# Minimizing Biased VM Selection in Live VM Migration

Suhib Bani Melhem*, Anjali Agarwal*, Nishith Goel† and Marzia Zaman†
*Department of Electrical and Computer Engineering
Concordia University, Montreal, Canada
Emails: {s_banim, aagarwal} @encs.concordia.ca
†Cistech Limited, Ottawa, Canada
Emails: nishith@cistech.ca, Marzia@cistel.com

***Abstract*— VM selection algorithm selects one or more VMs from the full set of VMs running on a given overload host, once a decision to migrate VMs from that host is made to achieve host/server consolidation and load balancing in cloud data centers while satisfying the QoS constraints. Presently, VM selection is a crucial decision for resource management in the cloud data center management, specially with high dynamic environment. In this paper, we propose two new VM selection algorithms, namely Minimum VM Migrated Count and Minimum migration time Minimum VM Migrated Count to avoid frequent SLA violation on the same VM. We propose new metrics to compare with other VM selection algorithms. We evaluate our proposed algorithms through CloudSim simulation on different types of PlanetLab real and random workloads. The experimental results demonstrate that the proposed algorithms show significant reduction in the Maximum number of VM migrated count and the degree of load balancing of VMs migrated count with the other state of the art algorithms.**

***Keywords*— Live Migration; VM selection; Virtual machine consolidation; Data center; CloudSim.**

## I. INTRODUCTION

Cloud computing is based on the concept of virtualization. Virtualization plays a vital role in managing and organizing access to the resource pool via a software layer called virtual machine monitor (VMM) or hypervisor. It hides the details of the physical resources and provides virtualized resources for high-level applications. Besides, it virtualizes all of the resources of a given host allowing several VMs to share its resources [1]. Xen [2], Microsoft Hyper-V [3], and KVM [4], are popular virtualization software. Virtualization also allows gathering several VMs into a single host using a technique called VM consolidation. Another capability provided by virtualization is live migration, which is the ability to transfer a VM between hosts with a close to zero downtime.

Dynamic live VM migration in the scope of resource management is becoming a crucial issue to emphasize on optimal resource utilization, maximum throughput, minimum response time, enhancing scalability, avoiding over-provisioning of resources and prevention of overload to make cloud computing successful. Thus, the performance of applications in large virtualized data centers is highly dependent on data center architecture and smooth network communication among VMs, while minimizing the communication burden to avoid congestion, latency, etc. The communication cost of a network can be reduced by minimizing the VMs migration between hosts. Therefore, clients and service providers need to build a cloud computing infrastructure that minimizes not only operational costs but also total network load. The key aspect that is directly related to network resource management in the data center is how to minimize network overhead and load balancing-VM migration, which is still an active area of research. Intelligent host underload/overload detection, VM selection, and VM placement are the primary means to address live VM migration issue.

This paper focuses on the second problem that considerably influences the VM migration process. The VM migration process not only makes the VM unavailable for a certain amount of time but also consumes the network and CPU resources from both source and destination hosts. The performance of other VMs that are running on source and destination hosts are also affected due to increased resource requirements during the VM migration process. The VM selection starts after a decision of a given host is considered overloaded. Then a particular VM selection algorithm should be applied to select one or more VMs to migrate from this host to other hosts.

The challenges in the VM selection are to reduce the power consumption, minimize SLA violation and to avoid performance degradation. However, some solutions based on the last observed utilization for decision making may cause unnecessary migrations, thus increasing the overhead: the energy for VM migration, the performance degradation of the hosted applications, and the extra traffic [5, 6].

In this paper, we propose two VM selection algorithms termed as Minimum VM Migrated Count (MiMc) and Minimum migration time Minimum VM Migrated Count (MmtMiMc).

This paper starts by introducing related works in Section II. Section III explains our proposed VM selection algorithms. The system model is discussed in detail in Section IV. Section V presents our experimental setup and performance metrics. In section VI experimental results are analyzed.

978-1-5386-1115-9/17/$31.00 ©2017 IEEE

Section VII shows the concluding remarks and future directions.

## II. Related Work

Over the last two decades, there has been major significant research in data center resource management and allocation during VM migration. Many overload/underload detection algorithms have been proposed to generate optimal computing resource utilization and energy consumption reduction along data center.

