}\right) \tag{1}$$

where a (VMT) is the access set of the VM.

$P(t)$ the energy demand at time t in memory depends on the mode of operating of the system and nodes of memory as well. More clearly, $P(t)$ defines as in equation (2)

$$P(t) = \alpha.\,|J\,(t)| + \beta.|MN - J(t)| \tag{2}$$

For saving mode of energy, define β and for standby mode define α for the Energy Consumption (EC) of nodes of memory. Therefore, the total energy consumption by the node of memory, EC from 0 to a final fixed time T *can be found by equation (3).*

$$EC = \int_0^T P(t)dt \tag{3}$$

Also, to minimize consumption, equation (4) can be used.

$$Min\ EC = \int_0^T |J\,(t)\,|dt \tag{4}$$

The current version to minimize the consumption of energy is lies between the NP COMPLETE. That is why some of the well-known scheduling algorithms have been defined from which it can is possible to reduce the consumption of energy used by memory.

F. MEMORY-AWARE VIRTUAL MACHINES SCHEDULING EXPLANATION

The execution sequence of VM in multi-core systems affects the consumption of energy of memory. For instance, if there are two processor cores and four memory-nodes in a computer system shown in Fig. 2(a) and 2(b). It shows the system has two different VM execution sequences and energy memory consumption. A shaded box shows a VM running in Fig. 1, which illustrates the VM ID and the blue box's access set. $J(t)$ shows the value at each unit time. It assumes that the energy demand of a memory-node for a unit time in standby mode is one. Energy-saving mode ignores memory's energy consumption that shows the memory's energy consumption per unit of time. The total memory energy consumption without memory energy management is 32 because all integrated nodes should be 8 times standby. This approach enables the total energy memory consumption to be reduced. The actual energy saving depends on the order in which the VM is executed. If VM is run in the order in Fig. 2(a), memory's total energy consumption shall be reduced to 24. The reason for our work is that it is possible to reduce the total energy consumption in memory by changing the timing of the given VM in multi-core systems. For example, if the sequence of execution of the same VM as shown in Fig. 2(b) is changed, the total energy demand was reduced to 16. The reduction in energy is the outcome in the reduction of the number of memory nodes in standby mode by resetting the VM execution sequence [24]. The global minimum value for energy consumption can be measured through equation (4) by considering all VMTs. Therefore, a new scheduling policy must be developed to rearrange the VM execution sequence to save further memory energy, and the aim of this VM memory plan is to lessen the standby memory nodes' number [25].

$$\mathrm{Min}_{\{t\}\in \mathrm{T}} \int_0^T |J\,(t)\,|dt \quad \text{when a} < \text{t} < \text{b}$$

$$\mathrm{Min}_{\{t\}\in \mathrm{T}} \int_0^T |J\,(t)\,|dt \quad \text{when1} < \text{t} < 8$$

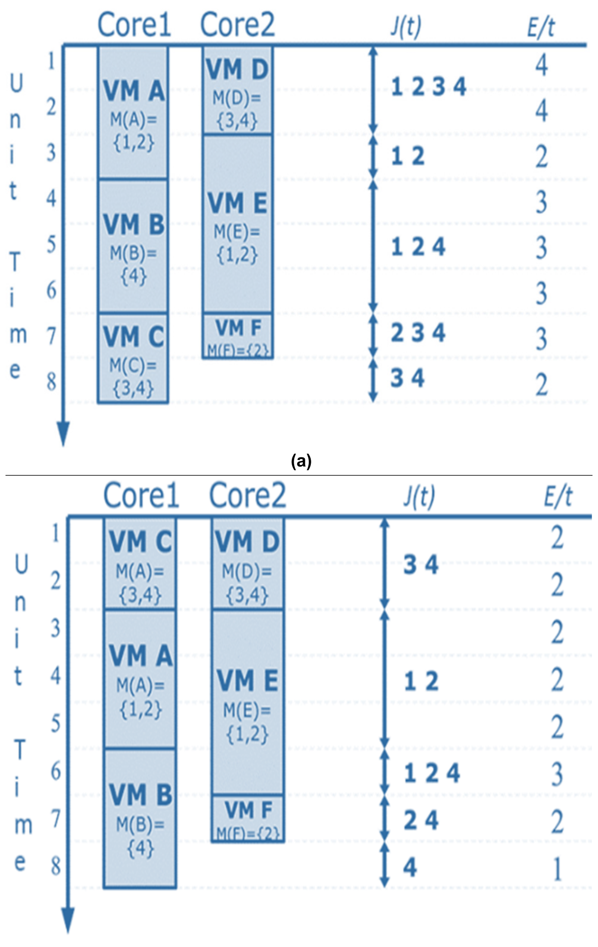


FIGURE 2. **(a). Energy consumption for different VM execution sequences Scenario 1. (b). Energy consumption for different VM execution sequences scenario 2.**

$$\text{Min} \rightarrow 17 \quad \text{when t} \rightarrow 2$$

A key consideration in the VM scheduler's design can be ensured by fairness between VMs shown in Fig. 1. A memory-aware scheduling functions are added to the Xen 's Credit scheduler in our memory energy management architecture instead of creating a new scheduler. Therefore, the scheduler selects a VM from the VM with credits and resides in the current queue. This rearranges the VM running queue temporarily and certifies that the memory scheduler has a similar impartiality level as the Xen credit scheduler [26].

