Hassan *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:63
https://doi.org/10.1186/s13677-020-00212-8

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Integrated resource management pipeline for dynamic resource-effective cloud data center

Hanan A. Hassan[1], Aya I. Maiyza[1*] and Walaa M. Sheta[1,2]

Abstract

Cloud computing is a popular emerging computing technology that has revolutionized information technology through flexible provisioning of computing resources. Therefore, efforts to develop an effective resource management approach have found that implementing efficient resource sharing among multiple customers that considers power saving, service-level agreements, and network traffic simultaneously is difficult. This paper proposes a practical integrated pipeline that can use various algorithms. The performance of each algorithm is evaluated independently to obtain the combination of algorithms that guarantees a resource-effective cloud data center framework. This integrated resource management pipeline approach would optimize performance based on several key performance indicators, such as power saving, network traffic, and service-level agreements, for either the whole system or the end-user. The performance of the proposed resource management framework was evaluated using a real testbed. The results demonstrated that the proactive double exponential smoothing algorithm prevents unnecessary migrations, the MMTMC2 VM selection algorithm improved the quality of service for end-users and reduced overall energy consumption and network traffic, and the medium-fit placement algorithm provided load balancing between active servers and decreased service level agreement violations. The performance comparison illustrated that the combination of these algorithms was considered to be the best solution toward a dynamic resource-effective cloud data center. Our results showed that energy consumption and the total number of migrations decreased by 16.64% and 49.44%, respectively.

Keywords: Cloud computing, OpenStack, Resource management, Service-level agreement, SLA violation, Power saving, Live migration, Cloud data centers

Introduction

Cloud computing has revolutionized resource sharing to provide several types of e-services to users. Cloud computing provisioning is based on virtualization techniques that obtain an abstract view of physical resources using the same interfaces. It provides several benefits, for example, the same physical infrastructure can be used for different runtime environments simultaneously. The rapid growth in cloud computing has increased the importance of cloud data centers (CDCs). Consequently, CDC administrators exert continuous effort to determine approaches to improve performance, maximize profit, increase multitenancy capabilities, and enhance infrastructure density. Thus, resource management (RM) has become a promising research area for CDCs.

A primary goal of RM is to simultaneously reduce resource utilization and comply with end-user service-level agreements (SLA) as much as possible. The complexity of the RM issue appears in the trade-off between energy saving and service level agreement violations (SLAVs), which is associated with the total number of migrations. For instance, virtual machine (VM)

*Correspondence: amaiyza@srtacity.sci.eg
[1]Informatics Research Institute, City of Scientific Research and Technological Applications, SRTA-City, Alexandria, Egypt
Full list of author information is available at the end of the article

consolidation often results in either low energy efficiency or violations of SLA constraints, depending on the overcommitment level [1, 2]. Moreover, a VM suffers from quality of service (QoS) degradation in two cases: (i) frequent migration and (ii) allocation on an overloaded server. Therefore, an effective RM must identify the most suitable solution toward a dynamic resource-effective CDC. The essential RM challenges are energy efficiency, SLAVs, load balancing, and network load [3].

Therefore, the principal contribution in this paper is the presentation of a resource management pipeline (RMP) approach to guarantee a resource-effective CDC framework. To achieve this, the following algorithms are performed as part of the RMP approach:

(i) The exponential smoothing algorithm [4] is performed as a proactive overload detection technique to minimize the number of migrations by estimating future loads, where the proactive (pre-overload-detection) method can decrease the total number of migrations better than the reactive (post-overload-detection) method. Moreover, overloaded servers generate a large amount of heat and thus increase the cost of cooling systems and CO_2 emissions [4, 5]. Therefore, some VMs must be migrated to comply with end-user SLAs. However, to reduce energy consumption in CDCs, the number of servers hosting VMs needs to be limited.

(ii) Two VM selection algorithms [6] are tested using proactive overload detection algorithms and a robust local regression (LRR) algorithm, a traditional predictive algorithm that was used in a previous study [7], to assess their performance separately and jointly. These VM selection algorithms improved end-user QoS by avoiding frequent migration.

(iii) The medium-fit (MF) VM placement algorithm [8] is used to decrease the SLAVs, improve energy efficiency, and provide load balancing between active servers. The MF VM placement algorithm outperforms the traditional VM placement algorithm, i.e., the modified best-fit decreasing (MBFD) algorithm. The goal of the MBFD algorithm is to minimize the number of active servers. In addition, the MBFD algorithm is less energy efficient than the MF algorithm, and it increases SLAVs and thus increases the number of VM migrations.

Every algorithm is practically implemented and evaluated separately. Then, all RMP stages are integrated to study the effect of this integration to determine the best combination. The combination of exponential smoothing overload detection algorithm, MMTMC 2 VM selection, and MF placement algorithms is considered the best solution in terms of the studied algorithms.

The remainder of this paper is organized as follows: In "Related work" section, related research is reviewed. In "RMP design" section, the RMP design is explained. In "Experimental configuration and evaluation methodology" section, the experimental design is illustrated, including experimental configurations, workload input, and performance metrics. In "Prediction assessment study" section, the prediction assessment study is presented. In "Experimental results and discussions" section, the experimental results and discussions are provided. Finally, "Conclusion" section concludes the paper.

Related work

RM in cloud environments has been studied extensively. Various RM studies have proposed methods to evaluate and improve energy consumption, SLA, network load, and load balancing in CDCs [4, 9–11].

Live migration is considered an essential component of any RM framework for CDCs. The VM migration problem can be divided into three sub-problems: (i) when to migrate, (ii) which VM to migrate, and (iii) where to migrate. In general, live migration methods can be categorized as reactive and proactive. However, there is another taxonomy in which live migration methods can be categorized as periodical VM placement (no overload or underload detection) [12, 13], threshold-based heuristics [14–20], and decisions based on statistical analysis of historical data [4, 21–23]. A taxonomy of VM migration methods used in RM approaches for CDCs is shown in Fig. 1. As can be seen, the reactive method is represented in threshold-based heuristics, and the proactive method is represented in decisions based on a statistical analysis of historical data.

First, an early attempt to improve the energy efficiency of a CDC was by periodic workload reallocation strategy [12, 13]. The authors did not provide an algorithm to determine the suitable time to optimize the VM placement, but the proposed method was periodically performed. In our study, we did not use this strategy because it causes a considerable number of migrations, and hence it will cause SLAVs for end-users.

Then, a fixed threshold-based heuristics strategy [14–20, 24] is considered an alternative solution by monitors the current behavior through occurring migration from underloaded or overloaded servers at a certain threshold value. But, the fixed utilization threshold strategy is based on a current load, and the variations in resource utilization cannot be supported. Therefore, Beloglazov et al. [23] found that keeping this threshold constant is not an intelligent solution as the workload utilization is in continuous change. They proposed an inter-quartile range (IQR) and median absolute deviation (MAD) algorithms to determine the utilization threshold value of a server dynamically. Servers are considered to

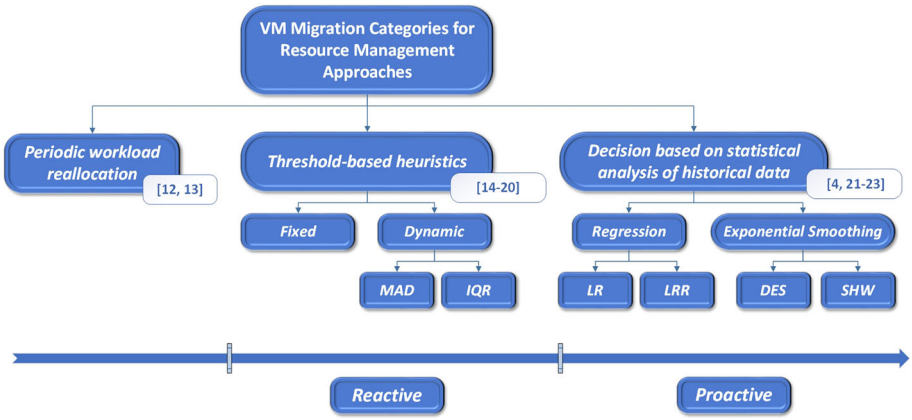

Fig. 1 Taxonomy of VM migration methods used in RM approaches for CDCs. The arrow at the bottom represents the direction of research depending on the pro-activity classification and represents the increase in complexity and performance improvement as well. (citing to previous efforts)

be overloaded if their current utilization is larger than this threshold. According to the dynamic cloud environment, the adaptive threshold is more efficient than the fixed threshold. Therefore, [18–20] proposed adaptive multi thresholds models. Li et al. [18] proposed a unique dynamic energy effective algorithm for VM consolidation based on multiple resource energy effective models. This algorithm aimed to reduce energy consumption with the QoS guarantee. They implemented a double threshold technique with multiple resource use in triggering the VM migration. The modified swarm optimization strategy (MPSO) avoids falling into the local optima that is a common problem in traditional heuristic algorithms. The results showed that the proposed algorithm minimized energy consumption and the number of VM migrations compared to the MBFD algorithm. Zhou et al. proposed a three threshold algorithm for VM consolidation based on triple and fixed thresholds in [25], in addition to triple and dynamic thresholds in [19]. Khattar et al. [20] proposed four adaptive thresholds framework for VM consolidation that consider energy consumption, SLA violations, memory, and bandwidth consumption. The results showed that four thresholds save more number of servers as compared to three thresholds.

Recently, most researches use decisions based on a statistical analysis of historical data strategy, because the migration occurs depending on the current and predicted utilization. Consequently, these techniques prevent unnecessary migrations that are initiated because of a massive spike in workload. Beloglazov and Buyya [23] proposed methods to predict future workloads, such as local regression (LR) and robust local regression (LRR) algorithms. These techniques predict CPU utilization for a server, then compared it to a dynamic threshold to detect overloaded servers.

Shaw and Singh [4] proposed simple and double exponential smoothing (DES) prediction algorithms to minimize the number of VM migrations. This algorithm determines whether migration must be performed and finds a suitable destination server for the selected VM, depending on the future load. The results showed that the proposed algorithm significantly minimized the number of migrations and energy consumption in addition to preserving the SLA. The results indicated that the DES technique performed the best. Energy consumption and SLAVs were decreased by 34.59% and 63.92%, respectively. Therefore, this study compares between DES and seasonal exponential smoothing algorithms where CDCs workloads may have seasonal patterns, such as intraday, intra-week, intra-month, intra-quarter, and intra-year.

