Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9783131, 1 page
https://doi.org/10.1155/2023/9783131

Retraction

Retracted: Sustainable Technical Debt-Aware Computing Model for Virtual Machine Migration (TD4VM) in IaaS Cloud

Wireless Communications and Mobile Computing

Received 12 December 2023; Accepted 12 December 2023; Published 13 December 2023

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] A. Vashistha, C. M. Sharma, R. P. Mahapatra, V. M. Chariar, and N. Sharma, "Sustainable Technical Debt-Aware Computing Model for Virtual Machine Migration (TD4VM) in IaaS Cloud," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 6709797, 12 pages, 2022.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 6709797, 12 pages
https://doi.org/10.1155/2022/6709797

Research Article

Sustainable Technical Debt-Aware Computing Model for Virtual Machine Migration (TD4VM) in IaaS Cloud

Avneesh Vashistha,[1] Chandra Mani Sharma,[2] Rajendra Prasad Mahapatra,[1] Vijayaraghavan M. Chariar,[2] and Navel Sharma[3]

[1]*Department of Computer Science & Engineering, SRM Institute of Science & Technology, Delhi-NCR Campus, Ghaziabad, Uttar Pradesh, India*
[2]*CRDT, Indian Institute of Technology Delhi, New Delhi, India*
[3]*Department of Computer Engineering & Informatics, Academic City College, Accra, Ghana*

Correspondence should be addressed to Chandra Mani Sharma; cmsharma.its@gmail.com and Navel Sharma; drnavel.sharma@gmail.com

Received 17 March 2022; Revised 6 April 2022; Accepted 8 April 2022; Published 25 April 2022

Academic Editor: Aruna K K

In the cloud, optimal CPU and memory utilization can lead to low energy consumption, which is an important aspect of green computing. However, constantly changing workloads may contribute to resource over- or underutilization. The former violates the service level agreement's quality of service constraints. The latter indicates that as workload decreases, virtual machine resource utilization decreases. They introduce difficult decision-making tasks when dynamically adapting (e.g., migrating) a virtual machine in order to maximize its resource utilization over time. To address these challenges, we propose a newer mathematical model called the technical debt-aware computing model for virtual machine migration (TD4VM). The model promotes a holistic approach to dynamic virtual machine adaptation for cloud service providers and addresses existing issues regarding logical aspects of virtual machine adaptation in a highly dynamic cloud environment, which includes a measurement mechanism and estimation guidelines for estimating future debt and utility. Technical debt-aware models make decisions based on VM operating costs, quality, minimizing SLA violations, and incurring technical debt. This approach connects decisions about virtual machine migration that affect overall utility over time. Our method can determine whether a virtual machine should be moved when it is over or underutilized based on its technical debt. The experimental results on a dataset obtained from the Materna-trace-1 demonstrate that the proposed approach outperforms other state-of-the-art methods on a variety of performance metrics. A numerical comparison shows that TD4VM outperforms the other approaches, with VM resource economies of 171.84%, 91.33%, 97.85%, and 93.89% for TD4VM, LRMMT, IQRMC, and IQRMMT, respectively. Additionally, we quantify the debt amassed using TD4VM and state-of-the-art techniques. When compared to LRMMT, IQRMC, and IQRMMT, which cost (in $) 0.77, 0.73, and 0.76, respectively, TD4VM accumulates the minimum debt of 0.17.

1. Introduction

Virtualization is the backbone of cloud computing and facilitates the creation of a required number of VMs on a physical machine for maximum resource utilization. The running of a data center may be minimized by shutting down the idle nodes or switching the state of idle nodes to low-power modes that result in a reduction of electricity consumption [1]. On-demand resource pooling enables the dynamic adaptation of resources (e.g., VMs) as per application requirements [2]. Moreover, virtual machine live migration may dynamically consolidate the minimum number of physical machines. The dynamic adaptation of the virtual machine could be strategic decisions that need to satisfy the QoS

constraints specified in the SLA and maximize the resource utilization of the underlying virtual machine. However, cloud dynamics lead to uncertainties; for example, the varying workload could be the reason for either underutilization or overutilization of a virtual machine [3]. Both cases are undesirable in the cloud environment. To address these issues, we propose an approach that leverages the technical debt-aware computing model for virtual machine migration (hence called TD4VM). This research paper is summarized as follows:

(1) This paper presents a technical debt-aware model for VM migration that leverages the principal of the technical debt (TD). In contrast, we explicitly map the TD in the context of VM migration, which allows us to build the technical debt-aware model

(2) We tailored Holt-Winters' multiplicative method, which accelerates our model for estimating the future debt of the VM

(3) We implemented the TD4VM model on the classical cloud framework, namely, CloudSim 5.0 [4]. Further, TD4VM decides whether to migrate a VM by considering long-term benefits (e.g., maximized VM utilization in the future)

(4) We compared our newly developed approach, TD4VM, with existing approaches based on resource utilization and accumulated debt over time

This paper has been structured as follows: Section 2 explores the related work. Section 3 showcases the problem statement and motivates the introduction of a technical debt-aware model for VM migration. In this section, we emphasize how TD4VM may be introduced at the VM level in IaaS cloud; elaborate Holt-Winters' multiplicative method for workload prediction; formulate a VM debt model driven by the technical debt metaphor; and present the VM utility model, which takes the advantages of the debt computing model for optimizing the VM migration decision and VM utility model. Section 5 explores an extensive discussion on the experimental evaluation and results. Section 6 explores the conclusion and proposes future work.

2. Related Work

Inefficient usage of VM resources affects the running cost of a cloud data center. When data is collected from several servers for six months for the purpose of the experiment, even though servers are rarely idle, utilization rarely reaches 100% [5]. Usually, the servers operate at 10–50% of their maximum utilization level, which leads to extra expenses on underutilization [1, 6]. Dynamic server migration and consolidation algorithms reduced resource consumption by 50% when compared with static resource allocation, resulting in a 20% reduction in SLA violations [7]. Although a framework for mapping between task and resource allocation can radically advance resource utilization, that results in improving the profit of primary cloud providers [8]. If resource allocation is done intelligently according to the user's requests, the global profit of providers can be maximized [9]. Based on the strategies used in VM placement, server consolidation, and load balancing in virtual resource management, a novel methodology takes robust decisions and makes the process easier to choose a more appropriate PM [10]. Agile, a lightweight, prediction-driven, and resource-scaling system, significantly reduced service level objectives (SLO) violations when compared to existing resource scaling mechanisms [11]. Green computing is a term that refers to reducing power consumption in a system by optimizing resource utilization [12]. Efficient code design and intelligent decision-making may aid in achieving the green computing objectives [13]. RAM and CPU utilization are inextricably linked to power consumption in a cloud computing environment [14].