G. SCHEDULING ALGORITHMS

Some well-known algorithms are available to schedule a VM's memory described in Fig. 3, such as RR, RM, EDF, LLF, and some state-of-the-art conventional schedulers are available, but every scheduler has its own terms and use. Most of the tasks in "Real-Time Operating Systems (RTOS)" are periodic. Periodic data are primarily in real-time from sensors, servo control, and monitoring systems. These periodic tasks use most of the processor's computer energy in RTOS. There are many simultaneous periodic tasks with different time constraints in a real-time control system. [27]. These time constraints include time of release (r_i), worst-case execution time (C_i), period (t_i), and deadline (D_i) for each task T_i. Real-time embedded systems have time limits related to the system output. The scheduling calculations are utilized to determine which assignment will be performed if more than one undertaking is accessible in the prepared line. The working framework must guarantee that each task is initiated at its right rate and complies with the time constraint [28].

1) STATIC OR OFFLINE SCHEDULING

"Offline scheduling algorithm" chooses a task to be performed with reference to a foreordained timetable, which is rehashed after a specific time interim. For instance, on the off chance that we have three assignments, T_a, T_b, and T_c, T_a will dependably perform first, and then at point T_b, and after that T_c.

2) DYNAMIC OR ON-LINE SCHEDULING

In "On-line scheduling", an assignment is performed concerning its need, which is resolved continuously as indicated by explicit principles and undertaking needs amid execution.

a: STATIC PRIORITY ALGORITHM

In "static priority," if the kth deployment of a T_1 task is higher than the kth deployment of T_2 as indicated by a predefined scheduling task. All the more formally, in the event that Task T_1 with $J(1, K)$ is higher than T_2 with $J(2, K)$, and T_n $J(n, K+1)$ will dependably have a higher need than $J(2,K+1)$. The rate monotonic scheduling algorithm is a standout amongst the best instances of a static algorithm.

Rate Monotonic (RM) Scheduling:

The algorithm "Rate Monotonic Scheduling (RM) " is a simple regulation that allocates priorities to various errands depending on their time period. This is an assignment with the shortest span, and an assignment with the longest span has the lowermost precedence for accomplishment. Since an assignment's time does not transform so that its precedence does not transform over time, the Monotonic rate is a fixed precedence algorithm. Primacies are managed before the accomplishment takes place, and overtime is not changed.

The algorithm works on preemptive regulation. Preemption takes place on a given processor when higher precedence assignments block lesser precedence assignments from the accomplishment. This blockage occurs in a given task set due to the precedence level of different assignments. Rate monotonic is a preemptive algorithm in which an assignment with a shorter time comes during accomplishment; it becomes higher precedence and can preempt a currently executed assignment. Priorities are allocated in RM according to the period. The precedence of an assignment is, contrariwise, commensurate with its timer. The assignment with the lesser period has the maximum precedence, and the assignment with the maximum period will be the bottommost. A certain

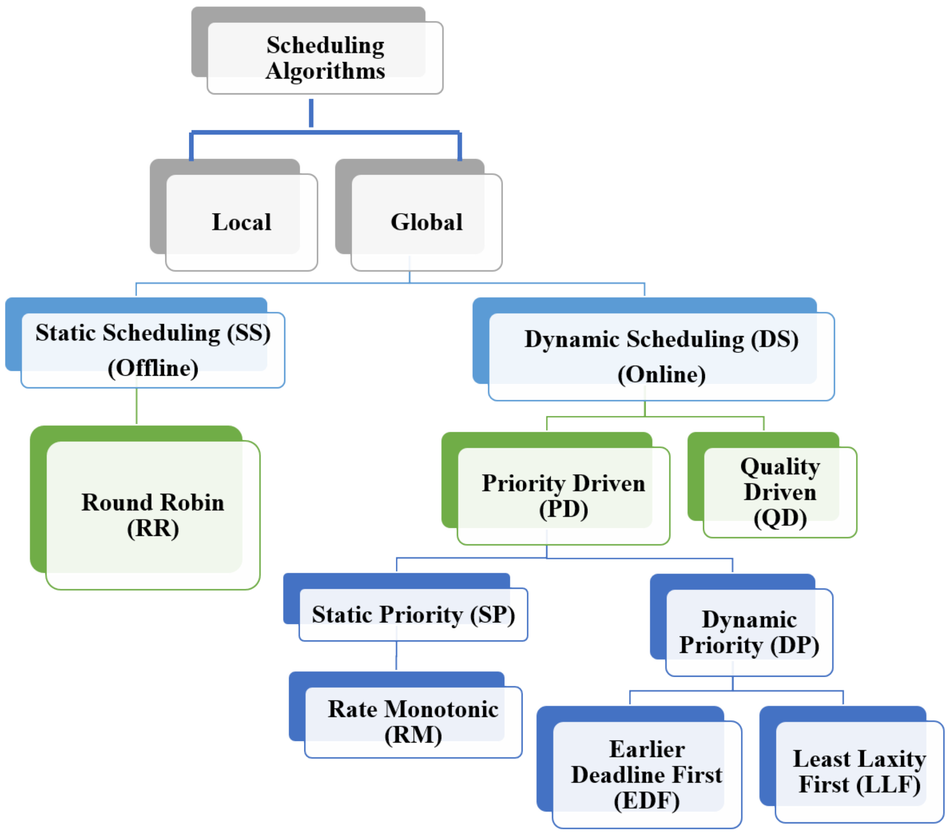


FIGURE 3. Scheduling Algorithms.

assignment is planned under a monotonous algorithm [29].

$$\sum\nolimits_{k=1}^{n} \frac{Ck}{Tk} \leq U\,(RM) = n(2^{\frac{1}{n}} - 1) \qquad (5)$$

where n is a number represented tasks, the RM scheduling comprises some assumptions that each job ought to have.