Most research on VM placement algorithms has used the best-fit decreasing (BFD) algorithm or a modified version of this algorithm [23] because these algorithms achieve high power savings. Recent research has used the MF algorithm to decrease SLAVs and the number of migrations for a homogeneous CDC environment [8], whereas the MF power-efficient decreasing (MFPED) algorithm is suitable for a heterogeneous data-center environment. This paper study the effect of the MF placement algorithm compared to the BFD placement algorithm using several overload detection and VM selection algorithms.

Table 1 summarizes publications on previous RM approaches for CDCs. These publications are classified according to their architecture, target, migration methods, performance metrics, and main conclusions. Table 2 illustrates the algorithms used for handling live migration problems for RM approaches. An analysis of this study concludes that the best RM approach

Table 1 Comparison of data centre RM approaches

Authors	Architecture	Target	When to migrate	Performance Metric	Conclusion
Nathuji and Schwan [12]	Centralized	minimize power consumption without performance penalties	- Periodic reallocation	- Power (*W*)	The overall energy consumption could be reduced significantly, up to 34%, without appreciable losses in performance.
Verma et al. [13]	Centralized	minimize power consumption, considering the VM migration cost	- Periodic reallocation	- Power (watts) - Energy (kilojoules) - Migration cost - Overall cost - Power savings (%)	pMapper was considered an efficient solution to minimize power consumption (less than 0.2% penalty).
Zhu et al. [15]	De-centralized - Pod controllers - Node controllers	improve workload management to ensure efficient use of data center resources	- Periodic reallocation - Threshold-based heuristics	- Response time (seconds) - Number of migrations	The integration of node and pod controllers improved performance by 32% and 23% over fixed allocation and over non-integrated controllers, and reduced migrations for high priority workloads.
Gmach et al. [16]	De-centralized - workload placement controller - migration controller	minimize power consumption, taking into account the Qos and the number of VM migrations	- Periodic reallocation - Threshold-based heuristics	- Migration overhead - CPU quality violations - Power consumption	The integration between reactive migration controller and periodic workload placement controller presented the best approach for power and SLA, but needs more migrations.
VMware Distributed Power Management (DPM) [17]	Centralized	minimize power consumption	- Fixed threshold heuristics		Fixed threshold heuristics are unsuitable for real systems with dynamic and unknown workloads.
Li et al. [18]	Centralized	minimize power consumption and QoS guarantee	- Dynamic threshold heuristics	- Energy - Number of migrations - Number of active physical servers - load balance degree	The double threshold with multi-resource utilization with the MPSO algorithm reduces energy consumption and improves the QoS.
Beloglazov et al. [24]	Centralized	minimize power consumption, taking into account QoS	- Fixed threshold heuristics	- Energy (*kWh*) - SLAVs (%) - Number of migrations	It is not a suitable decision for keeping the utilization threshold constant as the workload is in continuous change.
Beloglazov and Buyya [23]	De-centralized	minimize power consumption, taking into account QoS	- Decision based on statistical analysis of historical data	- Energy (*kWh*) - ESV - SLAVs - PDM (%) - Number of migrations	The proposed LR algorithm remarkably outperformed other dynamic VM consolidation algorithms.
Guenter et al. [21]	De-centralized	minimize power consumption, considering the trade-off between cost, performance, and reliability	- Decision based on statistical analysis of historical data	- Power saving - normalized daily energy savings (MWh)	Predicting demand was used to switch on servers before require and avoid switching on unnecessary servers.

Table 1 Comparison of data centre RM approaches

Authors	Architecture	Target	When to migrate	Performance Metric	Conclusion
Bobroff et al. [22]	Centralized	minimize power consumption	- Decision based on statistical analysis of historical data	- Time-averaged number of servers used - Capacity of overflow	The proposed algorithm used decreased the number of servers needed to support a certain SLA by 50% compared to static consolidation.
This work	De-centralized	minimize power consumption, respect overall and end-user's SLA, and eliminate unnecessary migrations	- Fixed threshold heuristics - Decision based on statistical analysis of historical data	- AITF (%) - AOTF (%) - Number of migrations - Energy saving (%)	Our combination of DES, MMTMC 2, and MF algorithms improved performance in power saving, QoS, and network traffic. The number of migrations reduced by 49.44% compared to default algorithms.

depends on the trade-off between energy efficiency, QoS, and network traffic. For instance, the threshold-based heuristics are the least complex algorithm. It can be tuned to be more power-saving and more QoS degrading using its parameter value. A decision based on a statistical analysis of historical data is more complex; by contrast, it is considered as a proactive method.

In this study, an efficient integrated RMP approach is proposed to obtain a resource-effective CDC framework. The main differences from the previous efforts are as follows:

(i) Several overload detection, VM selection, and VM placement algorithms are implemented and evaluated separately and integrated experimentally to determine the best combination in terms of power saving and SLA for end-users and overall system.

(ii) Partial load balancing between the active server is provided using the MF algorithm.

Table 2 Comparison of RM techniques

Authors	When to migrate (overload detection technique)	Which VM to migrate (VM selection technique)	Where to migrate (VM placement technique)
Nathuji and Schwan [12]	-	-	VirtualPower Management
Verma et al. [13]	-	-	PMaP Algorithm
Beloglazov et al. [24]	Fixed threshold	- Minimization of migration (MM) - Highest potential growth (HPG) - Random choice (RC)	Modified best fit decreasing (MBFD)
Beloglazov and Buyya [23]	- Inter-quartile range (IQR) - Median absolute deviation (MAD) - Local regression (LR) - Robust local regression (LRR)	- Minimum migration time (MMT) - Random selection (RS) - Maximum correlation (MC)	Power aware best fit decreasing (PABFD)
Guenter et al. [21]	- Short term load forecasting	-	-
Shaw and Singh [4]	- Simple Exponential Smoothing - Double exponential smoothing	Minimum utilization policy	Optimized destination host selection
Moges and Abebe [8]	local regression	-	- MF (for homogeneous) - MFPED (for heterogeneous)
This Work	- Fixed threshold - Robust local regression (LRR) - Double exponential smoothing (DES) - Seasonal holt winter (SHW)	- MMTMC - MMTMC 1 - MMTMC 2	- Modified best fit decreasing (MBFD) - Medium-fit (MF)

(iii) A real testbed is performed to evaluate all studied algorithms experimentally. In contrast, most presented studies simulated their algorithms using CloudSim.

RMP design

In this section, the proposed RMP approach realize a satisfactory dynamic resource-effective CDC framework is introduced. In addition, the RMP algorithms are described in detail.

RMP stages

Recently, the RM problem was split into four sub-issues [7]:

(i) Underload detection detects underutilized servers. All VMs on an underutilized server should be offloaded, and the server should be switched to sleep mode.
(ii) Overload detection detects whether a server is overutilized. If a server is overutilized, some VMs should be migrated to another active or reactivated server to avoid SLAVs.
(iii) VM selection selects VMs to be migrated when an overloaded server is detected.
(iv) VM placement finds suitable servers for the selected VMs.

Figure 2 shows the RMP approach to achieve a resource-effective CDC framework. First, underload and overload detection algorithms are executed periodically. If the server is underloaded, all VMs are offloaded from it, then switch it to sleep mode. If the server is overloaded, the selection algorithm is executed to choose VMs to migrate from it. Then, the VM placement algorithm is executed using the VMs list, depending on their resource utilization.

In this paper, we focus on overload detection, VM selection, and VM placement algorithms as essential stages of the RMP approach. We examine the best individual performance of each algorithm and then determine the best combination of the three algorithms to achieve superior performance in terms of power saving, SLA, and network traffic.

RMP algorithms

In this section, overload detection, VM selection, and placement algorithms are discussed. These algorithms represent critical phases in the dynamic integrated RMP framework, where the goal is to improve the overall performance of the RM CDC framework and respect end-users' SLA.

Overload detection algorithm

Various overload detection algorithms were presented in "Related work" section. The simplest algorithm is the averaging threshold-based (THR) algorithm, which detects an overloaded server when the average value of the last CPU utilization is more than a fixed value [14–20]. The LRR algorithm predicts CPU utilization and hence decreases SLAVs [23].

Many studies have used conventional DES, which performs better than simple exponential smoothing [26], to predict cloud workloads; however, conventional DES cannot model seasonality [27]. The Seasonal Holt-Winters

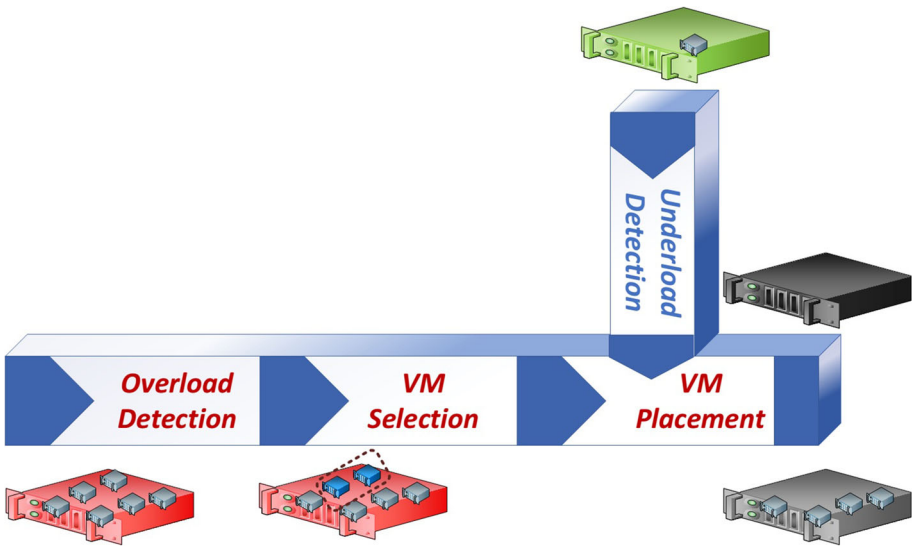

Fig. 2 RMP approach for resource-effective CDC framework. Red, green, grey, and black servers represent overloaded, underloaded, stable, and sleep-mode servers, respectively

(SHW) model can be used to forecast trends and seasonal workloads. SHW can also model workloads with a multi-seasonal pattern. CDC environments may have intra-day, intra-week, intra-month, intra-quarter, or intra-year seasonal patterns. Therefore, in this paper, SHW is evaluated to determine if it can accommodate intraday seasonal patterns. A comparative study is performed in the "Prediction assessment study" section using offline training used to investigate different distributions and select the suitable algorithm for a cloud computing environment, i.e., DES or SHW. Then, the THR and LRR algorithms are used as comparative algorithms to evaluate the suitable exponential smoothing algorithm as a part of the presented framework.