Authors in [7][8] proposed three policies. The first approach is called Minimization of Migrations (MM); in this approach, the minimum number of VMs is moved to underload hosts to reduce migration overhead. Descending VMs CPU utilization ordering step is implemented as the first step in this algorithm, after that a repeated scanning for the ordered list is performed to find the best candidate VMs to be migrated. The second algorithm is Highest Potential Growth (HPG). This policy migrates VMs, which have relatively the lowest value of CPU usage to reduce the total likelihood increase of the utilization and SLA violation. The third algorithm is Random Choice (RC) that chooses a VM to be moved according to a uniformly distributed discrete random variable whose values index a set of VMs allocated to a host.

Authors in [9] proposed two different algorithms. The first algorithm is Minimum Migration Time (MMT). In this method, a VM is chosen based on the value of the migration time, the less the better. Migration time can be easily computed as the amount of RAM utilized by the VM divided by the additional network bandwidth available for the host. The second algorithm is the Maximum Correlation (MC). In this approach, a correlation value is calculated. Whenever the value of the correlation between the resource usage by applications running on an oversubscribed host increases then the likelihood of overloading will be higher. So, the selection of the VMs to be migrated is based on the correlation of the CPU utilization with other VMs, where the highest correlation value is selected. To assess the prediction quality of the dependent variable the multiple correlation coefficient is used in multiple regression analysis.

Authors in [10] proposed two algorithms. The first algorithm is called the Median Migration Time (MedianMT). This method selects a given VM that requires the median time to complete a migration relatively to the other VMs allocated to the host. The second algorithm is the Maximum Utilisation (MaxU) that selects a VM to migrate from the overutilized host based on the largest possible usage of CPU that can be expected to minimize the number of migrations.

In literature, many more algorithms for the VM Selection have been proposed [11-14]. The existing VM selection algorithms focused on minimizing SLA violation on all the system and they ignore the frequent violation for the same VM, where a certain VM might be selected frequently to migrate from its overloaded host to another host based on the VM selection policies. In this paper, MiMc and MmtMiMc are proposed to avoid biased VM selection without maximizing the overall SLA violation.

## III. Proposed VM selection Policies

The process of migration not only makes the VM unavailable for a certain amount of time but also consumes the network and CPU resources from both source and destination hosts. This study proposes VM selection policies that resolve biased VM selection in live VM migration, resulting in a fair SLA violation on all the VMs while keeping the same percentage in the other metrics.

- Minimum VM Migrated Count (MiMc): The algorithm selects the VM to migrate from the host overloaded based on the minimum number of VM migrated count.
- Minimum Migration Time Minimum VM Migrated Count (MmtMiMc): The algorithm first selects VMs with the minimum amount of RAM to minimize the live migration time [10] and sorts them in increasing order. Then, out of the selected subset of VMs, the algorithm selects the VM with the minimum number of VM migrated count.

**Algorithm 1:** Minimum VM Migrated Count (MiMc) algorithm

```
Input: OverloadedHost.
Output: a VM to migrate.
min_migrated_count ← Max
selected_vm ← None
vmList ← OverloadedHost.getVmList()
foreach vm in vmList do
    migrated_count = vm.getMigrated_count
    if migrated_count < min_migrated_coun then
        min_migrated_count ← migrated_count
        selected_vm ← vm
return selected_vm
```

**Algorithm 2:** Minimum Migration Time Minimum VM Migrated Count (MmtMiMc) algorithm

```
Input: OverloadedHost, vms_ram_values.
Output: a VM to migrate.
min_migrated_count ← Max
selected_vm ← None
vmList ← OverloadedHost.getVmList()
vmList.sortDecreasing_vms_ram_values()
For (int i = 0; i < 4; i++)
vmList2[i] ← vmList[i]
foreach vm in vmList2 do
    migrated_count = vm.getMigrated_count
    if migrated_count < min_migrated_coun then
        min_migrated_count ← migrated_count
        selected_vm ← vm
return selected_vm
```

We compare proposed algorithms, MiMc and MmtMiMc, with three state-of-the-art VM selection algorithms, namely MC, MMT, and MU [9][10]. Besides, we investigate the impact of the four well-known host detection policies on the proposed algorithm. These VM host detection algorithms include:

- Averaging threshold-based algorithm (*thr*) computes the mean of the n last CPU utilization values and compares it to the previously defined threshold. The algorithm detects underload state if the average of the n last CPU utilization measurements is lower than the specified threshold.
- Median Absolute Deviation (*mad*) specifies a lower threshold empirically, while the upper threshold is calculated using the median of the absolute deviation from the medians of the CPU usage data sets.
- InterQuartile Range (*iqr*) is another approach to determine the upper threshold, while the lower threshold is determined empirically as before.
- Local Robust Regression (*lrr*) compares the maximum migration time to an expected value and weights it before deciding of overloading in the host.

