An Efficient VM Selection Strategy for Minimization of Migration in Cloud Environment

Nimisha Patel[1,2](✉) and Hiren Patel[3]

[1] Rai University, Ahmedabad, Gujarat, India
nimishaa_25@yahoo.co.in
[2] Sankalchand Patel College of Engineering, Visnagar, Gujarat, India
[3] LDRP Institute of Technology and Research, Gandhinagar, Gujarat, India
hbpatel1976@gmail.com

Abstract. Cloud Computing has been one of the most emphasized paradigms over the last few years. Increased usage of Cloud Computing has resulted into the augmentation of energy consumption and emission of carbon footprints in the environment. Many researchers have been working in the different directions to address these issues. Out of various facets, efficient allocation of Virtual Machines (VMs) on hosts could be one of the good paths to save energy of data center. Optimized VM allocation process is divided into two phases viz. (i) selection of VMs to be migrated and (ii) placement of VMs on the new host. During the selection phase, minimizing the number of VMs to be migrated would result into improvement in performance and reduction in SLA violation. In this research, we have proposed a modification in an existing Minimization of Migration algorithm. The existing algorithm works for two scenarios viz. (a) single VM selection and (b) multiple VM selections. We find the scope of enhancement in the existing algorithm, especially in the case of multiple VM selection. In such scenario, the existing algorithm selects a combination of VMs which is not the optimum. We propose our algorithm to optimally select the combination of VMs such that number of VMs to be migrated remains minimal and utilization of host, after migration, reaches nearer (and below) to an upper threshold value. The prospect of this research would to enhance utilization of hosts which would result in a reduction in a number of live hosts resulting in saving in energy consumption.

Keywords: Energy efficiency · VM migration · Cloud computing Consolidation

1 Introduction

Cloud computing (CC) has been a novel paradigm in the field of Information and Communication Technology which offers a different style of computing where users do not own their computing resources but rent and pay for them. NIST [1] defines it as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and

A. V. Deshpande et al. (Eds.): SmartCom 2017, CCIS 876, pp. 92–101, 2018.
https://doi.org/10.1007/978-981-13-1423-0_11

services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." Hence, resources are provisioned/released to/from the user on demand over the network and payment is made accordingly. This new paradigm has become very popular in recent times.

Out of various challenges faced by Cloud such as security, availability, performance etc., the issue of energy consumption has attracted the attention of many researchers. The issue is being addressed in many facets such as VM Consolidation, Load balancing, VM Migration etc. In this research, we aim to address the issue of energy consumption by optimally selecting (minimum number of) VMs to be migrated from one host to another in such a way that a minimum number of the host are to be kept on, resulting in a reduction in electrical power. Authors of [2] have addressed the issue of minimizing migration but it works fine for the cases where only one VM is to be migrated to consolidate the system. But, in real situation, where a number of hosts in a data center is huge and the number of VMs in a host is large, we need to migrate multiple VMs to stabilize the system. In such scenario, the existing method does not give an optimum solution. Hence, we find the scope of improvement in the existing method and we propose a modified version of minimization of migration algorithm to handle the cases of multiple VM selection.

Proper utilization of hosts leads into keeping an optimum number of hosts alive and rest to be idle resulting in saving in power consumption. VM consolidation is a process of appropriately distributing tasks on VMs and mapping or migrating VMs on or across hosts. The process of VM consolidation is divided among four phases [3] viz. (a) Overloaded host detection (b) VM selection from overloaded host (c) VM placement for selected VMs (d) Underloaded host detection. In this research, we propose a modified minimization of migration algorithm for VM selection (as mentioned in b) for optimally selecting the combination of VMs to be migrated. The Efficient method to select the best combination of VMs would result into significant improvement in power consumption.

The overall paper has been organized as follows. Section 2 discusses related work, followed by our proposed scheme in Sect. 3. Experimentations and results are described in Sect. 4. The overall research has been concluded in Sect. 5 followed by a list of references.

