International Journal of Industrial Engineering, 30(1), 67-77, 2023

SERCON-BASED TIMESTAMPED VIRTUAL MACHINE MIGRATION SCHEME FOR CLOUD

Nisha Chaurasia[1, *], Shashikala Tapaswi[2], and Joydip Dhar[2]

[1]Department of Information Technology
Dr. B R Ambedkar National Institute of Technology Jalandhar
Punjab, India
[*]Corresponding author's e-mail: chaurasian@nitj.ac.in

[2]Department of Information Communication Technology,
Atal Bihari Vajpayee Indian Institute of Information Technology and Management Gwalior
Madhya Pradesh, India

With the advent of cloud computing, the need for deploying multiple virtual machines (VMs) on multiple hosts to address the ever-increasing user demands for services has raised concerns regarding energy consumption. Considerable energy is consumed while keeping the data centers with a large number of servers active. However, in data centers, there are cases where these servers may not get utilized efficiently. There can be servers that consume sufficient energy while running resources for a small task (demanding fewer resources), but there can also be servers that receive user requests so frequently that resources may be exhausted, and the server becomes unable to fulfill requests. In such a scenario, there is an urgent need to conserve energy and resources, which is addressed by performing server consolidation. Server consolidation aims to reduce the total number of active servers in the cloud such that performance does not get compromised as well as energy is conserved in an attempt to make each server run to its maximum. This is done by reducing the number of active servers in a data center by transferring the workload of one or more VM(s) from one server to another, referred to as VM Migration (VMM). During VMM, time is supposed as a major constraint for effective and user-transparent migration. Thus, this paper proposes a novel VM migration strategy considering time sensitivity as a primary constraint. The aim of the proposed Time Sensitive Virtual Machine Migration (TS-VMM) is to reduce the number of migrations to a minimum with effective cost optimization and maximum server utilization.

Keywords: Cloud Computing; Server Consolidation; Virtual Machine Migration.

(Received on July 20, 2022; Accepted on September 22, 2022)

1. INTRODUCTION

Today's advanced form of computing understood as cloud computing, offers energy, cost, and effective computing to the world of technological advancement. Cloud computing environments provide high Quality of Service (QoS) for their customers, resulting in the necessity to deal with a power-performance trade-off (Beloglazov and Buyya, 2011; Beloglazov and Buyya, 2013). It facilitates data storage or server resources to be shared from a resource pool among multiple cloud customers and accessed by multiple applications. The main resource types in data centers include computing resources, storage resources, and network resources (Lin *et al., 2013*).

The term cloud in cloud computing implies the concept of computation that is done on-the-fly, with little human intervention. A cloud is defined as a place in a network infrastructure where Information Technology (IT) and computing resources (Zhang *et al.*, 2016), such as computer hardware, operating systems, networks, storage, databases, and even entire software applications, are available instantly, on-demand (Kumar *et al.*, 2014), (Huang *et al.*, 2016). Moreover, a cloud is based on different service models, which include the pay-per-service and the pay-per-use service models (Khalid *et al.*, 2013). Virtualization serves as an important element of cloud computing and enables the creation of multiple virtual instances, called Virtual Machines (VMs) (He *et al.*, 2014; Shen *et al.*, 2015), on a single host/server. The use of virtual machines allows users to securely gain administrative privileges within the guest operating system and customize the runtime environment according to their specific requirements (Mauch *et al.*, 2013). Figure 1 depicts a cloud where multiple virtual machines run on a physical server. These physical servers form the various distributed data centers and provide the facility to obtain access to the data center from the diverse locations forming the overall cloud.

In the cloud, multiple servers run consistently, encapsulating a number of virtual machines fulfilling the requested tasks. In addition, these virtual machines running on servers sometimes vary in their computation parameters to meet the requested service requirements. Even though visualization allows multiple VMs to run on fewer servers (in comparison to running a single host for each task), there are still a huge number of servers running in the cloud environment. The challenge here is to minimize the number of these servers to an extent such that resources and power are saved from being wasted. Thus, arises an urgent demand for performing server consolidation (Ferreto *et al.*, 2011; Mazumdar *et al.*, 2017; Abadi *et al.*, 2018). Server consolidation is an approach to ensure efficient usage of computer server resources in order to reduce operating costs (Speitkamp *et al.*, 2010). It allows the allocation of a maximum number of VMs on a minimum number of hosts and undesirable servers to be placed into a low-power state, switched off or devoted to the execution of incremental workload (Mastroianni *et al.*, 2013). Figure 2 shows the cycle for managing servers in the cloud involving consolidation as a phase.

