Monil and Rahman *Journal of Cloud Computing: Advances, Systems and Applications* (2016) 5:8
DOI 10.1186/s13677-016-0059-7

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

VM consolidation approach based on heuristics, fuzzy logic, and migration control

Mohammad Alaul Haque Monil[1] and Rashedur M. Rahman[2*]

Abstract

To meet the increasing demand of computational power, at present IT service providers' should choose cloud based services for its flexibility, reliability and scalability. More and more datacenters are being built to cater customers' need. However, the datacenters consume large amounts of energy, and this draws negative attention. To address those issues, researchers propose energy efficient algorithms that can minimize energy consumption while keeping the quality of service (QoS) at a satisfactory level. Virtual Machine consolidation is one such technique to ensure energy-QoS balance. In this research, we explore fuzzy logic and heuristic based virtual machine consolidation approach to achieve energy-QoS balance. A Fuzzy VM selection method is proposed in this research. It selects VM from an overloaded host. Additionally, we incorporate migration control in Fuzzy VM selection method that will enhance the performance of the selection strategy. A new overload detection algorithm has also been proposed based on mean, median and standard deviation of utilization of VMs. We have used CloudSim toolkit to simulate our experiment and evaluate the performance of the proposed algorithm on real-world work load traces of Planet lab VMs. Simulation results demonstrate that the proposed method is most energy efficient compared to others.

Keywords: Cloud, Datacenter, Dynamic virtual machine consolidation, CloudSim toolkit, Planetlab VM data, Fuzzy logic

Introduction

Cloud computing can be classified as a new era of computing which has revolutionized the IT industry with its pay-as-you-go services. Its dynamic provisioning of computing services by using Virtual Machine (VM) technologies provides opportunity for consolidation and environment isolation. Having the viable business prospect, all the tech-giants have already started providing cloud services. IT companies are now moving from traditional CAPEX model (buy the dedicated hardware and depreciate it over a period of time) to the OPEX model (use a shared cloud infrastructure and pay as one uses it). To enable and ensure the global growth of computing need, cloud service providing companies are now using warehouse sized datacenters to meet user demand which incurs considerable amount of energy. At the beginning of the cloud computing era, cloud service providers focused mainly on catering the computing demand that lead to expansion of cloud infrastructures; hence energy consumption. Therefore, energy consumption by data centers worldwide was risen by 56 % from 2005 to 2010 [4]. In 2010 it was accounted to be between 1.1 and 1.5 % of the total electricity use and carbon dioxide emissions of the ICT industry were estimated to be 2 % of the global emissions which was equivalent to the emissions of the aviation industry [4]. Additionally, an average size data center consumes as much energy as 25,000 households [1]. American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) estimated that infrastructure and energy costs contributed about 75 %, whereas IT contributed just 25 %

* Correspondence: rashedur.rahman@northsouth.edu
[2]Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh
Full list of author information is available at the end of the article

to the overall cost of operating a data center [2] in 2014. So, to cater the increasing need of computing, energy aware technique should be applied in cloud computing infrastructure otherwise the energy need will be huge and will be threatening to the environment [20]. To handle this problem, datacenter resource needs to be utilized in an efficient manner. An efficient approach will not only reduce the energy consumption but also keep the performance up to the mark. Both in hardware and software there are several techniques being used for energy consumption of a cloud system. In hardware level, Dynamic Component Deactivation (DCD) and Dynamic Performance Scaling (DPS) are two such techniques, while in virtualization level, several techniques have been proposed e.g., the Energy Management for Hypervisor-based VMs and Kernel-based Virtual Machine (KVM) [28].

VM Consolidation is one of the techniques which draw researchers' attention and is an active field of research in recent time. As we know that inactive host or host in sleep mode causes minimal energy; therefore, energy consumption can be reduced considerably. By adopting VM consolidation, more energy could be conserved by shutting down underutilized datacenters. However, to achieve this outcome, we need to consolidate different VMs in one server and migrate VMs from one host to the other which may lead to SLA (Service Level Agreement) violation. So, algorithms must be designed in such a way that not only reduces power consumption but also serves desired QoS (such as SLA). In a VM consolidation method, selecting the VM to migrate is a challenging job and researchers came up with different solutions. In real world the computation need is very dynamic; therefore, decision is dependent on several criteria. In our research we have applied fuzzy logic. When the overload will be detected, our proposed fuzzy logic and heuristic based algorithm will decide the VM to migrate from the source datacenter to achieve minimum energy consumption by keeping the SLA violation at minimal level.

