Efficient task optimization algorithm for green computing in cloud.

G, T., CH, D.C., VARMA, G.P.S. and MEKALA, M.S.

2023

Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

ARTICLE TYPE

Efficient Task Optimization Algorithm for Green Computing in Cloud

Thanmayatejaswi G[1] | Dileep Chakravarthy Ch[2] | G P S Varma[3] | M S Mekala*[4]

[1,2]School of Computer Science Engineering, SRKR Autonomous college, AP, India
[3,4]School of Computer Science Engineering, KL University, AP, India

Correspondence
*mekala. Email: dr.msmekala@email.com

Present Address
Present address

Abstract

Cloud infrastructure assets are accessed by all hooked heterogeneous network servers and applications to maintain entail reliability towards global subscribers with high performance and low cost is a tedious challenging task. Most of the extant techniques are considered a limited constraints like task deadline, which leads Service Level Agreement (SLA) violation. In this manuscript, we develop Hadoop based Task Scheduling (HTS) algorithm which considers a task deadline time, completion time, migration time and future resource availability of each virtual machine. The Intelligent System (IS) enabled with adaptive neural computation method to assess all above attributes. Specifically, the result of Prophecy Resource Availability (PRA) method has been used to assess the status of each Virtual Machine (VM), which helps to streamline the resource wastage and increase the response time with low SLA violation rate.

KEYWORDS:
Energy optimization; cloud computing; Hadoop system; measurement decision system

1 Introduction

Cloud Computing (CC) is an emerging technology opted by education institutes, industries and public sectors from past one decade. The cloud data centres (CDCs) are the backbone due to its huge scalable resource infrastructure includes Millions of server systems and their respective cooling hardware's. Those resources of CDC are apparently provisioned to various applications without latency in performance to users over the globe[1,2]. Each data centre has consumed approximately $10 sof MW$ of vitality for cooling and powering to the millions of server systems.

The objective of our manuscript is to develop a errand scheduling algorithm. It should be enable with errand execution time, vitality usage and resource usage rate to achieve high performance by balancing the workload. Underlying objective is - The tremendous enhancement of cloud user count management and diminishing errand execution time really meets the expected Quality of Service (QoS) of the user. Streamlining the workload balance rate among the VMs with good resource usage would be accomplished by consolidating the high resource used servers and assets wastage. The asset wastage is caused due to increase of idle assets count rate[3,4].

Our project mostly concentrates on below listed contributions. 1. Errand assignment approach emphasis on refining task completion and migration time with respective of vitality usage and asset usage rate, which also increases workload equilibrium ability rate. 2. The knowledge intelligence system evaluates the vitality and asset usage rate; these outcomes are plays an important role during decision making for task allocation.3. Develop a Prophecy Resource Availability (PRA) method to assess the

[0]**Abbreviations:** VM, Virtual Machine

VM assets usage rate before allocating the task, which gives stability in all aspects over the entire system.4. Novel framework design helps to reduce to processing time and cost of functionality, which shows high impact on vitality preservation and level of QoS without SLA violation.

2 Related Work

In[5], Min-Min approach has concentrated on efficient errand scheduling over cloud. It initially evaluates execution time of tasks and their respective assets results, which are less appropriate. In[6], QoS-aware cloud framework has designed as a fundamental phenomenon for errand scheduling label as QoS-Min-Min. It also identifies asset of same errands for errand scheduling as the response of requests, which outcomes are enhancing the execution time than traditional approaches. The depletion of power in data centers is due to laxity of memory, storage, CPU, and network interface. In their work Minimization of Migration (MM), they focus on lowering CPU utilization to reduce power consumption. They considered VM placement into 2 phases. In the first phase, the VM is given provisioning and is placed in host. In the second phase, current VM is optimized. They sort all VMs in an ascending order based on their weights and ladle of host to the VMs. By doing this method, they pleasantries that energy consumption gets decreased. The MM 50%-90% strategy centrals to 0.48 kWh of energy consumption.

In[7,8], the Max-Min algorithm initially estimates the capacity of the VM through the analysis of task execution time of each VM. It diminishes the errand and asset response time compared with existing systems. It has a drawback that, the lengthy errands have executed by low asset capacity system which outcomes are increases the makespan and cost of the system.