Figure 1 Schema of Cloud

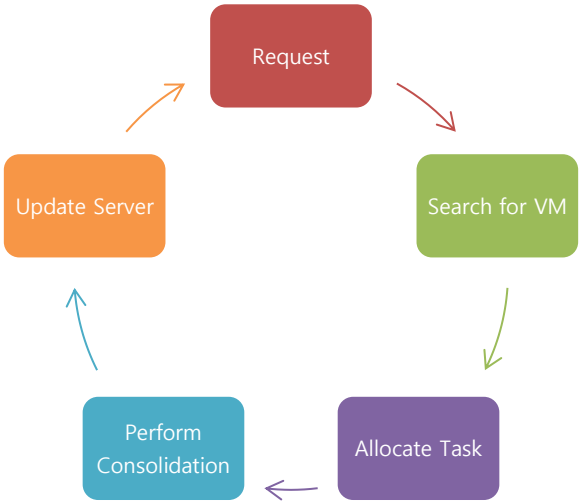

Figure 2 Workflow of a server in the cloud involving consolidation as a phase

Consolidation can save management costs because it is much easier to manage a small number of machines than a large number (Lee *et al.*, 2011). Consolidation can be done both in the server and virtual machine. When the consolidation is done by reducing the number of physical servers/machines (PMs) by maintaining the same number of VMs installed as a whole, it is known as server consolidation, while when the consolidation is done for the execution of multiple VMs on the same host to reduce power consumption, it is termed VM consolidation (Esfandiarpoor *et al.*, 2015). However, the terms may be used interchangeably.

For consolidation, when a server lacks or is overwhelmed by computation tasks, an attempt is made to migrate the task to a server (virtual machine, originally) capable of handling the resource demand and computation requirements. There are many issues affecting consolidation, including server and workload behavior, security restrictions requiring the co-location of certain application components, and power line redundancy restrictions (Srikantaiah *et al*., 2008). VM consolidation also raises several management issues because it tends to optimal exploitation of available resources while avoiding severe performance degradation due to resource consumption of co-located VMs (Corradi *et al.*, 2014). This is generally done by performing a VM migration process where the selected VM (originally the task) is transferred from one host to another to reduce the number of servers running in a data center. VM migration helps to successfully achieve various resource management objectives, such as load balancing, power management, fault tolerance, and system maintenance (Hsu *et al.*, 2014; Ahmad. *et al.*, 2015).

Time is considered a prominent factor during migration, such that users remain unaware of the change that occurred. The paper, therefore, presents a novice, time-sensitive VM Migration algorithm that aims at performing the minimum migration with effective cost optimization and maximal server utilization. The following sections are: Section II is a literature review of server consolidation; Section III concerns the proposed algorithm; Section IV is the experimentation and results which compare the proposed algorithm with the existing Sercon algorithm; and, lastly, Section V concludes the discussion. In respect of the importance of server consolidation, the next section discusses the related work in the field.

2. RELATED WORK

As cloud computing is becoming increasingly popular and more data centers are being built, reducing their power consumption is an important issue (Ho *et al.*, 2011). The section explores the work performed to achieve an ideal level of server consolidation. Much research has confirmed the high value of consolidation. The various consolidation schemes analyzed or proposed by the researchers can be categorized into three types:

1. Cost-based or performance-based (Abdulgafer *et al.*, 2009; Ye *et al.*, 2011; Beloglazov and Buyya, 2013): The researchers considered a particular aspect while performing the computation in the cloud for minimizing the cost of migration, CPU or disk usage or input/output events. They contributed to the consolidation by limiting these aspects, resulting in high performance and increasing the number of requests completed without violating the Service Level Agreement (SLA).
2. Parameter-based (Ho *et al.*, 2011; Lee *et al.*, 2011; Corradi *et al.*, 2014; Rao *et al.*, 2015): Researchers have focused on certain evaluation parameters, viz. memory, CPU requirement, etc. A parameter is pre-decided, and consolidation algorithms were proposed based on this parameter. The purpose of this parameter is to determine the most appropriate server to which an incoming workload can be allotted. These parameters are formally based on an NP-hard bin-packing problem, such as best-fit, first-fit, and heaviest-fit server findings.
3. Nature-inspired (Mastroianni *et al.*, 2013; Singh. *et al.*, 2013; Perumal *et al.*, 2016): Researchers in the nature-inspired category performed an in-depth study on the behavior of various instances from nature, such as Swarm V-formation and Honeybee Hive formation. The authors sought to learn how these flying insects coordinate among each other such that energy is conserved. In mirroring the insects and their behavior in the cloud scenario, they are considered as the servers or data centers, and their behavior is imitated as the behavior the cloud servers would adopt to conserve energy. Hence, according to the way the insects coordinate their work, the data centers in the cloud are categorized into active servers, fully loaded active servers, idle servers, and power-down servers. In the literature, it was found that managing the servers in these classes allows an easier assessment of how energy can be conserved.

One of the approaches mentioned in the literature is Sercon. Sercon is one of the known performance-based algorithms. Sercon aims to reduce the number of migrations and increase the number of released nodes such that performance is enhanced. Sercon considers CPU and memory as dominating resources and hence estimates a value lambda (λ) (CPU limit) from the total CPU and memory available. Using this, it calculates a score for each running VM and server. Based on the score, the algorithm determines the VM whose task is to be migrated as well as where it can be migrated. In contrast to Sercon, the proposed work proposes an improved algorithm where time is added as a major constraint for estimating the instant of migration. The proposed time-sensitive algorithm for VM migration (TS-VMM), similar to Sercon, considers the estimation of a particular resource (CPU or memory or bandwidth or disk) from its respective available resource pool. It calculates the score value for each node as well as for each VM in the cloud. From the estimated values of load, the least-loaded VM(s) is selected as the candidate for migration. A schedule is formed based on which, after execution of the algorithm, migration of the candidate VM(s) is initiated. If a VM during migration does not release the node, the schedule rolls back the migration

process and switches to another VM migration. In the presented algorithm, time is added as a critical measure dependent on which migration takes place. The next section hence discusses the proposed TS-VMM.