The remainder of this paper is organized as follows. In section Motivation, motivation has been clarified. In section Proposed method, our proposed methods and algorithms are given. In section Experimental setup, experimental setup is given and in section Experimental result, experimental result and comparisons have been presented. In section Related works, related works are discussed Finally, in section Conclusion, we have discussed about future work and concluded our paper.

Motivation

VM consolidation algorithm needs to be designed in such a way that there will be minimum energy consumption, minimum violation of SLA, efficient VM migration and minimum number of active hosts in a given time. VM migration causes SLA violation because when a VM is migrated from one host to other it has to transfer its primary memory to the destination host and in the transfer process the requested CPU could not be delivered as the VM will be in a transition state. For this reason, along with power consumption, we need to make sure that the number of VM migration is minimal which will in fact reduce the SLA violation. A desired VM consolidation approach will reduce energy consumption and as well as, it will reduce the negative impact on QoS. To address these issues, VM consolidation has been considered as a bin packing problem in some researches, e.g., [3, 17, 19]. On the other hand, there are researches where VM consolidation has been broken down in separate problems where bin packing solution is considered as one of the sub-problems of VM consolidations, i.e., VM placement [1, 2, 4, 8, 9, 12, 15, 16]. VM consolidation has been broken down in four sub-problems and dealt in researches are the followings:

1. Identify the under loaded datacenter to put them in sleeping mode by migrating all the VMs to other active datacenter (Under load detection).
2. Determine the host that is overloaded. Migrate some VMs from the identified overloaded datacenter to other datacenters while preserving QoS (Overload detection).
3. Decide the VM(s) that should be migrated (VM selection).
4. Place the selected VMs on other active or reactivated hosts\ (VM placement).

Breaking down into sub-problems has two key advantages. (1) Problems get simplified if it is divided into sub-problems and provides the opportunity to break the VM consolidation problem to four problems and devise separate algorithms. By doing that, performance of each algorithm can be measured and analyzed to investigate for identifying the better approach. As in this research we will mostly focus two sub-problems problems, one is host overload detection and another is VM selection. (2) It enables the option of distributed execution of the algorithms by executing the underload/overload detection and VM selection algorithms. Distributed VM consolidation makes the scaling easier. When a new node is added it automatically gets included in the algorithm which is essential for large-scale Cloud providers. These approaches are designed in CloudSim (an open source

Cloud Simulation designed by CLOUDS lab of University of Melbourne [5]). Researchers have developed their algorithms in CloudSim [1, 2, 4] which can be accessed and used for further research. However, there are places where the improvements could be done to yield better results by saving more power yet delivering the expected QoS. The driving factors which motivate to conduct this research are the following:

- For VM selection, there are several VM selection approaches are proposed in research [1, 2, 4], namely Maximum Correlation (MC), Minimum Migration Time (MMT), and Random Selection (RS). The maximum correlation (MC) approach selects the VM to migrate which has the highest correlation value among all the VMs of a host. The minimum migration time approach (MMT) selects the VM to migrate which has the least memory as it will be migrated faster. And the random selection approach (RS) selects the VM randomly from a host. The approaches offer different results. One method (MC) provides more power savings but lacks in QoS. Another method (MMT) provides better performance KPI, i.e., QoS incurring more power [4]. As the situation is uncertain and in real world the computation need is very dynamic, fuzzy logic can be applied with different inputs to achieve the tradeoff between energy consumption and QoS.
- Migration control can be applied to select the VM to migrate. Refraining from steady resource consuming VM migration can lead to better performance in dynamic VM consolidation [3]. But constantly high resource consuming VM should not be migrated as they consume large number of resources. So, migration control can be applied on two types of VMs; steady resource consuming VM and high resources consuming VMs. These phenomena can be taken into account while designing a VM selection method.
- To decide whether a host is overloaded or not, there are several algorithms proposed [1, 2, 4] (e.g., Inter Quartile Range (IQR): which decides the threshold of a host to be marked as overloaded using interquartile range, Median Absolute Deviation (MAD) uses median absolute deviation and THR provides threshold for a host to be marked as overloaded. Local Regression (LR) and Local Robust Regression (LRR) provide prediction of host utilization,). These statistical measures provide a threshold (IQR, MAD and THR) and prediction (LR and LRR) for a host to be identified as overloaded. In parallel, these algorithms rely on mean and standard deviation of resource utilization that gives an indication of future load of a VM which also can be an approach independently for overload detection [15]. However, mean and standard deviation is very much influenced by terminal values. Terminal values indicate the outlier values or the values that are too large or too small and do not represent the normal values. As VM's resource utilization can be very dynamic in real world, instead of mean we can use median and we can modify the formula for standard deviation using median instead of mean. An overload detection algorithm can be designed from this.
- When VM needs to be migrated to another datacenter in VM placement phase or underload detection phase, the destination host needs to be judged whether it will be overloaded in future by using overload detection method.

Proposed method

In this work we have designed Fuzzy VM Selection with migration control algorithm and Mean, Median & standard deviation based over load detection algorithm. However, before going in detail, overview of VM consolidation is presented. The algorithm below portrays the basic VM consolidation approach designed in CloudSim.

Algorithm 1 provides a basic flow of VM consolidation. At first the hosts are created. Then the VM data is taken as input. Based on the real life data of VM and cloudlets are created. Then VMs are assigned to host and cloudlet is assigned to VM. Based on dynamic consolidation technique, status is checked for every scheduled interval. For every scheduled interval, underload detection algorithm is executed and less utilized hosts are put into sleeping mode by transferring all VM to other active VM. Then overload detection is executed, and overloaded hosts are identified. At later steps, VM is selected from the overloaded hosts to migrate. Then those VMs are placed into available hosts or if needed a host is switched on from sleeping mode. After each iteration (the iteration time can be varied in CloudSim, most of the research have used 5 min as iteration interval [1, 2, 4]) a log is created to calculate energy consumption and QoS. At the end of the simulation, Energy consumption and QoS is shown. In next sections our proposed algorithms are discussed. More details of the iteration is discussed in section Experimental setup.

Algorithm 1. Basic VM consolidation

Input number of hosts;
Interface with real cloud data;
VM is created and assigned to hosts;
Cloudlet is created and assigned to VMs;
for every specified time interval
    Execute underload detection;
    Identify overloaded host through overload detection.
    VM is selected for migration from overloaded host.
    VM is placed in available datacenters.
    Preserve history and calculate QoS
end
Simulation ends and provides Energy consumption and other QoS value

A. Fuzzy VM selection with migration control

Fuzzy technique is an attractive approach to handle uncertain, imprecise, or un-modeled data in solving control and intelligent decision-making problems. Different VM selection methods offer different advantages. It will be worthy if we could generate a method which will have the benefits of all of them by combining them together. Fuzzy logic can be an ideal tool for this. It will consider all the options and depending on those options a fuzzy value will be generated based on the predetermined rule of inferences. To develop the fuzzy VM selection method we have selected three distinguished inputs and each of them offers some advantages over others and different researches have already proven them. Minimum migration time and Maximum Correlation can be found at [2, 4] and the idea steady resource consuming VM is adopted from [3]. The following subsections will be focusing on the variables we will be using as input to our fuzzy systems, member ship function generated, inference rules and algorithms for computation.