In[9], making high performance task scheduling with limited cores is an tedious challenging task. The author considered precedence attributes for taking a decision of mapping the errands to the VMs. The *task graph template* is labelled because of the author considered VM allocation for errand scheduling. This system didn't give an appropriate results because of lack of network consideration. The research scholars have to consider as a scope for further research stream. This drawback has been streamlined in[10], the author considered network devices and their stability for QoS and for reliability. In cloud simulation mechanism, every scholar has to remain two important factors like high data transfer rate and cost effectiveness. In[11,12], the author concentrates on task deadline and its cost as the attributes to select the VM for effective task scheduling. In[13], develops a new scheduling approach based on graph theory, were the author considered cost and the errand queue length; this approach did not allow re-submission of the failed tasks. The difficulties of errand scheduling have been discussed in this section. In[14], another heuristic approach for schematic programming schemes are developed over cloud frameworks; the outcomes have gloomy time complication.

In[15], the authors have concentrated on workload scheduling over cloud framework to streamline the issues. Those issues are caused due to vitality usage, node and machine unavailability. Consequently, the author has to deal with fixed attributes.

3 Proposed Architecture

The proposed framework has been classified into three parts for interactive task execution over cloud environment. *User area:* The framework starts with task assign towards V-instance-M via client, assets remain conveyed towards clients with plan of V-instances crusading of server farm. The client request has consider SLA measurements before presenting an errand towards the V-instance-M. *Resource area:* The asset section remain subsequently arranged towards VM & Physical Machine (PM) segment. With the advancement of virtual innovation, the VM area remains designed by remaining burdens of clients. In order to diminish VM processing expenses, clients can estimate their VMs dependent on their undertaking necessities. *Scheduling area:* The planning segment is classified with HTS calculation. The initial segment manages a positioning of all PMs in the cloud to discover PMt_{min} and PMt_{max} portion of VM by HTS calculation. The calculation compacts new undertakings to VM-M and coordinates VM on the PM. With objective towards debilitating the rendered assets in the cloud, assignments eat up to combine to least tally of PMs. Therefore, to reduce the utilization of vitality, inert hosts remain turned off. The subsequent part manages relocating the running VMs from the hosts that are completely used to spare the force utilization.

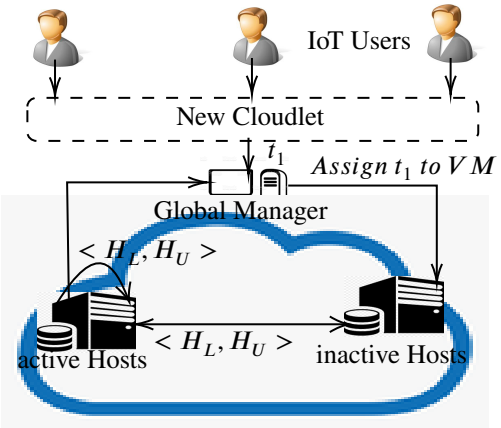


FIGURE 1 Proposed System Architecture

3.1 HTS Algorithm

Algorithm: DL enabled errand sorting
Input: < $host_{UUT}, host_{LUT}$ **with imbalanced utilization threshold>**
Output: Optimal Number of Feasible Tasks

Let Sort VM_{list} with processing time;
int T, $FT[t_i] = 0$, $NFT[t_i] = 0$, $WT[t_i] = 0$;
For {each VM_i in VM_{list}}{
$AH \leftarrow NULL$;
\For{each $host_j$ in $host_{list}$}{
\If{$!PM is\ suitable\ for\ VM_i$}{
Do nothing;\\
\Else{
$\text{host}_{LUT} \leftarrow VM_i$;
}
\If{Utilization > $host_{LUT}$}{
$host_{UUT} \leftarrow VM_i || VM_{list}$;
}
\Else{
$AH \leftarrow PM$;
}}}
\If{$AH \leftarrow PM$}{
Active the PM from inactive list;\\
}
\If{$AH \neq NULL$}{
Assign VMs $\rightarrow AH$ list;
}
\Else{
Inform $VM - M$;
}}
Return Feasible Tasks

TABLE 1

Initially un-level resource usage VMs are insist to migrated to suitable PM through optimal threshold usage rate of PMs, subsequently remain VMs of PM have sorted as non-decreasingly with processing time (line 2-4).The CPU usage of each PM might vary with other PMs. The PMs have been classified as PM_{UT} and PM_{LT} through the PM usage rate. If a PM has high CPU usage and violates PM_{UT} value, then such PM called as over-usage PM. Similarly, if PM violates PM_{LT} value,, then is called under-usage PM (line 5-18). The PMs comes under idle mode/VMs not yet assigned, then such PMs are switched off to diminish vitality usage (line 19-22). Legitimately, these PMs have to notify to the global manager (line 23-27). The HTS approach remain effectively yielded reduced number of deadline violation PM list.

