Automated Software Engineering (2022) 29:34
https://doi.org/10.1007/s10515-022-00332-2

A Novel Technique for Accelerating Live Migration in Cloud Computing

Ambika Gupta[1,2] · **Suyel Namasudra**[2]

Received: 1 September 2021 / Accepted: 15 February 2022 / Published online: 23 March 2022

Abstract

Currently, cloud computing is being used in many scientific areas like geoscience, DNA sequencing, healthcare, and many more. In a cloud computing environment, a Virtual Machine (VM) is a virtualized instance of any computer that can execute almost all the tasks of a computer. VM migration can be referred to as a task to move VMs from one physical machine to another physical machine. During VM migration, there are many issues, such as fault occurrence, seamless connectivity, and maintaining the quality of service. The cloud service provider has to anticipate the server downtime and various other delays like slow processing of user's request due to the occurrence of a fault, improper allocation of VMs, and many more. A reliable and advanced live migration optimization technique has been proposed in this work for a trustworthy cloud computing environment. There are three main algorithms in the proposed scheme considering the total migration time, namely Host Selection Migration Time (HSMT), VM Reallocation Migration Time (VMRMT), and VM Reallocation Bandwidth Usage (VMRBU). These algorithms support to enhance the performance of cloud computing environments by minimizing the migration time. The proposed scheme has been compared to some existing approaches, namely Kernel-based Virtual Machines (KVM) and Pareto Optimized Framework for Seamless VM Live Migration (POF-SVLM), to evaluate its performance. The results show that the proposed scheme reduces the total cores of CPU by 60-70%, downtime by 70-80%, data transfer rate by 40-50%, and migration time by 40-50%.

Keywords Virtual machine · Downtime · Data transfer · Migration time · Performance

Extended author information available on the last page of the article

1 Introduction

Cloud computing is one of the most useful technologies due to its advanced features like pay-per-use, scalability, flexibility, resiliency, and many more (Armbrust et al. 2010). It is very emerging, and nowadays, almost all IT companies are using cloud computing environments. There are mainly three service delivery models in cloud computing: (1) Infrastructure as a Service (IaaS), (2) Platform as a Service (PaaS), and (3) Software as a Service (SaaS). IaaS is the backbone of any cloud environment as it provides the entire infrastructure, PaaS provides a platform for developing and running applications, and SaaS allows cloud users to access a set of applications or software from cloud environments.

A cloud environment consists of three entities: (1) users, (2) data owner, and (3) Cloud Service Provider (CSP). Here, virtualization plays an important role as it provides the cloud clients or users with virtual storage and software. Various other computing services like database, networking, storage, and analytics are also available over the internet through virtualization (Sangpetch et al. 2017). Sometimes, virtualization technology also called a virtual machine monitor, requires a certain number of resources to work effectively. The execution of VMs is occurred using a software tool called hypervisor as depicted in Fig. 1. It creates a bridge between hardware and various guest Operating Systems (OSs) that effectively handle shared memory control (Zhang et al. 2018; Obasuyi and Sari 2015). There are major benefits of virtualization, such as flexible operations, cost efficiency, adaptability in transferring data, and reducing the risk of system failure (Bala and Chana 2012). To provide a

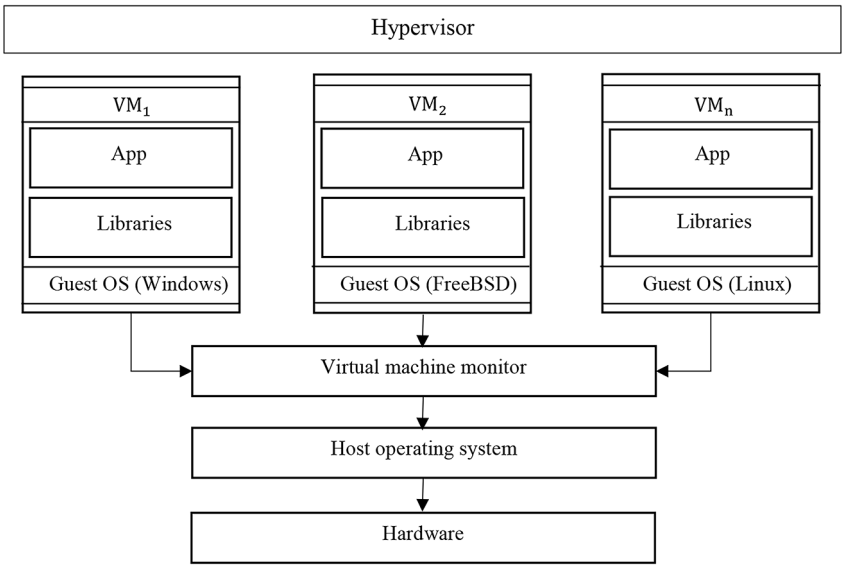

Fig. 1 Multiple guest OSs on the hypervisor

service to the cloud users on request, generally, VMs are migrated from one physical node to another, which is known as live migration (Malleswari and Vadivu 2019; Choudhary et al. 2017). As mentioned earlier, there are many issues in live migration like the performance of applications, fault occurrence during inter VM migration, seamless connectivity in network link, as well as security concerns. These challenges proactively occur in the systems, where cloud computing environments are used. As a result, some steps must be taken to determine how migration may be performed or what alternative options are available to address these issues. These problems of live migration must be resolved to increase the system performance and better functioning among many VMs (Motaki et al. 2021).

For resolving the problems of live migration, there are many existing techniques (Wang et al. 2021; Garrido et al. 2021; Namasudra 2020; Yuan et al. 2020; Gao et al. 2020, 2021; Sharma et al. 2021; Gómez et al. 2020; Hamdy et al. 2020; Namasudra et al. 2020; Suruliandi et al. 2021; Rauf et al. 2021; Agrawal et al. 2021; Aljunid and Huchaiah 2020; Ali et al. 2021; Kumar et al. 2021). The pre-copy migration consists of two phases: (1) warm-up phase, and (2) stop and copy phase (Hu et al. 2013). The warm-up phase uses the push technique in which foul memory pages are reconstructed on the host server. Foul memory represents data on a disc that is modified, but has not yet been written on the disc. It means that the upgraded or changed memory pages have been regenerated in the migration process. In the stop and copy phase, VM is stopped at the source node server (Cui et al. 2017). In the case of the post-copy migration method, at first, the VM is suspended at the host of the data center. (Hines et al. 2009). Then, VM is moved to the Target Server (TS) and starts execution. If any requests emerge for the memory page that is unavailable at the target server, it generates a page fault. In the hybrid-copy migration technique, there is a combination of pre-copy and post-copy migration techniques (Salfner et al. 2011). Here, at first, it works as a pre-copy live migration, and then, it uses the concept of post-copy migration technique. The main limitation of both these approaches, namely pre-copy and post-copy migration, is that the complete migration time consumption is increased, and live migration is used for allocating a VM from one physical node to another without turning it off (Huang et al. 2011; Deshpande et al. 2013). For significant improvement of the number of task allocations in the live migration process, a high accuracy-based cloud classifier model is used in (Zhang et al. 2021). There is a lot of overhead in this scheme in calculating the accuracy of cloud classification. In the KVM strategy (Deshpande et al. 2012), all the memory pages are initially transferred from the source to the target server, which results in large data transfer rate. The large data transfer rate generates additional overhead in the migration process. In 2020, Dhule et al. (Dhule and Shrawankar 2020) have proposed POF-SVLM to support low data transfer rate and lower downtime, for live VM migration. It uses a shared network between the source node and the destination node. By using this approach, it is easy to monitor the entire system for all the input and output requests. POF-SVLM reduces the migration time by 55–60%, but only a 20% reduction in CPU utilization time, which is another major concern.