1) Minimum migration Time

Minimum Migration Time (MMT) policy selects the VM which can be migrated within minimum time limit [2, 4]. The migration time is limited by the memory the VM is using and the spare bandwidth. At any moment t, The MMT with Migration Control policy finds VM x that will be selected for migration by the formula (1). *RAM(x)* is the Radom Access Memory (RAM) utilization of VM x and *RAM(y)* is the RAM utilization of VM y. NET_h means the available bandwidth for migration *and* V_h is the set of VMs of host h. So the this method comapares the migration time and selects the VM x with minimum migration time among all VMs reside in host h.

$$x \in V_h \mid \forall y \in V_h,\ \frac{RAM(x)}{NET_h} \leq \frac{RAM(y)}{NET_h} \qquad (1)$$

This policy gives us the lowest SLA among all VM selection models. Migration time will be considered as one input of our fuzzy system.

2) Correlation

This method works based on the idea that the higher the correlation between the resource usages by applications running on an oversubscribed server, the higher the probability of server being overloaded [14]. It means that if the correlation of the CPU utilization of VMs of a particular host is high then the probability of this host being overloaded is also high [4, 14]. Based on this research outcome, correlation is considered as a metric as it will provide the information about the VM(s) that is going to cause the host to be overloaded. It is a predictive measure and consequently it will safer if such a VM could be migrated to other host where it will not have higher correlation with other VMs. In the subsequent portion, it is described how the correlations of VMs are calculated. An augmented matrix is created for all VMs of host using last n cycles' CPU utilization and correlation value is calculated. The highest the correlation

value of VM, the higher the probability of that VM makes the host to be overloaded.

As described in [4], let there are n number of VMs. Let Y be one out of those n VMs for which we want to determine the maximum correlation with other $n-1$ VMs. Here our objective is to evaluate the correlation of Y with the rest of VMs. The $(n-1)*n$ augmented matrix is denoted by X. Each value in the matrix X represents the observed values of $(n-1)$ VMs and y vector represents $(n-1)$ observations of VM Y.

$$X = \begin{bmatrix} 1 & x_{1,1} & .. & x_{1,n-1} \\ . & . & & \\ . & . & & \\ 1 & x_{n-1,1} & . & x_{n-1,n-1} \end{bmatrix} y = \begin{bmatrix} y_1 \\ . \\ . \\ y_n \end{bmatrix} \tag{2}$$

A vector of predicted value of VM y is denoted by $\hat{y}$ and expressed in Eq. 3.

$$\hat{y} = Xb, \quad where \;\; b = \left(X^T X\right)^{-1} X^T y \tag{3}$$

As we can find the predicted vector ŷ of Y, the multiple correlation coefficient $(R_{Y,\ 1.....N-1})^2$ also can be determined as this is equal to the squared correlation coefficient of the observed values y and predicted values of ŷ of VM Y. So the correlation coefficient can be defined by Eq. 4. Here m_y and $m_{\hat{y}}$ are the sample mean of the values of y and $\hat{y}$.

$$R^2_{Y,\ X_1,...X_{n-1}} = \frac{\sum_{i=1}^{n} \left(y_i - m_y\right)^2 \left(\hat{y}_i - m\hat{y}\right)^2}{\sum_{i=1}^{n} \left(y_i - m_y\right)^2 \sum_{i=1}^{n} \left(\hat{y}_i - m\hat{y}\right)^2} \tag{4}$$

Now the multiple correlation coefficient can be easily found for any VM Xi by $R^2_{X_i,\ X_1,.....X_{n-i},\ X_{n+i}\ .....X_n}$. According to this method the VM that has the highest correlation with other VMs' CPU utilization will be migrated. More details of this method could be found in [1, 14].

3) Migration control metric for steady resource consuming VM

It has been proven that migration control provides better result in energy aware VM consolidation and this approach also saves the unwanted traffic load [3]. Migration control can be done in various ways. We can stop migrating the high CPU using VMs or we can restrict steady resourse consuming VM from migration. In this work we will take steady resource consumption as a non-migration factor. If a VM's requirement highly fluctuates over time, then it can cause the host to be overloaded. In dynamic VM consolidation approach, VMs are resized in each iteration according to their need. So when a VM requests CPU which is abruptly high then host may not have such CPU available at that time and SLA violation will occur. As VM migration is triggered from an overloaded host we do not want to to migrate such VM from the overloaded host whose demands of CPU is not changed suddenly. In other words, if a VM is steady resource consuming over some iteration it means that it will be the least possible VM to make this host overloaded and we can expect the same behavior in the next iteration. We have used standard deviation for calculation of steady state resource consumption. If the standard deviation is high it means that the CPU request changes abruptly and we can call VMs with low standard deviation as steady resource consuming VMs.