2 Related Work

Anton et al. [2] have been one of the few researchers who worked in the domain of achieving energy efficiency by proper resource allocation. Along with defining an architectural framework and survey of research in energy-efficient computing, the authors have proposed energy-efficient resource allocation policies and scheduling algorithms and claimed significant cost saving and improvement of energy efficiency. Minimization of Migrations (MM) is one of the key contributions by the authors in the direction of selecting a combination of VMs from an overloaded host for subsequent migration. The algorithm covers two aspects of migration viz. (a) single VM selection and (b) multiple VMs selection. The algorithm works efficiently for (a) but there seems to be a scope of improvement in (b) which is one of the main sources of motivation for this research. In [3], authors identified dynamic consolidation of VMs using live migration and switching idle nodes to sleep mode, as the mean to optimize resource usage and reduce energy consumption. Further, the authors propose heuristics for dynamic consolidation of VMs to claim a reduction in energy consumption and maintenance of SLA. Different statistical methods such as (a) Median Absolute Deviation (MAD) (b) Interquartile Range (IQR) (c) Local Regression (LR) and (d) Robust Local Regression (LRR), have been used for detection of the overloaded host. Various methods for VM selection such as (a) Minimum Migration Time (MMT) (b) Random Choice (RC) and (c) Maximum Correlation (MC) have been discussed. Power Aware Best Fit Decreasing (PABFD) policy has been proposed for VM placement. A research paper [4], aimed to maximize utilization and minimize cost by proper management of resources and allocation strategies. Authors proposed performance analysis based resource allocation scheme which follows the best fit strategy for efficient VM allocation. The execution time of CPU and performance of memory are considered as two important factors with a flexibility of assigning weight to resource. Subsequently, authors claimed improvement in resource utilization without compromising the allocation time. In the survey paper, [5] identified resource utilization and energy consumption as two important factors to be considered in the process of VM consolidation. To address the factors, authors have suggested having intelligent workload placement and relocation techniques. Authors have exhaustively described terminology involved such as virtualization, VM migration, VM consolidation (static, dynamic) etc. Authors of [6] proposed a QoS-aware VM consolidation approach (based on resource utilization history of VMs) to improve QoS metrics and energy consumption. Initially, the proposed algorithm detects an overloaded host. Then, VMs are selected for migration from these overloaded hosts. Next, detect underloaded hosts and select all VMs from them for migration. At the end, a new placement is searched for VMs to be migrated. Authors claimed a reduction in energy consumption and SLA violation.

3 Proposed Algorithm: Modified Minimization of Migration

The Minimization of Migration (MM) [2] policy selects the minimum number of VMs to migrate from overloaded hosts to reduce the CPU utilization under the upper utilization threshold. The algorithm tries to selects best possible VM combination that satisfies two conditions viz. (a) keep the number of VMs to be migrated as minimum as possible (b) keep the host utilization under and nearer to the upper threshold, after migration. The algorithm works well where the resultant VM is single. But, there are cases where single VM selection is not sufficient for bringing host utilization under the upper threshold. In such scenarios, MM algorithm selects one VM (by default) with the highest utilization and proceeds to make best possible combination by selecting next VM(s) for bringing the host utilization under the upper threshold. The main issue with the MM algorithm is that it does not provide the optimum combination of VMs which brings the host utilization nearer to the upper threshold. To address this issue, for the cases of multiple VM selection, we have modified existing MM algorithm to maximize utilization of host while keeping the number of VMs to be migrated minimal. We start the selection procedure for a number of VM to be migrated equal to 2 and check whether host utilization (after migration) goes below upper threshold or not. If not, the selection of VM proceeds for all nCr combinations incrementally, where *n* is total number of VMs in a host and *r* ranges from 2 to n.

To understand the scenario and to support our claim, we have taken following example values. It is worth noting here that we have taken the example of the scenario where more than one VMs are to be selected for migration. Current Host Utilization *hUtil* = 85%, Upper Threshold *THRESH_UP* = 70% (0.7), Total number of VMs in a host = 4, Utilization of all VM in the host, *vmUtil* = {13, 11, 9, 5} (sorted in decreasing order based on utilization).