We used the same VM placement method as in [13]. The VM allocation algorithm selects the destination host to receive the migrated VM, which causes the least increase in the power consumption. The algorithm relies on the traditional greedy algorithm to optimize the allocation of VMs.

## IV. SYSTEM MODEL

The target system is an IaaS environment, represented by a large-scale data center. The data center consists of a maximum of J heterogeneous hosts where each host contains multiple VMs. Multiple VMs can be allocated to each host through Virtual Machine Monitor (VMM). Besides, each host

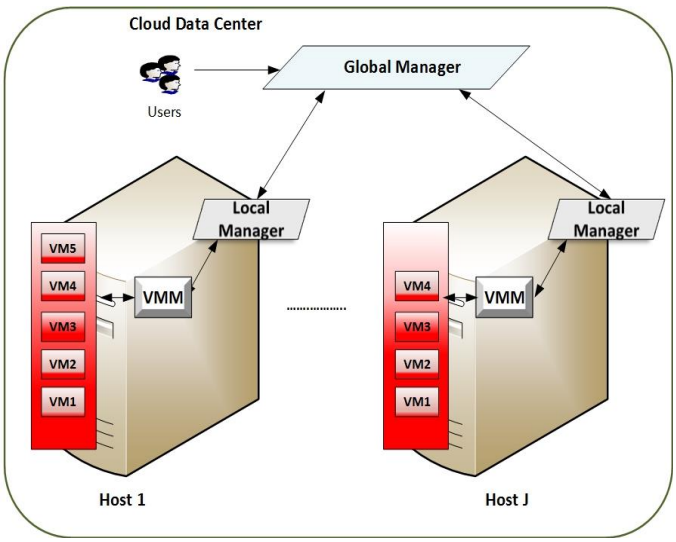


**Figure 1: System Model.**

**Table 1: Characteristics of the workload data (CPU utilization).**

| Workload Type | Date | Host | VMs | Mean (%) | SD (%) |
|---|---|---|---|---|---|
| Real (PlanetLab) | 03/03/2011 | 800 | 1052 | 12.31 | 17.09 |
| | 22/03/2011 | 800 | 1516 | 9.26 | 12.78 |
| | 03/04/2011 | 800 | 1463 | 12.39 | 16.55 |
| | 20/04/2011 | 800 | 1033 | 10.43 | 15.21 |
| Random | ---------- | 50 | 50 | ------- | ------- |

and VM are characterized by the CPU performance metrics defined in terms of Millions Instructions Per Second (MIPS), the amount of RAM and network bandwidth. The target system model is depicted in Figure 1 [15].

As shown in the Figure 1, the system model consists of global and local manager. Users submit their needs for provisioning of M heterogeneous VMs. The local managers, which are part of VM monitor (VMM), resides on each node and are responsible for keeping continuous monitoring of a node's CPU utilization, resizing the VM in accordance with their resource needs and making decision about when and which VMs have to be migrated from the node. The global manager resides on a master node and gathers information from the local managers to keep the check of the general view of the utilization of resources. The global manager gives commands for the optimization of the VM placement. VMMs do actual resizing, migration of VMs and changes in power states of the nodes.

## V. EXPERIMENTAL SETUP

To evaluate the efficiency of our algorithms with the existing algorithm, we have used the same experiment setup as used in [15] with some different workload. A data center has been simulated having J heterogeneous physical hosts and N virtual machines. The value of J and N depends on the type of workload which is specified in Table 1. In each workload, half of hosts are HP ProLiant ML110 G4 servers 1,860 MIPS each core, and the other half consists of HP ProLiant ML110 G5 servers with 2,660 MIPS each core. Depending on the CPU and memory capacity four types of single-core VMs are used: High-CPU Medium Instance: 2500 MIPS, 0.85 GB; Extra Large Instance: 2000 MIPS, 3.75 GB; Small Instance: 1000 MIPS, 1.7 GB and Micro Instance: 500 MIPS, 0.633 GB. The characteristics of these VM types are similar to Amazon EC2 instance types.