- Active jobs ought not to impart the resources to different jobs.
- Time-frames should be like the deadline. Deadline is deterministic.
- The executing jobs (required to execute) having top priority will perform the preemption of various other jobs.
- There is a need to assign priority to each job, as indicated by the RM approach.

Dynamic Priority Algorithms:

In "Dynamic Priority Algorithm (DPA)", distinctive tasks may have diverse needs next time, higher or lower than alternate assignments. Dynamic priority management is a sort of scheduling algorithm for the estimation of priorities during system execution. Dynamic priority scheduling aims to respond to development and to establish an optimum self-sustaining configuration. Depending on the complexity of a specific problem, it can be very difficult to build well-defined policies to accomplish such assignments.

Earliest Deadline First (EDF) Scheduling:

"Earliest First Deadline (EDF) " is the dynamic priority scheduler algorithm for embedded systems in real-time. This scheme accepts that the task execution is finished without burning through more time; it started executing the next job. At whatever point the processor becomes inactive, this protocol executes and chooses the work having the early finish time. The earliest deadline first selects an assignment by its deadline so that the priority of an assignment with the earliest deadline is higher than others. It means that an assignment's priority is inversely proportional to its absolute time limit. Since the absolute time limit for an assignment depends on the current time, every moment is a scheduling event in EDF, as the time limit for the assignment changes over time. Due to the earliest deadline at one moment, an assignment with a higher priority may have low priority at the next moment because of the early deadline of another assignment. EDF typically performs in preemptive mode, i.e., currently, the assignment's execution is preempted when another assignment becomes active with the earliest deadline. EDF is an optimal algorithm, representing that if an assignment set is feasible, it is planned by EDF.

Another thing is that EDF does not explicitly assume the periodicity of assignments so that it is independent of the time of the assignment and can therefore be used to plan aperiodic assignments. Choose one of them randomly if two assignments have the same absolute time limit. This scheme does not avoid Deadline-misses. Accordingly, despite the fact that no sufficient time is available for its completion, a task execution can be begun at the same iteration (that prompts deadline-misses).

Least Laxity First (LLF) Scheduling:

"Least Laxity First (LLF)" is the dynamic priority-scheduling algorithm for deployment level. It means that every moment is a planning event because the laxity of each assignment changes at every moment. An assignment that has less laxity at once may have higher priority than others at another moment. It implies maximum priority is given to the task having laxity of low-level. It is dictated by tie-breaking guidelines, tasks with a similar laxity run in a fixed manner.

LLF fulfills the situation if the real running-time is not declared until the task complete. Whereas, it is not start-time but finish-time predictable; if the genuine running-time is declared, that might be outlandish.

More formally, an assignment's priority is inversely proportionate to its laxity in terms of time. Since an assignment's laxity is defined as the urgency of it. It is described mathematically as

$$L_i = D_i - C_i \tag{6}$$

Here D_i is the time limit for the assignment, C_i is just the worst-case execution time (WCET), and L_i is the assignment's laxity. It implies that passivity is the time that remains before the time limit occurs after achieving the WCET. From the above equation, the laxity of an assignment in runtime has been included in the present time instant.

$$L_i = D_i - (t_i - C_i^R) \tag{7}$$

Here is the current instant of time and is the unsettled WCET of the assignment. By using the above equation, the laxity of each assignment is considered at every instant of time; then, the precedence is dispensed. One vital thing is that the laxity of a running assignment does not modify; it remains the same while the laxity of all other assignments is lessened by one after every one-time unit.

H. PROCESSOR UTILIZATION FACTOR (U)

For a given undertaking set of n occasional assignments, the utilization factor U of a processor is the measure of time-spent T_i to execute S_i. In such manner, the undertaking set for n occasional assignments playing out the errand set, when Si is an undertaking from the assignment set.

$$U = \sum\nolimits_{i=1}^{n} \frac{Ci}{Ti} \tag{8}$$

Ci/Ti is the time on the off chance means the processor's use is more important than others. This assignment set cannot be premeditated by any estimation. The utilization factor of the processor recounts the processor stack on a solitary processor. U = 1 implies 100 percent utilization of the processor [30], [31].

I. SCHEDULING OVERHEAD

The time-complexity is the quantity of tasks a protocol carries out to finish its operations (taking into account that every task takes a similar time-period). The calculation that carries out the operation in a minimum quantity of tasks is viewed as the effective one (from the perspective of time-complexity). In addition to time-complexity, The space intricacy of a calculation is the measure of memory/space taken by the protocol (while execution) and input length is denoted by n. It incorporates both the input space and the Auxiliary Space (ASP). ASP is an additional and tentative memory utilized by the protocol while its execution. Utilizing Big-O notation, space complexity of a protocol is usually communicated.

Scheduling is the most common VM service, and the scheduling algorithm's complexity is critically important. The overhead should therefore be kept low to prevent performance degradation. RM requires the least computation because all VM's are checked in the run queue. RM is less complex than EDF in some specific scenarios because the popularity of memory-nodes has to be taken into account in EDF [32]–[34].