Existing Method (MM) [2]: MM selects first VM (by default) irrespective of its utilization. That means, in this case, first VM with utilization = 13 is selected. Then, the method checks for all other VMs where the difference between (a) and (b) remains minimum and resulted in utilization under to THRESH_UP, where (a) hUtil - THRESH_UP = 85 − 70 = 15 (b) Addition of utilization of all VMs selected. So for the example values given, a combination of {13, 5} shall be selected as the total utilization is 18 where the difference is 3 [(13 + 5) − (85 − 70)] which is minimum and resultant utilization will nearer and under THRESH_UP [85 − (13 + 5) = 67, 67 < 70].

Our Proposal (Modified Minimization of Migration – M3): In our proposal, unlike existing method, we do not select the first VM (by default). On the contrary, we look for all the possible combinations of VMs ranging from 2 to n (until we achieve host utilization below THRESH_UP). In above example, possible combinations are as under. For VM = 2, VM combinations are {13, 11}, {13, 9}, {13, 5}, {11, 9}, {11, 5}, {9, 5}, For VM = 3, VM combinations are {13, 11, 9}, {13, 11, 5}, {11, 9, 5}, For VM = n = 4, VM combination is {13, 11, 9, 5}.

In our proposal, the algorithm selects a pair {11, 5} because while doing so difference between (a) and (b) remains minimum and nearer to THRESH_UP. (a) hUtil - THRESH_UP = 85 − 70 = 15 (b) Addition of utilization of all VMs selected (11 + 5 = 16). So, for this value, the difference is 1 (against 3 in existing method) and hUtil will be 69 (against 67 in existing method). For these case example, as we have identified a pair of VM from the first option only, hence the algorithm would not go for checking other combinations with VM = 3 or VM = 4.

Algorithm: Modified Minimization of Migration (M3)

```
Input: hostList Output: migrationList
for each h in hostList do
        vmList ← h.getVmList()
        vmList.sortDecreasingUtilization()
        hUtil←h.getUtil()
        bestFitUtil←MAX
        if hUtil – THRESH_UP > VM.getUtil() then goto Multiple_VM_Selection endif
Single_VM_Selection:
        while hUtil > THRESH_UP do
                for each vm in vmList do
                        if vm.getUtil() > hUtil − THRESH_UP then
                                t←vm.getUtil() − hUtil + THRESH_UP
                                if t < bestFitUtil then
                                        bestFitUtil←t
                                        bestFitVm←vm
                                endif
                        endif
                endfor
                hUtil←hUtil − bestFitVm.getUtil()
                migrationList.add(bestFitVm)
                vmList.remove(bestFitVm)
        endwhile
        goto Update
Multiple_VM_Selection:
        n ← h.getVmSize()
        for each total_Migration from 2 step 1 till n do
                bestFitUtil ← MAX
                bestFitCombination ← NULL
                for each Combination of VM starting from 2 step 1 till nCtotalMigration do
                    totalUtil ← 0
                    for each VM in Combination do
                            totalUtil = totalUtil + Combination.nextVm.getUtil()
                    endfor
                    if [totalUtil >= (hUtil – THRESH_UP)] AND [totalUtil <bestFitUtil] then
                            bestFitUtil ← totalUtil
                            bestFitCombination ← Combination
                    endif
                endfor
                if bestFitUtil != MAX then break endif
        endfor
        hUtil  ← hUtil – bestFitUtil
        migrationList.add(bestFitCombination.getAllVMs())
        vmList.remove(bestFitCombination.getAllVMs())
Update:
        if hUtil < THRESH_LOW then
                migrationList.add(h.getVmList())
                vmList.remove(h.getVmList())
        endif
endfor
return migrationList
```

