International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I7P224
Volume 70 Issue 7, 238-249, July 2022

Original Article

An Empirical Approach of Performance and Energy-Aware Scheduling [PEAS] Mechanism in the HPC-Cloud Model

Sharavana. K[1], Josephine Prem Kumar[2], Shivamurthy[3]

[1]*Department of Information Science and Engineering, HKBK College of Engineering, Bengaluru, India*
[2]*Department of Computer Science and Engineering, Cambridge Institute Of Technology, Bangalore, India*
[3]*Department of Computer Science and Engineering, VIAT, Muddenahalli, India*

[1]sharatanuj@gmail.com

Received: 17 May 2022 Revised: 02 July 2022 Accepted: 16 July 2022 Published: 18 July 2022

Abstract *- Cloud Computing provides the advantage of flexibility, elasticity, scaling, and customization to the HPC community as it attracts users that cannot afford to use the dedicated HPC infrastructure. HPC infrastructure is proven costly, as it requires upfront investment despite the advantage of processing the complex task. Interconnection of HPC and cloud environment creates the complex infrastructure for parallel computation and further creates a major issue in managing the makespan and energy performance trade-off. This research presents the PEAS (Performance and Energy-aware scheduling)-mechanism; PEAS is designed for parallel computation with task scheduling and optimal resource allocation at data centers. At first, a system model is designed for the parallel computing process; later, a novel and efficient scheduling algorithm is designed for task scheduling, and at last energy-aware mathematical model is designed for optimal energy utilization. PEAS are evaluated considering the HPC aware scientific workflow like cyber shake and montage workflow considering the evaluation parameter as Make span, Energy consumption, and Power utilization. Moreover, PEAS is proven to be more efficient than any other existing model available to date.*

Keywords *- Cloud Computing, HPC, Scientific Workflow, HPC Cloud.*

1. Introduction

Over the years, a computer has been used to improve performance due to the demand for high development in Information Technology (IT) for both industries and academics. The systems used for computation have data storage ability and strong computing efficiency. This is used widely in industries for support and workflow scientifically. Conversely, there is a vast improvement in the system's performance and power consumption.The improvised energy consumption leads to ecologically, extreme economic and technical escalations. Lately, Cloud Computing has been developed for equipping resources effectively and efficiently. The cloud's infrastructure management is centralized, leading to the users' on-demand access to the resources. These are also paid in a "pay as you go" way [1-2].

Moreover, due to these advantages cloud has become an alternating computer for a traditional cluster to execute the HPC applications. HPC(High-Performance Computing) cloud utilizes virtualization technologies to improvise server management and resource utilization; moreover, virtualization technology enables the various application to process isolated through VMs and makes it possible to combine several VMs to a particular PM (Physical Machine). Moreover, virtualization technologies provide various aspects of energy management.

Considering the infrastructure of the cloud is elastic, the growing count of users chooses the application to be organized, including business and scientific applications in the cloud. Many applications in the scientific domain are exhibited using the workflows of various other domains, including astronomy, increasing energy of physics, astrophysics, bioinformatics, etc. Cloud computing is made up of a computing model based on service and infrastructure, giving the user the demanding abilities of computation. Considering the modern model of virtualization used for job and resource allocation in dynamic. The cloud platform users are required to only provide a request to their system. Based on the user requests, the system automatically provides the necessary resources so that the user manages the cloud platform and emphasizes the focus mainly on the cloud application. For the cloud service provider, the main concern lies in the correct allocation of applications for scheduling and infrastructure of hardware such that the processing expenditure is decreased. The cloud computing system has a cast on a bigger scale, which leads to energy consumption at various levels of resource computation [3].

Huge scientific dataset analysis, massive communication-intensive firmly activities, and high-throughput computation have become key parts of HPC workloads in recent years. HPC clouds are rapidly expanding the application user base and platform options for running HPC workloads, ranging from in-house committed powerful computers to resource clusters with and without HPC-optimized interconnects and operating systems to resources with variable levels of virtualization to hybrid combinations that outsource segments of the task to the cloud. HPC users and cloud service providers face the difficult task of selecting the best framework based on a limited understanding of all shows, platform functionality, and target performance measures such as cost. As a result of this trend, there may be a discrepancy between the HPC application's necessary and available capabilities. One unfavorable scenario could arise in a part of the construction being overburdened while another is idle, resulting in long waits and lower total throughput. The existing HPC set of inputs isn't built to handle these problems. To operate well in a circumstance, fresh schedulers and strategies must be investigated [4].

However, scientific workflows are used for a higher scale in the infrastructures of the cloud. Normally, these workflows consist of thousands of tasks; hence, a higher count of computational resources is required during the execution period. These computational resources are provided for the cloud computing infrastructure. Although, the tasks in the workflow of scientific domain are inter dependent and have communication between them, which differs from the unwanted tasks. Hence, the cloud management system needs the resources to be assigned for the workflow relating to the executions. The resources that are used in the computation process are as virtual machines that are within the cloud platform [3]. These virtual machines are used in various applications for measuring different parameters for configuration, which have memories, the capacity of the disk, and many cores of the CPU. The scientific execution workflow in the cloud has a higher energy consumption; the virtual machines must be deployed in a manner that makes it energy efficient. Hence, energy consumption in the cloud platform has attained attention globally. The cloud's data center has enormous energy consumed for running monitors, servers, cooling systems, cooling fans, consoles, processors, network extensions, and other resources[4]. However, the energy consumption in the data centers causes pollution globally by emitting CO2 in enormous amounts which is the reason for greenhouse gases. Data centers cause 2 percent of the entire CO2 emission globally. As a result, execution deployment is done significantly for workflow in a scientific domain for energy awareness in the platform.

A technique for decreasing the consumption of power for any model of computation is more effectively used for the mechanism for power awareness. A widely used and renowned technique, DVFS (dynamic voltage frequency scaling), is used for dynamically trading off the energy's delays. Considering the applications of cloud computing, the most important aim is the efficient allocation of resources for growing performance for cloud information centers. Hence, performance enhancement and dealing with these constraints over time use various methodologies. The techniques used over the years include CEFT, called Earliest Constrained (Finish Time), Reliable hierarchy has driven scheduling, and dynamic voltage frequency scaling. Also, it has been observed that DVFS has achieved effective task load scheduling. It is a highly effective optimization technique for energy consumption deployed for the cloud system. This optimization of energy is attained by dynamic voltage scaling down. It also helps attain higher Quality of internet service by decreasing the energy consumption in the devices [5][6].

Although, these pre-existing methodologies have a definitive prototype and the cost for communication of the processors is extremely high during the use of these methodologies. In the paper [5], a scheme for optimizing runtime and energy is initiated for energy reduction by different task loads being scheduled for different embedded systems. Considering paper [6], different techniques for energy efficiency are debated, and the embedded systems need to enhance the computation's performance.