The exponential smoothing technique was implemented more than 60 years ago to analyze stock exchanges. This algorithm was developed to fit the seasonal pattern in time-series forecasting. The essential benefit of this algorithm is reduced memory requirements during training phase [28]. This scheme is based on the following equations, which measure the level, trend, and seasonality terms. Note, Eqs. (4) and (5) compute the weighted aggregation of the level, trend, and seasonality.

Level:

$$S_t = \alpha \left(X_t - I_{t-p}\right) + (1-\alpha).\left(S_{t-1} + T_{t-1}\right) \tag{1}$$

Trend:

$$T_t = \beta \left(S_t - S_{t-1}\right) + (1-\beta).T_{t-1} \tag{2}$$

Seasonality:

$$I_t = \gamma (X_t - S_t) + (1-\gamma)\, .I_{t-p} \tag{3}$$

Forecast:

$$\hat{Y}_{t,DES}(k) = S_t + k.T_t \tag{4}$$

$$\hat{Y}_{t,SHW}(k) = S_t + k.T_t + I_{t-p+k} \tag{5}$$

Where $X_1, X_2, ..., X_{t-1}$ denotes the CPU utilization history of the VM during time (t), N is the total number of samples, k is the index of the forecast samples (k = 1,..., K), and p is the seasonality length. The smoothing algorithm requires a training phase to adapt the level, trend, and seasonal smoothing factors (α, β, and γ).

Several methods can be used to obtain initial values for the level (S_t) and trend (T_t). Typically, S_1 is equal to X_1. Three valid equations for T_1 are as follows:

$$T_1 = S_2 - S_1 \tag{6}$$

$$T_1 = \left[(S_2 - S_1) + (S_3 - S_2) + (S_4 - S_3)\right]/3 \tag{7}$$

$$T_1 = (S_n - S_1)/(n-1) \tag{8}$$

By calculating the root mean square error (RMSE) using:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i)^2} \tag{9}$$

The optimal parameter values (α, β, and γ) can be obtained. Then, these values are used in the testing phase to predict the future CPU utilization of the VM.

A detailed description of DES and SHW implementation, tuning, and optimization is beyond the scope of this study. However, details can be found in the literature [4, 28]. In this paper, these algorithms were performed to predict the CPU utilization of a server. If the current and future values (sum of the predicted utilization values of its VMs) are overloaded, then migration will take place. Therefore, this method eliminates unnecessary VM migrations, where each migration is correlated with QoS degradation, which increases SLAVs.

VM selection algorithm

The VM selection algorithm determines suitable VMs to migrate from an overloaded server. Beloglazov et al.'s traditional selection algorithm is minimum migration time maximum CPU utilization (MMTMC) [7]. This algorithm first selects VMs with minimum migration time, and then selects the VM with the top CPU utilization to reduce overall server utilization as much as possible.

The main disadvantage of the original MMTMC algorithm is that it does not consider the SLA of the end-user when some VMs suffer from frequent migration, whereas others are slightly offloaded. Therefore, two VM selection algorithms have been proposed previously [6]. These algorithms are modified versions of the MMTMC algorithm [7]. The primary goal of these algorithms is to respect the end-user SLA, i.e., these algorithms consider all VMs with equal priority. Figure 3 illustrates the VM selection criteria of the MMTMC, MMTMC 1, and MMTMC 2 algorithms, where the added parts are shown in red.

The MMTMC 2 is superior because the threshold value can be controlled using the rate of growth to obtain the optimum solution [29].

As shown in Algorithm (2), it also nominates VMs with a lower amount of RAM to decrease the live migration time. Then, it chooses the VM with the maximum CPU utilization as long as the number of migrations is not exceeding a threshold value. This threshold value is incremented dynamically with time. It is calculated by the Eq. (10) that used in the neural network learning. Where $u_{thr}(t)$ is the current threshold value, $u_{initial}$ is the initial threshold value, λ is the rate of growth ($\lambda > 0$), and t is time. The start value ($u_{initial}$) and the rate of growth (λ) can be controlled to get the best solution [29]. For instance, the low growth rate decreases the number of migrations and hence reduces the network traffic; by contrast, the

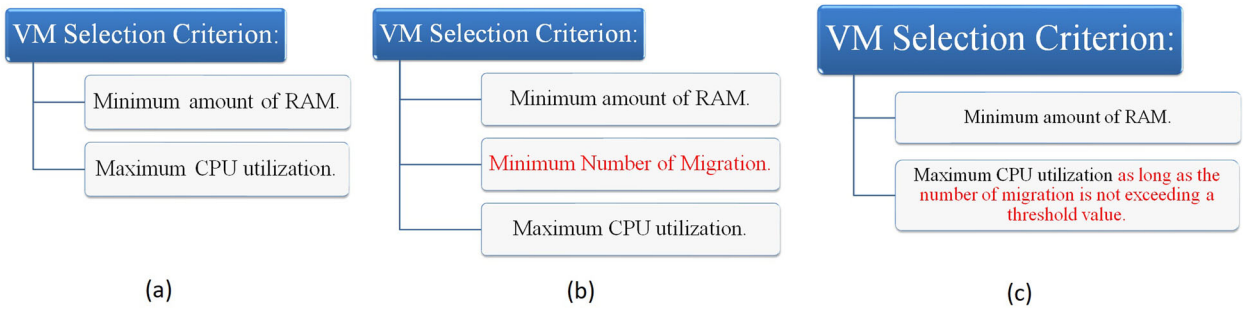

Fig. 3 Criteria of VM selection algorithms: **a** MMTMC, **b** MMTMC1, and **c** MMTMC2

low growth rate increases the overload time of the server. More details about these two algorithms are given in [6].

Algorithm 1: MMTMC 2 [6]

Input : *n*, *vms_cpu_map*, *vms_ram_map*, *vms_number_of_migration*, *current_threshold_number_of_migration*
Output: a VM to migrate

min_ram ← `min`(*values of vms_ram_map*)
max_cpu ← 0
selected_vm ← None
foreach *vm, cpu in vms_cpu_map* **do**
  if *vms_ram_map*[*vm*] > *min_ram* **then**
    continue
  if *vms_number_of_migration*[*vm*] > *current_threshold_number_of_migration* **then**
    continue
  vals ← *last n values of cpu*
  mean ← `sum`(*vals*) / `len`(*vals*)
  if *max_cpu* < *mean* **then**
    max_cpu ← *mean*
    min_number_of_migration ← *vms_number_of_migration*[*vm*]
    selected_vm ← *vm*
return *selected_vm*

$$u_{thr}(t) = \lceil u_{initial} \exp^{\lambda t} \rceil \tag{10}$$

The results demonstrated that a significant improvement in the end-user SLA was achieved using these two algorithms. In addition, there was a considerable improvement in energy consumption and network traffic using the MMTMC2 algorithm [6]. Therefore, in the current study, the performance of these two algorithms was evaluated compared to several overload detection and placement algorithms.

VM placement algorithm

The third issue for improving CDC performance is the VM placement algorithm, which selects a server for placing the selected VMs to be migrated. A traditional framework uses the MBFD placement algorithm [7], which attempts to minimize the number of active servers. This heuristic has low energy efficiency and increases SLAVs; thus, it increases the number of migrations.

Therefore, in this paper, the MF VM placement algorithm is used to reduce SLA violations and the number of migrations. It is based on bin-packing heuristics [8]. To evaluate the performance of this algorithm, we performed experiments using a real homogeneous testbed scenario. The MF bin-packing rule is defined as follows:

$$Allocated\_Server \equiv min_S |L_S - L_D| \tag{11}$$

where L_s is the CPU utilization level of server s. L_D is the desired CPU utilization level of a server given by

$$L_D \equiv (Thr_{overload} + Thr_{underload})/2 \tag{12}$$

where $Thr_{overload}$ and $Thr_{underload}$ are the overload and underload CPU utilization threshold values, respectively. In other words, the MF placement algorithm is based on the preference of a server whose resource level has a minimum distance from L_D. If $L_D = Thr_{overload}$, then Eq. (12) is similar to the best-fit algorithm. If $L_D = Thr_{underload}$, then Eq. (12) is equivalent to the worst-fit algorithm. Thus, Eq. (12) is considered to be the general form of the fit-decreasing placement algorithm. Therefore, we used this placement algorithm to improve QoS compared to the traditional MBFD placement algorithm.

Experimental configuration and evaluation methodology

In this section, the experimental setup, workload traces, and key performance indicators (KPIs) to evaluate the performance of the algorithms in the integrated RMP approach are presented.

Table 3 Testbed Specifications

Brand	Cores	Threads	CPU Type	RAM	RAM Type	Bandwidth
FUJITSU ESPRIMO P556/E85+	4	4	Intel® Core™ i5-6600 @ 3.3 GHz	4 GB	DDR4-133 MHz	10G Ethernet

Experimental setup

In this section, the experimental configurations, such as testbed specifications, algorithm parameters, and experimental sets used to evaluate the performance of the proposed RMP approach are illustrated. The testbed comprised five physical servers. Table 3 shows the specifications of the physical machines used in our experiments. One physical server was used as a controller server that ran all OpenStack core services and the OpenStack Neat global manager. The other four physical servers were used as compute servers running OpenStack Nova, as well as the data collectors and local managers of OpenStack Neat. Additional details about the testbed configuration can be found in the literature [6].

All setup parameters of the framework were set to their default values (with the exception of the setup parameters of the tested algorithm) according to each experiment. Table 4 details the parameter values used in all experiments. The parameters of the overload detection, VM selection, and VM placement algorithms were varied in each experiment according to Table 4 to facilitate a complete comparison of the algorithms. The LRR algorithm supports a safety parameter to control the sensitivity of the algorithm to potential overloads [7]. This parameter is equivalent to $Thr_{overload}$ in the THR overload detection algorithm, which detects an overloaded server if the mean of the last n CPU utilization values is greater than $Thr_{overload}$. Therefore, the results of the THR algorithm using $Thr_{overload}$ = 0.8, 0.9, and 1.0 are comparable to the results of LRR using a = 0.9, 1.0, and 1.1, respectively, in patterns and not values.

For the VM selection algorithm, the number of migration thresholds was calculated over 24 hours. The best result of the MMTMC 2 algorithm was achieved using λ equal to 0.001 and 0.0006 with (THR0.8, LRR1.1) and (THR0.9, LRR1.1); respectively, through trying and tuning. The performance of the MMTMC 2 algorithm was close to the performance of MMTMC 1 if λ was more than these values. Also, the servers will suffer from an overloaded workload if λ was less than these values due to the limited number of migrations.