3.2 Resource Integrated Modules

Errand Completion Time: The CPU accuracy is evaluated with MIPS. The MIPS refers to the performance reciprocally with completion rate; high-rank PM has high performance. The MIPS is evaluated with Eq. 1. During the evaluation of the errand deadline rate, the errand expected completion time is essential to derive. Therefore, the completion time is estimated with Eq. 2. Migration time is used to accomplish low latency with high performance; pre-estimation of migration time is an essential task. The outcome of this process plays an important role in making a decision before triggering the migration of resources. The migration time is usually calculated with Eq. 3. Prophecy Resource Availability (PRA) method emphasized (PM_i, PM_b, T, t_d) alludes behind event T has happened, earlier decision is processed through VM manager. Subsequently, the Markov decision method elements towards depict queuing model of active PMs using Eq. 4.

$$MIPS = \frac{Instructioncount}{ExecutiontimeX10^6} \tag{1}$$

$$Completion\,time = \frac{CPUclockcycles}{Clockrate} \tag{2}$$

$$Migration\,time = \frac{Amount\,of\ Resources}{CPUclockcycles \times Clockrate \times BD} \tag{3}$$

$$(S, \{A(s), s \in S\}, \{\tau[s,c], s \in S, c \in A(s)\}, \{p[t|s,c], s,t \in S, c \in A(s)\}) \tag{4}$$

$$CapacitVM_j = Core_j \times Core_{j,mips} + VM_{BD} \tag{5}$$

Where, $Core_j$ refers each VM core count VM_j, pe_{mips} refers core capacity of VM_j indicates in terms of million instruction per second, VM_{BD} remains a VM_j bandwidth capacity. The Performance, vitality relation (pvr) of each PM remains represents computation potentiality with Eq. 6. Where $PM.V_{sto}$, $PM.V_{BD}$, and $PM.V_{CPU}$ are alludes heap vitality cost for storage, bandwidth, cores of PM.

$$PM_{pvr} = \xi_t + \frac{(PM.core + PM.RAM) * PM.f_{max}}{(PM.V_{sto} + PM.V_{BD} + PM.V_{CPU}} \tag{6}$$

$$\xi_t = \left(\xi_s + \xi_{com}\right) \times \beta \cdot x_{ij} \times \left(f_{ij}(t)\right)^3 \tag{7}$$

Where, ξ_s ideal energy usage, ξ_{com} refer energy usage for computation, x_{ij} refer task allocation to j^{th} VM, $f_{ij}(t)$ refer execution frequency at time t and β refer constant amplification value. When there is no SLA default, the greatest burden hub can deal with

per unit time is the preparing intensity of the hub. The right expense and in-right expense of the asset are figuring as indicated by the heap forecast result and the cost of cloud hubs.

$$\underset{w=1}{\overset{W}{Min}}\left(\varpi_c^r\right) = \sum_{r=1}^{R} \lambda_r \times a_j + \omega \cdot \sum_{r=1}^{R}\sum_{j=1}^{J}\sum_{w=1}^{W} \lambda_r \times a_j \times \phi_w^j \times b_r \tag{8}$$

Where, λ_r threshold usage value, a_j refer task resource requirement rate, b_r resource availability rate. The total migration time of the migration scheme is

$$\mathbb{Z}_t = \sum_{r=1}^{R}\sum_{j=1}^{J} \frac{RD_r}{B_j} \tag{9}$$

Where, RD_r refer entail data, B_j refer demanded bandwidth of task.