IV. MATERIAL AND METHODS

A. SIMULATION PARAMETERS

One of the most important reenactment parameters is the number of processor cores and RAM in the system under consideration. The most extreme memory vitality usage, denoted as *Emax*, specifies the memory's energy consumption if no memory's energy management technique is used, and each memory node is therefore in the standby role. This study does not consider memory energy consumption during *Emax* read or write operations because the workload is heavily dependent. Table 1 indicates the different experimental configurations used for simulation in RTSim.

B. CPU AND I/O BURST TIME TRACE

The CPU and I/O assignment accomplishment time statistics are used to better imitate VM's performance features in the simulator. The CPU assignment accomplishment time is the time period a VM continues to run before it is transferred to off-state because of I/O occasions or the termination of a certain time cut. The I/O burst time is when a VM hangs tightly to finish the I/O occasions. The CPU accomplishment time, time of a VM and its I/O accomplishment time depend on the outstanding assignment at each VM's highlights.

C. FAIRNESS CRITERIA

The memory-aware VM scheduling has the same fairness level that can be found in Xen 's Credit scheduler. We will run twelve VM's on each core to see the results of the memory-aware VM scheduling on impartiality and try to compile the simple Kernel of Linux. Then we will measure the time that will be elapsed and show the fallouts in Table 2. As anticipated, the memory-aware scheduling of VM dispenses CPU time on all VM's evenly.

TABLE 1. **Experimental configurations.**

Setup Name	Configurations
System Configuration 1 (SC1)	8 Cores, 32GB RAM (16 Memory Nodes)
System Configuration 2 (SC2)	12 Cores, 48GB RAM (24 Memory Nodes)
System Configuration 3 (SC3)	16 Cores, 64GB RAM (32 Memory Nodes)
System Configuration 4 (SC4)	12 Cores, 24GB RAM (12 Memory Nodes)
System Configuration 5 (SC5)	12 Cores, 32GB RAM (16 Memory Nodes)
System Configuration 6 (SC6)	12 Cores, 40GB RAM (20 Memory Nodes)

TABLE 2. **Time taken by scheduling algorithms.**

Algorithms	Average (s)
No Memory-Aware Scheduling	870.83
Rate Monotonic	845.51
Earliest Deadline First	835.14
Least Laxity First	859.36

D. SIMULATION ARCHITECTURE

RTSim "Real-Time system Simulator" retains trail of $J(t)$ at every time t for a given replication interval and simulates the VM executions over the processor cores. The consumption of energy will be calculated by the series of $J(t)$ used. RTSim simulates the VM for four states, as shown in Fig. 4.

- Execute
- Runnable
- Expired
- Wait

RTSim maintains its separate queue for each state, and the VM will be kept in the same state. The state Runnable is also called the running state. RTSim repeats the following steps for each of the processors throughout the duration:

E. VIRTUAL MACHINE SCHEDULING

RTSim uses a scheduling algorithm to select one of the VM in the queue. $J(t)$ is updated using the selected VM access set if necessary.

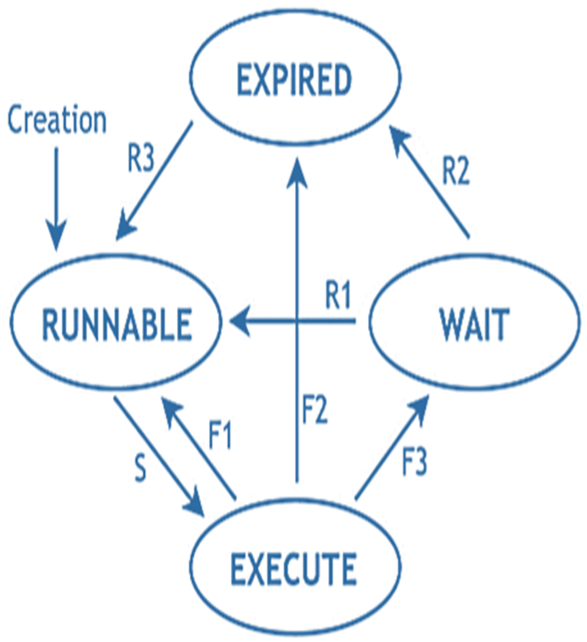


FIGURE 4. **RTSim activities.**

F. EXECUTION TIME

The CPU time track determines the time of execution of the selected VM, a parameter of the simulation is discussed earlier. The VM's state will then be reformed to EXECUTE (transition S).

G. EXECUTION

If the accomplishment time of the VM in the credit scheduler is by default equal to or greater than the time slice of 30ms, the VM is preempted. Otherwise, after consuming its CPU burst time, the VM will voluntarily leave the CPU. RTSim charges 100 VM credits per 10ms in both cases, just like Xen 's current credit scheduler.

H. POST-PROCESSING

When the VM voluntarily surrenders the CPU, the status of the VM is transformed to WAIT (transition F3). The subsequent state is altered on the basis of the unsettled VM credits that consume the entire slice of time. If the VM still has some credits, it switches to RUNNABLE (transition F1) again; otherwise, it switches to EXPIRED (F2). RTSim moves to the first procedure after the post-processing procedure to select the next VM to run.

A WAIT for VM expects some time that reenacts the sitting constricted time for I/O occasions. The holding uptime of the VM in RTSim is dictated by the I/O time track. After the expiry of the I/O burst time, the VM status will be changed to RUNNABLE on the off chance that it has remaining credits (progress R1); generally, EXPIRED (change R2) will be changed. Like the Xen credit scheduler, RTSim reloads all VM credits every 30ms. Once the credits are recharged, EXPIRED VM moves to the RUNNABLE state (transition R3).