Each experiment was tested three times to handle the inconsistencies caused by unpredictable influences of the

Table 4 Experimental parameters

RMP phase	Algorithm	Parameters	Definition	Values
Underload	THR	n	last CPU utilization values to be averaged	$n = 2$
detection		$THR_{underload}$	underload CPU utilization threshold	$THR_{overload} = 0.5$
Overload	THR	n	last CPU utilization values to be averaged	$n = 2$
detection		$THR_{overload}$	overload CPU utilization threshold	$THR_{overload} = 0.8, 0.9, 1.0$
	LRR	n	last CPU utilization values to be averaged	$n = 2$
		α	safety parameter	$\alpha = 0.9, 1.0, 1.1$
	DES	n	last CPU utilization values to be averaged	$n = 2$
		N	Input samples	$N = 30$
		K	Forecast sample	$K = 1$
	SHW	n	last CPU utilization values to be averaged	$n = 2$
		N	Input samples	$N = 30$
		K	Forecast sample	$K = 1$
VM	MMTMC	n	last CPU utilization values to be averaged	$n = 2$
selection	MMTMC 1	n	last CPU utilization values to be averaged	$n = 2$
	MMTMC 2	n	last CPU utilization values to be averaged	$n = 2$
		$u_{initial}$	The initial threshold value	$u_{initial} = 3$
		λ	The rate of growth	λ is an empirical value
VM	Modified best fit	$THR_{Placement}$	CPU threshold from total server's capacity	$THR_{Placement} = 80\%$
placement	Medium-fit (MF)	$THR_{Placement}$	CPU threshold from total server's capacity	$THR_{Placement} = 80\%$

CDC environment, including the initial VM placement, latency, and workload trace.

Four sets of experiments were conducted using the testbed. For the first three experimental sets, one algorithm was evaluated; thus, all other algorithms were set to their default configurations to determine the effect of each algorithm. For instance, the first experiment illustrated the influence of using the DES overload detection algorithm. Therefore, all other algorithms were set to their default configurations [7] (i.e., the MMTMC selection algorithm and MBFD placement algorithm). In addition, the fourth experimental set evaluated the integration of the three presented algorithms together and with the default algorithms. Table 5 shows the configurations for all experimental sets. The experimental sets are categorized as follows:

(i) ***Overload detection algorithm (Exp. 1):*** Experimental sets 1.a and 1.b illustrate the results of using THR and LRR algorithms as traditional overload detection algorithms to evaluate the influence of using the DES overload detection algorithm through experimental set 1.c. The DES aims to decrease the number of VM migrations based on future load. The results and discussion are presented in "Overload detection algorithm results" section.

(ii) ***VM selection algorithms (Exp. 2):*** Experimental set 2.a and 2.d illustrate the results of using the MMTMC selection algorithm as a traditional VM selection algorithm using THR and LRR overload detection algorithms. In addition, experimental sets 2.b, 2.c, 2.e, and 2.f illustrate the influence of using the MMTMC1 and MMTMC2 selection algorithms. These algorithms were proposed in [6], which evaluated the performance with other overload detection techniques, such as LRR. This algorithm aims to reduce frequent migration while respecting the customer's SLA. The results and discussion are presented in "VM selection algorithms results" section.

(iii) ***VM placement algorithm (Exp. 3):*** Experimental sets 3.a and 3.b illustrate the results of using the MBFD algorithm as a traditional placement algorithm using MMTMC and MMTMC 2 selection algorithms. In addition, experimental sets 3.c and 3.d illustrate the influence of using the MF placement algorithm. This algorithm aims to improve the

Table 5 Experimental plan / category

Experimental Set		Overload detection algorithms	VM selection algorithms	VM placement algorithms
Exp. 1	a	THR	MMTMC	MBFD
	b	LRR	MMTMC	MBFD
	c	DES	MMTMC	MBFD
Exp. 2	a	THR	MMTMC	MBFD
	b	THR	**MMTMC 1**	MBFD
	c	THR	**MMTMC 2**	MBFD
	d	LRR	MMTMC	MBFD
	e	LRR	**MMTMC 1**	MBFD
	f	LRR	**MMTMC 2**	MBFD
Exp. 3	a	THR	MMTMC	MBFD
	b	THR	**MMTMC 2**	MBFD
	c	THR	MMTMC	**MF**
	d	THR	**MMTMC 2**	**MF**
Exp. 4	a	THR	MMTMC	MBFD
	b	THR	**MMTMC 2**	MBFD
	c	**DES**	MMTMC	MBFD
	d	**DES**	**MMTMC 2**	MBFD
	e	THR	MMTMC	**MF**
	f	THR	**MMTMC 2**	**MF**
	g	**DES**	MMTMC	**MF**
	h	**DES**	**MMTMC 2**	**MF**

Note: The bold formatting indicates the presented algorithms as not traditional on the real RM CDC framework

overall data center SLA. The results and discussion are presented in "VM placement algorithm results" section.

(iv) ***Combination of the different algorithms (Exp. 4):*** This experimental set presents the results of using a combination of all the algorithms. This experimental set aims to determine the optimum combination that represents the best possible performance. The results and discussion are presented in "Combination results" section.

Workload input

We used workload traces that were proposed as part of the CoMon project, which is a monitoring infrastructure for PlanetLab [30]. These traces contained CPU utilization collected every five minutes from more than 500 places around the world [31].

A previous study [7] has used PlanetLab's workload tracers, where they filtered 33 traces over 24 hours. The selected traces were used to stress the system to be more overloaded by satisfying the following situations:(i) CPU utilization is greater than 80% at least 10% over the 24 hours, and (ii) CPU utilization is less than 20% at least 10% over the 24 hours. Therefore, we used these traces to simulate a practical workload.

Performance evaluation metrics

In this section, performance measurement metrics used to evaluate the proposed RMP approach are presented. For efficient evaluation, metrics were used to analyze the KPIs (overall energy consumption, QoS, and network traffic). The evaluation metrics are as follows:

(i) **Aggregated idle time fraction (AITF):** The AITF is the sum of idle time of all servers over the total time of all servers (24 hours), as shown in Eq. (13). An increase in AITF means saving more energy.

$$AITF = \frac{\sum_{s \epsilon S} t_i(s)}{\sum_{s \epsilon S} t_o(s)} \tag{13}$$

Where S is all servers, $t_i(s)$ is the idle time of the server s, and $t_o(s)$ is the overall time of the server s.

(ii) **Aggregated overload time fraction (AOTF):** The AOTF is the sum of overloaded time of all servers over the busy time of all servers (not in sleep mode), as shown in Eq. (14). A decrease in AOTF means that the overall system QoS is improved.

$$AOTF(u_{thr}) = \frac{\sum_{s \epsilon S} t_o(s, u_{thr})}{\sum_{s \epsilon S} t_b(s)} \tag{14}$$

Where $t_o(s, u_{thr})$ is the overload time of the server s calculated according to the overload threshold u_{thr} and $t_b(s)$ is the overall busy time of the host s.

(iii) **Total number of VM migrations:** This reflects the CDC network overhead.

(iv) **User-level performance degradation due to migration:** The User-level PDM is the performance degradation that occurred in a single VM during migration over the VM's total capacity [32].

(v) **Performance degradation due to migration (PDM):** The PDM is the mean value of User-level PDM for all VMs in the system, as shown in Eq. (15).

$$PDM = \frac{1}{M} \sum_{j=1}^{M} \frac{C_{dj}}{C_{rj}} \tag{15}$$

Where M is the number of VMs, C_{dj} is the estimate of the performance degradation of the VM j due to migrations, C_{rj} is the total CPU capacity required by the VM j during its lifetime.

(vi) **SLA Violations (SLAVs):** The SLAVs is the failure of providing QoS. There are two causes of SLAVs: the SLAVs due to overloading and migration. Therefore, it is calculated by multiplying AOTF with PDM [32], as shown in Eq. (16).

$$SLAVs = AOTF * PDM \tag{16}$$

(vii) **Energy and SLA violations (ESV):** The ESV measures energy consumption and the SLAVs value, where the resource management system aims to minimize both of them. Therefore, it is calculated by multiplying the SLAVs with the energy consumption [23], as shown in Eq. (17).

$$ESV = SLAVs * Energy \tag{17}$$

(viii) **Estimated energy saving percentage:** It is calculated using the AITF and AOTF performance metrics using an energy estimation model [7]. According to this model, the power consumptions are $450W$, $270W$, and $10.4W$ in the full-utilization, idle, and sleep states, respectively [33].

Prediction assessment study

In this section, we determine which exponential smoothing technique is suitable for cloud workloads. The prediction technique was performed on the CPU utilization of the VM rather than the server because users are most likely to use their VM in a specific period. Three different PlanetLab workloads were tested as samples.

Figure 4 illustrates the three workload samples used to evaluate the performance of the DES and SHW exponential smoothing techniques. These workload samples were selected because they represent the most common frequent user behaviors regarding the 33 filtered PlanetLab traces described in "Workload input" section.

First, a training phase was performed using 70% of the CPU utilization values over 24 hours. Equation (7) is used

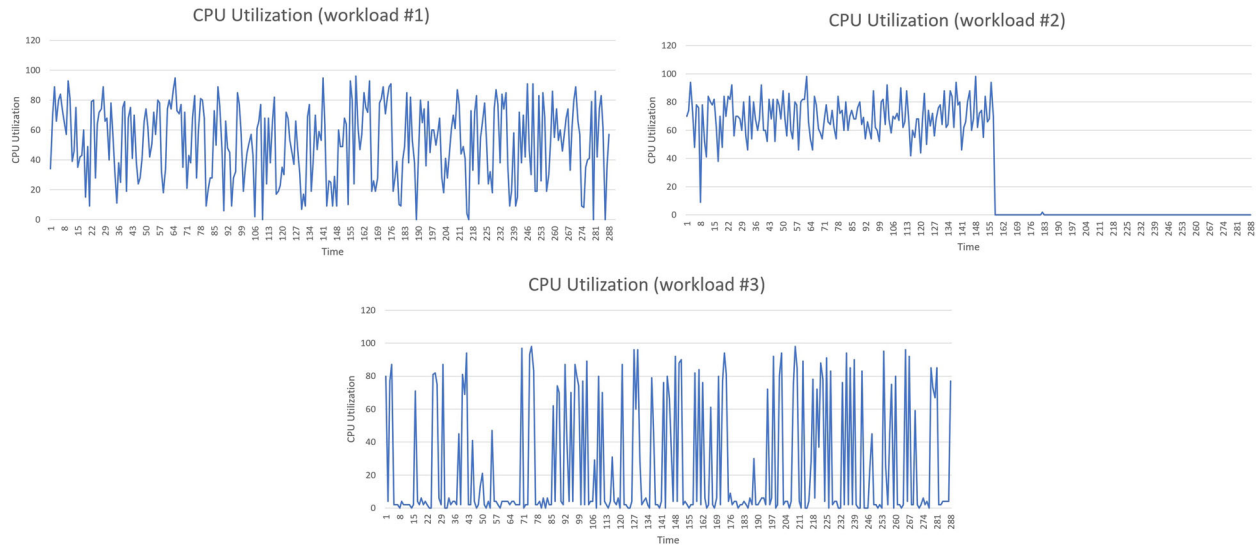

Fig. 4 Workload samples

in the initialization phase because it achieves the minimum RMSE in all tested workloads using the DES or SHW. Therefore, we used it in the training, testing, and operating phases. In the training phase, the prediction parameters (α and β for the DES algorithm and α, β, and γ for the SHW algorithm) were obtained, where these parameters produced the least RMSE.