3. PROPOSED APPROACH

Considering the scenario of VM migration (VMM) for server consolidation, the existing Sercon algorithm (Murtazaev *et al.*, 2011) is chosen as the basis for the proposed Time-sensitive VM Migration algorithm. The Sercon algorithm performs consolidation offline and inherits some properties of well-known heuristic algorithms for bin-packing, First-Fit, and Best-Fit (Murtazaev *et al.*, 2011). It aims at obtaining a minimum number of migrations, provided the least loaded nodes can be released. Taking into account the VM migration strategy, the following assumptions are made:

1. There are three levels defined for active servers, viz. under-utilized, normal, and over-utilized, according to their computational ability.
2. The running cost for each server is defined as a fixed and variable component.

$$C_{ji}(t) = C_{ji_0} + C_{ji}.t\ , \tag{1}$$

where $C_{ji}(t)$ is the cost function with a fixed component C_{ji_0} and variable component $C_{ji}.t$; i is the service assigned to server j; and t is the time of utilization of a server.

3. The cost is dependent on the time, t, for which the server is utilized.
4. The jobs performed are executed in parallel.

The overall cost optimization objective function is thereby formulated as follows:

$$min \sum_{j=1}^{m} \sum_{i=1}^{p} C_{ji}(t)\, n_k Y_{ji} \tag{2}$$

where $C_{ji}(t)$ is the cost function with a fixed cost, n_k is the number of servers in the k^{th} category (low as 1, medium as 2, and high as 3) among the total of m servers (n_k varies from 0 to l where l is a constant defining number of active servers in k category), and Y_{ji} is the indication that server j is assigned with service i among the total of p services arrived.

The objective function is followed such that

$$\sum_{j=1}^{m} Y_{ji} = 1, \forall i \in [1 \dots p] \tag{3}$$

$$\sum_{j=1}^{m} u_{ir} Y_{ji} \leq s_{jr} n_k, \forall i \in [1 \dots p], \forall r \in R \tag{4}$$

where u_{ir} defines the units of resources required by a service from resource pool R and s_{jr} defines the resource capacity of the server, and

$$n_k \in \{0,1\}, \forall k = \{1,2,3\}\ and\ Y_{ji} \in \{0,1\}, \forall \{i\ and\ j\} \tag{5}$$

This ensures that the objective function aims at minimizing the server costs such that each service is assigned at least once for execution and the server capacity does not exceed its limit because of multiple workloads from multiple jobs.

Sercon considers CPU and memory as dominating resources and estimates a value lambda (λ) (mean of CPU/memory limit) from the total CPU and memory available. Also, similar to the discussion in (Murtazaev A. *et al.*, 2011), the rank of an item (server) j, called score with (i) CPU limit (w_j) and (ii) memory limit (v_j), respectively, is calculated using surrogate weight as (shown in equation (6)):

$$s = \lambda.w_j + (1 - \lambda).v_j, \tag{6}$$

where

$$\lambda = \frac{\sum_{j \in m} w_j}{\sum_{j \in m} w_j + v_j}$$

on m servers. Based on the score calculated, all the servers and VMs are ranked, and the algorithm determines the VM to be migrated as well as the host where it can be migrated. In contrast to Sercon, the proposed work is an improved algorithm

where time is added as a critical constraint for estimating the instant of migration. The timestamp would regulate the periodic checks for the possibilities of VM migrations. This makes the VMs complete their tasks, avoiding SLA violations. In the cloud, the multiple tasks executing simultaneously inherit the possibility of migrations. The timestamp will help in avoiding the early decisions (may lead to frequent and redundant migrations) and late decisions (may lead to improper load balance on servers) to migrate the VM(s), as it allows a periodic check based on the proportionate time taken as an average of tasks completion time of all VMs, helping in energy minimization. The proposed time stamp algorithm for VM migration (TS-VMM), similar to Sercon, considers the estimation of a particular resource (CPU or memory or bandwidth or disk) from its respective available resource pool. It calculates the score value for each node as well as for each VM in the cloud. From the estimated values of load, the least-loaded VM(s) is selected as the candidate for migration. A schedule is formed based on which, after execution of the algorithm, migration of the candidate VM(s) is initiated. If a VM during migration does not release the node, the schedule rolls back the migration process and switches to another VM migration. Cloud, along with the migration, demands timely monitoring of executing servers such that any server which initially was running in the normal state starts demanding higher resources, making the server run overwhelmed can be handled. Hence, to address this, an average execution time of all the VMs is estimated, and it is added as the timestamp on which migration possibility is checked.