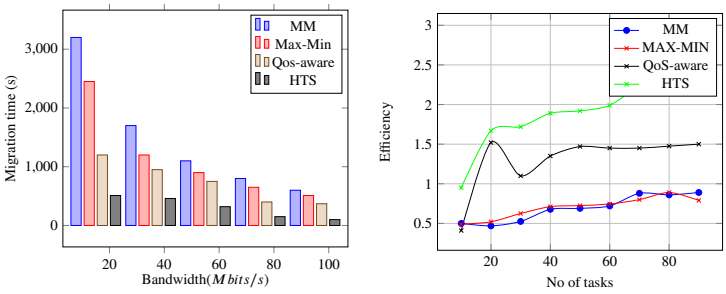
VM Computation Capacity estimates by using Eq. 5, which is essential to increase response time, diminishes VM migration count and also reduces vitality consumption. These factors are too primary under the infrastructure framework. We have designed Errand assignment approach based on task completion and migration time with respective vitality usage and asset usage rate. The knowledge intelligence system evaluates the vitality and asset usage rate during task allocation decision based on Prophecy Resource Availability (PRA) method assess VM assets usage rate and energy usage rate before allocating the task, which leads adequate stability in all aspects over the entire system. Equation 5 remains used to assess resource demand. Here, each PM resource demand, and its performance to be evaluated based on execution history and execution rate at time t. The equation 7 remain used to estimate the energy consumption of each PM/host. The host which remain potential in terms of execution cost, energy consumption, and should be less than the threshold value of equation 6 and 8 that PMs are be updated in active PM set and rest of the PMs in the inactive set. The equations 8 and 9 remain to play a vital role used to tackle migration operations to reduce the energy usage and its processing cost, such host, differentiated from not quality host or Physical Machine (PM).

4 Implementation and Experimental Analysis

To evaluate the presentation of benefit and assignment solidification computation in cloud plan, we used resource use, Service Level Agreement Violation (SLAV) rate and rented cost.

The normal CPU use, SLAV rate analysed. To diminish the trial blunder, the analysis was rehashed multiple times for each assessment index and accepts the normal value as the last assessment list esteem. Empirical outcomes explicate that, HTS algorithm has essentially degrades the deadline misses rate by at least 36 % and 10 % corresponded with Initial Distribution (IntiDis), Sercon scheduler and 16 % compared to QoS-Guide, Max-Min, and Min-Min scheduling algorithms.

Fig. 2 shows the connection among movement time, organize transmission capacity. The relocation time of various quantities of information square sizes are tried at various transfer speed. We assumed the transmission capacity 20, 40, 60, 80, 100 Mbit/s separately. The quantity of information squares of size 164M in the source hubs 10- 100 separately. The greatest estimation of the movement time is 3500, and this worth is determined under the state of 100 information squares and the transfer speed of 20 Mbit/s. The base relocation time is 250, determined under the state of 10 information squares with transfer speed of 100 Mbit/s.



(a) Migration time analysis based on bandwidth

(b) Execution analysis with efficiency rate

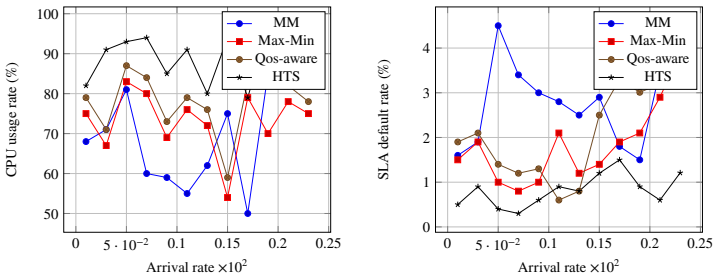
FIGURE 2 Migration and Execution analysis

The relocation time of 10 information squares is 83.7% lower than the movement time of 100 information squares under the data transfer capacity of 20 Mbit/s. The relocation time of 10 information squares is 81.7% lower than the relocation time of 100 information squares under the transmission capacity of 100 Mbit/s. As the transmission capacity increments, the estimation of relocation time of 100 information squares is drawing nearer to the estimation of the relocation time of 10 information squares.

The relocation time increments directly with the quantity of information square. The movement time diminishes exponentially as the accessible organize transfer speed increments straightly. The reason behind this is that the movement time is shorter

in higher connection transfer speeds also, littler the measure of information. As the forecast blunder increments, the asset request can not fit the pattern of a client request, so the all-out expense of the cloud framework increments step by step, which demonstrates the heartiness of our approach to the forecast blunder.