Figure 5 illustrates the square error values between the actual and predicted CPU utilizations using the DES (blue line) and SHW (red line) algorithms for the tested workloads. As shown, the DES overload detection algorithm outperformed the SHW algorithm according to the square error values.

Table 6 illustrates the RMSE values calculated using Eq. (9) over 24 hours while discarding the first 30 values, where the prediction algorithm depends on historical data. Therefore, we initially performed the THR algorithm, and then switched to the prediction algorithm during the experimental phase. As can be seen, the RMSE value of the DES algorithm is lower for the three workload samples, which means that seasonality is not suitable for our case. This result might be because there is no seasonality during the last 30 historical sample data. Thus, the DES algorithm was used subsequently to predict CPU utilization in overload detection techniques as a significant phase in the RMP approach.

Experimental results and discussions

In this section, we present the experimental results and discussions about the RMP approach. "Overload detection algorithm results", "VM selection algorithms results", and "VM placement algorithm results" sections illustrate the experimental results of the proposed algorithms compared to traditional algorithms in each stage of the proposed RMP CDC framework. In addition, in the "Combination results" section, the results of the combined proposed RM algorithms are presented.

Overload detection algorithm results

The first RMP stage involves the overload detection algorithm, as explained in the "RMP stages" section. It illustrates the influence of using the DES algorithm compared to the THR and LRR algorithms. Figure 6 shows the AITF, AOTF, and the number of VM migrations obtained by these algorithms.

For the THR algorithm, the mean AITF increased significantly from 22.54% to 32.94% with increasing mean AOTF (20.69% to 27.6%) by changing the threshold value from 0.8 to 1.0, respectively. In addition, the mean number of VM migrations decreased from 239.33 to 24.33 for the 80% and 100% thresholds, respectively.

Similarly, with the LRR algorithm, the mean AITF increased from 25.55% to 34.2% with increasing mean AOTF (24.98% to 30.65%) and a reduction in the mean number of VM migrations (198.33 to 29.67) as a result of changing the value of the safety parameter from 1.1 to 0.9.

Varying the threshold value of the DES algorithm from 0.8 to 0.9 increased the mean AITF value from 26.18% to 30.2% with increasing mean AOTF (26.92% to 29.77%) and a decreasing in the mean number of VM migrations (149 to 81). Clearly, this represents an improvement in energy savings, where the mean AITF increased from 26.18% to 30.2% for the DES (0.8 and 0.9) threshold values, respectively, compared to 22.54% to 26.69% and 25.55% to 29.07% for THR (0.8 and 0.9) and LRR (1.1 and 1.0), respectively. In addition, network traffic was improved,

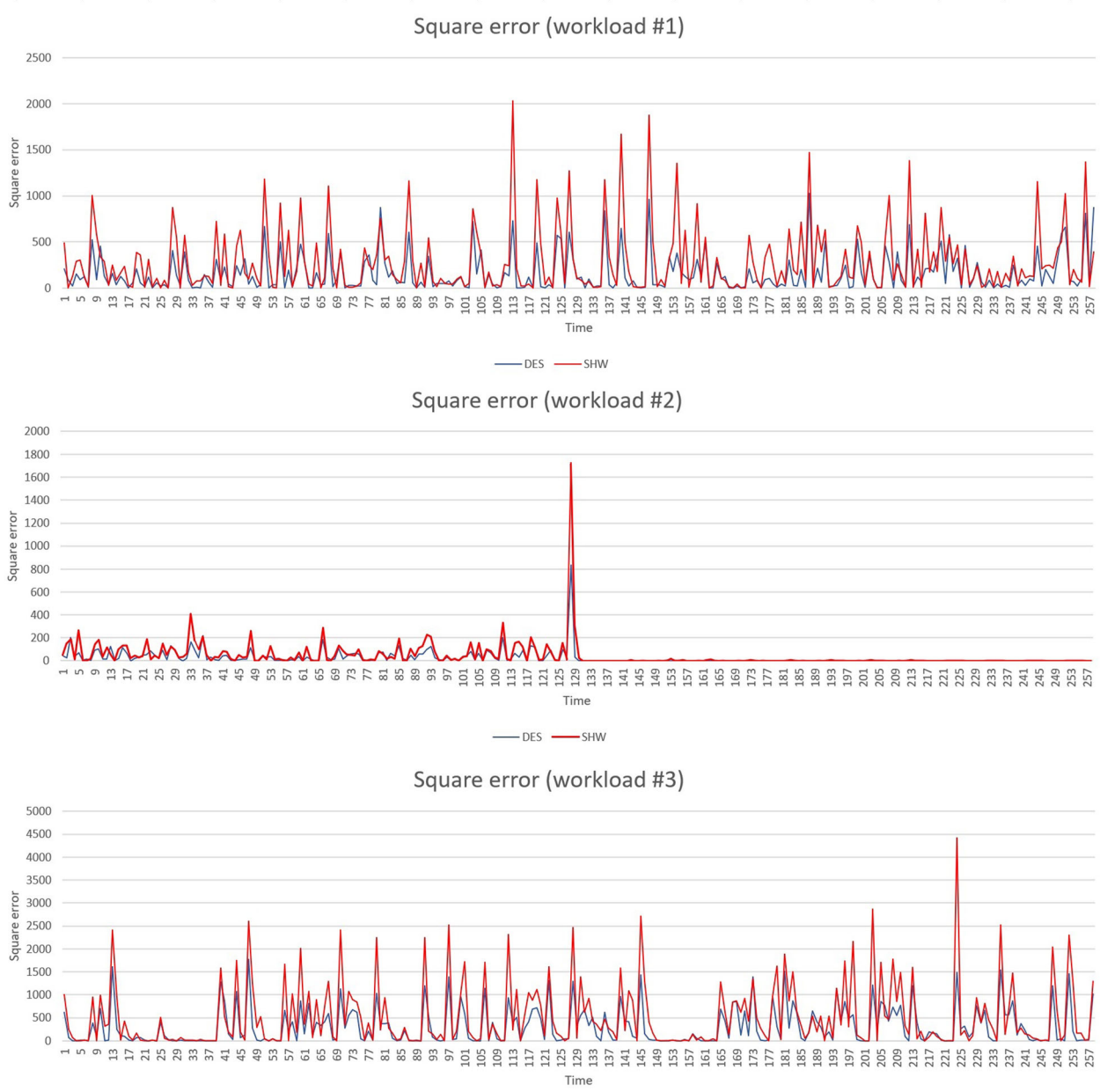

Fig. 5 Square error values between the actual and predicted CPU utilization for the three tested workloads (taking samples every 5 min)

where the mean number of VM migrations decreased compared to the two other algorithms. Thus, DES is an efficient algorithm to prevent unnecessary migrations.

Clearly, the DES algorithm achieved more energy savings than the other two algorithms using all overload detection algorithms. However, this improvement came at the expense of QoS degradation, where AOTF increased from 20.69% and 24.98% to 26.92% for THR 0.8, LRR 1.1, and DES 0.8, respectively, as shown in Fig. 6.

According to the results, THR 1.0, and LRR 0.9 achieved the highest estimated energy savings at the cost of substantial performance degradation (high AOTF). Therefore, these parameters not suitable for our objective in this study to obtain a compromise solution between power