3. Problem Statement

Idle servers consuming 70% of their peak power may lead to underprovisioned servers and affect the running cost of a cloud data center [15]. Moreover, the migration of the entire operating system between virtual machines with minimal service downtime results in cluster administration and allows separation of hardware and software considerations. It also strengthens clustered management into a single articulated management domain [16]. Based on the existing approaches for VM allocation, we have identified that the workload on virtual machines changes dynamically. Considering this dynamism, virtual machines may be provisioned according to the required number of running applications [17]. Elasticity, interoperability, and scalability are important characteristics that can be used to reduce the operating costs of cloud resources [17]. These characteristics result in VM dynamic allocation and/or releasing but may not prevent VMs from being under or overutilized [17, 18]. An underutilized VM incurs technical debt for the service provider. Overuse of VM results in resource scarcity for users and SLA violations, which are frequently accompanied by a financial penalty [17, 18]. A suboptimal virtual machine always carries a technical debt, and the reasons could be strategic, managerial, or even unintentional. To address these issues, we introduce a technical debt-aware model into the context of VM migration that also minimizes the VM computing cost while keeping SLA and quality of service (QoS) [1, 2]. In particular, this research paper presented a technical debt-aware computing model for virtual machine migration in the IaaS cloud.

3.1. Technical Debt. The most cited source of technical debt is Ward Cunningham's report "The WycashPortfolio Management System," released in the year 1992. In this paper, the term "debt" was used for the first time and described how debt is the result of a violation of good code and architecture practices during software development [19]. However, even before Ward Cunningham, the problem of software development and maintenance was mentioned by Meir Lehman in the 1980s. He emphasized the urgent need for disciplined software engineering; otherwise, as software development evolves, its complexities also increase unless work is done

to maintain or reduce them [20]. Earlier, the technical debt metaphor has been introduced in various fields such as software architecture [21–24], software code quality [25], software design [26], documentation [27], rework [28], software testing [29], refactoring [30–33], compliance debt [34], social debt [35], and agile development process [36].

3.2. Motivation: Why Take on Technical Debt? VM optimization may be improved if a VM is being migrated between hosts which are either underutilized or overutilized. We argued that VM migration decisions may carry technical debt and transformed from a liability to future values [17, 18, 21]. In this paper, we convince our peers that there is a need to consider the technical debt metaphor as one of the essential parameters for VM migration. A similar argument has already been given as "a little level of debt is not always a bad," as it can help to decide for VM migration [37]. We sight this statement as a valid statement for VM migration [21]. Unlike previous work, we put forward the fact that VM migration decisions should not only be QoS-aware but also be long-term value and technical debt-aware. In our view, a little debt is more acceptable than the decision of VM migration that could result in clearing technical debt. With this perspective, this paper introduces a technical debt-aware model for VM migration. In this model, we may lose short-term gains by choosing less attractive options or by solving problems in the short term. We considered two perspectives when taking on the technical debt model for VM migration.

(1) The decision to migrate VMs must be made strategically and with a long-term perspective in mind rather than a short-term one

(2) Future opportunities that VM migration decisions create

Furthermore, we view the investment in the VM migration decision as a loan that may occur with interest over time and signals a probable technical debt. This incurred technical debt needs to be identified, tracked, and managed as well for value creation and maximum VM utilization [21].

3.3. Technical Debt in Virtual Machine Migration under IaaS Cloud. In IaaS, technical debt is defined as the difference between the ideal and actual revenue generated by a virtual machine. We argue that a VM may be under- or overutilized during execution. This leads to technical debt that has to be monitored for prevention and must also be managed by proactive decisions to generate future value. Technical debt may also be used for the identification of a futuristic workload, which is just several requests that are being assigned to or executed by a virtual machine for a given time interval [18]. Based on the predicted workload, it may also be estimated in advance how many VMs would be required to process the predicted workload [17, 18].

3.4. Technical Debt Indicators

3.4.1. SLA Violation. Trust in the service provider could be seen in the form of SLA and could be the reason for unintentional debt on VM [17]. The penalty cost for each violation would be counted as interest in this case [18, 21]. Penalty to be imposed on the customer can be further classified as [38] fixed penalty: whenever a virtual machine is overutilized, a fixed penalty on resources per percentage of SLA violation has to be given by the service provider to the customer; when the service provider is not able to provide computing capacity for a certain amount of time, the delayed penalty is calculated as the delayed capacity × the amount of time. A proportional penalty is also calculated as an extended form of a delay-dependent penalty. Furthermore, the penalty must be calculated in terms of the proportion of each resource per percentage of the delay incurred in returning the capacity.

Table 1: Average operating cost of VM usage for all approaches over 7200 minutes.

VM resources	TD4VM	LRMMT	IQRMC	IQRMMT
Operating cost	1.41	1.56	1.56	1.56

3.4.2. Run-Time Decision. A VM may accumulate technical debt because of poor or bad run-time decisions for VM migration if the allocated VM may not be able to match current requirements [17, 18, 21].

3.4.3. VM Utility. From a utility point of view, a suboptimal VM is the result of under-/overutilization that accumulates technical debt on the VM, which could be either good or bad [17, 18].

4. Measuring Technical Debt in VM Migration

Because of the nature of the dataset, we apply Holt-Winters' multiplicative method, a time series forecasting method. This method has been used to predict the workload on a virtual machine. Based on the predicted data, we then present a VM debt computing model and a VM utility model in the context of VM migration.