To make the simulation based evaluation applicable, we evaluate the proposed VM selection approaches on random workload and three real-world [15] publicly available.

To compare the performance of our proposed algorithms with the existing algorithms we have considered five metrics. Three of them are previously defined in the literature, which are SLA violation, total energy consumption by the physical resources for executing variable workloads, and total number of VM migrations occurred either for hotspot mitigation or for VM consolidation. In this paper we propose two new metrics, which are the maximum number of VM migrated count and the degree of load balancing of VMs migrated count. All of the five metrics are precisely defined below:

- **Maximum number of VM migrated count:** higher number of VM migrated count increases violation on the VM, and results in performance degradation. Following equation can be used to calculate the Maximum number of VM migrated count during a given time interval.

$$migrated\ count(P, t1, t2) = Max(\int_{t1}^{t2} Mig_{VM1}(P), \int_{t1}^{t2} Mig_{VM2}(P), \dots, \int_{t1}^{t2} Mig_{VMn}(P)) \quad (1)$$

where $P$ represents the current placements of VM, $Mig_{VMn}(P)$ shows the number of migration of VM $n$ between time intervals $t_1 and\ t_2$ for the placement $P$.

- **Degree of load balancing of VMs migrated count:** a lower number of degree of load balancing reduces biased selection among VMs, resulting in a fair SLA violation on all the VMs. Degree of load balancing is calculated by the variance of the VMs migrated count as indicated in the following equation:

$$Degree\ of\ load\ balancing = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} (m_i - \bar{m})^2} \quad (2)$$

$$\bar{m} = \frac{1}{N} * \sum_{i=1}^{N} m_i \quad (3)$$

where $m_i$ represents migrated count of $VM_i$, N is the number of VMs, and $\bar{m}$ is average VM migrated count as calculated using equation (3).

- **SLA Violation:** It is defined that when the request for the CPU performance exceeds the available capacity, a violation of the SLA established between the resource provider and the customer occurs. SLA violation is calculated as shown in equation (4) [9]:

$$SLA\ Violations\ (SLAV) = \frac{1}{J} \sum_{x=1}^{J} \frac{T_{sx}}{T_{ax}} * \frac{1}{N} \sum_{i=1}^{N} \frac{Cd_i}{Cr_i} \quad (4)$$

where $J$ is number of hosts, $T_{sx}$ the total time that utilization of host x reach to 100%, and $T_{ax}$ is lifetime (total time that host is active) of host $x$. When host utilization reaches 100%, the applications performance is bounded by the host. N shows number of VMs, $Cd_i$ estimated as 10% CPU utilization of $VM_i$ in all migrations. $Cr_i$ is total CPU requested by $VM_i$ .

- **Number of VM migrations:** a higher number of VM migrations increases the network load, and results in performance degradation. Following equation can be used to calculate the number of migrations during a given time interval [16].

$$Migrations(P, t1, t2) = \sum_{j=1}^{J} \int_{t1}^{t2} Mig_j(P) \quad (5)$$

where $P$ represents the current placements of VMs, $J$ is the number of hosts, $Mig_j(P)$ shows the number of migration of Host $j$ between time intervals $t_1 and\ t_2$ for the placement $P$.

- **Energy Consumption:** In order to measure the power consumption of a given server at a time t with placement P, we can use following equation [17, 18].

$$E_x(P, t) = k * E_{max} + (1 - k) * E_{max} * U_x(P, t) \quad (6)$$

where $E_{max}$ is the power consumption of the server at 100% utilization, $k$ is the static power coefficient that is equal to the amount of power consumption by an idle processor. According to [17], an idle processor consumes 70% of the power consumed when its utilization is 100%. Therefore, in our experiments, $k$ is set to 70%. In this model, $U_x(\mathrm{P}, \mathrm{t})$ is the current CPU utilization of a server at time t, which has a linear relationship with the power consumption.

Total energy consumption of all the servers between time t1 and t2, can be calculated using equation (7).

$$Energy(P, t_1, t_2) = \sum_{x=1}^{J} \int_{t1}^{t2} E_x(P, t) \quad (7)$$

## VI. Experimental Results

We selected five performance metrics to compare the proposed algorithms with the existing algorithms, which are SLA violation, total energy consumption, the total number of VM migrations, and the newly proposed Maximum number of VM migrated count and Degree of load balancing of VMs migrated count. We compare with VM selection algorithms presented in [7-9] including MC, MMT, and MU among four well-known host detection algorithms in [8, 9, 15] including *iqr, lrr, mad, and thr*. The main goal of these experiments is to substantiate the threshold adaptability in hypothesis by evaluating the performance of the proposed algorithm across four workloads. The four workloads include three real workloads (20110303, 20110322 and 20110403) that traces from more than a thousand PlanetLab servers and one random workload.