3.1 Discussion of TS-VMM

Considering the scenario of VM migration (VMM) for server consolidation, the existing Sercon algorithm (Murtazaev *et al.*, 2011) is chosen as the basis for the proposed Time Stamp VM Migration algorithm. Thus, the complexity of TS-VMM is visualized as that of Sercon, which is $O(n^4)$. The Sercon algorithm performs consolidation offline and inherits some properties of well-known heuristic algorithms for bin-packing, First-Fit, and Best-Fit (Murtazaev *et al.*, 2011). It aims at obtaining a minimum number of migrations, provided the least loaded nodes can be released. The proposed algorithm takes into account the capacities of all physical as well as virtual servers. The migration efficiency with tolerable resource efficiency value of each server is defined, and a threshold is set (beyond which a server is restricted to run). Successful and unsuccessful migration counts (that occurred in the past) are maintained so as to make migration attempts limited. Also, since it is required to monitor the executing workload in a timely manner, the timestamp is incorporated, which checks for the possibility of migrations.

The comparison of the Sercon algorithm with the proposed TS-VMM algorithm is shown in Table 1.

Table 1. Comparison of Sercon and TS-VMM

Sercon Algorithm	TS-VMM Algorithm
Offline/Cold migration algorithm	Online/ Hot migration algorithm
Considers CPU and memory as a resource	Considers all resources
No threshold is defined for servers	The threshold is taken into account
A single check for migration	A periodic check for migration using timestamp

3.2 TS-VMM Algorithm

The working of the TS-VMM algorithm is as follows:

i. *Initialization of λ and score value*
The number of VMs and servers with their respective resource limit, the total number of migrations performed, unsuccessful migration attempts, and the time stamp for checking migration possibility are initialized, and the value of λ (mean of resource limit) and score (overall value of resource limit) is obtained from the particular resource (CPU/memory/disk/network) limit to keep track of each server. Initial migration efficiency (ME_0) is chosen as a lower bound above which migration is permitted. ME_0 is calculated on the basis of total migrations and the number of released nodes. It defines the percentage a VM contributes to releasing a node. A timestamp is associated with each defined virtual machine for checking the possibility of migration.

ii. *Limiting attempts of migration by specific VM*
The frequent migration of a particular VM is restricted by checking the unsuccessful migration attempts for it. The migration attempts are checked until the value for allowed migration attempts is greater than unsuccessful migration attempts. In this, if a VM is not migrating for the first time and have greater migration efficiency than the initial migration efficiency (ME_0), the updated migration efficiency is checked for it.

iii. *Attempting migration by ranking all VMs and servers using score*

After updating the migration efficiency, all the servers (nodes) are sorted in decreasing order using a score. Also, the candidate VM for migration is determined, the value of λ is calculated for all the VMs, and the score sorts and ranks these VMs in order of least residual capacity.

iv. Checking for successful and unsuccessful migrations with updated λ and score values

Using the updated values of λ and score, the feasibility of migration is checked. If it is a successful attempt, the migration process takes place using the pre-defined schedule. The migration efficiency is checked, and the residual capacity of the VM is estimated. Also, the new values of λ and the score for the new VM are calculated, incrementing the successful attempt count. While for unsuccessful migration, λ and the score for failed VM are calculated.

v. Timestamp for periodic checking of the updated status of servers with an attempt for migration possibility

The timestamp facilitates the instant at which migration possibility is checked. It learns the time for which each virtual machine is going to execute a task and estimates the average execution timestamp value, using which it performs a check for over/under utilization. This helps in the periodic monitoring of servers and estimating the servers that can be put to the idle state, ensuring consolidation. It also ensures that the same server does not get utilized repeatedly, as the continuous running of one server also degrades the performance.

The pseudo-code for the proposed algorithm (Algorithm 1) is as follows:

Algorithm 1: Time-Sensitive VM Migration

Require:

1. Number of servers, r is the resource capacity of VMs, Re is the total resource capacity of servers, Number of VMs on servers, Threshold (preset), Total number of Migrations, Unsuccessful Migrations, ME_0, i.e., Migration Efficiency value, Timestamp ts for task completion assigned to each VM, cl is CPU limit, and ml is Memory limit.
2. λ (defined as: r=(Re))
3. score = $|\lambda * r + (1 - \lambda) * (Re - r)|$

} Initialization of λ and score value

procedure TS-VMM

4. **while** (allowed_migration_attempts >= unsuccessful_migrations)
5. **if** (count! = 1 and migration_efficiency > ME_0) **then**
6. Calculate migration efficiency
7. **end if**
8. **end while**

} Limiting attempts of migration by specific VM

9. Calculate the score for each node and Sort in decreasing order.
10. Determine candidate VMs for migration
11. Calculate λ and the score for each VM and sort the respective score with the least residual capacity for each VM

} Attempting migration by ranking all VMs and servers using score

12. Check for successful VM migration.
13. **If** (success is received) **then**
14. Perform the migration based on the schedule defined and calculate the migration efficiency and calculate the new residual of VM.
15. Calculate the new value of λ and score for the VM to which workload has been transferred and increment the successful attempts count.
16. **else**
17. Calculate λ and the score of the VM which failed to migrate.
18. Increment the unsuccessful attempts.
19. **end if**

} Checking for successful and unsuccessful migrations with updated λ and score values

20. After Timestamp, ts, go to step 1 to calculate changed λ and score values and check for the status of servers as normal, under or overutilized.

end procedure

} Timestamp for periodic checking of the updated status of servers with an attempt for migration possibility

At time interval avg(t), the requirement of the consolidation process is checked. Moreover, it is assumed that consolidation may take place only at the time instance avg(t). Two types of events may occur (i) Selection of VMs for migration (ii) Release of VMs after migration. Thus, maximizing the number of physical servers to be put in the sleep state (minimizing energy consumption). In Figure 3, a sketch of the consolidation progression using the TS-VMM algorithm is depicted.