4.1. Workload Prediction on VM. Time series data is temporal and is used to predict future values based on previous values. The time component is an important variable and is involved in lots of prediction problems but is mostly limited to research labs rather than industrial applications [39]. As a result, we favor Holt-Winters' multiplicative method over other commonly used time series methods for workload prediction on a virtual machine [17, 18]. We preprocess the dataset for 10 days for training and the next 5 days of data for testing the accuracy of monitoring every 5 minutes. To predict the dataset values for the CPU, memory usage, and total resource usage, we fit Holt-Winters' multiplicative method and evaluate the prediction accuracy using the common accuracy metrics for CPU usage and memory usage. We have used MSE, MAE, and RMSE metrics, as shown in Table 1, to estimate the accuracy of forecasted values using the Holt-Winters method.

4.2. VM Debt Computing Model. To estimate the technical debt on a given virtual machine, we employed the notation *principal* and *interest* defined in the technical debt metaphor [17, 18, 21]. Furthermore, we systematically connect these notations for building our VM debt computing model as follows:

4.2.1. Principal. We are considering the principal as the invested cost of searching and selecting the new VM for performing VM migration operations. In particular, the principal can be calculated using the following equation:

$$Principal = \left(C_{cpu} * T\right), \tag{1}$$

where T indicates the time required for searching and migration and C_{cpu} represents the CPU execution cost for processing the VM migration operation.

4.2.2. Interest. The interest could be accumulated on the VM execution over some time when VM execution exhibits the underprovision or overprovision of its resource utilization. For the VM, the interest could be accumulated up to n future timestamps, which can be obtained from the difference between the actual VM resource provisioning and predicted resources. Moreover, the interest can be driven from two different cases of VM resource utilization as the following equation.

$$\text{Interest(VM)} = \left\{ \begin{array}{l} \sum_{R=1}^{m} \sum_{t=1}^{n} \left(\text{VM}^{R}_{\text{provision}} - \text{VM}^{R}_{\text{utilization}}\right) * C_{\text{exec}} \text{if} \text{VM}^{R}_{\text{provision}} > VM^{R}_{\text{utilization}} \\ \sum_{t=1}^{n} \left(\text{SLA}_{\text{constraints}} - \sum_{R=1}^{m} VM^{R}_{\text{utilization}}\right) * C_{p} otherwise \end{array} \right\}, \tag{2}$$

where R denotes the VM resources (*CPU* and *RAM*). C_{exec} and C_p are the resource execution cost and penalty, respectively.

Case 1. Interest for overprovision of VM resources. In this case, the resources provisioning on the VM is higher than the resource utilization [17, 18] ($VM^{R}_{\text{provision}} > VM^{R}_{\text{utilization}}$). The interest would be calculated as the execution cost of unused VM resources (VM_R) over n future timestamps, which is derived from the difference between the provisioned resources by the VM and the actual utilization of VM resources as shown in the top formula in equation (2) [17, 18].

Case 2. Interest for underprovision of VM resources. Unlike the previous case, e.g., overutilization of VM resources violates the *SLA constraints* due to more demand for VM resources (e.g., specified in the *SLA*) than the actual provisioned resources on the VM. The interest would be accumulated as the penalty cost against *SLA constraints* violation over n timestamps, as shown in the top formula in equation (2). Finally, the overall technical debt on the VM resources utilization can be estimated according to the principal and accumulated interest as

$$Debt_{VM} = Principal + Interest(VM). \tag{3}$$

4.3. VM Utility Model. In this section, we present a VM utility model for estimating the execution cost of VM resources. In this work, we consider CPU and RAM as the VM resources, and each resource has a different execution cost denoted by C^{R}_{exec}. We compute the VM resource execution cost using the following equation.

$$C(VM_{\text{exec}}) = \sum_{R=1}^{m} VM^{R}_{\text{provision}} * C^{R}_{\text{exec}}, \tag{4}$$

where R denotes the VM resources (*CPU* and *RAM)* and $VM_{\text{provision}}$ indicates the current provisioning of resources to the VM. Further, there may be an environmental condition, in which VM-provisioned resources could not satisfy the SLA constraints. Consequently, it causes the SLA violation, which is compensated by paying the penalty cost to VM resource violation. We calculate the penalty cost by

$$C\left(VM_{\text{penalty}}\right) = \left(SLA_{\text{constraints}} - \sum_{R=1}^{m} VM^{R}_{\text{provision}}\right) * C_{p}, \tag{5}$$

where $SLA_{\text{constraints}}$ denotes the specified constraints in the SLA and C_p is the penalty cost against the violation of SLA constraints. Based on the above functions, we are in the position to build the VM utility model, which is capable of quantifying the current (actual) utility of VM resources in the cloud environment using

$$U^{c}_{VM} = C(VM_{\text{exec}}) - C\left(VM_{\text{penalty}}\right). \tag{6}$$

4.3.1. Debt-Aware VM Utility. In the previous section, we formulated the VM debt computing model that would support the debt-aware VM utility model for making an economic-driven decision for dynamic VM allocation. Furthermore, optimizing the present value of the VM utility in the cloud data center will depend on the future value of the underlying VM utility [17, 18, 38]. In this regard, we adopted a predictive learning approach (time series forecasting method) for forecasting the VM resource

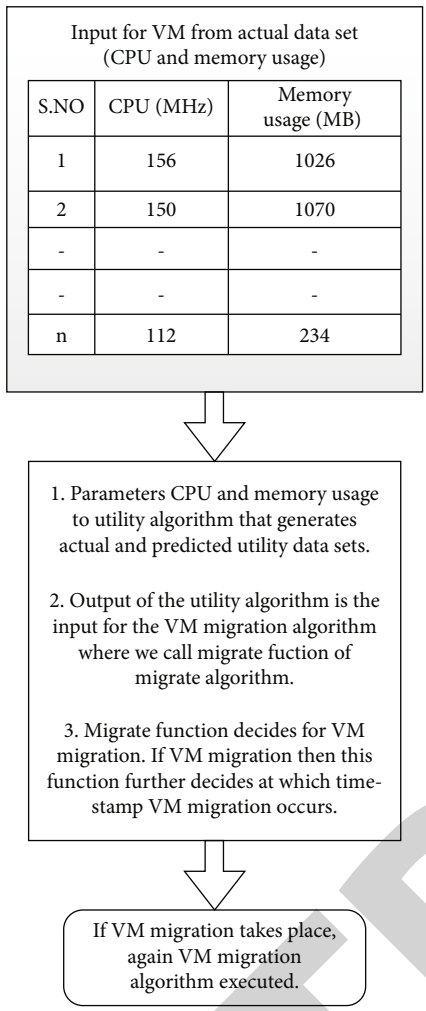

Figure 1: Schematic diagram showing the VM migration strategy.