To solve the concerns of the existing approaches, a novel scheme has been proposed in this work. In the proposed algorithm, the Master Server (MS) keeps tracking the entire processing of transferring the load from the Faulty Node (FN) to the target

server. The proposed technique is mainly based on three approaches, i.e. HSMT, VMRMT, and VMRBU, for minimizing the migration time, downtime, data transfer rate, and the total CPU cores used during migration. By shifting the load towards the master server and quick identification of faulty node, the overall migration time gets reduced. The proposed scheme is simulated using the CloudSim simulator, and many experiments are performed to evaluate the efficiency of the proposed scheme. Results and discussion show that this proposed novel scheme is better than the existing schemes.

The following are the key contributions of this paper:

1) In this paper, a novel approach has been proposed to solve the issues faced by VMs during live migration. It reduces the downtime, migration time, total cores of CPU, and data transfer rate to a great extent to make the overall system efficient.
2) The proposed novel approach breaks down the servers into two primary components: master server and Slave Server (SS). The master server always keeps track of the slave servers, which helps in identifying the fault immediately.
3) The performance of the proposed scheme has been evaluated by executing many experiments. Results and discussion show the efficiency of the proposed scheme over the existing schemes.

The rest of the paper is structured into several parts. Some related works and key findings from the existing approaches are described in Sect. 2. In Sect. 3, there is a description of the entire proposed scheme. The experimental results and their discussion are given in Sect. 4, and finally, Sect. 6 concludes the entire paper with some future work directions.

2 Related Works

There are many existing approaches for the live migration of VMs proposed by many academicians and researchers. Buyya et al. (Beloglazov and Buyya 2012) have presented three algorithms, namely Maximum Correlation Policy (MCP), Random Choice Policy (RCP), and Minimum Migration Time Policy (MMTP), which are used to migrate a VM to other host-assigned VM that requires minimal time to complete the migration process. In their scheme, the host gets shut down frequently. Xu et al. (2016) have proposed an approach for predicting the performance of the application on the basis of IaaS. They have considered hardware heterogeneity and interference-aware VM provisioning using online measured resource utilization. The performance of this scheme is low, if there is heterogeneity in the hardware system. Tao et al. (2016) have introduced a methodology related to energy consumption, which is based on bucket code to generate the token, i.e., a unit of bytes. Their Binary Graph Matching-based Bucket-code Learning Algorithm (BGM-BLA) uses a triple optimization model to reduce energy consumption, migration cost, and communication time between VMs. For comparison and analysis purposes, no open-source tool has been employed in this work. Wu et al. (2016) have proposed an approach for

cloud infrastructure based on storage area networks. The system developed in this approach can be applied to the private cloud. The model can predict the overhead during VM allocation using various variables like CPU utilization, I/O utilization, and VM CPU utilization. The limitation to this approach is that it has not been tested for various cloud infrastructures and hypervisors. An approach using different cloud platforms to improve the system efficiency in case of any fault tolerance is suggested by Ahmad and Andras (2019). This model uses open source software, namely Microsoft Azure and Amazon Elastic Compute Cloud (EC2), to measure the scalability of the entire system. Here, the response time varies even if the same experiments are executed in different clouds.

of cloud-based software services. Our technical scalability metrics are inspired by metrics of elasticity. We used two.

cloud-based systems to demonstrate the usefulness of our metrics and compare their scalability performance in.

two cloud platforms: Amazon EC2 and Microsoft Azure. O.

Moghaddam et al. (2020) have proposed an intelligent VM migration algorithm, i.e., Cellular Learning Automata-based Evolutionary Computing (CLA-EC), which minimizes the frequency of migrations and physical servers during the placement and replacement of physical servers. Thus, it reduces the power consumption at the data centers. However, the replacement of physical servers across data centers consumes more resources, which creates issues in many practical cases. Rajabzadeh et al. (2020) have suggested a resource monitoring method that efficiently uses memory and bandwidth by following a particular sequence. The sequence starts with testing the critical state of physical machines, then, defines the exact VM for migration, as well as applies the VM migration policy. This scheme continuously tracks load across the node servers, which generates overhead. Here, the node server provides the interaction between users and application.

A novel scheme using the concept of mirroring approach of continuous transmission of memory pages from source to target is suggested by Rajapackiyam et al. (2020). Their scheme includes an algorithm consisting of two layers, namely data mover and mirror block. In this scheme, a master node and a table are maintained to define a migrated VM list. This technique does not use the iterative transmission of data to avoid overhead. However, it faces high migration time. In (Zhang et al. 2013), the authors have mentioned that by using OpenStack, it is easy to maintain server consolidation and balancing of load across different nodes during VM live migration. To remove duplication of data blocks, a Static Chunking (SC) methodology is used in this scheme. It also saves the overall transmission time. The main limitation of this scheme is that it fails to optimize the deduplication time. The deduplication time is the amount of time to process the same message twice within a particular time limit. Sun et al. (2016) have proposed an enhanced serial migration strategy based on the post-copy migration strategy. They have developed queuing models for quantifying performance metrics, such as the average waiting time and the blocking ratio of all the migration requests. To manage large data size in video summarization, an algorithm based on deep bidirectional analysis has been proposed in (Hussain et al. 2020). However, this scheme is slow in its processing. Table 1 shows a comparison and illustrates brief details of existing schemes.

Table 1 Comparative study on existing approaches

Schemes	Key Findings	Advantages	Disadvantages
(Cui et al. 2017)	The pre-copy migration technique consists of two phases: (1) warm-up phase, and (2) stop and copy phase.	Here, all the memory pages are shifted from source to target server before the VM shuts down due to fault occurrence.	In this scheme, foul memory pages are re-constructed on the host server. Therefore, the migration time is high.
(Deshpande et al. 2012)	In KVM technique, an initial prototype model has been developed that allows migration of various VMs within a cluster.	This approach reduces the network traffic, which results in a 26% reduction in the migration time.	Here, all data stored in RAM is first tagged as dirty and must be re-located, which creates overhead during data transfer.
(Dhule and Shrawankar 2020)	Here, a network is shared between the source node and destination node to monitor the entire system.	This scheme uses a shared network to maintain the details of all input and output requests.	This model does not reduce the downtime to a great extent.
(Beloglazov and Buyya 2012)	This scheme deals with various methods like MCP, RCP, and MMTP, which are used to migrate a VM to other host-assigned VM.	It requires minimal time to identify the target server in the migration process.	Here, the host gets shut down frequently.
(Xu et al. 2016)	In this scheme, the resource is provided to the users using VM provisioning based on hardware heterogeneity and interference.	It supports predicting the performance of the application on the basis of IaaS.	However, predictable performance is sometimes low, when there is heterogeneity in the hardware system.
(Tao et al. 2016)	BGM-BLA is used in this scheme to optimize energy consumption.	This scheme reduces overall energy consumption and migration cost.	This scheme does not consider fault cases.
(Wu et al. 2016)	It can predict the overhead during VM assignment using various variables like CPU utilization, I/O utilization, and memory usage.	This scheme is suitable in the private cloud.	It cannot be applied in different cloud infrastructures.
(Ahmad and Andras 2019)	It uses different cloud platforms to improve system efficiency in case of any fault tolerance.	The performance of this scheme is high.	Here, the response time varies as per the cloud platforms.
(Moghaddam et al. 2020)	An algorithm is proposed to find out the replacement of physical servers during migrations of virtual machines.	The key benefit of this work is the minimal usage of energy during VM migrations.	This approach consumes more resources due to multiple replacements of physical servers.
(Rajabzadeh et al. 2020)	Here, an approach to monitor resources is proposed to efficiently use memory and bandwidth for migrating virtual machines.	The main advantage of this approach is that it reduces bandwidth usage during the VM migration process.	This algorithm generates overhead because of the continuous tracking of workload across hosts.