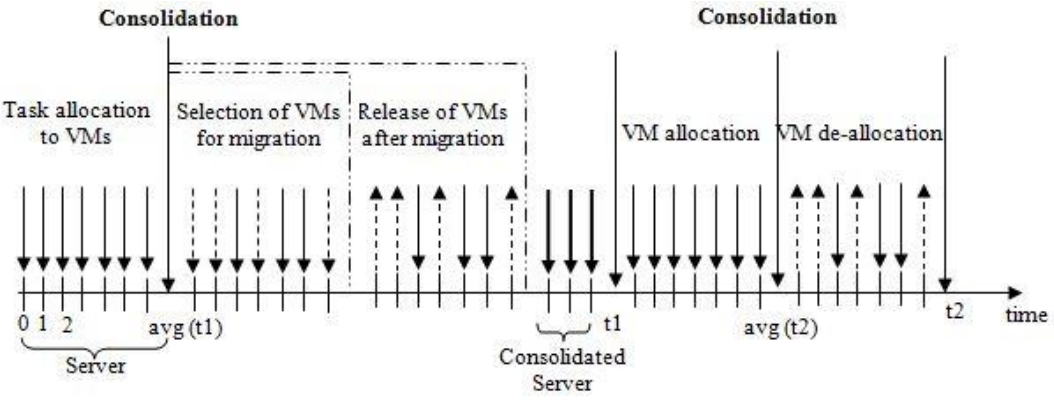

Figure 3. Consolidation progression in TS-VMM

Figure 4 presents the workflow of the Time Stamp Virtual Machine Migration scheme, following the strategy for consolidation as defined in Algorithm 1.

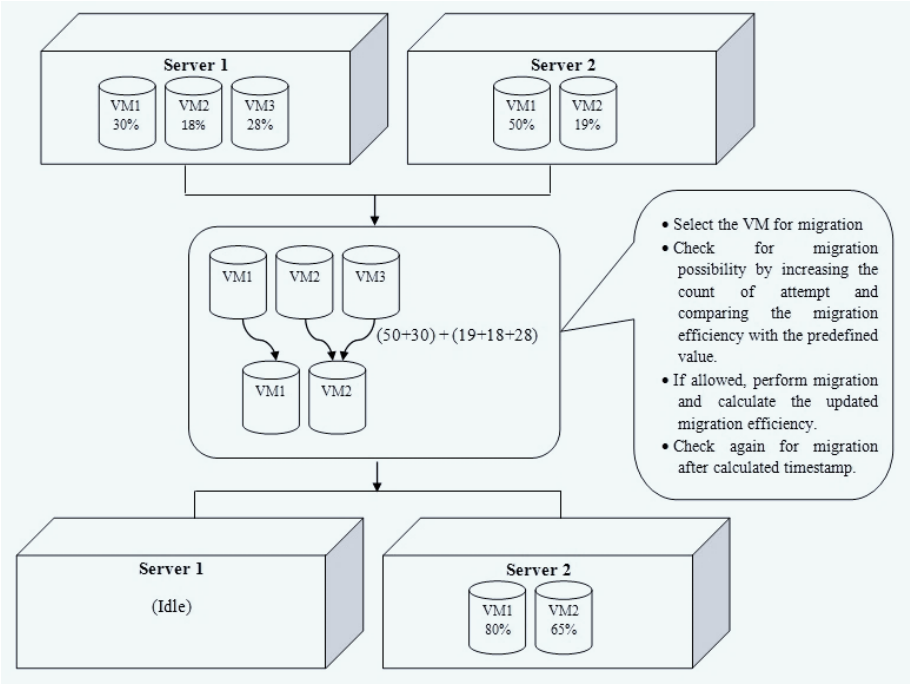

Figure 4. Workflow of TS-VMM

4. EXPERIMENTATION SETUP AND RESULTS

The analysis of the proposed TS-VMM algorithm has been performed using the widely preferred open-source software, OpenStack (Corradi A. *et al.*, 2014). The version of OpenStack chosen for deployment is Juno. The experimental environment consists of a Controller node, a Network node, and two Compute nodes. The conceptual architecture used for experimentation is three-node architecture with OpenStack networking, known as neutron networking, and is shown in Figure 5.