demand in the cloud data center. After that, we combined the forecasting method with our VM debt computing model to predict future debt up to n future timestamps, as shown in equation (3). Moreover, we calculate the future VM resource execution cost over n time steps based on the forecasted VM resources. Finally, we build the debt-aware VM utility model, which is capable of estimating the debt-aware predictive utility of the underlying VM as shown in the equation.

$$U_{VM}^{D} = \sum_{t=1}^{n} C(VM_{exec}) - Debt_{VM}. \quad (7)$$

4.4. Debt-Aware Decision for Dynamic VM Migration. Based on Sections 4.2 and 4.3, we developed algorithms for the technical debt-aware decision model for dynamic VM migration, which are based on the utility algorithm, VM migration algorithm, and migrate algorithm. The working of these algorithms has been shown in Figure 1.

As discussed in Figure 1, our first algorithm, the utility algorithm, generates the actual utility and predicted utility, bypassing CPU usage, memory usage, CPU provisioning, and memory provisioning parameters as inputs (from the actual dataset and the predicted dataset) to the utility function. In this function, the penalty is double (the sum of CPU and memory execution costs) and the principal are the fixed costs. The amount of CPU and memory provisioned depends on the current VM configuration. The following are the important steps:

(1) In line number 6, we calculate the VM's current utility, i.e., the VM running cost, which is just the sum of CPU and memory cost as calculated in lines numbers 4 and 5

(2) From lines 7 to 8, we estimate the incurred debt on CPU and memory

(3) From lines 9 to 14, we identify whether a VM is overprovisioned or underprovisioned. Interest is calculated if the VM is overprovisioned; otherwise, a penalty is calculated

(4) In line number 15, we calculate the accumulated technical debt

(5) Actual and predicted utilities are generated on lines 16 and 17

The output of the utility algorithm produces the actual and predicted utilities that would be the input for our next algorithm: *VM Migration*.

In the VM migration algorithm, we used two functions, VM migration and migrate. VM migration function takes two parameters *cu set* and *pu set*, which are the current utility set and the predicted utility set. In this algorithm, the following are the important steps:

(1) From lines 3 to 8, we find the average of the current utility denoted by cu_{avg} for the three timestamps

(2) From lines 9 to 11, we find the average of the predicted utilities, denoted by T_1, T_2, and T_3 for timestamps t_1; an average of t_1 and t_2; and an average of t_1, t_2, and t_3, respectively

(3) From lines 12 to 17, we are considering two special cases for the last two or one timestamp values, when either the predicted utilities for two or one timestamps left in place of three timestamps

(4) In line number 19, we call the function "Migrate" bypassing the values of timestamp current utility average (cu_{avg}), an average of predicted utilities T_1, T_2, and T_3 as parameters

(5) The function "Migrate" decides whether a VM is migrated or not. If the VM migration decision has been taken, then this function further decides at which timestamp VM migration occurs. Line numbers 11 and 12 of this function help decide the timestamp for VM migration

Input: Cpu& Memory Usages Set (Cpu_{Usage} & $Memory_{Usage}$) //Actual And Predicted Datasets Cpu& Memory Provisioning Set ($Cpu_{provisioning}$ & Memory $_{provisioning}$)
Constant Input: Cpu& Memory Executioncost ($Cpu_{executioncost}$ &Memory $_{executioncost}$)
Fixed Penalty Cost
Principal // Principal is the Rework Cost of VM
Output: Actual Utility/Predicted Utility Of VM
Utility (Cpu_{Usage} , $Memory_{Usage}$, $Cpu_{provisioning}$, $Memory_{provisioning}$)

1. Initialize: Interest =0
2. Penalty =0
3. **For (Each Time Stamp)**
4. $Cpu_{cost} = Cpu_{usage} * Cpu_{executioncost}$
5. $Memory_{cost} = Memory_{usage} * Memory_{executioncost}$
6. Current Utility = Cpu_{cost} + $Memory_{cost}$ //Calculating VM Running Cost
7. $Debt_{memory}$ = (Current $Memory_{provisioning}$ - $Memory_{usage}$) * $Memory_{executioncost}$
8. $Debt_{cpu}$ = (Current $Cpu_{provisioning}$ - Cpu_{usage}) * $Cpu_{executioncost}$
9. **If (($Cpu_{provisioning}$ + $Memory_{provisioning}$) > (Cpu_{Usage}+ $Memory_{Usage}$))** // Over-Provisioning
10. Interest =$Debt_{cpu}$ + $Debt_{memory}$
11. **Else** // Under-Provisioning
12. % of Penalty =100 – ((Cpu_{Usage} / $Cpu_{provisioning}$) + (Memory $_{Usage}$ / Memory $_{Provisioning}$))*50
13. Penalty = Fixed Penalty Cost * (% Of Penalty)
14. **End If**
15. Total Debt = Interest + Penalty + Principal // For Actual/Predicted Datasets
16. Actual Utility = Current Utility - Penalty // Generates Actual Utility
17. Predicted Utility = Current Utility - Technical Debt // Generates Predicted Utility
18. **End For**

Algorithm 1: Utility.

Input: Current Utility set (cu)
Predicted Utility set (pu)
Output: Decision of VM migration, if migrate then at which time stamp.
// t is the timestamp and T is the time-period of 3 time-stamps (t_1, t_2, t_3)
VM Migration (cu set, pu set) //function receiving current and predicted utility datasets as parameters

1. Initialize: time stamp t =0
2. **while (t <= n-1) //n is the total no of time stamps**
3. **if (t = = 0)**
4. $cu_{avg} = cu_t$
5. **else if (t = = 1)**
6. $cu_{avg} = (cu_t + cu_{t-1})/2$
7. **else**
8. $cu_{avg} = (cu_t + cu_{t-1} + cu_{t-2})/3$
9. $T_1 = pu_{t+1}$
10. $T_2 = (pu_{t+1} + pu_{t+2})/2$
11. $T_3 = (pu_{t+1} + pu_{t+2} + pu_{t+3})/3$
12. **if (t = = n - 2)**
13. $T_3 = -\infty$
14. **else if (t = = n-1)**
15. $T_3 = -\infty$
16. $T_2 = -\infty$
17. **end if**
18. **Migrate (t, cu_{avg}, T_1, T_2, T_3)//calling Migrate function**
19. t = t+1
20. **End while**

Algorithm 2: VM migration.