Table 1. Data set

Host ID	Host type	Host capacity	VM ID	Core per VM	VM capacity (MIPS)	VM utilization required (%)	MIPS required by VM
1	HpProLiantMl110G5Xeon3075	Total Core = 2 MIPS per Core = 2660 Total Capacity = 5320	101	1	2500	30	750
			102	1	2000	25	500
			103	1	1000	25	250
			104	1	500	30	150
			105	1	1000	20	200
			106	1	2000	35	700
			107	1	1000	35	350
			108	1	2500	25	625
			109	1	500	40	200
2	HpProLiantMl110G4Xeon3040	Total Core = 2 MIPS per Core = 1860 Total Capacity = 3720	201	1	2500	24	600
			202	1	2000	21	420
			203	1	1000	18	180
			204	1	500	17	85
			205	1	1000	13	130
			206	1	2000	7	140
			207	1	1000	35	350
			208	1	2500	30	750
			209	1	500	22	110
			210	1	2000	30	600

(continued)

Table 1. (*continued*)

Host ID	Host type	Host capacity	VM ID	Core per VM	VM capacity (MIPS)	VM utilization required (%)	MIPS required by VM
3	HpProLiantMl110G5Xeon3075	Total Core = 2 MIPS per Core = 2660 Total Capacity = 5320	301	1	2500	50	1250
			302	1	2000	60	1200
			303	1	1000	15	150
			304	1	500	18	90
			305	1	1000	17	170
			306	1	2000	12	240
			307	1	1000	26	260
			308	1	2500	28	700
			309	1	500	31	155
4	HpProLiantMl110G4Xeon3040	Total Core = 2 MIPS per Core = 1860 Total Capacity = 3720	401	1	2500	26	650
			402	1	2000	39	780
			403	1	1000	29	290
			404	1	500	21	105
			405	1	1000	14	150
			406	1	2000	20	400
			407	1	1000	10	100
			408	1	2500	27	675
			409	1	500	27	135
			410	1	2000	17	340

4 Experimentation and Results

In this section, we discuss the analysis of existing Minimization of Migration algorithm along with our proposal, Modified Minimization of Migration. For our experimentation, we have taken dataset as mentioned in Table 1. We have simulated a data center consisting of 4 hosts with 9 to 10 VMs on each. Each host is modeled to have 2 CPU cores with MIPS capacity of either 2660 or 1860 per core. Each VM requires 1 CPU core with utilization requirement as mentioned in Table 1. The CloudSim toolkit [7] has been used as a simulation platform because it supports modeling of on-demand virtualization-enabled resource and application management with a varying workload.

For the described simulation setup, we have carried out series of experimentation and generated the results as mentioned in Table 2 and Fig. 1.

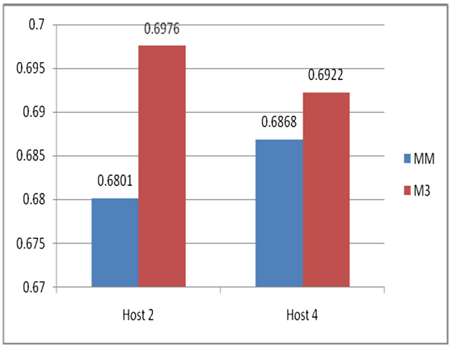

Fig. 1. Host utilization after migration

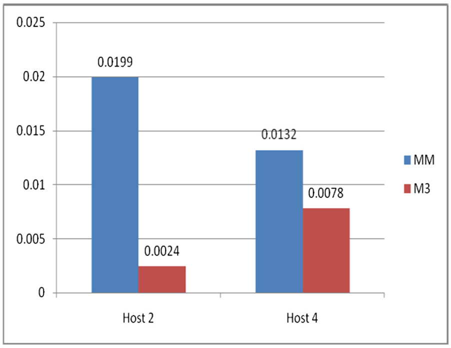

Fig. 2. Difference (threshold- host utilization after migration)

In our experimentation, the value of the upper threshold is taken as 0.7 which can be calculated dynamically using any of the available statistical methods such as Median Absolute Deviation, Interqurtile Range, Local regression etc. As can be seen from Table 2 that for the case of Host 1 and Host 3, there exists a solution with only one VM selection. So, for both these hosts, both the algorithm results in the same outcome. Hence, these results are not considered for comparison. But, for Host 2 and Host 4, the results are compared in Fig. 1. As shown in the figure, M3 gives resultant host utilization (after migration) nearer to the upper threshold value (here, 0.7). Hence we claim more optimized utilization of available hosts, without compromising the number of VMs to be migrated. Figure 2, the difference between the upper threshold and host utilization after migration has been shown. Lower the difference (nearer to the upper threshold) better the host utilization.