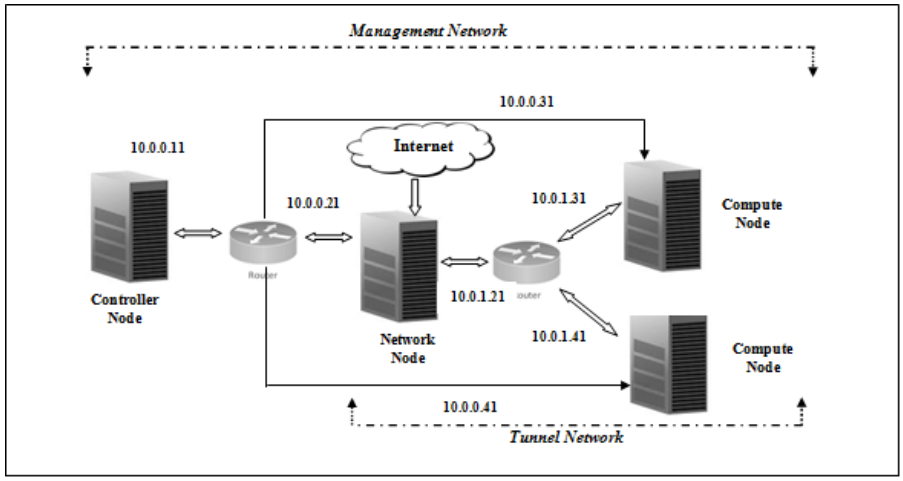

Figure 5. OpenStack Experimental Cloud Setup

The platform used for the deployment utilized Ubuntu 14.04 with 32-bit processors fulfilling the hardware requirements demanded. Two switches were also used for deploying the OpenStack networking services. The management network, i.e., the communication between all four nodes, is managed by one switch, while the Tunnel network, where Compute nodes communicate through Network nodes, is done by the second switch. One NIC required for the Controller node is used for the management network. Among the three NICs of the Network node, two NICs are used for the management network and the instance tunnel network, and one NIC is for accessing the external network. The two NICs of the Compute node are for the management and tunnel network. The details of the hardware requirements used for the experimental set-up are as follows (Table 2):

Table 2. Hardware Requirements for Neutron in OpenStack

Node	Specification
Controller Node	8GB RAM, 1-2 CPU, 100GB Storage, 1 NIC
Network Node	2GB RAM, 1-2 CPU, 50GB Storage, 3 NIC
Compute Node	8+GB RAM, 2-4+ CPU, 100+GB Storage, 2 NIC

The proposed Time-sensitive VM Migration (TS-VMM) algorithm is based on the effective Sercon algorithm for VM migration policy. Sercon has proved its significance against the most fundamental bin-packing algorithm, such as First-Fit Decreasing (FFD). The proposed algorithm, however, adds advancement to it. Along with the Sercon features, the algorithm addresses the virtual environment with a timestamp managed for each initiated virtual machine, estimating the time at which chances for migration are checked. The server categories are also maintained according to the running capabilities, viz., low, medium, and high-end. The proposed algorithm performs checking for migration while assigning new tasks.

The success of the proposed algorithm has been validated using OpenStack for live migration. In the initial state, two compute servers are be allocated with the VMs in the active state before the migration process. Thereafter, the migration of VMs from one server to another, where TS-VMM comes into action for deciding the time stamp (instance) at which this migration is expected to take place. Subsequently, the VMs that qualify for migration are made to transfer load under live migration. At last, the migrated VMs actively run on the new host (server).

The experimentation of the proposed Time-Stamp Virtual Machine Migration algorithm shows that it also provides the same functionality, along with similar accuracy, as the Sercon algorithm. The number of migrations, along with the number of VMs used and nodes released, exposes the out-performance of the algorithm presented. Although there are some comparative variations in migration efficiency, in most cases, it provides 100% efficiency. The preset CPU threshold value for migration is 0.7. The migration efficiency is calculated as follows:

$$ME = \frac{\text{total number of nodes released}}{\text{total number of migrations}} \times 100$$

The comparison of the TS-VMM, Sercon, and FDD can be clearly understood by the following comparison graphs. Figure 6 shows the comparison of the number of migrations performed with the number of VMs used for the scenario. It can be seen that FFD requires a greater number of VMs with the increase in the number of migrations compared to Sercon and TS-VMM, while TS-VMM consumes a similar number of VMs, indicating that it is able to compete with Sercon's performance.

Figure 7 shows the number of nodes released against the number of migrations performed. Again, here FDD requires more migrations as the number of nodes released increases, whereas Sercon and TS-VMM again use a lesser number of migrations while increasing the number of released nodes.

Figure 8 shows the comparison of FFD, Sercon, and TS-VMM concerning the number of VMs used against the number of nodes used for migrations. As TS-VMM operates similarly to Sercon, the number of VMs afforded by the servers remains almost similar to the VMs afforded by servers in Sercon.

Figure 9 shows the migration efficiency comparison between the three algorithms. It can be seen that the TS-VMM outperforms the other two algorithms. Although Sercon has 100% migration efficiency in many cases, TS-VMM seeks to ensure a consistent 100% efficiency in almost every case.