Input: Current timestamp
Average of Current Utility group (cu_{avg})
Average combination of Predicted Utilitygroup (T_1, T_2, T_3)
Migrate (t, cu_{avg}, T_1, T_2, T_3)
1. Initialize: mig= n+1
2. **if ($cu_{avg} \leq T_3$)**
3. No need to migrate
4. **else if ($cu_{avg} \leq T_2$)**
5. mig= t+3
6. **else if ($cu_{avg} \leq T_1$)**
7. mig= t+2
8. **else**
9. mig= t+1
10. **end if**
11. **if (mig ≤ n)**
12. 'migrate at timestamp' **mig**
13. **end if**

Algorithm 3: Migrate.

5. Experimental Results

To evaluate the proposed approach, we have conducted a quantitative experiment. The primary goal of this experiment is to validate the effectiveness of our approach against other existing approaches in the CloudSim simulator. Specifically, we aim to develop a technical debt-aware decision model for virtual machine migration. Besides, this model provides a mechanism for managing technical debt for the migration of virtual machines dynamically in the cloud data center. To evaluate the technical debt-aware approach, we aim to answer the following research questions:

RQ1: How accurately does the TD4VM approach predict resource utilization?

RQ2: Can the TD4VM approach outperform the other approaches based on resource utilization (e.g., CPU and RAM) and infrastructure operating cost?

RQ3: Can the TD4VM approach accumulate less debt than other existing approaches?

We have simulated our results on the CloudSim simulator, as it is a modern and advanced cloud simulation framework [18]. We used one data center, comprised of five hosts. Each host contains several virtual machines having a configuration according to the pricing scheme of GCP (n1-machine) [40]. Whenever a host is under- or overutilized, VMs from that host are migrated to the other hosts based on the technical debt-aware model. Our experiment was carried out on a machine equipped with an Intel Core i5 (2.8 GHz) Processor, 8 GB of RAM, and the Windows 10 operating system. In this experiment, the current usage of CPU and memory every five 5-minute time intervals (total 7200 minutes-5 day datasets) is collected from Materna-trace-1 [17, 18, 41] and used as input for TD4VM and other approaches.

5.1. Comparison with State-of-the-Art Approaches. We compared our TD4VM approach with the other classical approaches incorporated into the CloudSim framework, namely: IQRMMT, IQRMC, and LRMMT [1, 17].

(i) IQRMMT. IQRMMT is an adaptive threshold algorithm that selects the candidate VM with the minimum migration time relative to other VMs [3].

(ii) IQRMC. IQRMC is an adaptive threshold algorithm that selects the candidate VM with the maximum correlation with the other VMs [3].

(iii) Local regression based on the Loess method is used to fit a trend polynomial to the last observation of CPU utilization [3].

5.2. Performance Metrics. We employ the following metrics for evaluating our *TD4VM* approach against other existing approaches:

(1) Prediction accuracy. To assess the prediction accuracy of resource utilization using the Holt-Winters method, we used the MSE, MEA, and RMSE metrics

(2) Resource utilization. We examine resource utilization for 120 hours (7200 minutes) using equation (6). We plot the result of VM resource utilization against the actual allocated resources. Furthermore, we use the box plot to compare the resource utilization yields of all approaches

(3) Operating cost. We measure the infrastructure operating cost for 120 hours (7200 minutes) using equation (4). We plot the result of the infrastructure operating costs consumed by all approaches

(4) VM debt. We use equation (3) to figure out how much debt the VM has accumulated over 120 hours (7200 minutes). We plot the accumulated debt result of TD4VM against other state-of-the-art approaches

5.3. Results Discussion for RQ1. To answer RQ1, we provide the prediction accuracy of VM resource utilization using different metrics, as shown in Table 2. Furthermore, the

Table 2: VM resource prediction accuracy.

Resources	MSE	MAE	RMSE
CPU	0.01	0.55	0.10
RAM	0.01	0.08	0.11

obtained results show that the MSE, MAE, and RMSE are in fact comparatively low for both resources (CPU and RAM), and thus the accuracy is acceptable [21]. For a more detailed result, Figures 2 and 3 depict the CPU and RAM predictions, respectively. As we can see in Figures 2 and 3, there are some variations between the predicted and actual resource utilization, but the prediction captures the general patterns of the resource utilization, e.g., the spike between 500 and 1500 minutes.

For RQ1, we conclude that:

Answering RQ1: The VM resource prediction of the Holt-Winter method in our TD4VM approach is acceptably accurate. The variation between predicted and actual resource utilization is small, and most of the patterns are generally captured.

5.4. Results Discussion for RQ2. To answer RQ2, we plot the result of resource utilization and infrastructure operating cost yield by all approaches.

5.4.1. Resource Utilization-Based Assessment. To measure the resource utilization of all approaches, we conducted two sets of experiments considering different VM resource parameters. In the first experiment, we use only the CPU parameter, which is generally used by all classical state-of-the-art approaches in CloudSim [1]. Furthermore, to analyze the effectiveness of our TD4VM approach, we included RAM as an additional parameter in the second experiment. The obtained results from both experiments are discussed as follows.

(1) Measuring VM CPU Utilization. We assess the CPU utilization produced by TD4VM and other state-of-the-art approaches for executing the VM over 7200 minutes.

In Figure 4, the CPU utilization obtained from the TD4VM approach is higher than the CPU utilization yield achieved by other approaches. To conduct a more detailed review, we plot the CPU utilization measured at each timestamp for a period of 7200 minutes. In particular, we examined the variation of CPU usage in all approaches. The CPU utilization reported by the TD4VM approach is consistently better than other approaches. Furthermore, Table 3 provides an overview of the average CPU utilization of all approaches for 7200 minutes. Numerically, TD4VM achieved the highest average value of CPU utilization compared to other approaches. Finally, these experiments demonstrated that the TD4VM approach is more capable of making a long-term economic-driven decision for VM migration and outperforms other approaches. We also applied the Kruskal-Wallis test at a significant level of 5% and obtained results showing that the p-value is less than 0.05, which explains the significant difference in CPU utilization yield by all approaches.