Table 6 RMSE for DES and SHW algorithms

	DES	SHW
Workload #1	11.80	15.90
Workload #2	4.71	6.22
Workload #3	16.48	21.51

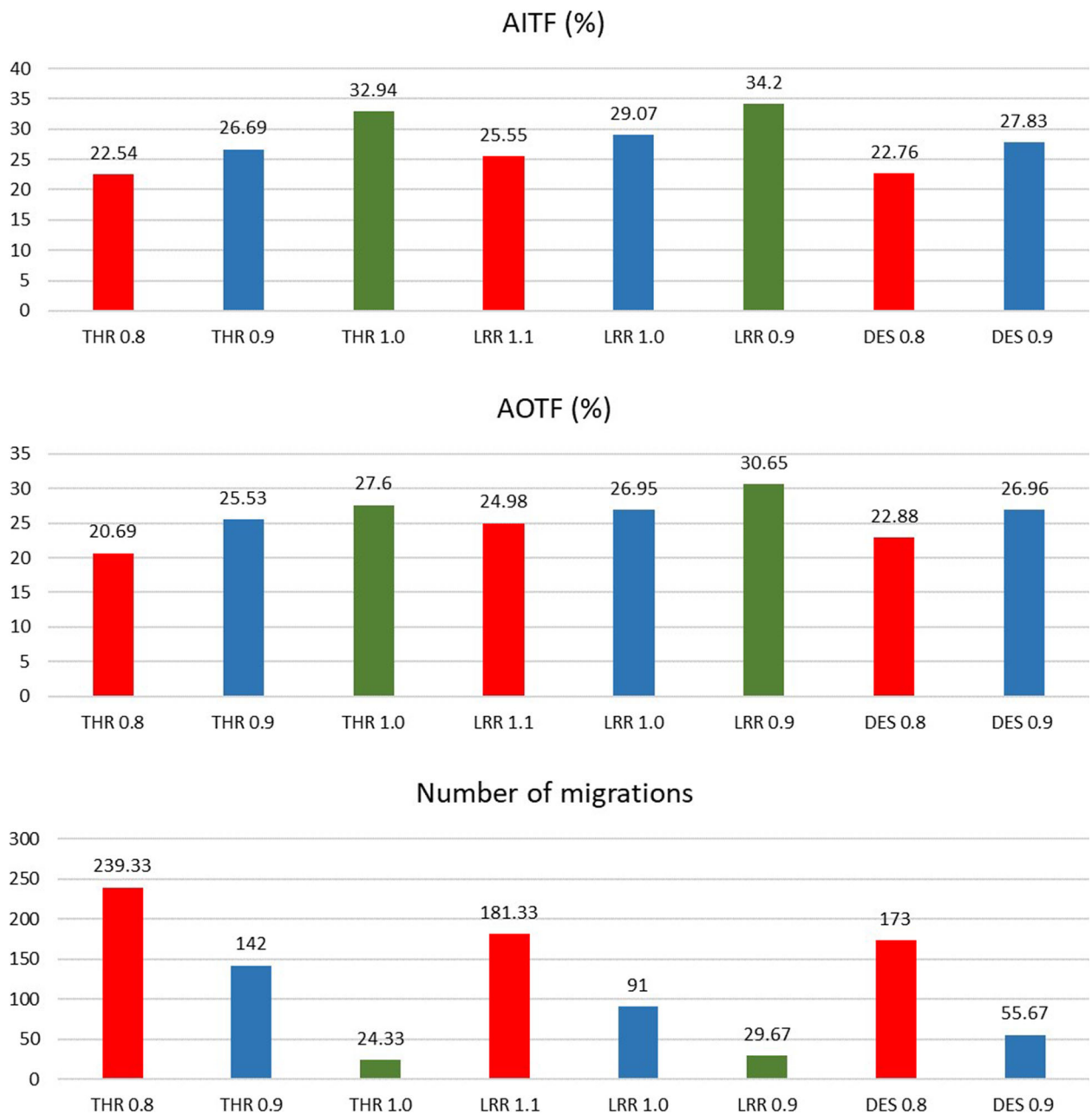

Fig. 6 Results of different overload detection algorithms: Mean values of AITF, AOTF, and the total number of VM migrations using THR, LRR, and DES overload detection algorithms with the MMTMC VM selection algorithm and MBFD algorithm. For instance, the green bar represents the parameter that substantially considerate power saving

saving, SLAs, and the number of migrations. As a result, DES 1.0 was not performed in this experiment set.

VM selection algorithms results

The second RMP stage involves the VM selection algorithm. The influence of using the MMTMC 1 and MMTMC 2 VM selection algorithms with traditional overload detection and VM placement techniques is determined. Figure 7 shows the AITF, AOTF, and the number of migrations for these algorithms compared to the MMTMC algorithm using the THR and LRR overload detection algorithms.

For the MMTMC 1 algorithm, the results demonstrate that overall performance was convergent to that of the MMTMC algorithm in terms of energy efficiency, overall SLA, and the total number of migrations. How-

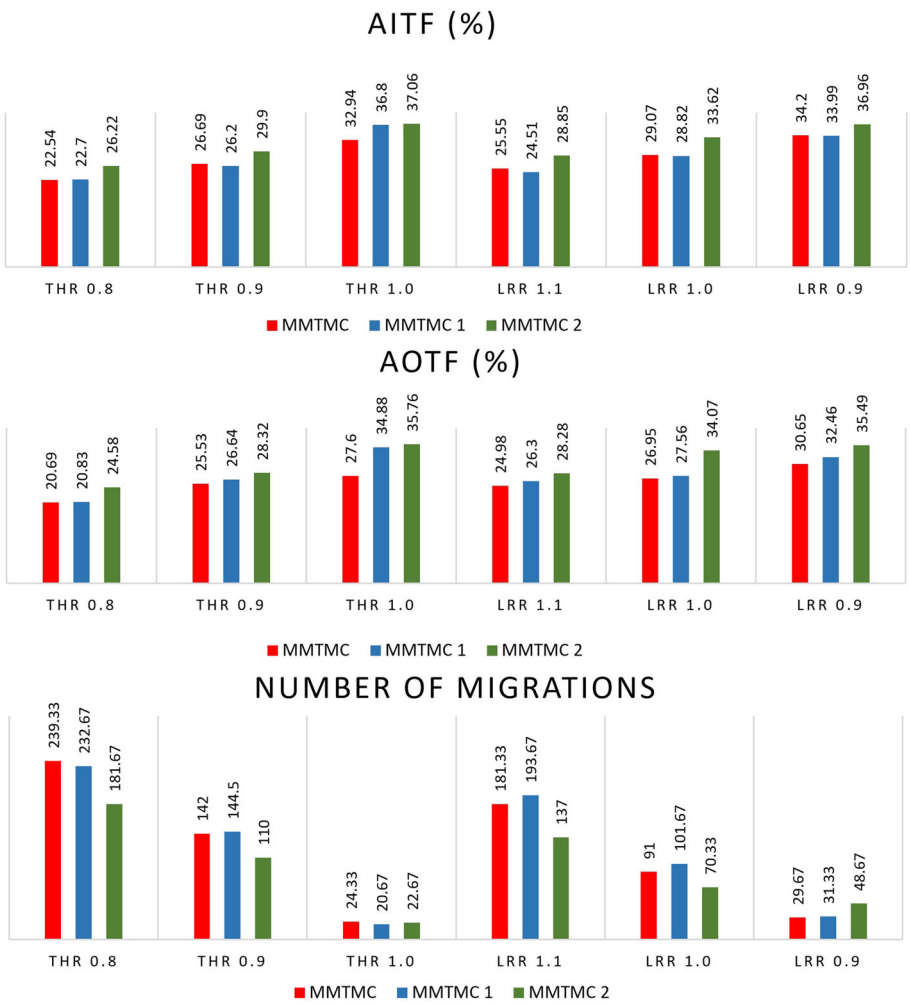

Fig. 7 Results of different VM selection algorithms: Mean values of AITF, AOTF, and the total number of VM migrations using different VM selection algorithms for the THR and LRR overload detection algorithms

ever, with the MMTMC 2 algorithm, there was only a slight improvement in energy savings and a slight drop in SLA. In contrast, there was a significant decrease in network overhead, where the mean number of VM migrations was reduced. For instance, with THR 0.8, the mean AITF increased from 22.54% to 26.22%, the mean AOTF increased from 20.59% to 24.58%, and the mean number of VM migrations decreased from 239.33 for the MMTMC algorithm to 181.67 with the MMTMC 2 algorithm.

The influence of the MMTMC 1 and MMTMC 2 algorithms was unclear with THR 1.0 and LRR 0.9 because overload detection occurred rarely, and most VM migrations occurred due to underloaded servers. Therefore, the contribution of these algorithms improved the framework in the presence of overloaded servers. THR 1.0 and LRR 0.9 cared slightly about the SLA of end-users while saving more power than using other overload detection configurations. The AITF increased to 37.06% and 36.96% using THR 1.0 and LRR 0.9, respectively, which means that the compute servers were in a low power state for approximately 37% of the overall active time of all compute servers. This power-saving percentage is considerable, especially as the testbed was designed to be overloaded.

Figure 8 illustrates the user-level performance degradation due to migration using the MMTMC, MMTMC 1, and MMTMC 2 algorithms with the THR 0.8 overload detection algorithm. As can be seen, remarkable diversity was observed in user-level PDM among VMs with the MMTMC algorithm. This diversity occurred due to the variation in the number of migrations per VMs. For instance, the fourth VM was migrated 31 times, while the 17^{th} VM was migrated only three times. Thus, the user-level PDM of these VMs was 0.2466% and 0.0218%, respectively. In addition, convergence was observed between the user-level PDM with the MMTMC 1 and MMTMC 2 algorithms.

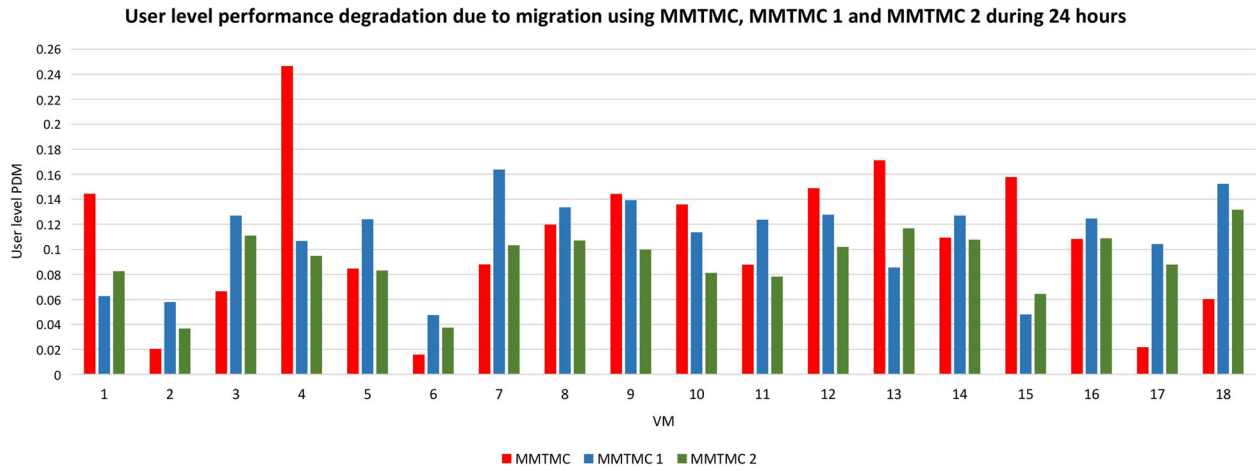

Fig. 8 User-level performance degradation due to migration (PDM) results: User-level PDM values using the MMTMC, MMTMC 1, and MMTMC 2 algorithms for THR 0.8 overload detection algorithm over 24 hours

Most of the user-level PDM and Number of migrations values were closed to each other and hence close to the mean value except the 1^{st}, 2^{nd}, 6^{th} and 15^{th} VMs using the MMTMC 1 algorithm. It is because this algorithm discards the zero CPU utilization from the selected list; as described in [6], due to the useless migration. In addition, the user-level PDM has decreased using the MMTMC 2 algorithm compared to the MMTMC 1. Even though this decrease is slight, but it will improve the SLA of the end-user and will decrease the overhead network traffic in terms of the system.

Table 7 illustrates the mean results of the MMTMC, MMTMC 1, and MMTMC 2 VM selection algorithms using the THR 0.8 overload detection algorithm over a 24 hour period. Despite the fact that the overall PDM for the three algorithms was convergent, all users were provided with convergent QoS using the MMTMC 1 and MMTMC 2, as shown in Fig. 8. The ESV value was reduced using the MMTMC2 compared to the other algorithms. The goal of the RM system is to reduce energy and SLAVs significantly; therefore, the ESV metric is suitable to evaluate both energy consumption and QoS. This means that the MMTMC2 outperformed other algorithms in terms of the SLA of the end-users and the system.