Further, to understand the impact of migration on power consumption, we analyzed the same empirically. Table 3 and Fig. 3 summarize the same. As can be seen from the table and figure, the resultant power gets reduced after migration in both the methods.

Table 2. Results

Host ID	Host utilization (before)	Algorithm	VM selection type	VM selected	Host utilization (after)	Difference (threshold-host utilization after migration)
Host 1	0.7002	MM	Single	104	0.6720	0.0280
		M3	Single	104	0.6720	0.0280
Host 2	0.9046	MM	Multiple	208, 204	0.6801	0.0199
		M3	Multiple	202, 207	0.6976	0.0024
Host 3	0.7923	MM	Single	308	0.6607	0.0393
		M3	Single	308	0.6607	0.0393
Host 4	0.9745	MM	Multiple	402, 403	0.6868	0.0132
		M3	Multiple	401, 406	0.6922	0.0078

Table 3. Power consumption

Host ID	Power consumption before migration (Watts)	Power consumption after migration (Watts)	
		MM	M3
Host 1	108.0075	107.4398	107.4398
Host 2	133.0914	124.2043	124.9032
Host 3	111.6917	107.2143	107.2143
Host 4	134.4892	124.4731	124.6882

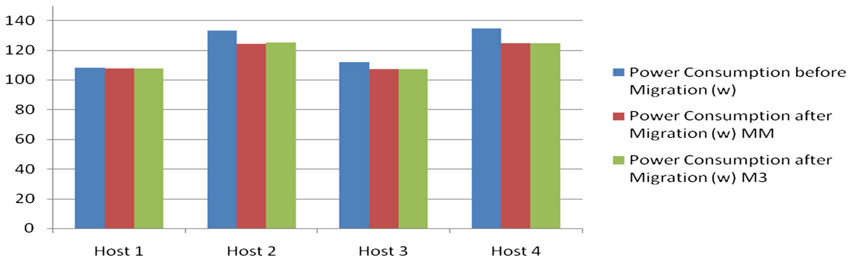

Fig. 3. Power consumption

But, there is a negligible increase in power consumed by M3 as compared to MM. So, we may conclude that without significant increase in power consumption, one may achieve higher utilization using M3.

5 Conclusion

The issues of energy consumption and carbon emission by Cloud data center have attracted the attention of many researchers in recent time. In this research, we have modified an existing algorithm which claimed to minimize the VM migration while keeping the hosts optimally utilized. Our contribution through this research is to identify the best combination of VMs to be migrated in such a way that (a) number of VMs to be migrated is minimizes and (b) host utilization after migration remains below and nearer to the upper threshold. Our results show that the proposed algorithm provides an optimum solution as compared to the existing method.

References

1. Mell, P., Grace, T.: The NIST definition of cloud computing (draft). NIST Special Publication, vol. 800, p. 145 (2011)
2. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comput. Syst. **28**(5), 755–768 (2012)
3. 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. Concurr. Comput.: Pract. Exp. **24**(13), 1397–1420 (2012)
4. Lee, H.M., Jeong, Y.S., Jang, H.J.: Performance analysis based resource allocation for green cloud computing. J. Supercomput. **69**(3), 1013–1026 (2014)
5. Ferdaus, M.H., Murshed, M.: Energy-aware virtual machine consolidation in IaaS cloud computing. In: Mahmood, Z. (ed.) Cloud Computing, pp. 179–208. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10530-7_8
6. Horri, A., Mozafari, M.S., Dastghaibyfard, G.: Novel resource allocation algorithms to performance and energy efficiency in cloud computing. J. Supercomput. **69**(3), 1445–1461 (2014)
7. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw.: Pract. Exp. **41**(1), 23–50 (2011)