(2) Measuring VM Resource Utilization (CPU and RAM). We analyze the overall VM resource utilization which includes the CPU and RAM over 7200 minutes.

Figure 5 illustrates the VM resource utilization (CPU and RAM) for 7200 minutes. We can see that the TD4VM approach achieves higher utilization of VM resources than other state-of-the-art approaches. Furthermore, we conduct a more comprehensive review by examining the VM resource utilization at each timestamp of 7200 minutes. The TD4VM approach consistently better utilizes the VM resources than other approaches. In addition, we looked at how much VM resources each approach used for 7200 minutes, as shown in Table 4. A numerical comparison was conducted, in which the IQRMC approach has better VM resource utilization than other existing approaches (such as LRMMT and IQRMMT), but when comparing IQRMC with our approach TD4VM, we see that the TD4VM outperforms all other state-of-the-art approaches. We also applied the Kruskal-Wallis test at a significant level of 5% [12], and the obtained results show that the p-value is less than 0.05, which informs the significant difference in CPU utilization yield by all approaches.

5.4.2. Operating Cost-Based Evaluation. We examine the operating cost consumed by all approaches for executing a VM over 7200 minutes in the cloud data center. The VM operating cost could be optimized (e.g., in terms of minimal invested cost) by making a strategic decision for VM migration; otherwise, it impacts the VM resource utilization. Table 1 provides an overview of the average operating cost yield by all approaches over 7200 minutes. Notably, our TD4VM approach provides an economic-driven decision for dynamic VM migration. Consequently, TD4VM consumes fewer operating costs than other state-of-the-art approaches [17, 18].

For RQ2, we conclude that:

Answering RQ 2 From the above discussion, we conclude that the TD4VM approach has better VM resource utilization than other approaches. Moreover, TD4VM approach provides long-term based economic-driven decision for dynamic VM allocation that maximizes the overall resource utilization and minimize the VM execution cost.

5.5. Results Discussion for RQ3. We measure the debt accumulated by our TD4VM approach and other state-of-the-art approaches for executing a VM over 7200 minutes in the cloud data center.

The cost of unused VM resources or the penalty cost for SLA violation during VM execution are two potential sources of debt. From Figure 6, we observe that the increased value of debt negatively impacts the VM resource utilization, as shown in Figure 5. As a result, the approach that accumulates less debt could be more efficient for maximizing the VM's resource utilization. To conduct a more detailed review, we plot the accumulated debt on the VM execution

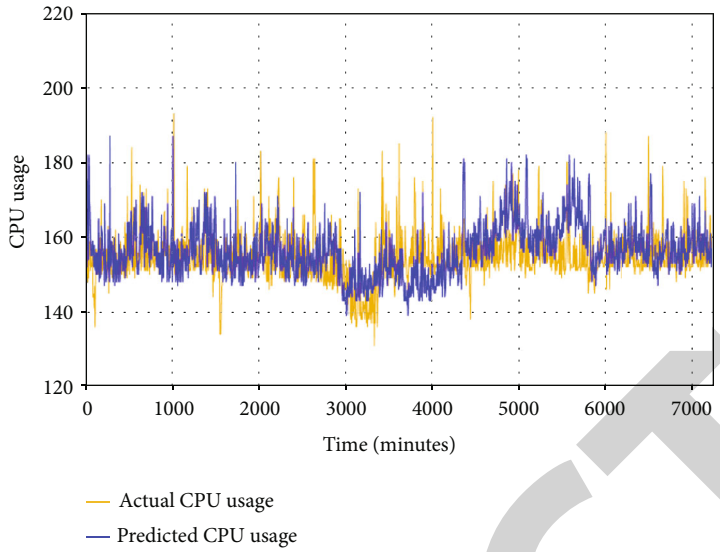

Figure 2: CPU usage prediction.

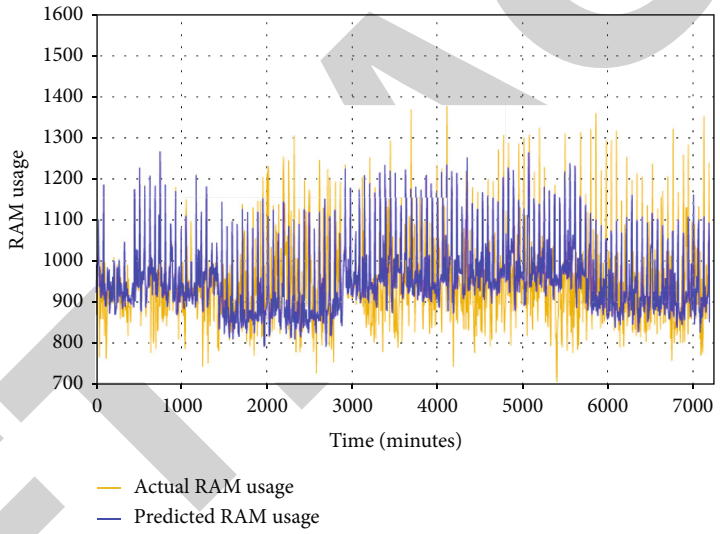

Figure 3: RAM usage prediction.

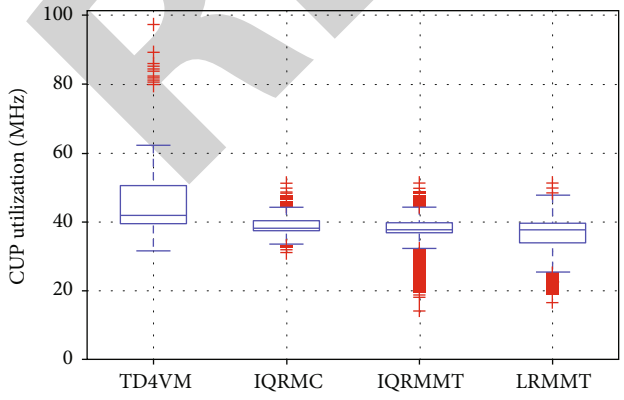

Figure 4: CPU utilization yield by all approaches.

for each timestamp of 7200 minutes. In particular, we examine the pattern of accumulated debt using all approaches. Figure 6 shows that the TD4VM approach incurs less debt than other existing approaches. Furthermore, we compute the average debt accumulated by all approaches for executing a VM over 7200 minutes, as shown in Table 5. In terms of numbers, TD4VM has the lowest minimum value (cost in dollars) of accumulated debt compared to other approaches. Finally, we conclude that the TD4VM outperforms other approaches. We also applied the Kruskal-Wallis test at a significant level of 5% and obtained results that show that the p -value is less than 0.05, which explains the significant difference in accumulated debt yield by all approaches.