Let us consider a host h and V_h be the set of VMs in host h. $CPU_u(x_t)$ is the CPU utilization of VM X at time t. $CPU_u(x_{t-1})$, $CPU_u(x_{t-2})$ ..... $CPU_u(x_{t-n})$ are the CPU utilizations of previous n time frames of VM X. Migration control parameter can be given by Eq. 5. Here $\text{CPU}_{\text{average}}$ means average CPU utilization in last n time frames. The standard deviation of CPU usage of VM X can be determined by this equation. This parameter will surely indicate the fluctuation of CPU usage of the particular VM X.

$$stdev = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(\text{CPU}_i - \text{CPU}_{\text{average}}\right)^2} \tag{5}$$

4) Fuzzy Membership function

A FIS (Fuzzy Inference System) is developed to provide fuzzy VM selection decision using three metrics as input. Member ship function

needs to be defined to develop the FIS. We are using 4 linguistic variables including VMselection as output. Range of these membership function is chosen from the real cloud simulation data of PlanetLab. In order to do the so, we have run the simulation and collected data of all these variables and proportioned to decide the range. As the ranges have been collected from real world data by doing statistical proportion operation (e.g. top 30 % values are high) for deciding different level (i.e. high, medium and low), using trapezoidal membership function is logical as it deals with ranges with flat region better. As the range of values should be counted as medium or low or high, not a peak value, triangular function is being not used and sigmoid function is not used as it does not define the flat region like trapezoidal function does. Membership function of the linguistic variables are given below:

- RAM: *T(RAM) = {Low, Medium, High}*
- Correlation: *T(Correlation) = {Low, Medium, High}*
- Standard Deviation: *T(Stdev) = {Low, Medium, High}*
- VM selection: T(*Vmselection*) = {Low, Medium, High, Very High}

Equation for the Trapezoid membership function [27] can be expressed as Eq. 6.

$$\mu(z) = \begin{cases} 0 & , \quad z \le a \\ \left(\frac{z-a}{b-a}\right) & , \quad a \le z \le b \\ 1 & , \quad b \le z \le c \\ \left(\frac{d-z}{d-c}\right) & , \quad c \le z \le d \\ 0 & , \quad d \le z \end{cases} \tag{6}$$

All the membership function graphs (Figs. 1, 2, 3 and 4) of the linguistic variables are given below and Table 1 shows the type of the membership function, the equation and the parameters.

As mentioned earlier, values and ranges of these membership functions are generated by heuristic approach. The source is 1-day (among 10 days) data of thousands of VM data of PlanetLab cloud network [25] (more discussed in section Experimental setup). For example, to deduce the standard deviation membership function, we have generated standard deviation value utilization of each of the thousand VMs. As these trace contains 288 (every 5 min from 1 day) data per VM, total sample size is about 288,000 (288 trace data*1000 VMs). Using a window size of 10, standard deviation is calculated for the total data and by doing ration the high, medium and low ranges are selected. The minimum migration time and correlation is also done in same way.

5) Fuzzy Inference Rule

Fuzzy inference rules are generated from the given linguistic variables. We have given equal weight on the variables to influence the VM selection value. If RAM is low it gets high priority as it makes the migration faster. If correlation is high then it gets high priority in migration as the higher the correlation is, the higher the probability of overloading the host. Finally, if the standard deviation is high then it will get high priority in migration compared to