1.1. Motivation and contribution of Proposed Work

HPC applications executed on cloud infrastructures include workflow applications like scientific workflow. Cloud computing is an integrated paradigm for the entire on-demand virtualization, and future users consume the platform, infrastructure, and software service. However, the dynamic computing services are backed up using the data centers used by the virtual machine environments for isolation and integration purposes. Also, performance has an essential part that is considered. Therefore, in this proposed paper, we include the mechanism for performance awareness considering parallel computation. This assists the performance improvement. Additionally, the contribution of the proposed research work has been stated below:

- We develop a technique for the optimal allocation of resources and HPC cluster scheduling of tasks, including the proposed method based on performance. We focus on the time and energy as a challenge, and the process is optimized using multiple processors.
- An algorithm is developed based on scheduling priority because constraint on priority analysis is burdensome. After which, this algorithm is used for task scheduling.
- Considering the parallel task, we design a mechanism for task scheduling used for the parallel task scheduling task. This is simultaneously accompanied by execution; further, more is coded for executing tasks.

- In parallel computation, the parameter constraints are considered time and energy, and makespan optimization is performed. Therefore, this improves performance.
- Furthermore, an evaluation is performed on PEAS considering the cyber shake workflow of the scientific domain along with four variations. This evaluation is performed by a virtual machine variation and a comparative analysis of the already existing models.
- The proposed model is used to optimize the allocation of resources considering constraints in the data centers. Similarly, the optimization of PEAS is used for task scheduling and is further used for attenuating energy consumption.

This specific work has been organized like other research, and the initial section emphasized parallel computation and the requirement of parallel computation, including the relating steps for optimization of makespan. Furthermore, the initial section has a subsection that focuses on the contribution of the research. The next section emphasizes the different methodologies and techniques that already exist and are utilized for enhancing performance, including the drawbacks that are present. Section three introduces the mechanism of PEAS, including the optimization model makespan with the allocation of resources and scheduling mechanism for tasks. Also, section four evaluates the proposed PEAS mechanism.

2. Related Work

HPC cluster performance is enhanced through designing an efficient task scheduling mechanism; hence this section mainly focuses on the different scheduling schemes to enhance the performance of cloud computing

The processor core consists of performance and power states, which are managed using an operating system and a hypervisor with an advanced configuration and a power interface. Considering ACPI for industries, the management interface is based on standard power [7]. Performance states are normally expressed in voltage and frequency pairs used for energy consumption, whereas the workloads operate in the core. PS combined with an improvised performance level is termed P0; this indicates a large voltage and a large frequency which leads to high power consumption. Increase Performance states lead to a low-performance level, which indicates that a decrease in voltage and a decrease in frequency lead to lower power consumption. A transfer in the processor core is from one to another PS within the DVFS. Although the transfer in the performance states sustains latency, this is normally on the scale of microseconds and is negligible, whereas the comparative analysis of execution duration for applications and on a scale of hours as well as minutes [8]. To help the power states for conserving energy, whereas idle class is where the core is present. The core has an active state that is represented by C0, where the performance is done by the instructions that are given at the level of performance that is specific to a power state that is used.

Conversely, C1 is the initial performance state in the idle state. When C1 is the core mode, it can be converted to C0, avoiding latency. After that, power consumption is reduced compared to being in an active mode, specifically C0. In high power states, the core is set to an inactive state. The method of conservation used is an aggressive power, indicating the combination of a clock and power. However, the core of the power states is increased compared to a specific value which causes a delay to return to CO. The tasks started in the system for real-time settings and are dynamic in the requirement to meet the deadlines, after which the return to performance states is more than the observation at C1, which could lead to attenuated performance [9].

In [10], periodic scheduling is used by heuristics for homogeneous multiprocessing and is independent of the tasks performed in real-time. Considering this method, the main priority is assigned to these tasks for the process of execution in comparison to various other tasks. However, the necessary parts and the smallest deadline process are assigned high priority. The technique used here has slack time. This happens during the starting completion of parts required for task scheduling that is optional at low processor speed by using DVFS. Although the method used here is not suitable for various applications considering workflow, this is assumed for independent tasks. Paper [11] is similar, considering the methods that are introduced. Using a target system, the homogenous multiprocessor is given as a starting approach. Although this application is assumed with the approach with the component tasks that are given, the precedence constraints and data independence are used for the tasks. Considering these two approaches, the heterogeneous processor using the core of DVFS is introduced.

However, the normal challenge of scheduling a platform that could be shared in the cloud has more pressing issues concerning mapping application sets based on tasks to fix the resources used for computation that finishes tasks within specified constraints. Also, it is essential to finish the given task considering the deadline. Therefore, reducing delinquency in the tasks and the response time of the consumed power of the given resources introduces some scheduling methodologies [12]. However, a challenge is faced while scheduling a Complete-NP. Many researchers in this field have focused on scheduling, considering energy efficiency (EE) heuristics [13] [14]. This method has used DVFS. In the paper [15], considering non-critical jobs, the execution time has been improvised using slack time as well as reduced consumption of energy in DVFS using a homogenous cluster, with no maximization of the time for execution of complete tasks. However, the jobs are

considered precedence constraints among the specified tasks while no deadlines are assumed. Although implementing a technique for aware scheduling for load and thermal in the cloud was introduced, the servers could be overloaded. Also, the DVFS is utilized to manage power for physical machines and the host-based on a single core [16].

In paper [17], a new technique is used for the workflow based on real-time, considering the applications used for cloud scheduling. This is used in the scheduling gaps with an approximate formulation and the underlying computations for the resources used in computing during the heterogeneous time. Although their mechanisms have considered the effect of input error on the tasks defined in a component application, there is no use of an energy-aware DVFS or heuristic approach.

On the contrary, the author of the paper [18 - 20] has initiated the technique for scheduling in which the DVFS is leveraged considering the system core with help from the processors that are heterogeneous, resulting in a balance in performance and energy consumption. The two methods used for scheduling are assumed to be in a batch and online mode. The online mode has differing computations along with tasks, and the constraints on time and arrival time exist together. In considering batch mode, the tasks that are performed happen in batches even via every core, the heterogeneous processors and DVFS are assumed, and the workload has simple, independent tasks. Further, a computation that is inappropriate is utilized.

3. Proposed Methodology

In the HPC Cloud environment, the data center encounters many parallel task execution requests as HPC-aware application requires efficient task scheduling to obtain maximum performance. Moreover, the performance of HPC clusters depends on the efficient scheduling of jobs.