For RQ3, we conclude that:

Answering RQ3: From the above discussion, we observed that the TD4VM approach accumulates less debt, which is a good sign for maximizing the VM resource utilization. The

Table 3: Average CPU utilization of all approaches over 7200 minutes.

VM resource	TD4VM	LRMMT	IQRMC	IQRMMT
CPU (MHz)	44.56	35.93	38.90	37.04

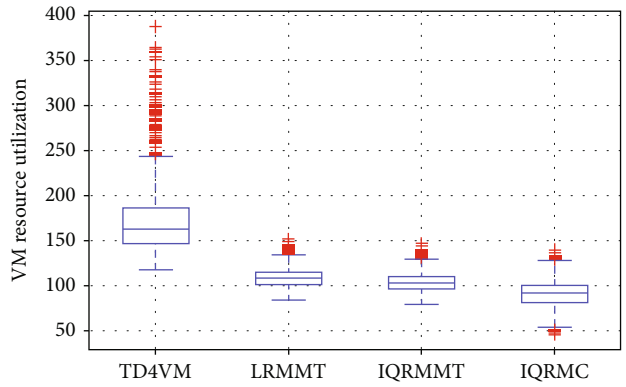

Figure 5: VM resource utilization yield by all approaches.

Table 4: Average VM resource utilization of all approaches over 7200 minutes.

Resource	TD4VM	LRMMT	IQRMC	IQRMMT
VM resources	171.84	91.33	97.85	93.89

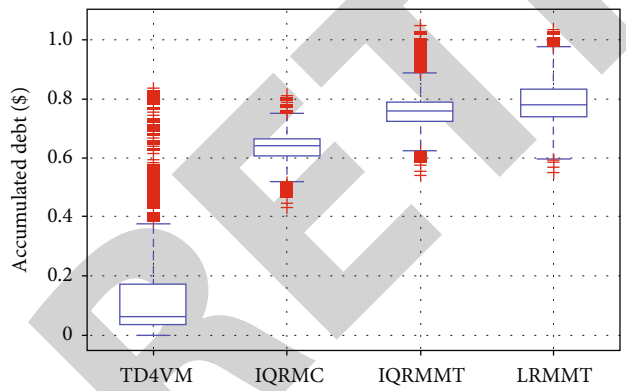

Figure 6: Operating cost consumed by all approaches.

Table 5: Average debt accumulated by all approaches over 7200 minutes.

VM resource debt	TD4VM	LRMMT	IQRMC	IQRMMT
Accumulated debt ($)	0.17	0.77	0.73	0.76

comparative discussion shows that the TD4VM accumulates less debt than other state-of the-art approaches.

5.6. Summary of Analysis. The uncertainty in a cloud environment may introduce resource contention among others. The data centers' first and foremost motive is to reduce the running cost as much as possible. Reducing running costs results in maximizing revenue. The running cost of a data center may be controlled and minimized if resource usage can be predicted in advance. Scalability and elasticity motivate the adoption of a cloud computing IaaS model that enables different benefits from the economies of scale in the cloud. Moreover, since it is impossible to achieve a perfect mapping between resource consumption and resource provisioning, existing mechanisms are usually unaware of incurring technical debt.

6. Conclusion and Future Work

We introduced the technical debt-aware approach (TD4VM), which reduces virtual machine operating costs, SLA violations, and technical debt. Technical debt can be defined as the difference between optimal and suboptimal VM migration decisions. Our approach connects VM migration decisions and accrues technical debt over time, reducing overall utility. Our model is capable of estimating the accumulated technical debt carried by a virtual machine well in advance and also assisting in VM migration decisions to address the issue of virtual machine under-/overutilization. We intend to extend this model in additional ways, such as

(1) To extend the technical debt-aware model by introducing additional mechanisms such as self-adaptivity. A self-adaptive system can analyze its own behavior and adjust the current system in order to achieve its objectives and manage its operations autonomously in the face of uncertainty. We currently consider CPU and memory utilization as parameters for VM migration in our current approach. Apart from fault tolerance, delay, security, and virtual machine recomposition, there are additional uncertainties that could be addressed in future research.

(2) Our technical debt-aware approach adapts VMs based on future values but ignores past values. We can extend this approach by taking into account historical values. We can deduce possible ways to improve the feasibility of our current system based on the pattern observed for previous values.

Data Availability

Data is available with authors and can be provided on request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] Z. Zhou, M. Shojafar, M. Alazab, J. Abawajy, and F. Li, "AFED-EF: an energy-efficient VM allocation algorithm for IoT applications in a cloud data center," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 2, pp. 658–669, 2021.

[2] M. Aslam, S. Bouget, and S. Raza, "Security and trust preserving inter- and intra-cloud VM migrations," *International Journal of Network Management*, vol. 31, no. 2, article e2103, 2021.

[3] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[4] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[5] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp. 826–831, IEEE, Melbourne, VIC, Australia, 2010.

[6] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[7] X. Fan, W. D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 13–23, 2007.

[8] H. Jin, L. Deng, S. Wu, X. Shi, H. Chen, and X. Pan, "MECOM: live migration of virtual machines by adaptively compressing memory pages," *Future Generation Computer Systems*, vol. 38, pp. 23–35, 2014.

[9] C. Chatfield, "The Holt-Winters forecasting procedure," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 27, no. 3, pp. 264–279, 1978.

[10] P. V. Avneesh Vashistha, "Economic driven model for virtual machine allocation in cloud data center," *International Journal on Emerging Technologies (IJET)*, vol. 11, no. 4, pp. 269–273, 2020.

[11] S. Kumar, R. Bahsoon, T. Chen, and R. Buyya, "Identifying and estimating technical debt for service composition in SaaS cloud," in *2019 IEEE International Conference on Web Services (ICWS)*, pp. 121–125, IEEE, Milan, Italy, 2019.