From the simulation results depicted in Figure 2 and Figure 3, it is completely obvious that the proposed algorithms significantly outperform the other algorithms in terms of Maximum number of VM migrated count and Degree of load balancing of VMs migrated count for all the real workload

traces among four host detection policies.

Figure 2 shows that MiMc VM selection algorithm reduces Maximum number of VM migrated count metric up to 41.03%, 68.92%, and 66.19% as compared to VM selection policies *mc*, *mmt*, and *mu* respectively when the work load is 20110303. There is almost the same reduction in the 20110322 and 20110403 workloads. Figure 2 also shows that MmtMiMc VM selection algorithm reduces Maximum number of VM migrated count metric up to 12.50%, 52.03%, and 52.52% as compared to VM selection policies *mc*, *mmt*, and *mu* respectively when the work load is 20110303. There is almost the same reduction in the 20110322 and 20110403 workloads.

Figure 3 shows that the Degree of load balancing of VMs migrated count metric is reduced up to 4.63%, 46.48%, and 38.62% for MiMc as compared to the VM selection policies mc, mmt, and mu respectively when the work load is 20110303. It should also be noted that almost the same reduction is obtained in the 20110322 and 20110403 workloads. Figure 3 also shows that for the proposed MmtMiMc algorithm, the Degree of load balancing of VMs migrated count metric is reduced up to 25.85% and14.96% for VM selection policies mmt, and mu respectively when the work load is 20110303. It should also be noted that almost the same reduction is obtained in the 20110322 and 20110403 workloads.

Figure 4 compares the SLA violation of the two proposed algorithms with the ones in the literature. The proposed MmtMiMc algorithm reduces SLA violation up to 32.14%, 25.85%, and 14.96% for mc, mu, and MiMc respectively when the work load is 20110303. But mmt algorithm still outperforms as the best among the others.

Figure 5 shows that the proposed algorithms MiMc and MmtMiMc outperform *mmt* and *mu*, and are similar to the *mc* algorithm in terms of number of VM migration. The proposed MiMc and MmtMiMc VM selection algorithms reduce number of VM migrations metric up to 14.42%, and 20.91% as compared to the VM selection policies *mmt*, and *mu* respectively when the work load is 20110303. There is almost the same reduction in the 20110322 and 20110403 workloads.