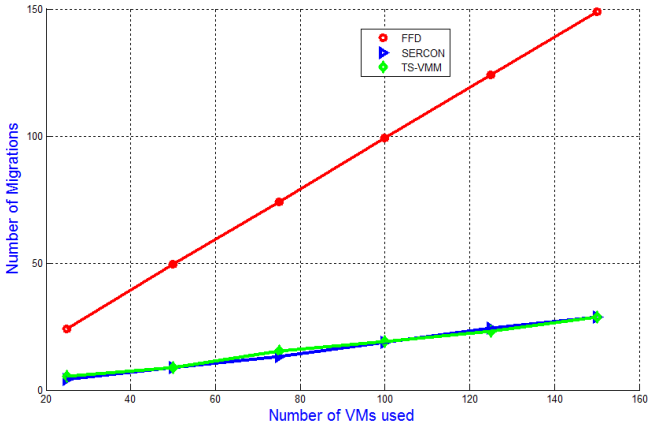

Figure 6. Relationship between Number of migrations and Number of VMs used in TS-VMM

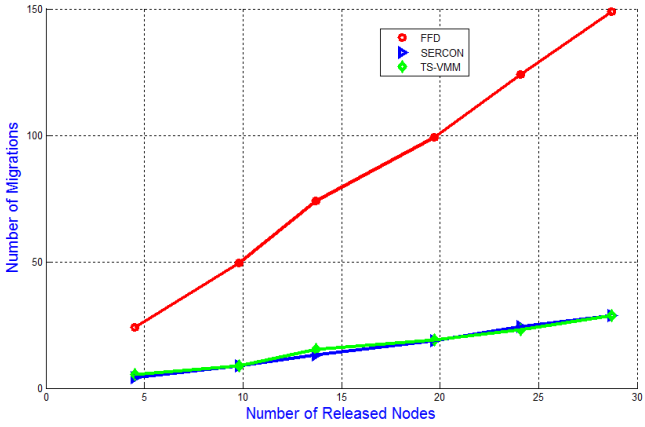

Figure 7. Relationship between Number of migrations and Number of released nodes in TS-VMM

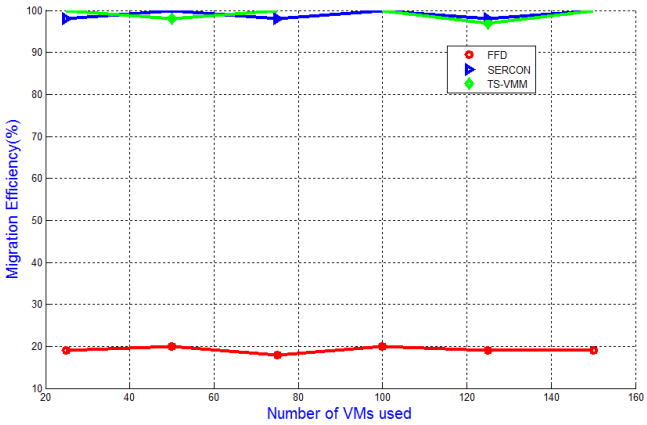

Figure 8. Relationship between Number of nodes and Number of VMs used in TS-VMM

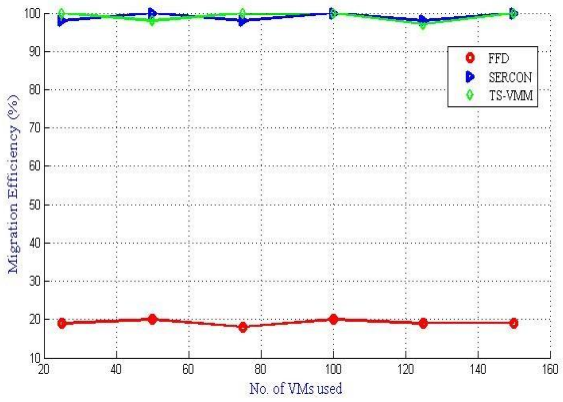

Figure 9. Relationship between Migration Efficiency and Number of VMs used in TS-VMM

5. CONCLUSION

The paper proposes a new time-sensitive VM migration policy, termed TS-VMM. Based on the Sercon algorithm, the new algorithm has proved its advancement. The TS-VMM algorithm incorporating timestamp adds an advantage to the working scheme by defining the time for each VM task completion, estimating the average time at which it checks for consolidation possibility. When the comparison between TS-VMM, Sercon, and FFD is made for the performance evaluation by estimating the number of migrations against the number of VMs used, number of migrations against the number of nodes released, migration efficiency against the number of VMs used, and migration efficiency against the number of nodes released, it is

observed that TS-VMM competes with Sercon with approximately 98.7% efficiency, while FFD scores a maximum of 20% migration efficiency.

REFERENCES

Abadi, R. M. B., Rahmani, A. M., and Alizadeh, S. H. (2018). Server Consolidation Techniques in Virtualized Data Centers of Cloud Environments: A Systematic Literature Review. *Software: Practice and Experience*, 48(9): 1688-1726.

Abdulgafer A. R., Marimuthu P. N., and Habib S. J. (2009), Network Redesign through Servers Consolidations. *Proceedings of Information Integration and Web-based Applications and Services (iiWAS2009)*.

Ahmad, R. W., Gani, A., Hafizah, S., Hamid, A. B., Shiraz, M., Yousafzai, A., and Xia, F. (2015). A Survey on Virtual Machine Migration and Server Consolidation Frameworks for Cloud Data Centers. *Journal of Network and Computer Applications*, 52:11–25.