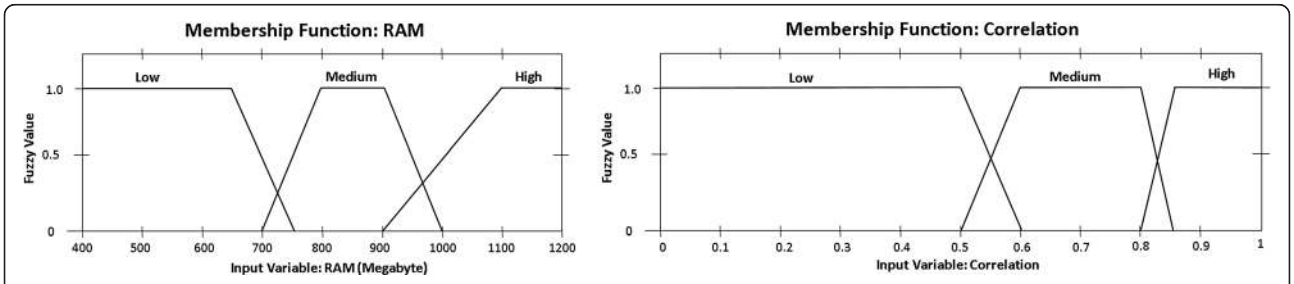

Fig. 1 Membership function : Ram and Correlation

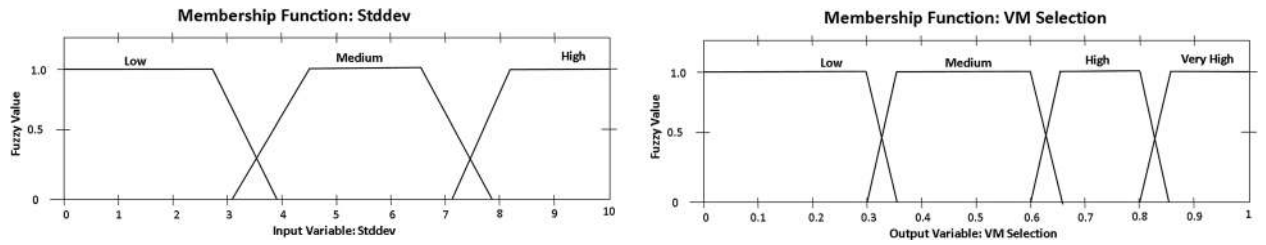

Fig. 2 Membership function Standard deviation and Fuzzy output: VM Selection

its steady state counterparts. The following Table 2 depicts the inference rules.

B. Fuzzy VM selection with migration control Algorithm

Combination of Fuzzy VM selection method and migration control is given in Eq. 7 and Eq. 8. These equations indicate that a VM will be nominated for migration if it produces lower CPU usage than the migration control threshold and possesses highest fuzzy output value. If all VMs of an overloaded host produce more CPU usage than the migration control threshold, then the VM that produces highest fuzzy output value will be migrated. It is described in detail below.

Here the VM x is selected for migration if the fuzzy output value of VM x is greater than all other VMs. However, there is a condition that is as follows. If the current time is t and in last n cycles CPU utilization of VM x is $CPU_u(x_t)$, $CPU_u(x_{t-1})$, $CPU_u(x_{t-2})$ … $CPU_u(x_{t-n})$, then the average CPU utilization must be less than CPUthreshold to satisfy migration control, i.e., not to migrate the highly utilized VMs. The Eq. 8 means if the average CPU utilization is above threshold for all the VMs then the VM x with maximum fuzzy output value will be selected for migration.

$$x \in V_h \mid \forall y \in V_h,\ \ Fuzzy\ Output(x) \geq Fuzzy\ Output(y)$$

Only if;

$$\frac{[CPU_u(x_t) + CPU_u(x_{t-1}) + CPU_u(x_{t-2}) + \ldots CPU_u(x_{t-n})]}{(n+1)} < CPU_{thresold} \tag{7}$$

However, if every VM vm satisfies the following condition that means average utilization is more than the threshold,

Algorithm 2. Fuzzy VM selection Algorithm (FS)