VM placement algorithm results

The third RMP stage involves the VM placement algorithm. The influence of using the MF VM placement algorithm is explained and compared to the MBFD algorithm. Figure 9 illustrates the obtained AITF, AOTF, and number of migrations for the MBFD and MF placement algorithms.

Using the MF placement algorithm improved performance in terms of both power savings and network traffic, where the mean AITF increased from 22.54% using the MBFD algorithm to 23.77% using the MF algorithm for the MMTMC selection algorithm. In addition, the mean number of VM migrations was reduced from 239.33 to 177.33. In contrast, there was a slight decrease in QoS, where the mean AOTF increased from 20.69% to 21.55%. This result was expected because the MF placement algorithm achieves partial load balancing between the active server, where the selected VMs allocated in the servers have average utilization. This technique allows an underloaded server to switch to sleep mode. In addition, this technique prevents a nearly-overloaded server from becoming more overloaded, and then the system migrates VMs from these servers. In contrast, the MBFD algorithm is based on allocating the selected VMs to the server with the highest CPU utilization if resources are available.

Table 7 Mean experimental results of the presented VM selection algorithms using THR 0.8 overload detection algorithm over 24 hours

Algorithms	PDM (%)	AOTF (%)	SLAVs ($\times 10^{-5}$)	Energy consumption (KWh)	ESV ($\times 10^{-3}$)	No. of migrations
MMTMC	0.1072	20.69	22.19	31.73	7.0404	239.33
MMTMC 1	0.1093	20.83	22.77	31.57	7.1884	232.67
MMTMC 2	**0.0907**	24.58	22.31	30.43	**6.7880**	**181.67**

The bold formatting highlights the improved metrics

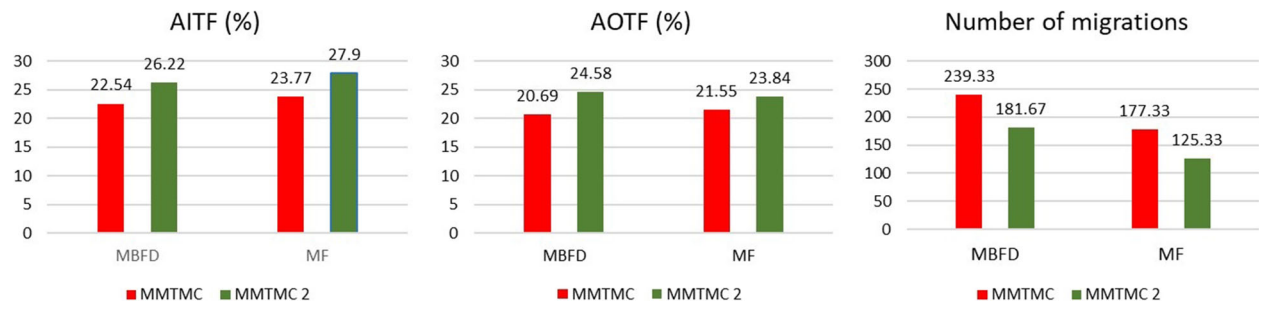

Fig. 9 Results of different VM placement algorithms: Mean values of AITF, AOTF, and the total number of VM migrations using MF placement algorithms compared with MBFD algorithms using the THR 0.8 algorithm and the MMTMC and MMTMC 2 selection algorithms

In terms of the MMTMC 2 selection algorithm, the results are consistent with those in the previous section. This selection algorithm with any overload detection or placement algorithm improved performance in terms of power savings, network traffic, and the SLA of the end-users. The mean AITF increased from 26.22% using the MBFD algorithm to 27.9% using the MF algorithm for the MMTMC selection algorithm. In addition, the mean number of VM migrations decreased from 181.67 to 125.33. Contrary to expectations, there was a slight improvement in QoS, where the mean AOTF decreased from 24.58% to 23.84%. Therefore, the MF algorithm was considered the best placement algorithm compared to the MBFD algorithm. As a result, the MF algorithm was recommended to be the candidate algorithm to represent the third RMP stage to achieve the resource-effective CDC.

In the next section, we present the results of combining the proposed overload detection, VM selection, and VM placement algorithms as essential stages in the integrated RMP approach. We found that the best combination of these algorithms guarantees a resource-effective CDC framework.

Combination results

In this section, we discuss combining the DES algorithm with the MMTMC 2 VM selection and MF VM placement algorithms to achieve the best practical integrated pipeline solution in terms of the proposed algorithms. In addition, the results are compared to all combinations of algorithms. Figure 10 illustrates the obtained AITF, AOTF, and the number of VM migrations for the combination of the MBFD and MF placement algorithms, MMTMC and MMTMC 2 VM selection algorithms, and THR 0.8 and DES 0.8 overload detection algorithms.

First, the MMTMC 2 algorithm demonstrated the best performance in terms of the VM selection algorithm. In terms of power savings, DES 0.8 with the MBFD algorithms performed best, where the mean AITF was 28.76%. This result was expected because the MBFD placement algorithm is based on only reducing power as much as possible.

In terms of QoS, it makes sense that the DES algorithm increased SLA violations because it is based on the decision to migrate VMs if the actual and predicted CPU utilization values are overloaded. Therefore, the possibility that the server will stay in an overloaded state increased. From the results, it is clear that the mean AOTF increased from 24.58% to 29.1% as a result of changing overload detection from the THR to the DES algorithm. However, the MF placement algorithm contributed to solving this problem by reducing the mean AOTF values to 23.84% and 24.86% using the THR and DES overload detection algorithms, respectively.

In terms of network traffic, the DES algorithm avoids unnecessary migration, where the mean number of VM migrations decreased to 120.67 and 121 using the MBFD and MF placement algorithms, respectively.

However, integration between the DES 0.8 overload detection and MBFD placement algorithms achieved the best performance in terms of power savings and network traffic; however, this integration was considered the worst in terms of QoS. Therefore, the combination of the DES 0.8 overload detection, MMTMC 2 selection, and MF placement algorithms provided the best performance in terms of power savings, QoS, network traffic, and end-user's SLA. Thus, we consider that this combination is suitable for realizing a dynamic resource-effective CDC framework.

Table 8 illustrates the estimated energy saving percentage allowed by our framework using a combination of overload detection, VM selection, and VM placement algorithms. In terms of power savings, the best combination is the DES overload detection algorithm, MMTMC 2 selection algorithm, and MF placement algorithm. This combination achieved energy savings of 17.23%. However, as discussed previously, this combination was only best relative to energy savings and network traffic. The MF algorithm achieved slightly better energy saving performance than the traditional algorithm using the THR 0.8 overload detection algorithms, where energy savings increased from 13.38% and 14.21% to 16.03% and 17.08%

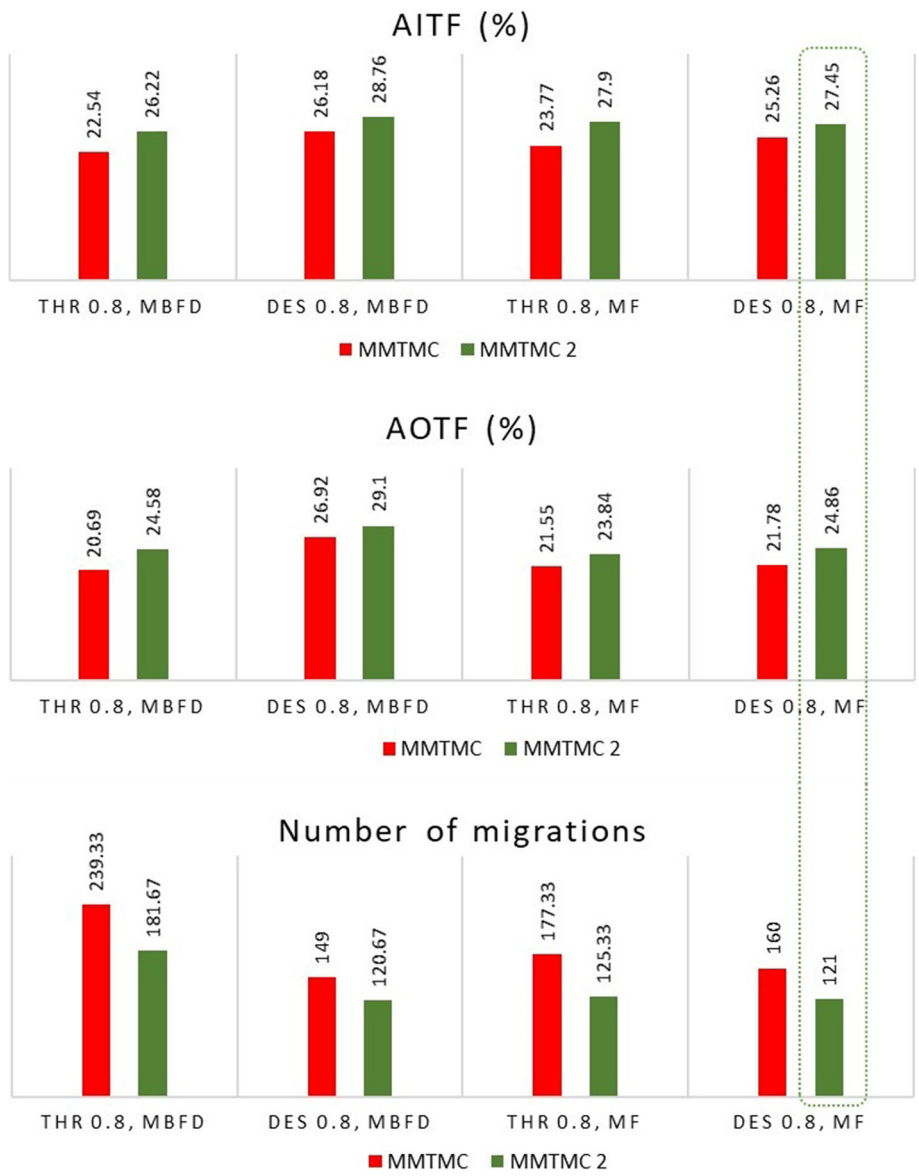

Fig. 10 Combination results: Mean values of AITF, AOTF, and the total number of VM migrations using combinations of overload detection, VM selection, and VM placement algorithms (The green loop represents the best performance)

for MMTMC and MMTMC 2, respectively. However, it is better to also respect QoS. In addition, our objective is to achieve dynamic resource-effective CDC; thus, the combination of DES and MMTMC 2 with MF comes the closest to satisfying this goal.