3.1. Preliminary analysis and designing of Initial Model

This section focuses on the preliminaries of the proposed model that involve the model related to task, models of resource, and models concerning power. The dissipation of power is given mathematically using the following equation, wherein consumption of power is expressed as $\mathscr{p}$,

$$\mathscr{p} = \mathbb{a} f C S^2 \tag{1}$$

Considering equation (1), the activity factor is denoted as $\mathbb{a}$, and load capacitance is denoted as$C$, the voltage supply is expressed as S. The frequency of the clock is given as f. However, when the voltage supplies, as well as the frequency of the clock, are in an ideal state, they are expressed as S. Also, the supply voltage and frequency are related to each other such that $S \propto f^{\alpha}$where α is the given constant. However, the speed of execution is assumed to be proportional to the frequency of the given clock, which is expressed as R. We further assume, $S = bf^{\beta}$and $R = cf$. The consumption of power is mathematically expressed using the following equation,

$$\mathscr{p} = \aleph R^{a} \tag{2}$$

This equation is expressed with the help of equation 1, which is expressed using equations 3 and 4.

$$\aleph = (c^{2\delta+1})^{-1} a b^2 B C \tag{3}$$

$$a = 2\delta + 1 \tag{4}$$

Considering the parallel task n, we take any graph $G = (S, E)$ in which the set of tasks is expressed as U and expressed as $S = \{1,2,3, \dots, n\}$. In this, the constraints on priority are denoted as E. Furthermore, the relation of $arc(x, y)$ form $xtoy$ shows that the task y has not been started until the task x is completed. The count of processors that has been requested x is expressed as $\mathcal{P}_x$ a parallel task that is specified using $\mathcal{P}_x$ as well as s_x task work. However, the data center has p some identical processors. For the execution of any task x, for any $\mathcal{P}_x$ having p processors within the data center that could be allocated. Therefore, stating that parallel computation occurs.

Considering a scheduling application that is parallel u which is represented using a task graph $I_1, I_2, \dots, I_v$ for a given processor l in a data center. However, many task graphs are assumed to be single-task graphs. Also, the power applied to the task is expressed as H given in the equation stated below, which $R_x = \mathscr{p}_x^{\frac{1}{a}}$ is used to indicate the speed of execution of the task x. Furthermore, it has been observed that $\mathscr{p}_x = \varphi R_x^a$. The time of execution for the given task using the equation stated below. The processor $\mathcal{P}_x$ has the same speed of execution t_x.

$$t_x = \left({}^{t_x}/_{P_i}\right)^{a^{-1}} \tag{5}$$

The energy that is necessary for the execution of the task is given using the equation stated below, which W_x denoted the work that has been done for the performance of the task x given by the equation below

$$E_x = R_x{}^{a-1} W_x \quad (6)$$

However, it should be observed in a processor that is real. The speed of execution and the frequency of the clock only assumes very discrete values, W_x which is the earlier work that has been computed.

3.2. Designing Problem statement

For the analysis of huge cluster datasets in machines that have been utilized for executing tasks, there are various tasks for the clusters considering parallel computing. Furthermore, these tasks must have a resource that is efficient and heavy. Assuming such situations, we emphasize developing a mechanism to allocate resources. After the resource is allocated, the task scheduling has to be optimal; this is for achieving a design algorithm for task scheduling [21 -24]. After which, an optimization model of makespan is developed. Therefore, to achieve this, we assume that the number n of parallel tasks has ρ constraints and the task has a size of $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$ having the necessity for execution of task as $r_1, r_2, \ldots, r_n$ for m processors in such situations that the length for scheduling is reduced. The consumption of energy does not exceed E. Therefore, the proposed work emphasizes resulting in energy efficiency that is optimized.

3.3. Mathematical modeling for parallel computing in an HPC environment

Considering a parameter W that is used to indicate the performed work for parallel tasks n is given by the equation stated below

$$W = w_1 + w_2 + \cdots + w_n = \mathcal{P}_1 r_1 + \mathcal{P}_1 r_2 + \cdots + \mathcal{P}_n r_n \quad (7)$$

Furthermore, the optimal length is denoted as T', and the bare minimum energy needed is denoted as G' for scheduling optimally. However, the computation for the lower bound is proposed using the equation stated below by makespan being minimum along with energy consumption as a constraint.

$$T' \geq \left(\frac{m}{\tilde{G}} \left(\frac{W}{m} \right)^{a} \right)^{1/(a-1)} \quad (8)$$

We also compute the lower bound using the equation stated below by the length of the schedule being considered a constraint.

$$G' \geq m \left(\frac{W}{m} \right)^{a} \left(\tilde{T}^{a-1} \right)^{-1} \quad (9)$$

3.4. ERAHPC-Cloud (Efficient resource allocation in HPC cloud environment)

This section emphasizes the development of a general optimization model considering four distinct resources: disk, CPU, the bandwidth of the network, and memory. In this case, the allocation occurs over machines and time. We propose a model for allocating resources for the parallel task that is non-dependent in which every task is assigned the same amount of power and speed. However, this technique is used to connect task scheduling. In this case, the resource is denoted as $\mathbb{T}$the capacity of resource on the machine, which is given as $\mathbb{e}_{\mathbb{T}}^{\mathbb{k}}$. Similarly, the task $\mathbb{k}$demands the resources that are denoted as $\mathbb{f}_{\mathbb{l}}^{\mathbb{T}}$. The task schedule encoding variable needs to be defined, and the allocation of resources.

Considering $\mathbb{A}_{kl}^{\mathbb{t}}$ which is the variable indicator and if the allocation of the task to $\mathbb{k}$ a machine at a set time duration, the variable indicator is set to 1. Furthermore, the task $\mathbb{l}$ allocates $\mathbb{Z}_{\mathbb{k}\mathbb{l}}^{\mathbb{t},\mathbb{v}}$ units for a machine $\mathbb{k}$and the type of resource $\mathbb{t}$ at $\mathbb{v}$ the time. Also, consider the task $\mathbb{l}$ for the machine $\mathbb{k}$. The allocation of the resources should be able to satisfy all four cases.

In the first case, the capacity of the machine $\mathbb{k}$ cannot be exceeded by the resources, as stated below

$$\sum_{\mathbb{l}} \mathbb{Z}_{\mathbb{k}\mathbb{l}}^{\mathbb{t},\mathbb{v}} \leq \mathbb{c}_{\mathbb{k}}^{\mathbb{t}} \; \forall \mathbb{k}, \mathbb{v}, \mathbb{t} \quad (10)$$

In the second case, inactive tasks do not permit the allocation of resources

$$\mathbb{q}_{\mathbb{l}} \leq \mathbb{Z}_{\mathbb{k}\mathbb{l}}^{\mathbb{t},\mathbb{v}} \leq \mathbb{f}_{\mathbb{k}}^{\mathbb{t}} \quad (11)$$

For all $\mathbb{k}, \mathbb{v}, \mathbb{t}, \mathbb{l}$

The third case is performed to avoid cost; the allocation of resources occurs until the completion of tasks.

$$\sum_{\mathbb{v}=Init_{\mathbb{l}}}^{Init \;\; \mathbb{l}+tenu} \mathbb{Z}_{\mathbb{k}\mathbb{l}}^{\mathbb{t},\mathbb{v}} = \begin{cases} tenu_{\mathbb{k}} \mathbb{k} = \mathbb{k}_{\mathbb{l}}^{*} \\ oothermachine \end{cases} \quad (12)$$

The fourth case is performed such that the task's duration is dependent on the allocation of resources and placement of tasks.