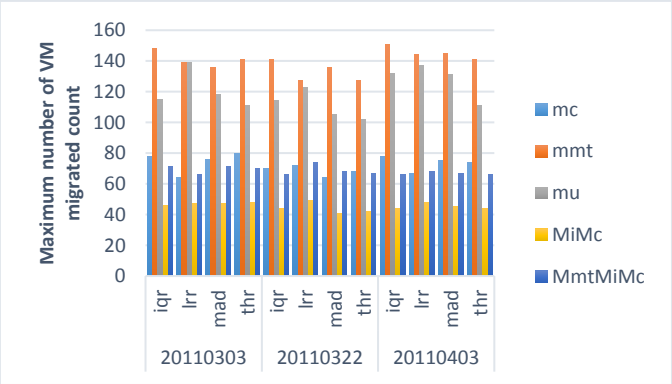


**Figure 2: Maximum number of VM migrated count for real workload traces.**

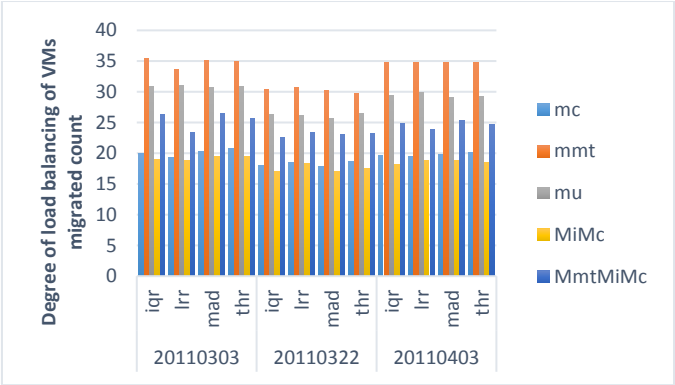


**Figure 3: Degree of load balancing of VMs migrated count for real workload traces.**

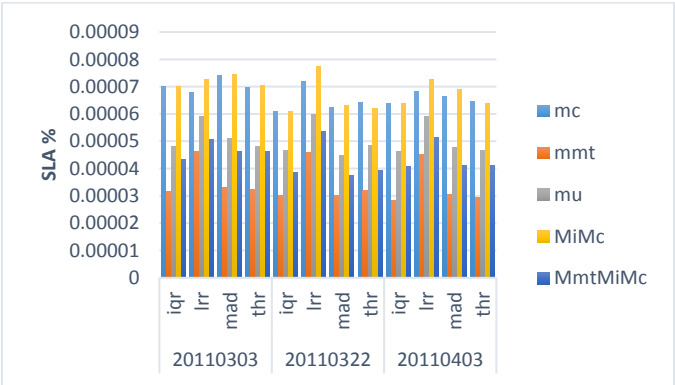


**Figure 4: SLA violation for real workload traces.**

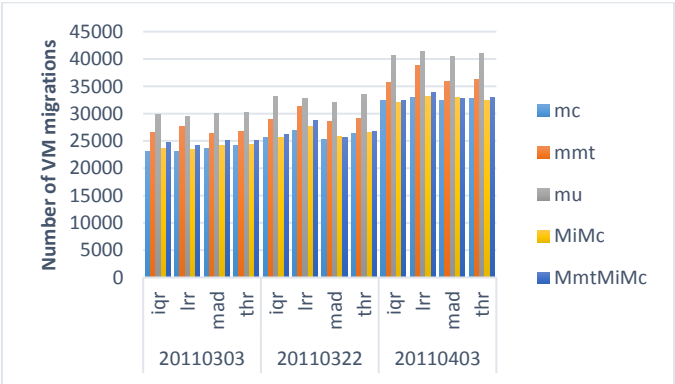


**Figure 5: Number of VM migrations for real workload traces.**

It can be seen from Figure 6 that the proposed algorithms are slightly better than mmt, and mu and similar to mc in terms of the energy consumption.

From the simulation results depicted in Figure 7 and Figure 8, it is completely obvious that the proposed algorithms significantly outperform the other algorithms in terms of Maximum number of VM migrated count and Degree of load balancing of VMs migrated count for the random workload.

The proposed MiMc VM selection algorithm reduces Maximum number of VM migrated count metric up to 25.41%, 52.11%, and 38.75% as compared to VM selection

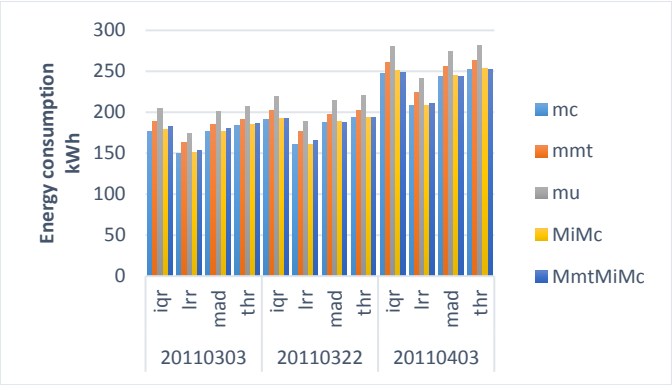


**Figure 6: Energy consumption for real workload traces.**

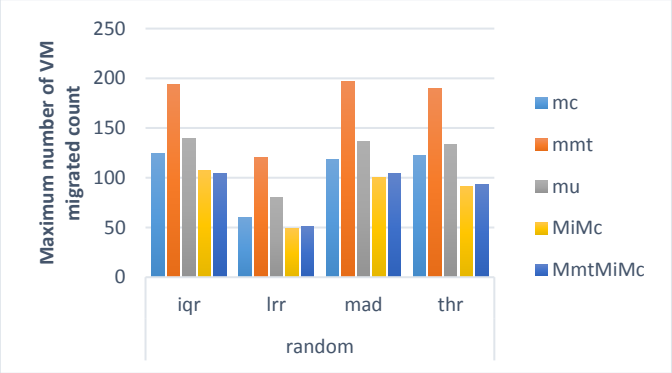


**Figure 7: Maximum number of VM migrated count for a random**

policies *mc*, *mmt*, and *mu*, respectively. It reduces Degree of load balancing of VMs migrated count metric up to 89.74%, 97.44%, and 90.67% as compared to VM selection policies *mc*, *mmt*, and *mu*, respectively.