Table 1 (continued)

Schemes	Key Findings	Advantages	Disadvantages
(Rajapackiyam et al. 2020)	Here, continuous transmission of memory pages from source to target machine is performed using the mirroring technique.	This technique does not use the iterative transmission of data for minimizing overhead.	This scheme faces high migration time.
(Zhang et al. 2013)	In this scheme, the authors have used OpenStack to easily maintain a server, server consolidation, and balancing of load across different nodes during live migration.	This scheme minimizes the overall transmission time.	This approach fails to optimize the deduplication time.
(Sun et al. 2016)	It proposes an enhanced serial migration strategy based on the post-copy migration strategy.	In this work, there is a reduction in the average waiting time.	Multi-threading is used in this approach, which makes the system more complex.

use a technical measurement of the scalability

Table 2 Description of notations

Abbreviation	Description
RS_i	i^{th} running server
N	Total number of running server
SS_i	i^{th} slave server
M	Total number of slave server
VM_i	i^{th} VM
R_i	i^{th} resource
k	Processed $User_{Request}$ on the TS
m	replica of k at the MS
Ack	Acknowledgment
$Max_{Capacity}$	Maximum capacity of the node
$Max_{Capacity}(R_i)$	Maximum capacity of the node forR_i
$MS_{VM_{Execution_{State}}}$	Execution state of VM at the MS
$FN_{VM_{Execution_{State}}}$	Execution state of VM at the FN
$SS_{VM_{Execution_{State}}}$	Execution state of VM at the SS
$TS_{VM_{Execution_{State}}}$	Execution state of VM at the TS
$User_{Request}$	User's request
$MS_{Received_{User_{Request}}}$	User request received at the MS
$TS_{Received_{User_{Request}}}$	User request received at the TS
$FN_{Send_{User_{Request}}}$	User request send from the FN
$MS_{Mem_{Pg}}$	Memory pages of the MS
$FN_{Mem_{Pg}}$	Memory pages of the FN
$TS_{Mem_{Pg}}$	Memory pages of the TS
$Produced_{Result_{Received}}$	Processed result received by the user

3 Proposed Scheme

In the proposed scheme, a network is developed that consists of two clusters linked by an interconnection network. Each cluster consists of a MS and SSs. If the required resource is available, at least one TS from a set of slave servers is selected as a process migration server. Here, the entire workload is transferred to the master server, when a fault arises. The MS manages every request and starts maintaining the replica copy of memory pages. To transfer the entire load to the TS, at first, it identifies the available resources of TSs. Then, transfers the execution state of the VM to the selected TS. After that, all memory pages are shifted to the TS. The TS now manages the entire workload of the FN, and provides a response to the MS. There is a regular shifting of memory pages from the FN to the TS, and the MS is responsible for maintaining each replica copy of the FN. It is easy to transfer all the memory pages immediately because of parallel processing, which supports maintaining replica copies at the MS and starts searching for the next TS to continue the processing. Figure 2 depicts the creation process of a MS from all running servers based on the ability to process all user requests. The MS checks the existence of the FN. Then, a TS is selected with the highest capacity node from all SSs. After the selection of the TS, the execution state of the VM is transferred from the MS to the TS. Now, all memory pages are processed on TS. After that, the user's request and memory pages are loaded on the TS, and the request is processed on the TS. Then, the replica on the MS is generated, and finally, a response is sent to the user.

There are mainly three algorithms in the proposed scheme, namely HSMT, VMRMT, and VMRBU, and an objective function. HSMT optimizes the data trans-

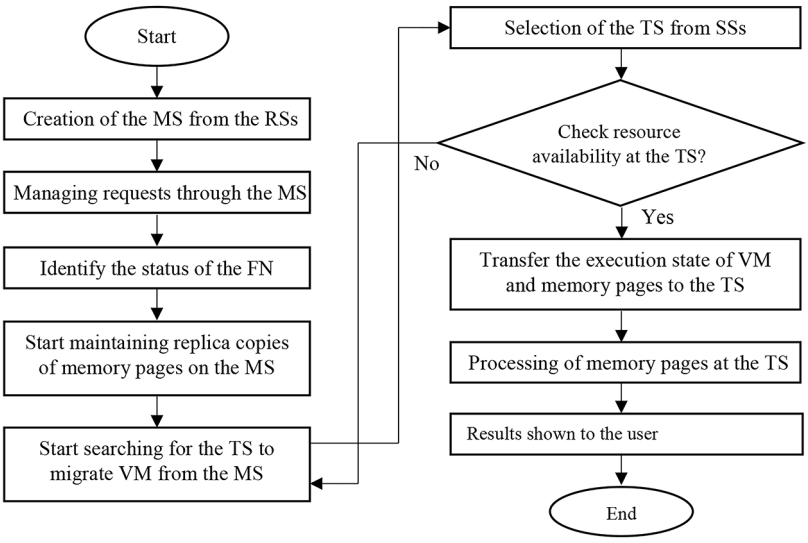

Fig. 2 Framework of the proposed approach to migrate the entire workload from the FN to the TS

fer rate and downtime, VMRMT optimizes the migration time, VMRBU optimizes the actual bandwidth consumption to reduce the consumption of the overall CPU core, and the objective function is responsible for overall optimization of processing of users' requests from the FN to the TS. The proposed work ensures low data transfer rate, so indirectly it consumes fewer CPU cores. There is no traffic congestion because of the gradual shifting of load, which supports quick processing and results in less downtime and less migration time.

3.1 Host Selection Migration Time

This algorithm starts by receiving all users' requests, and forwards them for creating VMs, which supports to continuously track all nodes. Here, the MS is selected among the Running Servers (RSs) on the basis of the maximum capacity of a particular node, so that it must accept many requests during any unexpected fault. When a fault occurs, the processing of the user request gets stuck on the SS, and the SS stops responding by giving an acknowledgment to the MS. This triggers the MS that a fault has occurred. Now, the MS tracks the node, where the process gets stuck and marks that particular SS as FN. After that, every request of the FN is managed by the MS. The proposed approach helps in the quick identification of the FN by sending a broadcast message to all SSs. If all SSs give acknowledgments to the MS, then, there is no need for migration. Then, the execution state of the FN is transferred to the MS along with all memory pages for further processing of the user request. The VMs are then stopped at the FN and the memory pages are returned to the MS. This approach easily handles the load of the FN through the MS and provides a better approach by consuming less time, which results in low downtime. Due to the gradual transfer of data, the overall data transfer rate gets down that results in low network congestion. Therefore, it decreases the downtime, and memory pages are quickly transferred from the FN to the MS. Algorithm 1 represents the working of HSMT and Table 2 represents the details of the notations used in this paper.