I. ALGORITHM

Compared to global scheduling, one benefit of partitioned scheduling is that it is possible that single processor-based response time analyses can be replicated after assigning agents to portioned processors. Since the assignments on M identical processors are partitioned, it is NP-hard by nature.

//

PQPA: Partitioned Quick convergence Processor demand Analysis

B&B: Branch and Bound

DF: Deadline First

//

If x is the set of all agents and R ⊆ X, then function f =DF-B&B PQPA∗(R, N_0) performs the previous parametric single processor EDF schedulability analysis on just agents of subset S starting from point N_0. The function objective is to maximize U as it is defined for set P. Let $(N_j)_{j=1,M}$ be the set of, initially empty, M partitions. Our algorithm consists of the following steps:

Partitioned Multiprocessor Scheduling for Memory Aware Energy Management

```
$N_{cur} = N^l$
while P $\neq \emptyset$ do
select agent p $\in P$ with the smallest deadline at point $N_{cur}$;
for j = 1, . . ., $M$ do $N_j = f*(T_j \cup \{x\}, N_{cur})$;
Assign agent p to partition $T_k$ with maximum $U(N_k)$, put $N_{cur} = N_k$, and remove p from P;
```

f explores the N-dimensional at each repetition starting from the result found in the preceding iteration. This guarantees that transfer one agent to a partition does not risk the schedulability of other partitions.

V. EXPERIMENTAL RESULTS

In this segment, we present the aftereffects of reproduction. All estimations are taken from a normal of 50 reenactments, each with a reproduction time of 10,000ms.

A. BASE MEMORY'S ENERGY CONSUMPTION

This part examines the effect of $Ebase$ system size on $Emax$. The number of installed memory nodes directly affects the $Emax$ of a given system; refers to that number as |MN|. $Ebase$, however, depends on $j(t)$. Since $j(t)$ is the set of memory nodes that run VM's simultaneously, it is directed to both., The average number of VM memory nodes referred to as VMn, and the number of system cores referred to as $Ncore$. So,

$$\frac{Ebase}{Emax} \propto \frac{Ncore\ X\ VMn}{|MN|} \tag{9}$$

This study accepts the SM4 to set design and 16 VMs per core solidification ratio. Fig. 5 demonstrates that our memory controls executive engineering, sparing memory vitality even without a programming calculation. In every system, the relative energy saving is not the equivalent. In this trial, a similar design of the entrance set is utilized, and VMn is hence the equivalent in all cases. Then,

$$\frac{Ebase}{Emax} \propto \frac{Ncore}{|MN|} \tag{10}$$

In any setup of the entrance set, we see that memory energy savings will enhance as the consolidation ratio upsurges. The expansion in the consolidation ratio builds the quantity of VMs in the running line. This empowers every calculation to locate a VM that is increasingly reasonable for scheduling, in this way sparing vitality.

Fig. 6 demonstrates the information available in TABLE 2 that time consumption by specific scheduling algorithms with and without memory awareness. Also, Fig. 7 describes the energy consumption with the specific

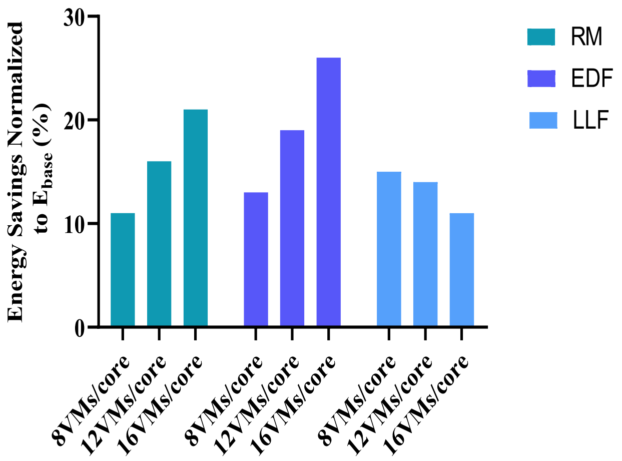


FIGURE 5. E_{base} Equalized to E_{max} for Different Computer Systems.

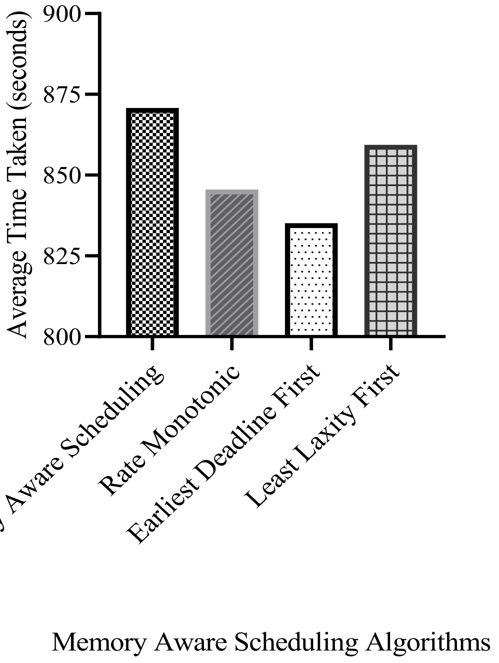
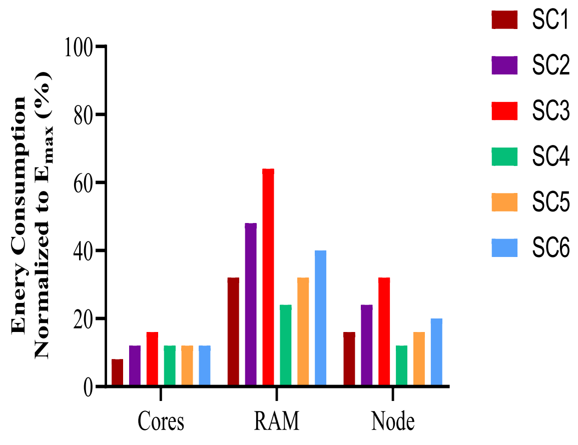