Beloglazov A. and Buyya R. (2011), Energy Efficient Allocation of Virtual Machines in Cloud Data centers. *Proceedings of IEEE/ACM Cluster, Cloud and Grid Computing*.

Beloglazov, A. and Buyya, R. (2013). Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data centers under Quality-of-service Constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(7): 1366–1379.

Corradi, A., Fanelli, M., and Foschini, L. (2014). VM Consolidation: A Real Case based on OpenStack Cloud. *Future Generation Computer System*, 32: 118–127.

Esfandiarpoor, S., Pahlavan, A., and Goudarzi, M. (2015). Structure-aware Online Virtual Machine Consolidation for Data Center Energy Improvement in Cloud Computing. *Computers and Electrical Engineering*, 42: 74–89.

Ferreto, T. C., Netto, M. A. S., Calheiros, R. N., and C. A. F. De Rose. (2011). Server Consolidation with Migration Control for Virtualized Data Centers. *Future Generation Computer Systems*, 27: 1027-1034.

He, L., Zou, D., Zhang, Z., Chen, C., Jin, H., and Jarvis, S. A. (2014). Developing Resource Consolidation Frameworks for Moldable Virtual Machines in Clouds. *Future Generation Computer Systems*, 32: 69-81.

Ho, Y., Liu, P., and Wu, J. (2011), Server Consolidation Algorithms with Bounded Migration Cost and Performance Guarantees in Cloud Computing. *Proceedings of IEEE Utility and Cloud Computing*, 154–161.

Hsu, C., Slagter, K. D., Chen, S., and Chung, Y. (2014). Optimizing Energy Consumption with Task Consolidation in Clouds. *Information Sciences*, 258: 452-462.

Huang, H., Peng, R., Feng, Z., and Zhang, M. (2016). A Cloud Workflow Modeling Framework Using Extended Proclets. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 23(4): 216–234.

Khalid, U., Ghafoor, A., Irum, M., and Shibli, M. A. (2013), Cloud based Secure and Privacy Enhanced Authentication and Authorization Protocol. *Proceedings of Elsevier Knowledge Based and Intelligent Information and Engineering Systems - KES2013*, 22: 680–688.

Kumar, S., Manvi, S., and Shyam, G. K. (2014). Resource Management for Infrastructure as a Service (IaaS) in Cloud Computing: A Survey. *Journal of Network and Computer Applications*, 41: 424–440.

Lee, S. and Sahu, S. (2011). Efficient Server Consolidation considering Intra-Cluster Traffic. *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 2011)*.

Lin, J., Zha, L., and Xu, Z. (2013). Consolidation Cluster Systems for Data Centers in the Cloud Age: A Survey and Analysis. *Frontiers of Computer Science*, 7(1): 1–19.

Mastroianni, C., Meo, M., and Papuzzo, G. (2013). Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data centers. *IEEE Transactions on Cloud Computing*, 1(2): 215–228.

Mauch, V., Kunze M., and Hillenbrand, M. (2013). High Performance Cloud Computing. *Future Generation Computer Systems*, 29: 1408–1416.

Mazumdar, S. and Pranzo M. (2017). Power Efficient Server Consolidation for Cloud Data Center. *Future Generation Computer Systems*, 70: 4–16.

Murtazaev, A. and Oh, S. (2011). Sercon: Server Consolidation Algorithm using Live Migration of Virtual Machines for Green Computing. *IETE Technical Review*, 28(3): 212–231.

Perumal, B. and Murugaiyan, A. (2016). A Firefly Colony and Its Fuzzy Approach for Server Consolidation and Virtual Machine Placement in Cloud Data Centers. *Journal of Advances in Fuzzy Systems*, 2016: 1–15.

Rao, K. S. and Thilagam, P. S. (2015). Heuristics based Server Consolidation with Residual Resource. *Future Generation Computer Systems*, 50: 87–98.

Shen, D., Luo, J., Dong, F., Fei, X., Wang, W., Jin, G., and Li, W. (2015). Stochastic Modeling of Dynamic Right-Sizing for Energy-Efficiency in Cloud Data Centers. *Future Generation Computer Systems*, 48: 82–95.

Singh A. and Hemalatha M. (2013). Cluster based Bee Algorithm for Virtual Machine Placement in Cloud Data Center. *Journal Theoretical and Applied Information Technology*.

Speitkamp B. and Bichler M. (2010). A Mathematical Programming Approach for Server Consolidation Problem in Virtualized Data centers. *IEEE Transactions on Services Computing*, 3(4): 266–278.

Srikantaiah S., Kansal A., and Zhao F. (2008), Energy Aware Consolidation for Cloud Computing. *Proceedings of the Power Aware Computings and Systems (HotPower'08)*, San Diego California.

Ye, K., Jaing, X., Huang, D., Chen, J., and Wang, B. (2011), Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments. *Proceedings of IEEE Cloud Computing*, 267–274.

Zhang, S., Qian, Z., Luo, Z., Wu, J., and Lu, S. (2016). Burstiness-Aware Resource Reservation for Server Consolidation in Computing Clouds. *IEEE Transactions on Parallel and Distributed Systems*, 27(4): 964–997.