Algorithm 1: Host selection migration time

Input: $User_{Request}$

Output: $MS_{Mem_{pg}}$

```
Start
    for ∀ User_Request
        for ∀ VM_i
            for ∀ RS_i
                for i = 1 to N
                    if RS_i > Max_Capacity
                        MS ← RS_i
                        return MS
                    else
                        i ← i + 1
                end for
            end for
            for ∀ SS_i
                for i = 1 to M
                    SEND broadcast message to all SS_i
                    REPLY with an Ack from the SS_i to the MS
                    if (Ack ! = True)
                        FN ← SS_i
                    else
                        i ← i + 1
                        Faulty node does not EXIST
                end for
            end for
            MS_VMExecution_State ← FN_VMExecution_State
            MS_Received_User_Request ← FN_Send_User_Request
            MS_Mem_Pg ← FN_Mem_Pg
            STOP VM_i at FN
            return MS_Mem_Pg
        end for
    end for
Stop
```

3.2 VM Reallocation Migration Time

After completing the execution of Algorithm 1, Algorithm 2 executes a search operation for the maximum capacity node from the existing SSs and marks it as the TS. The node with maximum capacity, which has sufficient resources is selected for hosting the load from the MS. After that, the execution state of the MS is transferred to the TS. If sufficient resources are not available, the current TS is discarded and the next TS is chosen for execution as shown in Fig. 2. Then, the user's request is transferred from the MS to the TS along with memory pages. The running VMs are suspended after shifting all the memory pages at the TS. This approach consumes less timespan for migrating the VMs because of the quick transfer of the load from the FN to the appropriate TS with the help of the MS by maintaining all tracks and replica copies of memory pages. This reduces the overall time period for the entire migration process. The functioning of VMRMT optimizes the migration time by immediately shifting all workloads from the MS to the TS.

Algorithm 2: VM reallocation migration time

Input: SS, and R_i

Output: $TS_{Mem_{Pg}}$

1. Start
2. for $\forall\ SS_i$
3. for $i = 1$ to M
4. if $SS_i > Max_{Capacity}(R_i)$
5. $SS_{VM_{Execution_{State}}} \leftarrow MS_{VM_{Execution_{State}}}$
6. RECORD i^{th} SS as TS
7. else
8. $i \leftarrow i + 1$
9. end for
10. $TS_{VM_{Execution_{State}}} \leftarrow MS_{VM_{Execution_{State}}}$
11. $TS_{Received_{User_{Request}}} \leftarrow MS_{Received_{User_{Request}}}$
12. $TS_{Mem_{Pg}} \leftarrow MS_{Mem_{Pg}}$
13. SUSPEND VM_i on MS
14. return $TS_{Mem_{Pg}}$
15. end for
16. Stop

3.3 VM Reallocation Bandwidth Usage

After the completion of Algorithm 2, VM starts to migrate from the FN to the TS. At first, it loads the user's request along with the memory pages on the TS. After that, the response of the processed user's request is stored in a variable k in step 6. The response to that user's request must be displayed to the requested user, when the fault has been fixed and the memory page and workload have been shifted from the FN to the TS. When any produced result fails before reaching the user, it can be handled by the replica (in step 7) of the response maintained at the MS. The data transfer in VMRBU is consistent since all memory pages operate on the MS, which prevents the formation of dirty pages. This approach consumes less CPU usage for migrating the VMs from the FN to the TS through the MS. Algorithm 3 represents all the steps in which the user's request is not lost during migration and it is managed through the MS, if any fault occurs on the FN.

Algorithm 3: VM reallocation bandwidth usage

Input: $User_{Request}$
Output: k or m

1. Start
2. for ∀ $User_{Request}$
3. INITIALIZE $k = 0$
4. LOAD $TS_{Received_{UserRequest}}$
5. LOAD $TS_{Mem_{Pg}}$
6. k = Processed $User_{Request}$ on the TS
7. m = Replica of k at MS
8. if $Produced_{Result_{Received}} == True$
9. return k
10. else
11. return m
12. end for
13. Stop

3.4 Objective Function

The objective function identifies the optimum utilization of CPU cores, and minimizes the downtime, migration time, and data transfer rate during the live migration process. In this work, the cumulative time to migrate a VM is the weighted sum of the HSMT, VMRMT, and VMRBU as given in Eq. (1) (Mousavi et al. 2017). This helps the algorithms to predict the time of migration near to reality and guarantees the algorithm's extensibility and adaptability. If variables A, B and C represent the times taken to execute Algorithms 1, 2, and 3, respectively, to change the effects of these three variables in the optimization objective function, weight coefficients, α, β and γ, are used. The sum of values of these coefficients must be equal to 1 for maintaining the result normalized as all the variables contribute to one entity, i.e., the total time consumed.

$$Totaltimeconsumed = \propto \times A\,(HSMT) + \beta \times B\,(VMRMT) + \gamma \times C\,(VMRBU) \quad (1)$$

These three coefficients can efficiently balance the optimization of the HSMT, VMRMT, and VMRBU. Datacenter administrators can adjust these three coefficients to have different HSMT, VMRMT, and VMRBU as shown in Eq. (1). When $\propto = 0.4$, $\beta = 0.4$ and $\gamma = 0.2$, the performance of these proposed algorithms is better as it reduces execution time for the entire migration process and to process the user request. The implementation of these algorithms shows optimization of the total migration time that leads to improvement of the objective function.

4 Performance Analysis

This section represents the performance analysis of the proposed scheme. The experiments regarding the above algorithms are performed on a set of hardware and software as discussed in Sect. 4.1.

4.1 Experimental Setup

The experiments are executed on a Dell XPS 8940 desktop. The hardware specifications of this computer system are Intel Core i7-11700 processor (11th generation CPU, 4.9 GHz), Windows 10 Pro OS (64-bit), 64 GB RAM, 1 TB SSD, and 2 TB HDD. Java 17 and CloudSim 3.0.3 are installed on the computer system for executing experiments (Calheiros et al. 2011). In the CloudSim, the CPU utilization is managed on the basis of uniform distribution of load. As represented in Table 3, there are different configurations of VMs to define specific attributes, such as length of instruction, input/output file size, the number of processors required, etc., for responding to user requests and other tasks. Datacenter defines a set of hosts with their hardware configurations, such as memory, CPU cores, capacity, and many more, and the provision for allocating these resources is based on the availability of the number of hosts and other resources (Zhang et al. 2015).

In the experiments, three different types of VMs are considered, i.e., VM-1, VM-2, and VM-3. The configurations of these VMs are given in Table 3. There is an automated environment for live migration by using the Bash script (LeCun et al. 2010). The Bash script automates live migration after running VM-1 for 5 min. Similarly, the automation of live migration takes place after running the setup of VM-2 and VM-3 for 15 and 20 min, respectively. The start and end times of the live VM migration are recorded in a log file for data transfer, and downtime is measured by sending an ICMP echo request from the client computer to network hosts.

4.2 Results and Discussion

To evaluate the performance of the proposed scheme, a variety of workloads, namely (1) compress, (2) compiler, (3) derby, (4) crypto, (5) scimark.small, (6) mpegaudio, (7) scimark.large, (8) Sunflow, (9) serial, (10) XML, 11) ApacheBench, 12) MySQL, 13) startup, and 14) TensorFlow, are considered in multiple iterations of VM live migration setups, i.e. VM-1, VM-2, and VM-3. The data transfer rate, CPU utilization, VM downtime, and migration time, are calculated for 14 migrations of each form of various workloads on VM1, VM2, and VM3 setups for the proposed scheme, as well as for the existing schemes, i.e. KVM and POF-SVLM.