$$tenu_k = max\left(\begin{array}{c} \frac{\mathbb{h}_{\mathbb{k}}^{cpu}}{\sum_{\mathbb{v}} \mathbb{Z}_{\mathrm{k}_l^*\mathbb{l}}^{cpu}}, \frac{\mathbb{h}_{\mathbb{k}}^{disk}}{\sum_{\mathbb{v}} \mathbb{Z}_{\mathrm{k}_l^*\mathbb{l}}^{disky,\mathbb{v}}} \\ forall \frac{\mathbb{h}_{\mathbb{k}}^{disk}}{\sum_{\mathbb{v}} \mathbb{Z}_{\mathrm{k}_l^*\mathbb{l}}^{disky,v}} \\ \frac{\mathbb{h}_{\mathbb{k}}^{disk}}{\sum_{\mathbb{v}} \mathbb{Z}_{\mathrm{k}_l^*\mathbb{l}}^{disky,v}} \end{array}\right) \quad (13)$$

Equation 13 indicates that the numerator has the entire resource requirement wherein the denominator gives the rate of the resource. Furthermore, the CPU cycle shows the resource requirement, and the rate indicates the cycle or bandwidth. However, the allocation of the cores and the memory is done only if a task of input is present for the machine. The local disk receives the output, and the bandwidth is always provided. It has also been assumed that the task allocation is done by peak (memory size) because the task's runtime would become worst in case a memory size that is less is provided. Considering the mechanism mentioned above, resources including disk, networks, memory, and CPU are scheduled [25-27]. Whereas the availability of the machine dynamically requires to be specified and known. Therefore, a resource monitor is developed where resource use is observed. The demand for resources is also developed as follows:

Before execution, it is essential to know the input's size and the task's location. Further, a recurrent job, updating the software or checking, occurs by statistical measuring. Furthermore, it can be observed that tasks in the form of phases perform the same computation for different data partitions of similar resource usability. Therefore, a few demanded tasks could be applied to other tasks. Later, If the above steps and the methodologies do not match, the task, in this case, is overestimated. Due to cost underestimation, the process could become slow. Although, overestimation results are expensive. Therefore, resource monitors are used where the resources that are not used are monitored. However, considering randomness, allocation is not standardized. In this case, allocation is enforced, where tokens are used.

3.5. ETSHPC-Cloud (Efficient task scheduling concerning HPC Cloud)

An algorithm for task scheduling is developed, and an optimization model makespan is designed while considering time and energy constraints. This algorithm assists in parallel execution for the reduction of makespan. Here, scientific workflows are assumed as the algorithm's input. Utilization of energy, cost execution, makespan as well as utilization of resources is constantly monitored and utilized for analysis in the future.

PEAS Algorithm

Step1: Start

Step2: Considering the P^{th}task list $N = V_1, V_2, \ldots\ldots V_p$ with a time duration of $[P, PEAS(N)]$

Step3: A task is given and V is split into (M+1) groups as

$$I_1 = [V_1, V_2, \ldots\ldots V_{k1}]$$
$$I_2 = [V_{k1+1}, V_{k2+1}, \ldots\ldots V_p]$$
$$I_m = [V_{km+1} + 1, V_{km+1}, \ldots\ldots V_p]$$

Step4: $V_{l_1}, V_{l_1}, \ldots\ldots U_{lm}$ is the sequence of tasks in the schedule that are complete. Furthermore, the task V_{lb} is finished at a given time sub_b and simultaneous schedule I_{b+1} for the remaining task execution.

Step5: Time duration $[P, PEAS]$ is split into different smaller intervals as stated below $[sub_o, sub_1]$ …… $[sub_{m-1}, sub_m]$, $[sub_m, sub_{m+1}]$ where sub_p=0

Step6: The task that is active in above mentioned small intervals is $I_1 V I_2 I V_{m+1} - \{V_{l_1}, V_{l_1}, \ldots\ldots V_{lm}\}$.

Step7: The initial time stated is $p$the task I_1 is scheduled for execution in which the count of resources is utilized r_1. Therefore, the resources that are utilized ($\mathbb{Q}$) in the smaller intervals $[sub_o, sub_1]$:

$$r_1 = \mathbb{Q}_1 + \mathbb{Q}_2 + \cdots . + \mathbb{Q}_{k_1} \leq o$$

This indicated that many tasks are scheduled at the same time for execution

Step8: During the completion of a task V_{lm}, the execution I_{m+1} is scheduled. Here, the value m is small so that the completion of the task V_{lm} does not have to be scheduled. Therefore, the count of tasks sub_m is as follows:

The resources utilized sub_m are

$$r_{d+1} = r_m + \mathbb{Q}_{lm} + \mathbb{Q}_{lm+1} \ldots . + \mathbb{Q}_p$$

Step9: Therefore, the count of resources that are utilized is diminished. This decreases the further energy consumption, and the improvised scheduling and allocation of resources decrease the time for execution.

Step10: End

Therefore, the count of resources is diminished with completing the tasks [28]. Once the scheduling of tasks is completed, we develop the optimized model makespan by considering the constraints of time and energy [29].

3.6. EHPC-CLOUD (Energy modeling on HPC Cloud Environment)

To resolve the scheduling of tasks considering the constraint of energy, we propose the equation

$$\begin{aligned}\tilde{G} = \mathbb{Q}_1 t_1 r^{1-\frac{1}{c}} + \mathbb{Q}_2 t_2 r^{1-\frac{1}{c}} + \cdots \\ + \mathbb{Q}_p t_p r^{1-\frac{1}{c}} \\ = Y r^{1-\frac{1}{c}}\end{aligned} \quad (14)$$

The entire work performed is denoted as Y the supply of power r. Using the equation 14, the consumption of power is computed, which is formulated by equation 15

$$r = \left(\frac{\tilde{G}}{Y}\right)^{c/(c-1)} \quad (15)$$

We also compute the speed of execution for the task, which is stated in equation 16

$$U = \left(\frac{\tilde{G}}{Y}\right)^{c/(c-1)} \quad (16)$$

The time of execution for the task k is calculated using the equation 17

$$v_k = t_k \left(\frac{Y}{\tilde{G}}\right)^{1/(c-1)} \quad (17)$$

In the equation given above, we assumed o the number of tasks with the time of execution as $\{v_k = v_1, v_2, \ldots, v_p\}$, the resource allocation of these tasks is for parallel tasks. Considering task list 1 to P, task 1 is scheduled, implying that various tasks can be scheduled for execution simultaneously. Furthermore, the task after $l+1$ is not considered though these are small tasks after the task l is finished. $l+1, l+2 \ldots . m$ Are the resources that are allocated, in which $\mathbb{Q} - \mathbb{Q}_l + \mathbb{Q}_{l+1} + \mathbb{Q}_{l+2} + \cdots . + \mathbb{Q}_{m+1} > o$.