FIGURE 6. Time consumption (seconds).



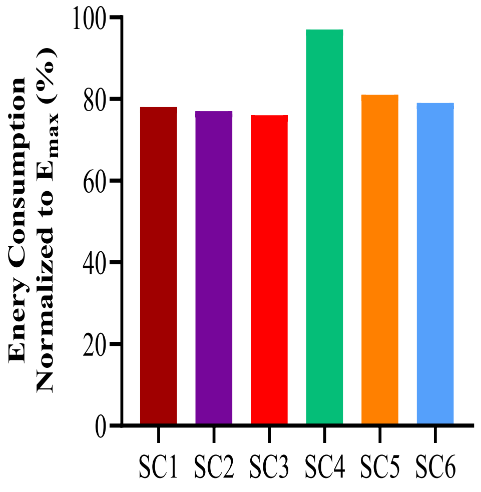


FIGURE 7. **Energy savings for SM4 with RM, EDF, and LLF algorithms.**

configuration of hardware resources like CPU, RAM, and cores, etc. This study figures out that the identified scheduling algorithms will save more energy for all access set configurations in relation to *Ebase*. The significant energy savings were achieved with the configuration of the access set SM4. Consequently, the amount of energy savings falls. To save more energy, the access size of each VM can be reduced with the help of memory-aware scheduling algorithms. The simulation results, with the help of RTSim, indicate that the EDF scheduling algorithm is more energy-efficient and shown the same results when deployed on Xen Credit scheduler.

VI. DISCUSSION

Thanks to the rapid development and technological breakthroughs of hardware, server energy efficiency has significantly improved. However, in many scenarios, energy reduction through memory aware real-time scheduling on the virtual machine in the multi-cores server is still an area of interest. Those approaches dynamically adjust the running state for a single resource (i.e., CPU or memory) or focus on single workloads. They also consider the interdependence of different resources and their impact on power consumption.

In view of the experiments, it is discovered that memory-sharing among VMs greatly impacts memory-energy utilization. It is observed that the utilization of memory in the Isolated Driver Domain in making shared-memory can mitigate significant issues. With the increase in the level of server-consolidation, the protocol's performance (against efficient energy utilization) further enhances. It is additionally observed that to accomplish more energy saving, energy-aware memory-management is fundamental by diminishing the overall memory-nodes (utilized by the VMs).

The multi-core-based non-virtualized processing has identical scheduling issues in decreasing the utilization of memory & energy. For this situation, the presented approach is also appropriate (considering VMs as operating-systems & VMs as processes). For productively using the computational-assets' expanding performance abilities, our research focuses on virtualized-computing as it is more alluring than non-virtualized-computing.

VII. CONCLUSION AND FUTURE WORK

This research simulates a feature to Xen's Credit scheduler called a memory-aware scheduling algorithm to reduce energy consumption. A simulator RTSim that can evaluate the memory and energy consumption in actuality. It is also used to examine the consumption of energy of memory in various systems. In addition, this paper describes the scheduling algorithms like RM, EDF, and LLF based memory-aware in VMs.

It can be concluded that memory-aware administration diminishes memory and energy demand efficiently, and also, EDF scheduling algorithms save more energy than the other identified scheduling algorithms. In particular, EDF take 835.14 sec for scheduling and it can save approximately 58.3 percent of memory's energy than conventional systems that would not benefit from memory-aware energy management. The energy efficiency of the algorithms continues to improve as the level of server consolidation increases. We implement the EDF scheduling algorithm in Xen 's credit scheduler to see if the simulation fallouts can be simulated on real systems or not. Results of simulation and deployment are equated, and similar results are achieved. We observed that shared memory between virtual machines pointedly impacts memory's energy consumption based on hardware resource configuration. This study also figures out that to reduce energy consumption (with the primary focus on the competition of all real-time tasks before the deadline), energy-aware memory management is essential. RTSim and deployment presently do not aid virtual machine relocation among processor cores, as Xen stereotypically controls virtual machine migration to benefit from the processor caches. We plan to provide aid to virtual machines' migration and develop different migration policies to save memory energy further.

REFERENCES

[1] T. Chen, Y. Zhang, X. Wang, and G. B. Giannakis, "Robust workload and energy management for sustainable data centers," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 651–664, Mar. 2016.

[2] C. Y. Lu and H. M. Lin, "Balancing of servers based on sampled utilization ratio and corresponding power consumption," U.S. Patent 9 857 865, Jan. 2, 2018.

[3] T. Pan, W. Qin, T. Huang, F. Yang, E. Xinhua, and H. Li, "Towards power-aware network function virtualization on multi-core processors," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 1–2.