Figure 3(a), 3(b), and 3(c) show the comparisons of data transfer rate among existing approaches and proposed HSMT for 14 different workloads on three VM setups, i.e. VM-1, VM-2, and VM-3, during live migration. The proposed HSMT approach helps in keeping the data transfer rate low, as a result, HSMT outperforms traditional

Table 3 Configurations of VM setup

Configuration	VM-1	VM-2	VM-3
CPU speed (MIPS)	250	300	500
I/O file size (MB)	10,000	20,000	30,000
RAM (MB)	613	870	1740
Bandwidth (bits/s)	500	700	1000
No. of CPU (cores)	2	3	2

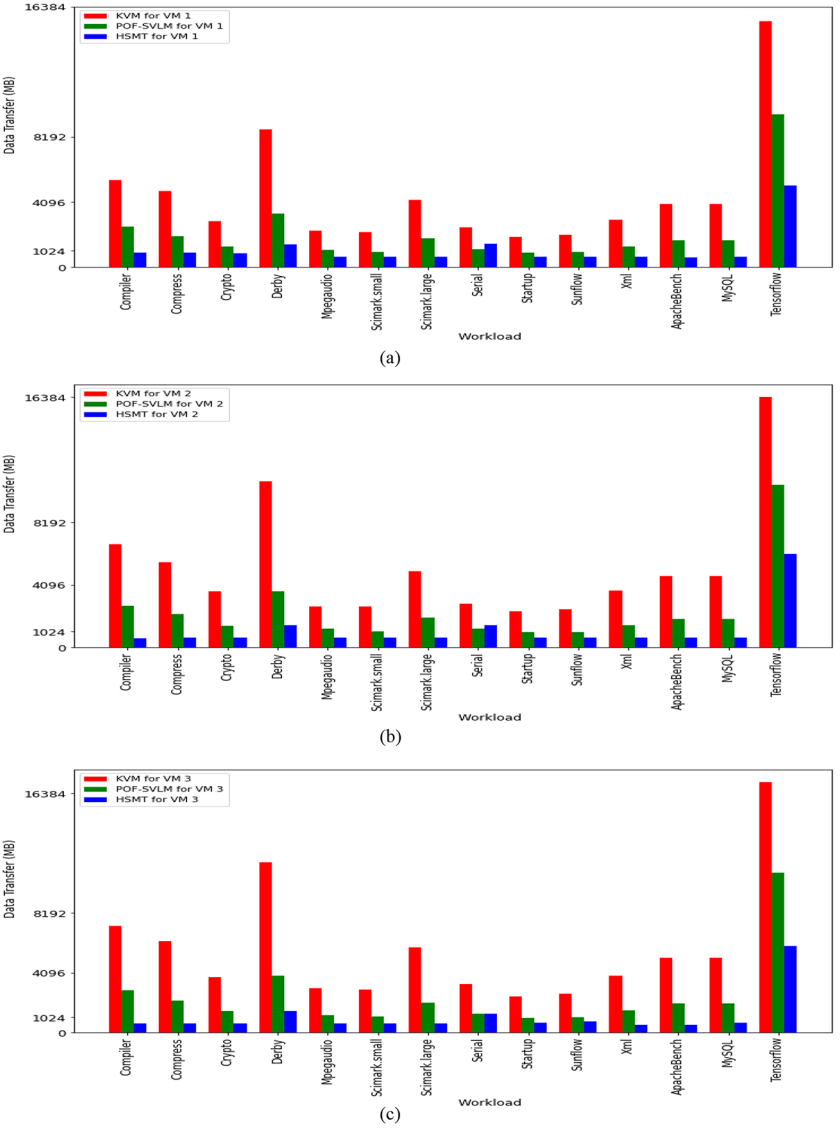

Fig. 3 Comparison of data transfer among KVM, POF-SVLM, and HSMT. (a) VM-1 setup (b) VM-2 setup, and (c) VM-3 setup

KVM and POF-SVLM approaches. The proposed approach transfers data to the MS, and then, makes all memory pages available at the TS. In this way, the total data transmission rate has been reduced by 40-50%. In the case of derby and TensorFlow workloads, the workload is large. To accommodate this large workload, the VM memory size is increased as the live migration phase maintains a copy of memory

pages from the original node (FN) to the destination node (TS). The more data transfer rate, the more time it takes to migrate the VM.

Figure 4(a), 4(b), and 4(c) display the comparisons of VM downtime of the proposed HSMT and already developed approaches, namely KVM and POF-SVLM. HSMT helps in reducing downtime for the migration process by quickly identifying

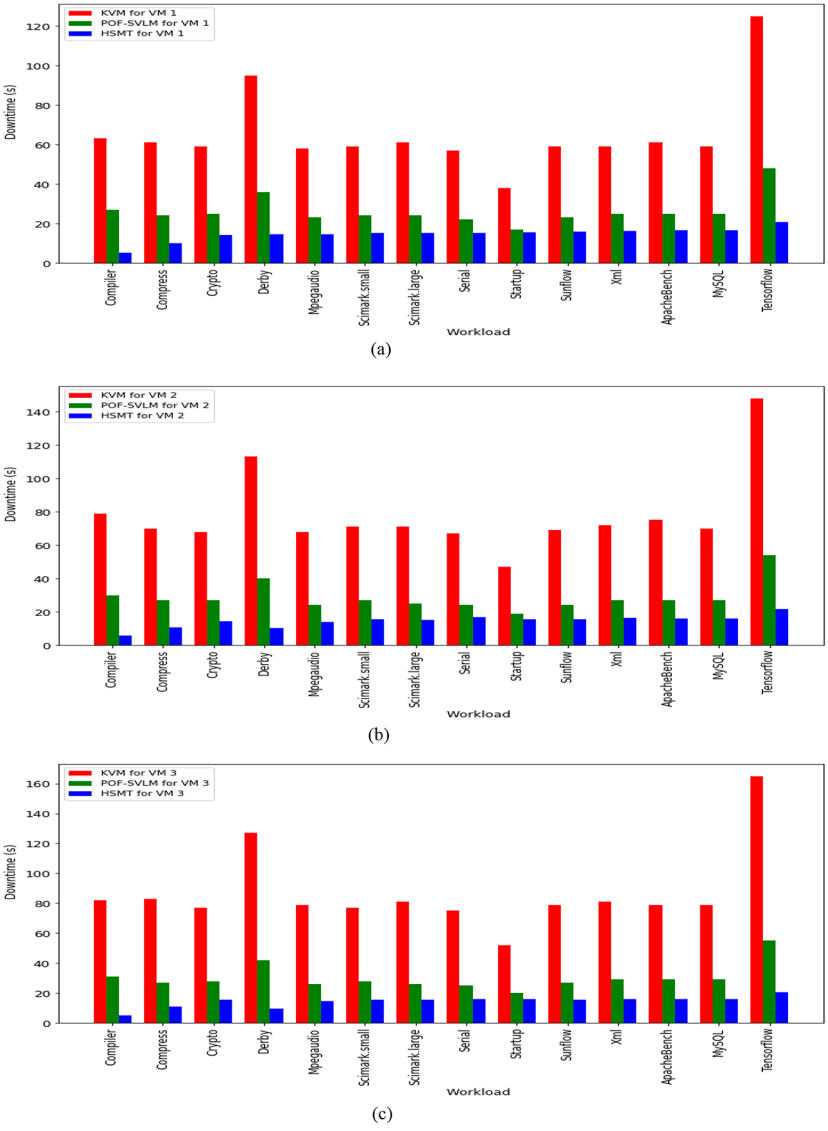

Fig. 4 Comparison of downtime among KVM, POF-SVLM, and HSMT. (a) VM-1 setup (b) VM-2 setup, and (c) VM-3 setup

the FN. Here, by executing the automated bash script on the system, the initialization of the workloads process and the live migration process are executed, and a log file is maintained for the entire experimental procedure to secure the start time and end time of the script execution. In the existing approaches, the downtime is more because all the memory pages are transferred to the target machine and VM takes a long time