Therefore, the allocation of tasks is executed at the same time. Although, the tasks are not considered after $l+1$. However, energy consumption is a constraint, and resource allocation is optimized. Considering the length of resource allocation as C(($t_1, t_2, \ldots, t_p$,) for the task o with the execution time. Furthermore, the length of the resource is computed

$$V = C\left(t_1, t_2, \ldots, t_p\right)\left(\frac{Y}{\tilde{G}}\right)^{1/(c-1)} \quad (18)$$

We also compute the ratio of performance which is stated in equation 19

$$\alpha \leq \frac{C\left(t_1, t_2, \ldots, t_p\right)}{Y/o} \quad (19)$$

However, we see $\forall z \geq 0$ when $v_k = z t_k$ then C($v_1, v_2, \ldots, v_p$) $= z$C $(t_1, t_2, \ldots, t_p,)$. Therefore, we formulate the equation 20

$$\begin{aligned}V = C\left(v_1, v_2, \ldots, v_p\right) \\ = C\left(t_1, t_2, \ldots, t_p\right)\left(\frac{\tilde{G}}{Y}\right)^{c/(c-1)}\end{aligned} \quad (20)$$

Further, based on the lower bound, it results in the optimized allocation of resources based on energy given in equation 21

$$\alpha = \frac{V}{V^*} \leq \frac{C\left(t_1, t_2, \ldots, t_p\right)}{Y/o} \quad (21)$$

For the optimization of makespan, the constraint of time has to be considered; initially, the technique mentioned above is implemented to schedule the task along with the execution time, and the power consumption is stated as $v_k = t_k Y^{1/(c-1)}$ and $r = \frac{1}{Y^{c(c-1)}}$ respectively. However, when V' is denoted for the length of the allocation of resources and V' is scaled for the consumption of energy considering the factor $\frac{1}{G^{1/(a-1)}}$. Therefore, equation 22 is used for the computation of allocation

$$\tilde{V} = \frac{V'}{G^{1/(c-1)}} \quad (22)$$

This is further expressed as:

$$G = \left(\frac{V'}{\tilde{V}}\right)^{c-1} \quad (23)$$

Furthermore, the ratio for performance is required to be defined to resolve the energy consumption using the allocation length [30]. The problem relating to energy and the constraint on the resource length for many processors has been considered here. Furthermore, the consumption of energy is computed using the equation 24

$$G = \left(\frac{C(t_1, t_2, \ldots., t_o)}{\tilde{V}}\right)^{c-1} Y \quad (24)$$

The ratio of the performance is given by the equation 25

$$\alpha \leq \left(\frac{C(t_1, t_2, \ldots., t_o)}{Y/o}\right)^{c-1} \quad (25)$$

It has been observed that $V' = C(v_1, v_2, \ldots., v_p)$where it is also assumed as $C(t_1, t_2, \ldots., t_p)\left(\frac{Y}{\tilde{G}}\right)^{1/(c-1)}$. Therefore, the consumption of energy is derived and expressed using the following equation 26

$$G = \left(\frac{V'}{\tilde{V}}\right)^{b-1} = \left(\frac{C(t_1, t_2, \ldots., t_p)}{\tilde{V}}\right)^{c-1} Y \quad (26)$$

Therefore, using the lower bound situation, the ratio of performance is derived in equation 27

$$\alpha = \frac{G}{G^*} \leq \left(\frac{C(t_1, t_2, \ldots., t_p)}{Y/o}\right)^{c-1} \quad (27)$$

Furthermore, consumption of power is given as

$$r = \left(\frac{C(t_1, t_2, \ldots., t_p)}{\tilde{V}}\right)^{c} \quad (28)$$

4. Performance Evaluation

The demand for resources in cloud computing has been increasing rapidly over the years because of its flexibility, cost efficiency, and various uses. It can be accessed easily at any given time and place. Further, using the HPC cluster plays an important role with the cloud to perform HPC-aware applications. However, cluster performance directly depends on the workload and task scheduling. Hence, these devices must have a higher performance considering the rapidly increasing demand [31]. Consumption of energy is high in these devices of computation which hinders performance.

Furthermore, an essential constraint in this is makespan. Therefore, for the optimization of these objectives, it is necessary to introduce heterogeneous PEAS computing devices for the efficient reduction in energy consumption and also give a higher performance. Moreover, PEAS is considered a simulator [19][21]. This is modified and adapted for the HPC cloud environment.

4.1. Dataset Details for HPC application evaluation

The runtime evaluation is performed at various instances at 25, 50, 100, and 1000. The results are graphically represented by considering the time of execution, count of tasks, and energy consumption. The total consumption of power, as well as run time, is evaluated by the use of various parameters that are explained and evaluated in the section expressed below. The proposed PEAS model is evaluated on the cybershake and montage scientific dataset [17]. The various parameters that are considered include the total time of simulation, the sum of power, the average power, and energy consumption [32-36]. These different parameters are evaluated at various cybershake instances and montage instances.

4.2. Designing Instances for HPC Cloud evaluation

For cybershake, every instance has a varying value of cybershake and montage value. Instance A is defined when the cybershake is 30, and the virtual machine is 25. The cybershake being 50 and the virtual machine at 50 is the instance B. Instance C is defined as the cybershake at 100 and the virtual machine at 100. The cybershake and virtual machine both have values of 1000 at the instance D. Considering the evaluation of montage, these different parameters are evaluated for the same montage instances. In the montage, instance A is defined when the montage value is 25 and the value of the virtual machine is 25. When the values of both montage and the virtual machine are at 50, it is at instance B. Instance C is given as the value of montage and virtual machine as 100. The montage value at 1000 and the virtual machine at 100 is at the instance D.

5. Results and Analysis

This graphical evaluation section is performed by comparing various parameters at different instances for both cybershake and montage. The various parameters include the total time of simulation, the sum of power, the average power, and energy consumption.

5.1. Makespan evaluation

The graph below shows the simulation time at various instances with a comparative analysis. A table is also given showing the values of the models at various instances. It has been observed that the simulation time increases with an increase in the workload. The model proposed is stable.

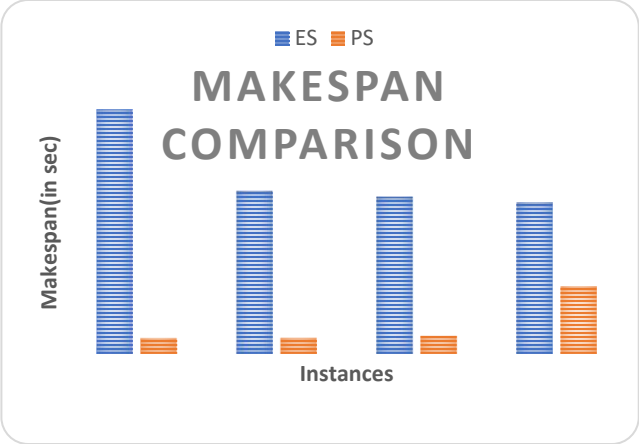

Fig. 1 Makespan comparison for cybershake workflow

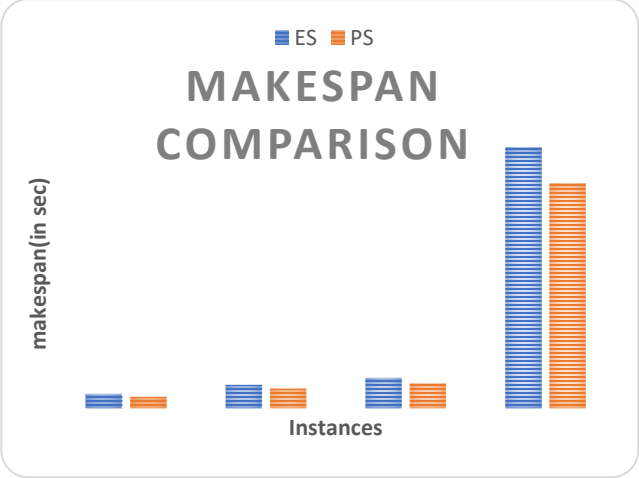

Fig. 2 Makespan comparison for montage workflow

5.2. Average Power

The tables and graphs below show the values and the comparative analysis at various instances. This comparative study is performed for cybershake as well as a montage. The proposed model results in being more efficient.