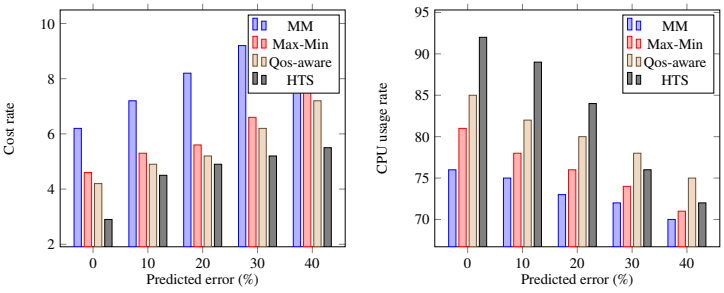
Figure 2 (b) delineates better outcomes for HTS in examination with different calculations.The recreation results show the effectiveness of the R-IS model-based HTS algorithm has enhanced while expanding task count than the Min-Min calculation. Here, we can observe comparable execution (at task count $\leqslant 90$). Moreover, with bigger number of undertakings (> 90), extra recreation results showed that the proposed HTS created essentially improved outcomes in contrast with Min-Min calculation. Fig. 3 (a) exhibits CPU usage for various periods under various load. It tends to be seen from the exploratory outcomes that the use of CPU assets under the three calculations changes with time. By computing the normal, we can realize that the HTS calculation has the most elevated normal CPU usage under high workload. The HTS calculation can make more full use of assets. Under the high load, the normal CPU usage is 24.1% higher than the QoS-aware calculation and 15.2% higher than the Max-Min calculation. The purpose behind this circumstance is that the QoS calculation and the Max-Min calculation don't precisely foresee the asset request, and don't think about the issue of the group asset cost enough.



(a) CPU usage rate vs arrival rate (b) SLA default rate vs arrival rate

FIGURE 3 CPU usage and SLAV analysis with arrival rate

The SLAV rate towards various time-spans over high load is appeared in Fig. 3 (b). The Max-Min calculation has a SLAV of 5.5% at arrival rate 0.05. This is on the grounds that the estimation of the heap is lacking, and the heap change can't be successfully managed with, bringing about a quick increment in the SLAV rate. At arrival rate 0.09, the quantity of assets that fulfill the heap is dispensed to lessen the SLAV rate. By figuring the normal CPU usage, asset allotment, and SLAV rate, and HTS calculation has reduced normal SLAV rate under high workload. The normal SLAV rate of the HTS calculation is 47.4% lower than the QoS-aware calculation and 70.3% lower than the Max-Min calculation. The effect of burden expectation blunder on the complete expense of a cloud is appeared in Fig. 4 . The all-out expense of cloud increments with the expansion of the forecast mistake under the three calculations. At the point when the forecast mistake is 0, the all-out expense of the HTS calculation is the littlest, and the QoS-Aware calculation is better than the Max-Min calculation. At the point when the forecast blunder is 40%, the aggregate cost of the HTS calculation increments by 1.34. The normal cost of HTS calculation is 22.5% lower than the QoS-Aware calculation and 35.8% lower than the Max-Min calculation. Fig. 4 shows the effect of burden forecast blunder on the CPU usage of a cloud. As the forecast mistake expands, the all-out expense of a cloud diminishes under the 3-calculations. At the point when the expectation blunder is 0, the CPU use of the HTS calculation is the most elevated, which is 94.2%. The QoS-Aware calculation is superior to the Max-Min calculation. When the expectation mistake is 30%, the CPU usage of the QoS-Aware calculation is higher than that of the HTS calculation. The normal CPU use of the HTS calculation is 2.9% higher than the QoS-Aware calculation and 13.2% lower than the Max-Min calculation. In view of the above outcomes, the outcomes exhibits that the absolute expense of the HTS calculation is diminished by 45.3% contrasted and the QoS-Aware calculation, the SLAV rate is diminished by 47.4%, the estimation of the asset assignment is decreased by 21.7%, and the worth of the CPU usage is expanded by 24.1%. The absolute expense of the HTS calculation is 51.2% lower than that of the Max-Min calculation, the quantity of SLAV is decreased by 70.3%, the estimation of the asset portion is decreased by 16.5%, and the estimation of the CPU usage is expanded by 15.2%.