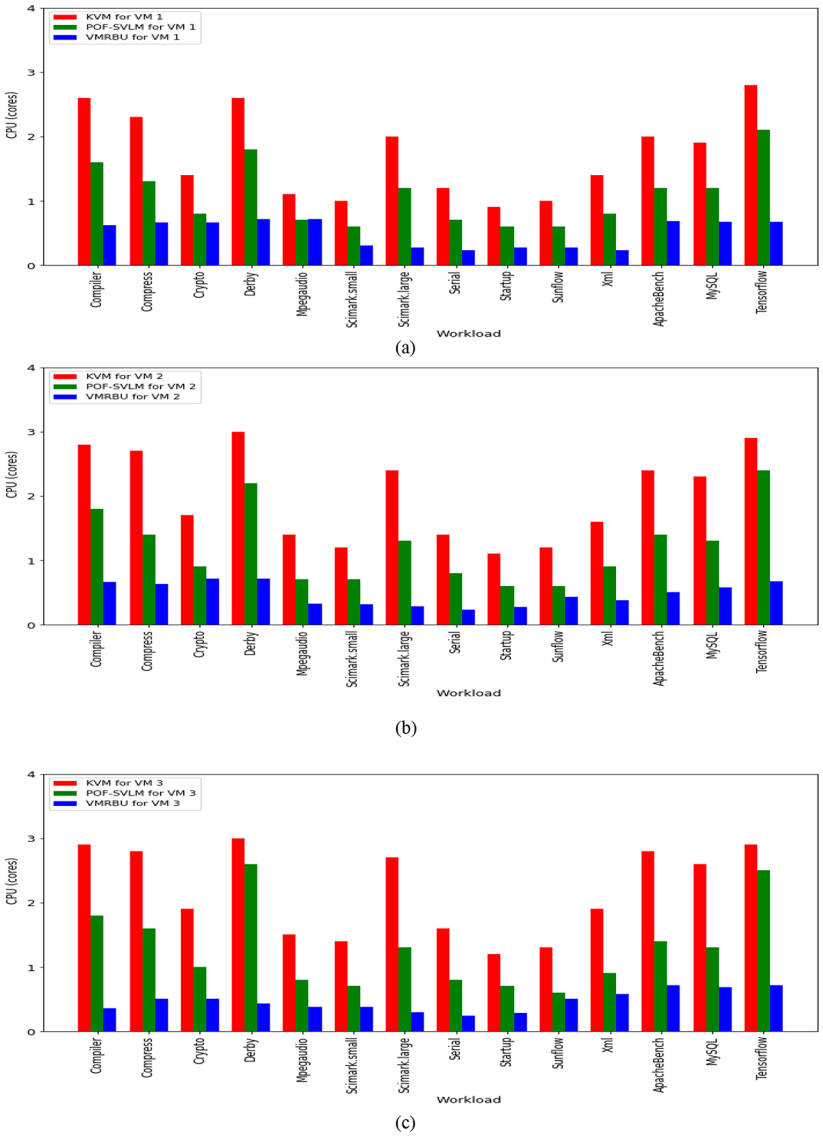

Fig. 5 Comparison of CPU utilization among KVM, POF-SVLM, and VMRBU. (a) VM-1 setup (b) VM-2 setup, and (c) VM-3 setup

to resume at the target machine. The results show an improvement as the downtime of the proposed approach is in the range of 0–20 s, which is less than 60 s, the time taken by the existing POF-SVLM approach. Thus, it reduces the total downtime by 70-80%. Further, it is also recorded that since the memory size of VM is large for workloads like derby and Tensorflow, the downtime is relatively high.

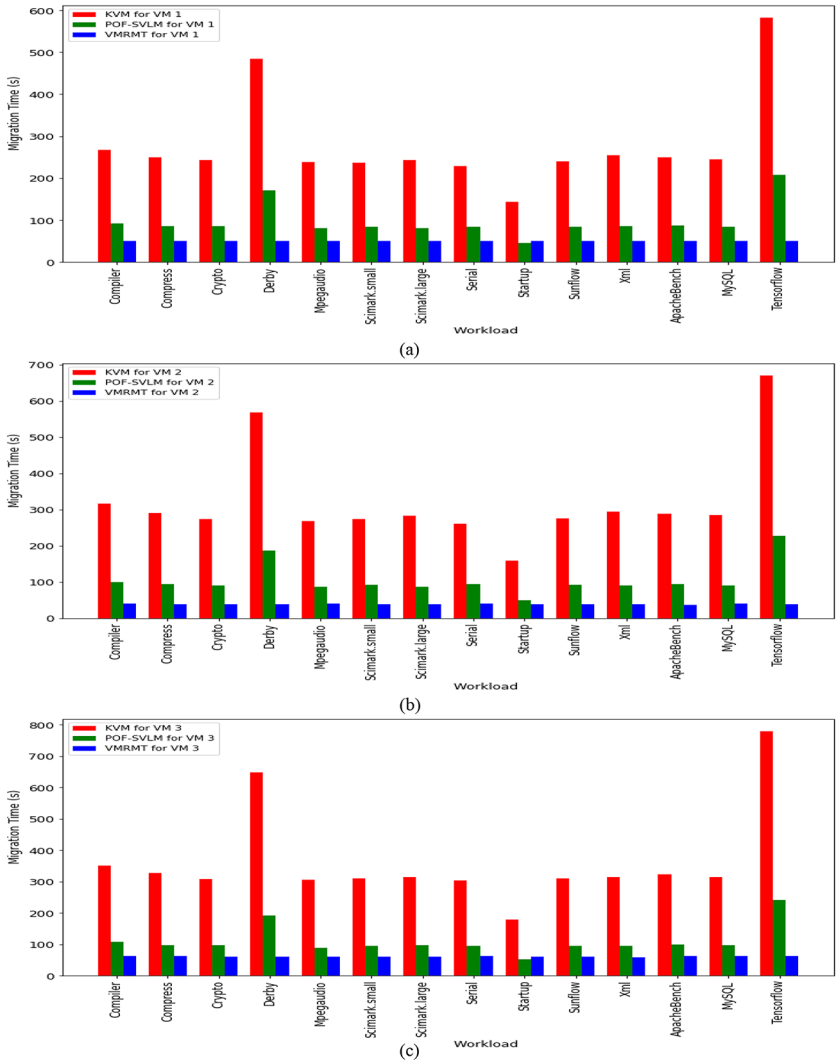

Fig. 6 Comparison of migration time among KVM, POF-SVLM, and VMRMT. (a) VM-1 setup (b) VM-2 setup, and (c) VM-3 setup

Figure 5(a), 5(b), and 5(c) show the CPU utilization for various workloads with different VM setups for live VM migration. In this experimental procedure, the number of CPU cores utilized during live migration is considered as a parameter. It can be easily seen in Fig. 5(a), 5(b), and 5(c) that the CPU consumption has reduced by 60-70% due to VMRBU. When VM starts to migrate from the FN to TS, in VMRBU, the data transfer is consistent because the production of dirty pages is avoided by keeping all the memory pages running on MS. Whereas, KVM and POF-SVLM send all the dirty pages to the destination. Therefore, a large number of dirty pages are transferred. These existing schemes take additional time to resume VM at the destination node. Thus, the utilization of CPU cores is more.