[4] J. Chase and R. Doyle, "Energy management for server clusters," in *Proc. 8th Workshop Hot Topic Operating Syst.*, IEEE Computer Society, May 2001, p. 0165.

[5] H. Usui, L. Subramanian, K. K.-W. Chang, and O. Mutlu, "DASH: Deadline-aware high-performance memory scheduler for heterogeneous systems with hardware accelerators," *ACM Trans. Archit. Code Optim.*, vol. 12, no. 4, pp. 1–28, Jan. 2016.

[6] K. Xie, X. Huang, S. Hao, and M. Ma, "Distributed power saving for large-scale software-defined data center networks," *IEEE Access*, vol. 6, pp. 5897–5909, 2018.

[7] M.-L. Chiang and T.-T. Hou, "A scalable virtualized server cluster providing sensor data storage and Web services," *Symmetry*, vol. 12, no. 12, p. 1942, Nov. 2020.

[8] E. M. Dow, J. P. Gilchrist, S. K. Schmidt, and C. J. Stocker, IV, "Virtual machine collaborative scheduling," U.S. Patent 9 971 625, May 15, 2018.

[9] G. Raghu, N. K. Sharma, S. G. Domanal, and G. R. M. Reddy, "Memory-based load balancing algorithm in structured peer-to-peer system," in *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*. Singapore: Springer, 2018, pp. 431–439.

[10] W. Tian, M. He, W. Guo, W. Huang, X. Shi, M. Shang, A. N. Toosi, and R. Buyya, "On minimizing total energy consumption in the scheduling of virtual machine reservations," *J. Netw. Comput. Appl.*, vol. 113, pp. 64–74, Jul. 2018.

[11] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits," *Proc. IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[12] J. Roeder, B. Rouxel, S. Altmeyer, and C. Grelck, "Energy-aware scheduling of multi-version tasks on heterogeneous real-time systems," in *Proc. 36rd Annu. ACM Symp. Appl. Comput. (SAC)*, 2021.

[13] M. A. El Sayed, E. S. M. Saad, R. F. Aly, and S. M. Habashy, "Energy-efficient task partitioning for real-time scheduling on multi-core platforms," *Computers*, vol. 10, no. 1, p. 10, Jan. 2021.

[14] P. Mohanamuraly and G. Staffelbach, "Hardware locality-aware partitioning and dynamic load-balancing of unstructured meshes for large-scale scientific applications," in *Proc. Platform Adv. Sci. Comput. Conf.*, Jun. 2020, pp. 1–10.

[15] A. Mascitti, T. Cucinotta, M. Marinoni, and L. Abeni, "Dynamic partitioned scheduling of real-time tasks on ARM big. LITTLE architectures," *J. Syst. Softw.*, vol. 173, Mar. 2011, Art. no. 110886.

[16] J.-W. Jang, M. Jeon, H.-S. Kim, H. Jo, J.-S. Kim, and S. Maeng, "Energy reduction in consolidated servers through memory-aware virtual machine scheduling," *IEEE Trans. Comput.*, vol. 60, no. 4, pp. 552–564, Apr. 2011.

[17] H. Yoda, N. Shimomura, Y. Ohsawa, Y. Kato, S. Shirotori, M. Shimizu, K. Koi, T. Inokuchi, H. Sugiyama, S. Oikawa, B. Altansargai, M. Ishikawa, A. Tiwari, and A. Kurobe, "The pursuit of saving energy consumption of memory systems by MRAMs, from STT-MRAM to voltage-control spintronics memory (VoCSM)," in *Proc. IEEE Int. Magn. Conf. (INTERMAG)*, Singapore, Apr. 2018, p. 1, doi: 10.1109/INTMAG.2018.8508840.

[18] H. Yoda, E. Kitagawa, N. Shimomura, S. Fujita, and M. Amano, "The progresses of MRAM as a memory to save energy consumption and its potential for further reduction," in *Proc. Symp. VLSI Circuits (VLSI Circuits)*, Kyoto, Japan, pp. 104–105, Jun. 2015, doi: 10.1109/VLSIC.2015.7231365.

[19] S. Aqueel and K. Khare, "Design and FPGA implementation of DDR3 SDRAM controller for high performance," *Int. J. Comput. Sci. Inf. Technol.*, vol. 3, no. 4, pp. 101–110, Aug. 2011.

[20] N. Chatterjee, M. Shevgoor, R. Balasubramonian, A. Davis, Z. Fang, R. Illikkal, and R. Iyer, "Leveraging heterogeneity in DRAM main memories to accelerate critical word access," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2012, pp. 13–24.

[21] J. Ahn, C. Kim, J. Han, Y. R. Choi, and J. Huh, "Dynamic virtual machine scheduling in clouds for architectural shared resources," in *Proc. 4th USENIX Workshop Hot Topics Cloud Comput.*, vol. 12, 2012, pp. 1–6.

[22] B. R. Chang, H. F. Tsai, and C. M. Chen, "Evaluation of virtual machine performance and virtualized consolidation ratio in cloud computing system," *J. Inf. Hiding Multimedia Signal Process.*, vol. 4, no. 3, pp. 192–200, 2013.

[23] D. G. Lago, E. R. M. Madeira, and D. Medhi, "Energy-aware virtual machine scheduling on data centers with heterogeneous bandwidths," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 83–98, Jan. 2018, doi: 10.1109/TPDS.2017.2753247.