From the simulation results depicted in Figure 7 and Figure 8, it is completely obvious that the proposed algorithms significantly outperform the other algorithms in terms of Maximum number of VM migrated count and Degree of load balancing of VMs migrated count for the random workload. The proposed MiMc VM selection algorithm reduces Maximum number of VM migrated count metric up to 25.41%, 52.11%, and 38.75% as compared to VM selection policies *mc*, *mmt*, and *mu*, respectively. It reduces Degree of load balancing of VMs migrated count metric up to 89.74%, 97.44%, and 90.67% as compared to VM selection policies *mc*, *mmt*, and *mu*, respectively.

Figure 7 and Figure 8 show that MmtMiMc VM selection algorithm reduces Maximum number of VM migrated count metric up to 23.77%, 57.50%, and 36.25% as compared to VM selection policies mc, mmt, and mu, respectively. It reduces Degree of load balancing of VMs migrated count metric up to 87.68%, 96.91%, and 88.78% as compared to VM selection policies mc, mmt, and mu, respectively.

It can be seen from Figure 9 that the proposed algorithms are similar to the *mc*, and *mu* in terms of SLA violation. It should be noted that the performance of the proposed algorithms is not much worse than that of *mmt* algorithm.

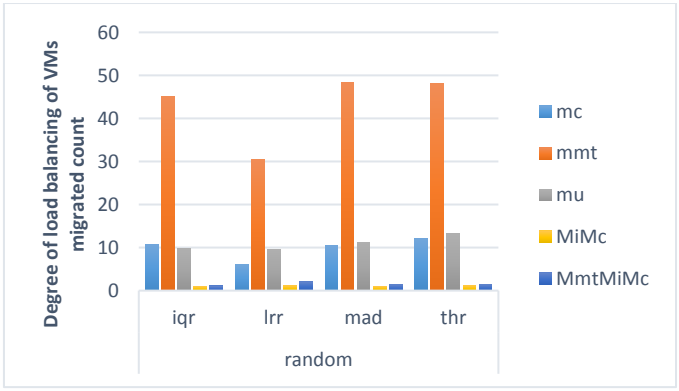


**Figure 8: Degree of load balancing of VMs migrated count for a random workload trace.**


**Figure 9: SLA violation for a random workload trace**

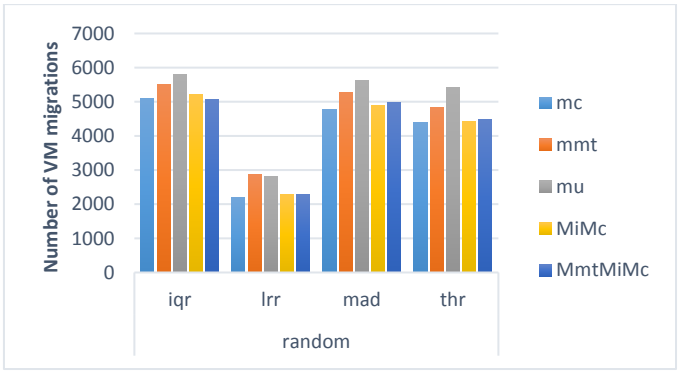


**Figure 10: Number of VM migrations for a random workload trace**

Figure 10 shows that the proposed algorithms MiMc and MmtMiMc outperform *mmt* and *mu*, and are similar to the *mc* algorithm in terms of number of VM migration. MiMc and MmtMiMc algorithms reduce number of VM migrations metric up to 8.35%, and 17.93% for VM selection policies: *mmt*, and *mu* respectively.

It can be seen from Figure 11 that the proposed algorithms are similar to the *mc*, *mmt* and *mu* in terms of the energy consumption for a random workload trace.

It should be noted that the improvements in both Maximum number of VM migrated count and Degree of load balancing of VMs migrated count have direct impact on the performance degradation for each VM in the cloud data


**Figure 11: Energy consumption for a random workload trace.**

center. These improvements have impact on avoiding VM migration frequently resulting in a fair SLA violation on all the VMs.