Figure 6(a), 6(b), and 6(c) show that the migration time is less than 50 s except for extremely heavy workloads. Here, the cloud users are unable to detect the migration time because the entire load of the FN is quickly managed through the TS. By using VMRMT, the total migration time is reduced by 40-50% in comparison to the existing approaches. In the case of KVM, it transfers all memory pages and keeps recording every memory page in the VM memory as foul memory. POF-SVLM uses a shared network to send all the information regarding input and output requests to the target machine, which generates a longer delay in migration time. Among all the comparisons made, there is a significant change, when workloads of derby, serial, and TensorFlow, are executed as the workload is more because of the large VM memory size. The final results show an improvement in the overall time consumption of the live migration process by reducing the data transfer rate, CPU utilization, migration time, and VM downtime.

5 Conclusions and Future Works

In a cloud computing environment, sometimes, it is required to move a running VM to various servers during live migration without disconnecting the client or application. Accelerating the live migration process can be helpful to manage data centers for correct and efficient functioning. In this paper, the main emphasis is given on the live migration challenges, such as transfer of data, time consumed for migration, CPU utilization, and downtime. To solve these problems, this paper proposes a novel scheme using three main algorithms, namely HSMT, VMRMT, and VMRBU. Here, in the first phase, the master server is created and all user's requests are handled through the master server. In the second phase, an appropriate target server is selected, and at last, the user's requests are processed and the result of it is shown to the user by maintaining a replica copy of memory pages at the master server. The experimental results demonstrate that the proposed scheme is better than the existing schemes, and it reduces the total CPU cores by 60-70%, downtime by 70-80%, data transfer rate by 40-50%, and migration time by 40-50%. In the future, the proposed scheme can be extended by developing a novel scheme for deallocating the virtual machines, when users are removed from the system. In addition, there is a huge scope to identify failures that occur during live migration in a cloud computing environment.

References

Agrawal, D., Minocha, S., Namasudra, S., Gandomi, A.H.: "A robust drug recall supply chain management system using hyperledger blockchain ecosystem," Comput. Biol. Med. 140, 2021. DOI: https://doi.org/10.1016/j.compbiomed.2021.105100

Ahmad, A.A.S., Andras, P.: "Scalability analysis comparisons of cloud-based software services," J. Cloud Computing: Adv. Syst. Appl. 8, 1, 2019. DOI:https://doi.org/10.1186/s13677-019-0134-y

Ali, H.M., Liu, J., Bukhari, S.A.C., Rauf, H.T.: "Planning a secure and reliable IoT-enabled FOG-assisted computing infrastructure for healthcare," Cluster Comput. 24, 2021. DOI:https://doi.org/10.1007/s10586-021-03389-y

Aljunid, M.F., Huchaiah, M.D.: Multi-model deep learning approach for collaborative filtering recommendation system. CAAI Trans. Intell. Technol. **5**(4), 268–275 (2020) ",",

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010) ",",

Bala, A., Chana, I.: Fault tolerance-challenges, techniques and implementation in cloud computing. Int. J. Comput. Sci. Issues **9**(1), 288–293 (2012) ",",

Beloglazov, A., Buyya, R.: 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 **24**(13), 1397–1420 (2012) ",",

Calheiros, R.N., Ranjan, R., Beloglazov, A.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Pract. Experience **41**(1), 23–50 (2011) ",",

Choudhary, A., Govil, M.C., Singh, G., Awasthi, L.K., Pilli, E.S., Kapil, D., "A critical survey of live virtual machine migration techniques," *Journal of Cloud Computing, Advances, Systems and Applications*, vol. 6, no.1, 2017. DOI: https://doi.org/10.1186/s13677-017-0092-1

Cui, Y., Yang, Z., Xiao, S., Wang, X., Yan, S.: Traffic-aware virtual machine migration in topology-adaptive dcn. IEEE/ACM Trans. Networking **25**(6), 3427–3440 (2017) ",",

Deshpande, U., Kulkarni, U., Gopalan, K., "Inter-rack live migration of multiple virtual machines," In *Proceedings of the 6th international workshop on virtualization technologies in distributed computing*, ACM, Delft, Netherland, 2012, pp 19–26

Deshpande, U., Schlinker, B., Adler, E., Gopalan, K., "Gang migration of virtual machines using cluster-wide deduplication," In *Proceedings of 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, ACM, Delft, Netherland, 2013, pp. 394–401

Dhule, C., Shrawankar, U.: POF-SVLM: Pareto optimized framework for seamless VM live migration. Comput. Springer-Verlag GmbH Austria **102**(8), 2158–2183 (2020) ",",

Gao, Z., Zhang, H., Dong, S., Sun, S., Wang, X., Yang, G., Wu, W., Li, S., de Albuquerque, V.H.C.: Salient object detection in the distributed cloud-edge intelligent network. IEEE Netw. **34**(2), 216–224 (2020) ",",

Gao, J., Wang, W., Liu, Z., Billah, M.F.R.M., Campbell, B., "Decentralized federated learning framework for the neighborhood: A case study on residential building load forecasting," In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, ACM, Portugal, 2021, pp. 453–459

Garrido, L.D., Sanz, J.J.G., Mestras, J.P.: Foundations for the design of a creative system based on the analysis of the main techniques that stimulate human creativity. Int. J. Interact. Multimedia Artif. Intell. **7**(2), 199–211 (2021) ",",

Gómez, A.B., Sánchez, J.L.L., Aguilar, M.A.: Blockverse: A cloud blockchain-based platform for tracking in affiliate systems. Int. J. Interact. Multimedia Artif. Intell. **6**(3), 24–31 (2020) ",",

Hamdy, M., Helmy, S., Magdy, M.: Design of adaptive intuitionistic fuzzy controller for synchronisation of uncertain chaotic systems. CAAI Trans. Intell. Technol. **5**(4), 237–246 (2020) ",",

Hines, M.R., Deshpande, U., Gopalan, K.: Post-copy live migration of virtual machines. ACM SIGOPS Operating System Review **43**(3), 14–26 (2009) ",",

Hu, W., Hicks, A., Zhang, L., Dow, E.M., Soni, V., Jiang, H., Bull, R., Matthews, J.N., "A quantitative study of virtual machine live migration," In *Proceedings of the ACM cloud and autonomic computing conference*, ACM, Miami, Florida, USA, 2013, pp. 1–10

Huang, D., Ye, D., He, Q., Chen, J., Ye, K., "Virt-LM: A benchmark for live migration of virtual machine," In *Proceedings of 2nd ACM/SPEC International Conference on Performance Engineering*, ACM, Karlsruhe, Germany, 2011, pp. 307–316