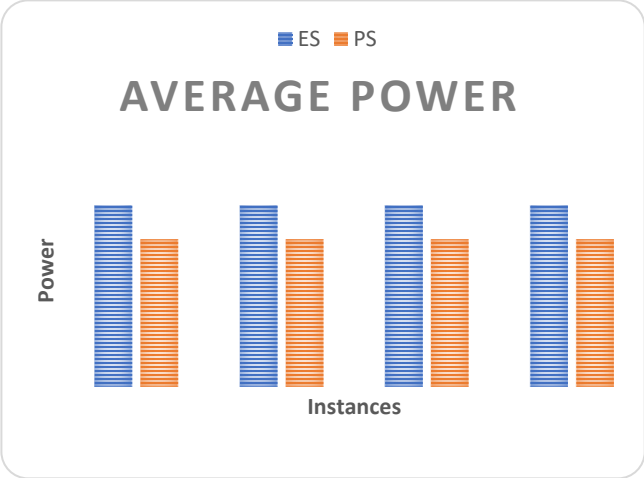

Fig. 3 Power comparison of cybershake workflow

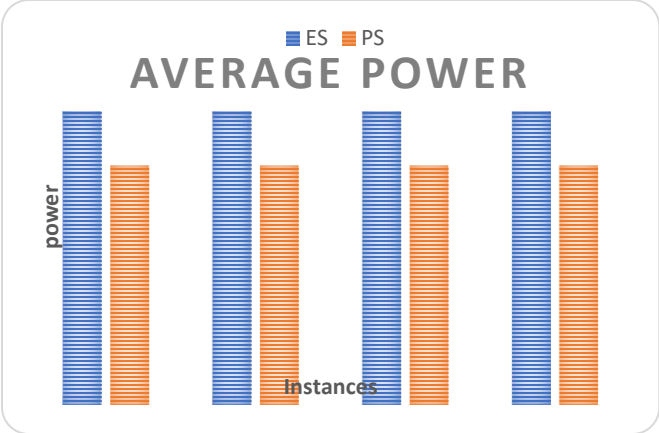

Fig. 4 Power Comparison of montage workflow

5.3. Energy utilization comparison

The graphs given below show a comparative analysis of the energy consumption parameter. In comparison, the table lists the detailed values of the models. The consumption is seen to vary largely in comparison to the two models.

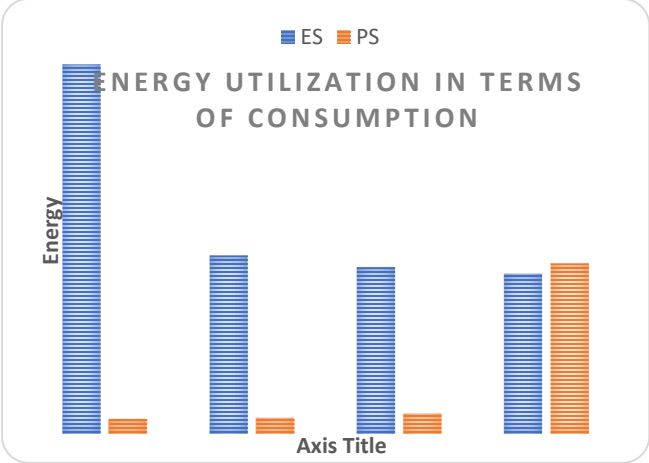

Fig. 5 Energy utilization evaluation through consumption comparison on cybershake workflow

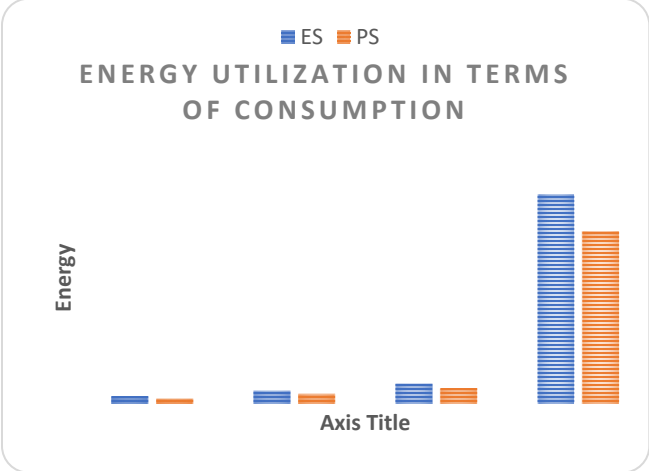

Fig. 6 Energy utilization in terms of consumption of energy on montage workflow

5.4 Improvisation over the existing model

This section discusses the improvisation over the existing model in terms of Energy utilization, Average Power, and makespan; table 1 presents the improvisation of the existing model with different instances. Table 1 presents the improvisation of PEAS over the existing model for cybershake workflow. In the case of Energy utilization, for instance, A and B PEAS improvise their model by 96.02% and91.06 %, respectively. Similarly, for instance, in C and D, PEAS observes 87.83 and 88.70 % improvisation, respectively. In the case of Average Power, PEAS observes nearly 18% improvisation over the existing model. Furthermore, in the case of makespan, instances A and B observe improvisation of 93.63 and 90.20, respectively. Similarly, C and D PEAS observe improvisation of 88.48% and 55.58%, respectively.

Table 1. Improvisation of PEAS over existing model considering cybershake workflow

HPC aware application	Energy utilization (in percentage)	Average Power(in percentage)	Makespan(in percentage)
Instance A	96.02111549	18.61438	93.63633
Instance B	91.06874	18.61436	90.20219
Instance C	87.8325	18.61433	88.48776
Instance D	88.70662	18.61412	55.58233

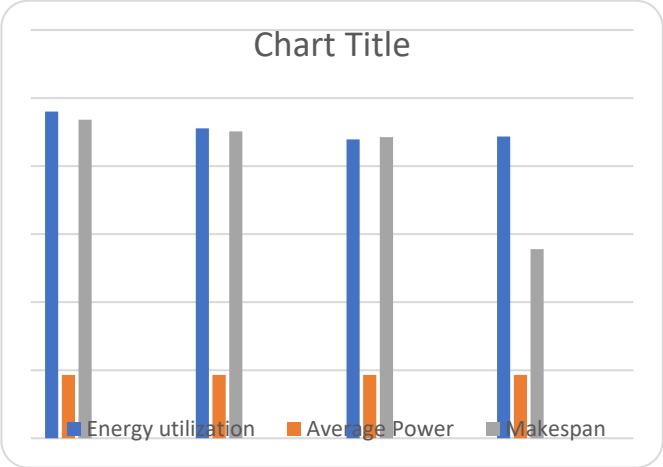

Fig. 7 Graphical improvisation on cybershake workflow

Table 2 presents the improvisation of the proposed PEAS over the existing model considering montage workflow. For instance, in A and instance B, PEAS observes improvisation of 30.37% and 26.95%; similarly, in C and D, PEAS observes improvisation of 23.64 %and 17.56 %, respectively. Further, in the case of all instances, PEAS observes improvisation of nearly 18%. Considering the makespan as a parameter, PEAS observes improvisation of 19.12 % and 16.32 % for instances A and B. Further, for Instance C and Instance D, PEAS observes improvisation of 18.40 % and 13.72%, respectively.