## VII. Conclusion

We present Minimum VM Migrated Count (MiMc) and Minimum migration time Minimum VM Migrated Count (MmtMiMc) algorithms that resolve biased VM selection in live VM migration. The proposed algorithms avoid frequent SLA violation on the same VM in cloud data center by selecting the VM to migrate from the overloaded host based on VM migrated count. The proposed algorithms determine which VMs will be selected to migrate from the overloaded host to underloaded host to achieve server consolidation and load balancing. The experimental results show that the proposed algorithms can minimize maximum number of VM migrated count and degree of load balancing of VMs migrated count significantly compared to the most commonly used MC, MMT and MU algorithms, resulting in a fair SLA violation on all the VMs, while keeping the same percentage in the other defined metrics.

The existing VM selection approaches focused on minimizing the number of VM migrations, reducing performance degradation, reducing the number of physical machines, and the data center energy consumption. It should be noted that no proactive criteria exist for live WAN migration that minimizes the number of the IP reconfigurations. It is known that if the time needed for IP reconfiguration for all migrated VM users increases, then there will be an increase in the interruption of service, network overhead and performance degradation.

## Acknowledgement

The authors would like to acknowledge the financial support provided by Natural Sciences and Engineering Research Council (NSERC) in collaboration with Cistech Limited, Canada.

## References

[1] T. Veni and S. Bhanu. “A survey on dynamic energy management at virtualization level in cloud data centers,” Computer Science & Information Technology, pp. 107-117, 2013.

[2] Xen, “Xen Hypervisor,” (2017), [online]. Available: http://www.xenproject.org/developers/teams/hypervisor.html [Accessed: June 10, 2017].

[3] Microsoft, “Hyper-V Server,” (2017), [online]. Available: https://technet.microsoft.com/library/hh831531.aspx [Accessed: June 10, 2017].

[4] Linux, “Kernel-based Virtual Machine (KVM),” (2017), [online]. Available: http://www.linux-kvm.org/page/Main_Page [Accessed: June 10, 2017].

[5] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, and C. A. F. De Rose, “Server consolidation with migration control for virtualized data centers,” Elsevier Future Generation Computer System, vol. 27, pp. 1027–1034, 2011.

[6] I. Takouna, E. Alzaghoul, and C. Meinel, “Robust virtual machine consolidation for efficient energy and performance in virtualized data centers,” IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing (CPSCom), pp. 470-477, 2014.

[7] A. Beloglazov and R. Buyya. “Energy efficient allocation of virtual machines in cloud data centers,” 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), pp. 577-578, 2010.

[8] A. Beloglazov, J. Abawajy and R. Buyya. “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” Elsevier Future Generation Computer Systems, vol. 28, no. 5, pp. 755-768, 2012.

[9] A. Beloglazov. “Energy-efficient management of virtual machines in data centers for cloud computing,” Ph.D. thesis, The University of Melbourne, 2013.

[10] A. Beloglazov and R. Buyya. “OpenStack neat: A framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds,” Concurrency and Computation: Practice and Experience, vol. 27, no. 5, pp. 1310-1333, 2015.

[11] S. Sohrabi and I. Moser “The effects of hotspot detection and virtual machine migration policies on energy consumption and service levels in the cloud,” Elsevier Procedia Computer Science, vol. 51, pp. 2794-2798, 2015.

[12] K. Shahzad, A. I. Umer and B. Nazir “Reduce VM migration in bandwidth oversubscribed cloud data centers,” IEEE 12th International Conference in Networking, Sensing and Control (ICNSC), pp. 140-145, 2015.

[13] M. A. H. Monil and R. M. Rahman “Implementation of modified overload detection technique with VM selection strategies based on heuristics and migration control,” IEEE/ACIS 14th International Conference in Computer and Information Science (ICIS), pp. 223-227, 2015.

[14] M. Al-Ayyoub, Y. Jararweh, M. Daraghmeh and Q. Althebyan “Multiagent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure,” Springer Cluster Computing, vol. 18, no. 2, pp. 919-932, 2015.

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

[16] S. Mustafa, B. Nazir, A. Hayat, and S. A. Madani “Resource management in cloud computing: Taxonomy, prospects, and challenges,” Computers & Electrical Engineering, vol. 47, pp. 186-203, 2015.

[17] A. Beloglazov and R. Buyya “Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers,” Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, p. 4, 2010.

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