Hussain, T., Muhammad, K., Ullah, A., Cao, Z., Baik, S.W., de Albuquerque, V.H.C.: Cloud-assisted multiview video summarization using CNN and bidirectional LSTM. IEEE Trans. Industr. Inf. **16**(1), 77–86 (2020) “,”,

Kumar, A., Shah, K., Namasudra, S., Kadry, S.: A novel elliptic curve cryptography based system for smart grid communication. Int. J. Web Grid Serv. **17**(4), 321–342 (2021) “,”,

LeCun, Y., Cortes, C., Burges, C.J., Mnist handwritten digit database at&t labs. Available: <background-color:#FF3300;uvertical-align:super;>http://yann.lecun.com/exdb/mnist/.</background-color:#FF3300;uvertical-align:super;><uvertical-align:super;>,</uvertical-align:super;><uvertical-align:super;> </uvertical-align:super;>2010 [Accessed on 10 June 2021]

Malleswari, T.Y.J.N., Vadivu, G.: “Adaptive deduplication of virtual machine images using AKKA stream to accelerate live migration process in cloud environment,” J. Cloud Computing: Adv. Syst. Appl., 8, 1, 2019. DOI:https://doi.org/10.1186/s13677-019-0125-z

Moghaddam, M.J., Esmaeilzadeh, A., Ghavipour, M., Zadeh, A.K.: Minimizing virtual machine migration probability in cloud computing environments. Cluster Comput. **23**(1), 3029–3038 (2020) “,”,

Motaki, S.E., Yahyaouy, A., Gualous, H., Sabor, J.: A new weighted fuzzy C-means clustering for workload monitoring in cloud datacenter platforms. ” Cluster Computing **24**(4), 3367–3379 (2021) “, ,

Mousavi, S., Mosavi, A., Várkonyi-Kóczy, A.R., Fazekas, G.: Dynamic resource allocation in cloud computing. Acta Polytech. Hungarica **14**(4), 83–104 (2017) “,”,

Namasudra, S.: “Fast and secure data accessing by using DNA computing for the cloud environment,” IEEE Trans. Serv. Comput., 2020. DOI:https://doi.org/10.1109/TSC.2020.3046471

Namasudra, S., Chakraborty, R., Majumder, A., Moparthi, N.R.: Securing multimedia by using DNA based encryption in the cloud computing environment. ACM Trans. Multimedia Comput. Commun. Appl. **16**(3), 1–19 (2020) “,”,

Obasuyi, G., Sari, A.: Security challenges of virtualization hypervisors in virtualized hardware environment. Int. J. Commun. Netw. Syst. Sci. **8**(8), 260–273 (2015) “,”,

Rajabzadeh, M., Haghighat, A.T., Rahmani, A.M.: New comprehensive model based on virtual clusters and absorbing Markov chains for energyefficient virtual machine management in cloud computing. J. Supercomputing **76**(3), 7438–7457 (2020) “,”,

Rajapackiyam, E., Subramanian, A.V., Arumugam, U.: Live migration of virtual machines using mirroring technique. J. Comput. Sci. **16**(4), 543–550 (2020) “,”,

Rauf, H.T., Gao, J., Almadhor, A., Arif, M., Nafis, M.T.: “Enhanced bat algorithm for COVID-19 short-term forecasting using optimized LSTM,” Soft. Comput., 25, 2021. DOI:https://doi.org/10.1007/s00500-021-06075-8

Salfner, F., Troger, P., Polze, A., “Downtime analysis of virtual machine live migration,” In *Proceedings of DEPEND the Fourth International Conference on Dependability*, Nice, France, 2011, pp. 100–105

Sangpetch, A., Sangpetch, O., Juangmarisakul, N., Warodom, S.: “Thoth: Automatic resource management with machine learning for container-based cloud platform,” In *Proceedings of the 7th International Conference on Cloud Computing and Services* Science, ACM, Porto, Portugal, 2017, pp. 75–83

Sharma, P., Moparthi, N.R., Namasudra, S., Vimal, S., Hsu, C.H.: “Blockchain-based IoT architecture to secure healthcare system using identity-based encryption,” Expert Syst., 2021. DOI:https://doi.org/10.1111/EXSY.12915

Sun, G., Liao, D., Anand, V., Zhao, D., Yu, H., “A new technique for efficient live migration of multiple virtual machines,” *Future Generation Computer Systems*, vol. 55, no. C, pp. 74–86, 2016

Suruliandi, A., Kasthuri, A., Raja, S.P.: Deep feature representation and similarity matrix based noise label refinement method for efficient face annotation. Int. J. Interact. Multimedia Artif. Intell. **7**(2), 66–77 (2021) “,”,

Tao, F., Li, C., Liao, T.W., Laili, Y.: BGM-BLA: A new algorithm for dynamic migration of virtual machines in cloud computing. IEEE Trans. Serv. Comput. **9**(6), 910–925 (2016) “,”,

Wang, C., Yang, G., Papanastasiou, G., Zhang, H., Rodrigues, J.J.P.C., de Albuquerque, V.H.C.: Industrial Cyber-Physical Systems-Based Cloud IoT Edge for Federated Heterogeneous Distillation. IEEE Trans. Industr. Inf. **17**(8), 5511–5521 (2021) “,”,

Wu, H., Ren, S., Garzoglio, G., Timm, S., Bernabeu, G., Chadwick, K., Noh, S.: A reference model for virtual machine launching overhead. IEEE Trans. Cloud Comput. **4**(3), 1–14 (2016) “,”,

Xu, F., Liu, F., Jin, H.: Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud. IEEE Trans. Comput. **65**(8), 2470–2483 (2016) “,”,

Yuan, G., Li, J., Fan, H.: Evaluating the robustness of image matting algorithm. CAAI Trans. Intell. Technol. **5**(4), 247–259 (2020) “”, ,

Zhang, J., Han, S., Wan, J., Zhu, B., Zhou, L., Ren, Y., Zhang, W.: "IM-Dedup: An image management system based on deduplication applied in DWSNs," Int. J. Distrib. Sens. Netw., 9, 7, 2013. DOI:https://doi.org/10.1155/2013/625070

Zhang, R., Su, X., Wang, J., Wang, C., Liu, W., Lau, R.W.H.: On mitigating the risk of cross-VM covert channels in a public cloud. IEEE Trans. Parallel Distrib. Syst. **26**(8), 2327–2339 (2015) ",",

Zhang, F., Liu, G., Fu, X., Yahyapour, R.: A survey on virtual machine migration: challenges, techniques, and open issues. IEEE Commun. Surv. Tutorials **20**(2), 1206–1243 (2018) ",",

Zhang, J., Liu, P., Zhang, F., Iwabuchi, H., A. A. d. H. e. A. de Moura, de Albuquerque, V.H.C., "Ensemble meteorological cloud classification meets internet of dependable and controllable things," *IEEE Internet of Things*, vol. 8, no. 5, pp. 3323–3330, 2021

Authors and Affiliations

Ambika Gupta[1,2] · Suyel Namasudra[2]

✉ Suyel Namasudra
suyelnamasudra@gmail.com

[1] Department of Computer Engineering and Applications, GLA University, Mathura, India

[2] Department of Computer Science and Engineering, National Institute of Technology Patna, Bihar, India