Table 2. Improvisation of PEAS over the existing model

HPC aware application	Energy utilization (in percentage)	Average power (in percentage)	Makespan (in percentage)
Instance A	30.37861	18.61515	19.12682
Instance B	26.95522	18.61467	16.3259
Instance C	23.64363	18.61456	18.40231
Instance D	17.56556	18.61412	13.72573

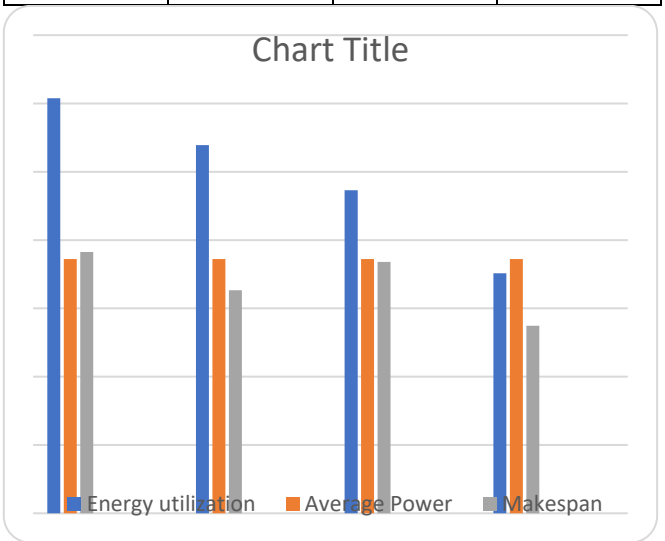

Fig. 8 Improvisation of PEAS over the existing model in montage workflow

6. Conclusion

In the past few years, commercial HPC users have moved on to clouds due to cost and alternatives to the dedicated clusters; the performance of HPC clusters highly depends on task scheduling, and recent schedulers lack optimal resource utilization. Hence, this research proposes a PEAS mechanism that helps optimal task scheduling and resource utilization. PEAS aims at optimizing the makespan and energy consumption. PEAS are evaluated considering the HPC-aware scientific workflow cybershake and montage considering the energy consumption, makespan, and Average Power. Further, a comparative analysis is carried out by designing four instances, i.e., instance A, instance B, instance C and Instance D. These instances comprise a certain number of VM and host. Comparative analysis indicates that PEAS observes marginal improvisation of up to 96% for all instances in terms of energy utilization, up to 19% improvisation in average power consumption, and up to 93% improvisation in makespan cybershake workflow. Similarly, for montage workflow, PEAS observes improvisation of up to 30% in terms of energy utilization, up to 19% improvisation in Average Power, and up to 19% improvisation in terms of makespan.

Although PEAS observes marginal improvisation for task scheduling which enhances the performance of HPC Clusters, other parameters need to be considered. Hence future work lies in cost optimization with a fault-tolerant approach.

References

[1] J. Emeras, S. Varrette, V. Plugaru, and P. Bouvry, "Amazon Elastic Compute Cloud (EC2) versus In-House HPC Platform: A Cost Analysis," in *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 456-458, 2019. Doi: 10.1109/TCC.2016.2628371.

[2] A. Pupykina, and G. Agosta, "Survey of Memory Management Techniques for HPC and Cloud Computing," in *IEEE Access*, vol. 7, pp. 167351-167373, 2019. Doi: 10.1109/ACCESS.2019.2954169.

[3] D. Dhinakaran, and P. M. Joe Prathap, "Preserving Data Confidentiality in Association Rule Mining Using Data Share Allocator Algorithm," *Intelligent Automation & Soft Computing*, vol. 33, no. 3, pp. 1877–1892, 2022. Doi:10.32604/iasc.2022.024509.

[4] B. Murugeshwari, D. Selvaraj, K. Sudharson and S. Radhika, "Data Mining with Privacy Protection Using Precise Elliptical Curve Cryptography," *Intelligent Automation & Soft Computing*, vol. 35, no. 1, pp. 839–851, 2023.

[5] A. C. Zhou, J. Lao, Z. Ke, Y. Wang, and R. Mao, "FarSpot: Optimizing Monetary Cost for HPC Applications in the Cloud Spot Market," in *IEEE Transactions on Parallel and Distributed Systems*. Doi: 10.1109/TPDS.2021.3134644.

[6] D. Dhinakaran, D. A. Kumar, S. Dinesh, D. Selvaraj, and K. Srikanth, "Recommendation System for Research Studies Based on GCR," *International Mobile and Embedded Technology Conference (MECON)*, Noida, India, pp. 61-65, 2022. Doi: 10.1109/MECON53876.2022.9751920.

[7] A. Saad, and A. El-Mahdy, "HPC Cloud Seer: A Performance Model Based Predictor for Parallel Applications on the Cloud," in *IEEE Access*, vol. 8, pp. 87978-87993, 2020. Doi: 10.1109/ACCESS.2020.2992880.

[8] VMware, "Host Power Management in Vmware Vsphere 5.5," *Tech. Rep. EN*- 001262-00, VMware Inc, 2013.

[9] A. Mazouz, A. Laurent, B. Pradelle, and W. Jalby, "Evaluation of CPU Frequency Transition Latency," *Comput. Sci. - Res. Dev*., vol. 29, no. 3, pp. 187–195, 2014. *Crossref*, http://dx. doi.org/10.1007/s00450-013-0240-x.

[10] R. Schöne, D. Molka, and M. Werner, "Wake-up latencies for Processor Idle States on Current X86 Processors," *Comput. Sci. - Res. Dev*, vol. 30, no. 2, pp. 219–227, 2015. *Crossref*, http://dx.doi.org/10.1007/s00450-014-0270-z.

[11] J. Kolodziej, "Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems," *Springer*, 2012.

[12] M.R. Garey, D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," *W. H. Freeman and Company*, 1979.

[13] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, and B. Luo, "Cost and Energy Aware Scheduling Algorithm for Scientific Workflows with Deadline Constraint in Clouds," *IEEE Trans. Serv. Comput*., vol. 11, no. 4, pp. 713–726, 2015. *Crossref*, http://dx.doi.org/10.1109/TSC. 2015.2466545.