[24] S. Sotiriadis, N. Bessis, and R. Buyya, "Self managed virtual machine scheduling in cloud systems," *Inf. Sci.*, vols. 433–434, pp. 381–400, Apr. 2018.

[25] X. Li, P. Garraghan, X. Jiang, Z. Wu, and J. Xu, "Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1317–1331, Jun. 2018, doi: 10.1109/TPDS.2017.2688445.

[26] A. Pegkas, C. Alexakos, and S. Likothanassis, "Credit-based algorithm for virtual machines scheduling," in *Proc. Innov. Intell. Syst. Appl. (INISTA)*, Thessaloniki, Greece, 2018, pp. 1–6, doi: 10.1109/INISTA.2018.8466305.

[27] A. Alhammad, S. Wasly, and R. Pellizzoni, "Memory efficient global scheduling of real-time tasks," in *Proc. 21st IEEE Real-Time Embedded Technol. Appl. Symp.*, Apr. 2015, pp. 285–296, doi: 10.1109/RTAS.2015.7108452.

[28] H.-E. Zahaf, A. E. H. Benyamina, R. Olejnik, and G. Lipari, "Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms," *J. Syst. Archit.*, vol. 74, pp. 46–60, Mar. 2017.

[29] G. Xie, G. Zeng, X. Xiao, R. Li, and K. Li, "Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3426–3442, Dec. 2017.

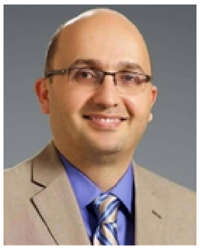
[30] A. Bhuiyan, Z. Guo, A. Saifullah, N. Guan, and H. Xiong, "Energy-efficient real-time scheduling of DAG tasks," *ACM Trans. Embedded Comput. Syst.*, vol. 17, no. 5, pp. 1–25, Nov. 2018.

[31] X. Wang, Z. Li, and W. M. Wonham, "Optimal priority-free conditionally-preemptive real-time scheduling of periodic tasks based on DES supervisory control," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 7, pp. 1082–1098, Jul. 2017.

[32] A. Sufian, A. Banerjee, and P. Dutta, "Survey of various real and non-real-time scheduling algorithms in mobile ad hoc networks," in *Industry Interactive Innovations in Science, Engineering and Technology*. Singapore: Springer, 20018, pp. 121–133.

[33] H. Alhussia, S. J. Abdulkadir, N. Zakaria, A. Patel, and A. Alzahrani, "Practical performance analysis of real-time multiprocessor scheduling algorithms," *J. Fundam. Appl. Sci.*, vol. 10, no. 2S, pp. 60–73, 2018.

[34] R. Tyagi and S. K. Gupta, "A survey on scheduling algorithms for parallel and distributed systems," in *Silicon Photonics & High Performance Computing*, R. Schmidt and A. García-Ortiz, Eds. Singapore: Springer, Sep. 2020, pp. 51–64.



MOHAMMAD A. ALQUDAH received the B.S. degree in mathematics from Yarmouk University, the M.S. degree in applied mathematics from the Jordan University of Science and Technology, Jordan, and the M.S. and Ph.D. degrees in mathematics from Central Michigan University, Mount Pleasant, MI, USA, in 2008 and 2010, respectively. He was an Assistant Professor with the Department of Mathematics and Statistics, Northwood University, Midland, MI, USA, where was a Faculty Member, from January 2011 to September 2015. From June 2011 to September 2015, he was the Chairperson with the Department of Mathematics, Northwood University. Since September 2015, he has been a Faculty Member with the Department of Basic Sciences, German Jordanian University, where he is currently an Associate Professor. His main research interests include numerical analysis, computational and applied mathematics, computer aided geometric design, neural networks, and computing techniques.

IQRA AHMED received the bachelor's degree in computer sciences. She is currently pursuing the Master of Philosophy degree in computer science with the Department of Computer Sciences, Institute of Management Sciences, Lahore, Pakistan. Her research interests include the Internet of Things, energy management, optimization, and real-time scheduling.

FAHAD AHMAD received the Ph.D. degree in computer science from the National College of Business Administration and Economics, Lahore, Pakistan, in 2017. He worked as an Assistant Professor with the Department of Computer Sciences, Kinnard College for Women, Lahore. He is currently working with the Department of Basic Sciences, Deanship of Common First Year, Jouf University, Sakaka, Saudi Arabia. His research interests include machine learning, deep learning, medical image processing, quantum computing, mathematical modeling, information security, fuzzy logic, theory of approximation, and splines.

SHAHID NASEEM received the Doctor of Philosophy degree in computer science from the National College of Business Administration and Economics, Pakistan. He is currently working as an Assistant Professor with the University of Education, Lahore, Pakistan. He is also an active researcher. He is actively involved in taking different undergraduate and postgraduate courses and performing research activities at UE. His research interests include emotional intelligence, fuzzy logic, the Internet of Things, quantum computing, and blockchain.

KOTTAKKARAN SOOPPY NISAR is currently a Full Professor of applied mathematics with Prince Sattam bin Abdulaziz University, Saudi Arabia. He has published more than 400 research articles in various international journals. His current research interests include special functions, fractional calculus, fluid dynamics, inequalities, CAD, machine learning, SAC OCDMA, and multidisciplinary applications of mathematics. He is an editorial board member of many reputed journals. He is the winner of number of awards, including PSAU Research Excellent Award and Riemann Young Researcher Award.