[12] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "An energy-efficient VM prediction and migration framework for overcommitted clouds," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 955–966, 2018.

[13] B. Wang, F. Liu, and W. Lin, "Energy-efficient VM scheduling based on deep reinforcement learning," *Future Generation Computer Systems*, vol. 125, pp. 616–628, 2021.

[14] Y. Liang, Z. Hu, and K. Li, "Power consumption model based on feature selection and deep learning in cloud computing scenarios," *IET Communications*, vol. 14, no. 10, 2020.

[15] C. Li, Y. Hu, L. Liu et al., "Towards sustainable in-situ server systems in the big data era," *ACM Sigarch Computer Architecture News*, vol. 43, no. 3S, pp. 14–26, 2015.

[16] S. S. Gill, P. Garraghan, V. Stankovski et al., "Holistic resource management for sustainable and reliable cloud computing: an innovative solution to global challenge," *Journal of Systems and Software*, vol. 155, pp. 104–129, 2019.

[17] N. Rodrigo, R. Rajiv, R. César, and B. Rajkumar, "Cloudsim: a novel framework for modeling and simulation of cloud computing infrastructures and services," pp. 1–9, 2009, http://arxiv.org/abs/0903.2525.

[18] M. Tomczak and E. Tomczak, "The need to report effect size estimates revisited. An overview of some recommended measures of effect size," *Trends in Sport Sciences*, vol. 1, no. 21, pp. 19–25, 2014.

[19] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 119–128, IEEE, Munich, Germany, 2007.

[20] B. Song, M. M. Hassan, and E. N. Huh, "A novel heuristic-based task selection and allocation framework in dynamic collaborative cloud service platform," in *2010 IEEE second international conference on cloud computing technology and science*, pp. 360–367, IEEE, Indianapolis, IN, USA, 2010.

[21] Z. Zhu, J. Bi, H. Yuan, and Y. Chen, "SLA based dynamic virtualized resources provisioning for shared cloud data centers," in *2011 IEEE 4th International Conference on Cloud Computing*, pp. 630–637, IEEE, Washington, DC, USA, 2011.

[22] M. Mishra and A. Sahoo, "On theory of vm placement: anomalies in existing methodologies and their mitigation using a novel vector based approach," in *2011 IEEE 4th International Conference on Cloud Computing*, pp. 275–282, IEEE, Washington, DC, USA, 2011.

[23] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes, "{AGILE}: elastic distributed resource scaling for {infrastructure-as-a-service}," in *10th International Conference on Autonomic Computing (ICAC 13)*, pp. 69–82, San Jose, CA, 2013.

[24] A. Strunk and W. Dargie, "Does live migration of virtual machines cost energy?," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 514–521, IEEE, Barcelona, Spain, 2013.

[25] L. Chen and H. Shen, "Consolidating complementary VMs with spatial/temporal-awareness in cloud datacenters," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 1033–1041, IEEE, Toronto, ON, Canada, 2014.

[26] A. Mosa and N. W. Paton, "Optimizing virtual machine placement for energy and SLA in clouds using utility functions," *Journal of Cloud Computing*, vol. 5, no. 1, pp. 1–17, 2016.

[27] C. M. Sharma and H. Kumar, "Architectural framework for implementing visual surveillance as a service," in *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 296–301, IEEE, New Delhi, India, 2014.

[28] M. M. Lehman, "Programs, life cycles, and laws of software evolution," *Proceedings of the IEEE*, vol. 68, no. 9, pp. 1060–1076, 1980.

[29] N. Brown, Y. Cai, Y. Guo et al., "Managing technical debt in software-reliant systems," in *In Proceedings of the FSE/SDP workshop on Future of software engineering research*, pp. 47–52, New York, 2010.

[30] Z. Li, P. Liang, and P. Avgeriou, "Architectural debt management in value-oriented architecting," in *Economics-Driven Software Architecture*, pp. 183–204, Morgan Kaufmann, 2014.

[31] H. Liu, F. Zhong, B. Ouyang, and J. Wu, "An approach for QoS-aware web service composition based on improved genetic algorithm," in *2010 International conference on web information systems and mining*, vol. 1, pp. 123–128, IEEE, Sanya, China, 2010.

[32] M. Younas, D. N. Jawawi, I. Ghani, T. Fries, and R. Kazmi, "Agile development in the cloud computing environment: a systematic review," *Information and Software Technology*, vol. 103, pp. 142–158, 2018.

[33] K. Krishnaiyer, F. F. Chen, and H. Bouzary, "Cloud Kanban framework for service operations management," *Procedia Manufacturing*, vol. 17, pp. 531–538, 2018.

[34] F. Shull, "Perfectionists in a world of finite resources," *IEEE Software*, vol. 28, no. 2, pp. 4–6, 2011.

[35] R. L. Nord, I. Ozkaya, P. Kruchten, and M. Gonzalez-Rojas, "In search of a metric for managing architectural technical debt," in *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, pp. 91–100, IEEE, Helsinki, Finland, 2012.

[36] R. Haas, R. Niedermayr, and E. Juergens, "Teamscale: tackle technical debt and control the quality of your software," in *In2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*, pp. 55-56, IEEE, Montreal, QC, Canada, 2019.

[37] E. Moreira, F. F. Correia, and J. Bispo, "Overviewing the liveness of refactoring for energy efficiency," in *Conference Companion of the 4th International Conference on Art, Science, and Engineering of Programming*, pp. 211-212, New York, 2020.

[38] A. Fox, R. Griffith, A. Joseph et al., *Above the clouds: A berkeley view of cloud computing*, vol. 28, Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, 2009.

[39] J. Fritzsch, J. Bogner, A. Zimmermann, and S. Wagner, "From monolith to microservices: a classification of refactoring approaches," in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, pp. 128–141, Springer, Cham, 2018.

[40] R. Kazman, Y. Cai, R. Mo et al., "A case study in locating the architectural roots of technical debt," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, pp. 179–188, IEEE, Florence, Italy, 2015.

[41] B. Ojameruaye and R. Bahsoon, "Systematic elaboration of compliance requirements using compliance debt and portfolio theory," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 152–167, Springer, Cham, 2014.