[14] L. Wang, S.U. Khan, D. Chen, J. Kolodziej, R. Ranjan, C. Xu, and A. Zomaya, "Energyaware Parallel Task Scheduling in a Cluster," *Future Gener. Comput. Syst*., vol. 29, no. 7, pp. 1661–1670, 2013. *Crossref*, http://dx.doi.org/10.1016/j.future.2013.02.010.

[15] Y. Mhedheb, F. Jrad, J. Tao, J. Zhao, J. Kolodziej, and A. Streit, "Load and Thermalaware VM Scheduling on the Cloud," in: Proceedings of the 13th *International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'13)*, pp. 101-114, 2013. *Crossref*, http://dx.doi.org/10.1007/978-3-319- 03859-9_8.

[16] K. Mizotani, Y. Hatori, Y. Kumura, M. Takasu, H. Chishiro, and N. Yamasaki, "An Integration of Imprecise Computation Model and Real-Time Voltage and Frequency Scaling," in: Proceedings of the 30th *International Conference on Computers and Their Applications (CATA'15)*, pp. 63–70, 2015.

[17] H. Yu, B. Veeravalli, Y. Ha, and S. Luo, "Dynamic Scheduling of Imprecise Computation Tasks on Real-Time Embedded Multiprocessors," in: Proceedings of the 2013 *IEEE 16th International Conference on Computational Science and Engineering (CSE'13)*, pp. 770–777, 2013. *Crossref*, http://dx.doi.org/10.1109/CSE.2013.118.

[18] Malcolm Atkinson, Sandra Gesing, Johan Montagnat, and Ian Taylor, "Scientific Workflows: Past, Present and Future," *Future Generation Computer Systems*, vol. 75, pp. 216-227, 2017.

[19] C.C. Lin, Y.C. Syu, C.J. Chang, J.J. Wu, P. Liu, P.W. Cheng, and W.T. Hsu, "Energy Efficient Task Scheduling for Multi-Core Platforms with Per-Core DVFS," *J. Parallel Distrib. Comput*, vol. 86, pp. 71–81, 2015. *Crossref*, http://dx.doi.org/10.1016/j.jpdc. 2015.08.004.

[20] W. Long, L. Yuqing, and X. Qingxin, "Using CloudSim to Model and Simulate Cloud Computing Environment," 2013 *Ninth International Conference on Computational Intelligence and Security*, Leshan, pp. 323-328, 2013.

[21] K. Sudharson, and V. Parthipan, "A Survey on ATTACK – Anti-Terrorism Technique for Adhoc Using Clustering and Knowledge Extraction, Advances in Computer Science and Information Technology," *Computer Science and Engineering*, CCSIT 2012, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer, Berlin, Heidelberg, vol. 85, pp. 508-514, 2012.

[22] Sharavana. K, Asghar Pasha, and Dr. Josephin Premkumar K, "Approach for Deploying the Hybrid Cloud in Diverse Open Source Tools," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 20, no. 3, pp. 25-34, 2018.

[23] D. Dhinakaran, P.M. Joe Prathap, D. Selvaraj, D. Arul Kumar, and B. Murugeshwari, "Mining Privacy-Preserving Association Rules based on Parallel Processing in Cloud Computing," *International Journal of Engineering Trends and Technology*, vol. 70, no. 30, pp. 284-294, 2022. Doi: 10.14445/22315381/IJETT-V70I3P232.

[24] S. Arun, and K. Sudharson, "DEFECT: Discover and Eradicate Fool Around Node in Emergency Network using Combinatorial Techniques," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-2, 2020. Doi: https://doi.org/10.1007/s12652-020-02606-7.

[25] J. Aruna Jasmine, V. Nisha Jenipher, J. S. Richard Jimreeves, K. Ravindran, and D. Dhinakaran, "A Traceability Set Up Using Digitalization of Data and Accessibility," 2020 3rd International Conference on *Intelligent Sustainable Systems (ICISS)*, pp. 907-910, 2020.

[26] N. Partheeban, K. Sudharson, and P.J. Sathish Kumar, "SPEC- Serial Property Based Encryption for Cloud," *International Journal of Pharmacy & Technology*, vol. 8, no. 4, pp. 23702-23710, 2016.

[27] K. Sudharson, and V. Parthipan, "SOPE: Self-organized Protocol for Evaluating Trust in MANET Using Eigen Trust Algorithm," 2011 3rd *International Conference on Electronics Computer Technology*, Kanyakumari, India, pp. 155-159, 2011.

[28] L. Rahul, and K. Sharavana, "Deployment of Virtual HPC Clusters on Demand from Volunteer Computing Resources," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 4, 2014.

[29] D. Dhinakaran, and P.M. Joe Prathap, "Ensuring Privacy of Data and Mined Results of Data Possessor in Collaborative ARM, Pervasive Computing and Social Networking," *Lecture Notes in Networks and Systems*, Springer, Singapore, vol. 317, pp. 431-444, 2022. Doi: 10.1007/978-981-16-5640-8_34.

[30] I. Colonnelli, B. Cantalupo, I. Merelli, and M. Aldinucci, "StreamFlow: Cross-Breeding Cloud With HPC," in *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1723-1737, 2021. Doi: 10.1109/TETC.2020.3019202.

[31] K. Sudharson, M. Akshaya, M. Lokeswari and K. Gopika, "Secure Authentication scheme using CEEK technique for Trusted Environment," 2022 *International Mobile and Embedded Technology Conference (MECON)*, Noida, India, pp. 66-71, 2022.

[32] A. Fernandez, "Evaluation of the Performance of Tightly Coupled Parallel Solvers and MPI Communications in IAAS from the Public Cloud," in *IEEE Transactions on Cloud Computing*. Doi: 10.1109/TCC.2021.3052844.

[33] K. Sudharson and S. Arun, "Security Protocol Function Using Quantum Elliptic Curve Cryptography Algorithm," *Intelligent Automation & Soft Computing*, vol. 34, no. 3, pp. 1769–1784, 2022.

[34] D. Dhinakaran and P.M Joe Prathap, "Protection of Data Privacy from Vulnerability Using Two-Fish Technique with Apriori Algorithm in Data Mining," *Journal of Supercomputing*, 2022. *Crossref*, https://doi.org/10.1007/s11227-022-04652-8.

[35] Margesh Keskar, Dhananjay D Maktedar, "Evolutionary Computing Driven ROI-Specific Spatio-Temporal Statistical Feature Learning Model for Medicinal Plant Disease Detection and Classification," *International Journal of Engineering Trends and Technology*, vol. 70, no. 6, pp. 165-184, 2022. *Crossref*, https://doi.org/10.14445/22315381/IJETT-V70I6P220.

[36] Mandeep Singh, Shashi Bhushan, "CS Optimized Task Scheduling for Cloud Data Management," *International Journal of Engineering Trends and Technology*, vol. 70, no. 6, pp. 114-121, 2022. *Crossref*, https://doi.org/10.14445/22315381/IJETT-V70I6P214.