Conclusion

In this paper, an integrated RMP approach was proposed to achieve a dynamic resource-effective CDC framework. This framework provides practical solutions for CDC performance bottlenecks by splitting the RM issue into three

Table 8 Estimated energy saving percentage using combinations of overload detection, VM selection, and VM placement algorithms

Algorithm	MMTMC	MMTMC 2
THR 0.8, MBFD	13.38%	15.74%
DES 0.8, MBFD	15.49%	17.23%
THR 0.8, MF	14.21%	**17.08%**
DES 0.8, MF	15.29%	16.64%

The bold formatting indicates the best combination result in terms of energy saving (THR 0.8-MMTMC 2-MF)

stages, i.e., overload detection, VM selection, and VM placement, where the related algorithms directly affect a VM's SLA. The DES overload detection algorithm was used to eliminate useless migration by making migration decisions based on the predicted CPU utilization value. In addition, two algorithms were proposed to improve the SLA of the end-user in CDCs, and an MF placement algorithm was employed to improve the overall SLA and provide load balancing between active servers in CDCs. The experimental results have demonstrated that the combination of DES overload detection, MMTMC 2 VM selection, and MF VM placement algorithms positively impacted power savings, reduced network bottlenecks, and improved QoS, where the number of migrations was reduced by 49.44% compared to the default configuration.

In real-world CDCs, stable and efficient solution algorithms are required to ensure energy saving that maximizes the service provider's profit and providing QoS that satisfies end-user requirements. Although the proposed framework and algorithms can solve these issues, heterogeneous large-scale CDCs were not covered. Thus, in the future, scalability and diversity will be tested for this system using CloudSim, and seasonality will be studied over 10 days of workload traces. In addition, dynamic solution methods based on machine learning algorithms to predict workloads will be investigated and developed to form a multi-objective optimization approach. Finally, a different priority end-user with multi-levels of SLAs for CDCs will be investigated and evaluated.

Abbreviations

AITF: Aggregated idle time fraction; AOTF: Aggregated overload time fraction; BFD: Best-fit decreasing; CDCs: Cloud data centers; DES: Double exponential smoothing; ESV: Energy and SLA Violations; IQR: Inter-quartile range; KPIs: Key performance indicators; LR: Local regression; RMSE: Root mean square error; MAD: Median absolute deviation; MF: Medium-fit; MFPED: MF power efficient decreasing; MMTMC: Minimum migration time maximum CPU utilization; MBFD: Modified best-fit decreasing; MPSO: modified swarm optimization strategy; PDM: Performance degradation due to migration; PABFD: Power aware best fit decreasing; QoS: Quality of service; RM: Resource management; RMP: Resource management pipeline; LRR: Robust local regression; SHW: Seasonal holt winter; SLAs: Service level agreements; SLAVs: Service level agreement violations; VM: Virtual machine

Acknowledgements

Not Applicable.

Authors' contributions

The authors had worked equally during all this paper's stages. All author(s) read and approved the final manuscript.

Funding

This research work was supported by the Egyptian Academy of Scientific Research and Technology (ASRT) JESOR Grant. This project was entitled 'Resource-effective Cloud Data Center: Developing a dynamic data center management tool'.

Availability of data and materials

The data required to support these findings cannot be shared at this time as the data also forms part of an ongoing project.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Informatics Research Institute, City of Scientific Research and Technological Applications, SRTA-City, Alexandria, Egypt. [2]CECS Department, University of Louisville, Kentucky, USA.

Received: 15 June 2020 Accepted: 4 November 2020
Published online: 23 November 2020

References

1. Carrega A, Repetto M (2020) Coupling energy efficiency and quality of service for consolidation of cloud workloads. Comput Netw:107210. https://doi.org/10.1016/j.comnet.2020.107210
2. Li H, Zhu G, Zhao Y, Dai Y, Tian W (2017) Energy-efficient and qos-aware model based resource consolidation in cloud data centers. Clust Comput 20(3):2793–2803. https://doi.org/10.1007/s10586-017-0893-5
3. Mustafa S, Nazir B, Hayat A, Madani SA, et al (2015) Resource management in cloud computing: Taxonomy, prospects, and challenges. Comput Electr Eng 47:186–203. https://doi.org/10.1016/j.compeleceng.2015.07.021
4. Shaw SB, Singh AK (2015) Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migrations and energy consumption in cloud data center. Comput Electr Eng 47:241–254. https://doi.org/10.1016/j.compeleceng.2015.07.020
5. Beloglazov A, Buyya R (2010) Energy efficient allocation of virtual machines in cloud data centers. In: 10th IEEE/ACM International Conference On Cluster, Cloud and Grid Computing (CCGrid). IEEE. pp 577–578. https://doi.org/10.1109/CCGRID.2010.45
6. Maiyza AI, Hassan HA, Sheta WM, Sadek NM, Mokhtar MA (2017) End-user's sla-aware consolidation in cloud data centers. In: 2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT). IEEE. pp 196–204. https://doi.org/10.1109/ISSPIT.2017.8388641
7. Beloglazov A, Buyya R (2015) Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds. Concurr Comput Pract Experience 27(5):1310–1333. https://doi.org/10.1002/cpe.3314
8. Moges FF, Abebe SL (2019) Energy-aware vm placement algorithms for the openstack neat consolidation framework. J Cloud Comput 8(1):2. https://doi.org/10.1186/s13677-019-0126-y
9. Liu Z, Wang X (2012) A PSO-based algorithm for load balancing in virtual machines of cloud computing environment. In: International conference in swarm intelligence. Springer, Berlin. pp 142–147. https://doi.org/10.1007/978-3-642-30976-2_17
10. Domanal SG, Guddeti RMR, Buyya R (2017) A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment. In: IEEE transactions on services computing. IEEE. https://doi.org/10.1109/TSC.2017.2679738
11. Li H, Zhao Y, Fang S (2020) CSL-driven and energy-efficient resource scheduling in cloud data center. J Supercomput 76(1):481–498. https://doi.org/10.1007/s11227-019-03036-9. Springer
12. Nathuji R, Schwan K (2007) Virtualpower: coordinated power management in virtualized enterprise systems. ACM SIGOPS Oper Syst Rev 41(6):265–278. https://doi.org/10.1145/1294261.1294287. ACM
13. Verma A, Ahuja P, Neogi A (2008) pmapper: power and migration cost aware application placement in virtualized systems. In: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware. Springer-Verlag Inc, New York. pp 243–264. https://doi.org/10.1007/978-3-540-89856-613
14. Gmach D, Rolia J, Cherkasova L, Belrose G, Turicchi T, Kemper A (2008) An integrated approach to resource pool management: Policies, efficiency and quality metrics. In: IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN). IEEE. pp 326–335. https://doi.org/10.1109/DSN.2008.4630101
15. Zhu X, Young D, Watson BJ, Wang Z, Rolia J, Singhal S, McKee B, Hyser C, Gmach D, Gardner R, et al. (2008) 1000 islands: Integrated capacity and workload management for the next generation data center. In: 2008 International Conference On Autonomic Computing (ICAC'08). IEEE. pp 172–181. https://doi.org/10.1109/ICAC.2008.32
16. Gmach D, Rolia J, Cherkasova L, Kemper A (2009) Resource pool management: Reactive versus proactive or let's be friends. Comput Netw 53(17):2905–2922. https://doi.org/10.1016/j.comnet.2009.08.011
17. (2013) VMware Distributed Power Management: Concepts and Usage, Tech. Rep. https://www.vmware.com/techpapers/2008/vmware-distributed-power-management-concepts-and-1080.html. Accessed 5 July 2019

18. Li H, Zhu G, Cui C, Tang H, Dou Y, He C (2016) Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing. Computing 98(3):303–317. https://doi.org/10.1007/s00607-015-0467-4. Springer
19. Zhou Z, Hu Z, Li K (2016) Virtual machine placement algorithm for both energy-awareness and SLA violation reduction in cloud data centers. Sci Program 22(3):974–983. https://doi.org/10.1155/2016/5612039. Hindawi
20. Khattar N, Singh J, Sidhu J (2020) An energy efficient and adaptive threshold VM consolidation framework for cloud environment. Wireless Personal Communications. Springer. https://doi.org/10.1155/2016/5612039
21. Guenter B, Jain N, Williams C (2011) Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In: INFOCOM 2011 Proceedings. IEEE. pp 1332–1340. https://doi.org/10.1109/INFCOM.2011.5934917
22. Bobroff N, Kochut A, Beaty K (2007) Dynamic placement of virtual machines for managing sla violations. In: 10th IFIP/IEEE International Symposium On Integrated Network Management. IEEE. pp 119–128. https://doi.org/10.1109/INM.2007.374776
23. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr Comput Pract Experience 24(13):1397–1420. https://doi.org/10.1002/cpe.1867
24. Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Futur Gener Comput Syst 28(5):755–768. https://doi.org/10.1016/j.future.2011.04.017
25. Zhou Z, Hu Z, Song T, Yu J (2015) A novel virtual machine deployment algorithm with energy efficiency in cloud computing. J Cent South Univ 22(3):974–983. https://doi.org/10.1007/s11771-015-2608-5. Springer
26. Shaw SB, Kumar C, Singh AK (2017) Use of time-series based forecasting technique for balancing load and reducing consumption of energy in a cloud data center. In: 2017 International Conference on Intelligent Computing and Control (I2C2). IEEE. pp 1–6. https://doi.org/10.1109/I2C2.2017.8321782
27. Taylor JW (2010) Exponentially weighted methods for forecasting intraday time series with multiple seasonal cycles. Int J Forecast 26(4):627–646. https://doi.org/10.1016/j.ijforecast.2010.02.009
28. Gardner Jr, Everette S (2006) Exponential smoothing: The state of the art–part II. Int J Forecast 22(4):637–666. https://doi.org/10.1016/j.ijforecast.2006.03.005
29. Exponential growth. https://en.wikipedia.org/wiki/Exponential_growth. Accessed 12 Aug 2019
30. Park K, Pai VS (2006) Comon: a mostly-scalable monitoring system for planetlab. ACM SIGOPS Oper Syst Rev 40(1):65–74. https://doi.org/10.1145/1113361.1113374
31. (2012) The PlanetLab traces. http://github.com/beloglazov/planetlab-workload-traces. Accessed 12 Sept 2019
32. Yadav SBS, Kalra M (2020) Energy-aware vm migration in cloud computing. In: Proceedings of International Conference on IoT Inclusive Life (ICIIL) 2019. Springer, NITTTR Chandigarh, India. pp 353–364. https://doi.org/10.1007/978-981-15-3020-3_32
33. Meisner D, Gold, BT, Wenisch TF (2009) Powernap: eliminating server idle power. In: ACM Sigplan Notices, vol. 37, no. 1. ACM. pp 205–216. https://doi.org/10.1145/1508244.1508269

Publisher's Note