(a) Predicted error (%) vs Cost rate (b) Predicted error (%) vs CPU usage rate

FIGURE 4 Predicted error vs CPU usage

5 Conclusion

In this paper, we proposed an errand, asset, and vitality utilization enhancement algorithm, where the unused cases are switch off to spare vitality. To limit the vitality utilization of machines, we proposed a powerful asset provisioning calculation to select the dynamic VMs and solidify the VMs to PMs. To guarantee the QoS pace of our asset and errand solidification approach, we proposed an information mindful procedure to decrease the quantity of VM relocations. Along these lines, we can accomplish the goal of limiting asset wastage and vitality utilization of HTS without yielding the nature of system administrations. Reenactment results have exhibited the adequacy of our proposed plot. The trial results show that the HTS calculation has certain preferences regarding cost, asset usage, and SLAV rate.

References

1. Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai,QoS prediction for service recommendation with deep feature learning in edge computing environment, in Mobile Networks and Applications. NewYork,NY, USA: Springer, 2019, pp. 1-11.
2. C. Zhang, R. Huang, and J. Zhang,Distributed adaptive consensus tracking of unknown heterogenous linear systems via output feedback, in Proc. 35th Chin. Control Conf., Chengdu, China, Jul. 2016, pp. 8008-8013.
3. H. Gao, S. Mao, W. Huang, and X. Yang, Applying Prophecy model checking to financial production risk evaluation and control: A case study of Alibaba's Yu'e Bao, IEEE Trans. Comput. Social Syst., vol. 5, no. 3, pp. 785-795, Sep. 2018.
4. G. Patel, R. Mehta, and U. Bhoi,Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing, Procedia Comput. Sci., vol. 57, pp. 545-553, Jan. 2015.
5. Y. Yin, S. Aihua, G. Min, X. Yueshen, and W. Shuoping,QoS prediction for Web service recommendation with network location-aware neighbor selection, Int. J. Softw. Eng. Knowl. Eng., vol. 26, no. 4, pp. 611-632, 2016.
6. B. A. Al-Maytami, P. Fan, and A. Hussain,I-MMST: A new task scheduling algorithm in cloud computing, in Proc. Int. Conf. Intell. Comput. Cham, Switzerland: Springer, 2018.
7. F. U. Xiao, Task scheduling scheme based on sharing mechanism and swarm intelligence optimization algorithm in cloud computing, Comput. Sci., vol. 45, no. 1, pp. 303-307, 2018.
8. Mekala, M S and Viswanathan, P., 2019, Equilibrium Transmission Bilevel Energy Efficient Node Selection Approach for Internet of Things, Wireless Personal Communications, Oct, 108 (3), pp. 1635-1663.
9. J. Y. Maipan-uku, A. Muhammed, A. Abdullah, and M. Hussin, Maxaverage: An extended max-min scheduling algorithm for grid computing environtment, J. Telecommun., Electron. Comput. Eng., vol. 8, no. 6, pp. 43-47, Sep. 2016.
10. H. Aziza and S. Krichen, Bi-objective decision support system for task- scheduling based on genetic algorithm in cloud computing, Computing, vol. 100, no. 2, pp. 65-91, Feb. 2018.
11. K. Baital and A. Chakrabarti, Dynamic scheduling of real-time tasks in heterogeneous multicore systems, IEEE Embedded Syst. Lett., vol. 11.1, pp. 29-32, Mar. 2018.
12. Mekala, M S and Viswanathan, P., 2019, Energy-efficient virtual machine selection based on resource ranking and utilization factor approach in cloud computing for IoT, Computers & Electrical Engineering, 73, pp.227-244.
13. Mahammad Shareef Mekala P. Viswanathan (2020) A survey: energy-efficient sensor and VM selection approaches in green computing for X-IoT applications, Int. J of Computers and Applications, 42:3, 290-305.
14. M. Al-Khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, Improving fog computing performance via fog-2-fog collaboration, Future Gener. Comput. Syst., vol. 100, pp. 266-280, Nov. 2019.
15. L. Chiaraviglio, F. DAndreagiovanni, An approach to balance maintenance costs and electricity consumption in cloud data centers, IEEE Trans. Sustain. Comput., vol. 3, no. 4, pp. 274-288, May 2018.