***Input**: overloaded host h, window size n*
***Output:** Virtual Machine to be migrated, VM_m*

1. *Input Overloaded host h ;*
2. *VM_h = GetMigratableVm(h);*
3. *VM_{hex} = ExcludeVmInMigration (VM_h);*
4. *Utilm (VM_{hex})=UtilizationMatrix(VM_{hex});*
5. *Metric(n) =CorrelationCoefficient(UtilM (VM_{hex}));*
6. *for each VM V_i of VM_{hex}*
7. *CPU_{hist}=GetMcParamFromCpuHistory(V_i);*
8. *CPU_{mc}=GetMigrationControl(CPU_{hist});*
9. *STDEV(V_i)=StandardDeviation(CPU_{hist});*
10. *RAM (V_i)=GetRam(V_i);*
11. *MC (V_i)= Metric(V_i);*
12. *$Output_{fuzzy}$= EvaluateFuzzy(STDEV(V_i), RAM (V_i), MC (V_i))*
13. *If $Output_{fuzzy}$ is highest till now*
14. *$VM_{highest} = V_i$;*
15. *if $CPU_{mc} < CPU_{threshold}$ then $VM_m = V_i$;*
16. *End;*
17. *End;*
18. *If VM_m is null: $VM_m = VM_{highest}$;*
19. *Return VM_m ;*

Table 1 Memberhip functions

Variables	Parameter		
	Level	*Function Type*	*Parameter*
RAM	Low	Trapezoidal	a = 0 b = 0 c = 650 d = 750
	Medium	Trapezoidal	a = 700 b = 800 c = 900 d = 1000
	High	Trapezoidal	a = 900 b = 1100 c = 1800 d = 1800
Correlation	Low	Trapezoidal	a = 0 b = 0 c = .5 d = .6
	Medium	Trapezoidal	a = .5 b = .6 c = .8 d = .85
	High	Trapezoidal	a = .8 b = .85 c = 1 d = 1
Stdev	Low	Trapezoidal	a = 0 b = 0 c = 3 d = .3.75
	Medium	Trapezoidal	a = 3.25 b = 4 c = 6.75 d = 7.5
	High	Trapezoidal	a = 7.5 b = 8.5 c = 100 d = 100
VM Selection	Low	Trapezoidal	a = 0 b = 0 c = .3 d = .35
	Medium	Trapezoidal	a = 0.3 b = 0.35 c = .6 d = .65
	High	Trapezoidal	a = .6 b = .65 c = .8 d = .85
	Very High	Trapezoidal	a = .8 b = .85 c = 1 d = 1

$$\frac{[CPU_u\,(vm_t) + CPU_u\,(vmx_{t-1}) + \ldots + \ CPU_u\,(vm_{t-n})]}{(n+1)}$$
$$\geq CPU_{thresold}$$

then this technique will select the VM that produces the highest fuzzy output value;

$$x \in V_h \mid \forall y \in V_h, \ \ Fuzzy\ Output(x) \geq Fuzzy\ Output(y) \tag{8}$$

The *Algorithm 2* depicts how Fuzzy VM Selection algorithm (FSMC) works. There are two inputs of the algorithm: the host h and window size n *(*CloudSim Default window size has been used*)*. The overloaded host is detected by previous phase: Overload detection. After having the host h at step-1, at step-2, *GetMigratableVm(h)* function pulls all the VM which are currently placed on that host h. At step-3, *ExcludeVmInMigration* function excludes all the VM which are already in migration for that host and assigned to VM_{hex} . At step-4, the function *UtilizationMatrix* calculates utilization matrix and stores at *UtilM* (VM_{hex}). At step-5, function *CorrelationCoefficient* calculates correlation coefficient based on *UtilM* (VM_{hex}) *and stores at Metric(n).* At step-7, for each VM V_i, CPU usage history of V_i is fetched using the function *GetMcParamFromCpuHistory* (V_i) for last n iteration as per CloudSim settings. At Step-8, Migration control parameter is calculated. To determine the steadiness of a VM's CPU usage, at Step-9, *STDEV*(V_i), Standard deviation is calculated using *StandardDeviation* function *from* CPU_{hist}. At Step-10, current usage of RAM will be fetched for V_i and will check for if the one is the lowest up to now. At Step-11, Correlation for this VM will fetched. At Step-12, fuzzy output $Output_{fuzzy}$ is determined using *EvaluateFuzzy* function where inputs of this function are *STDEV*(V_i), *RAM* (V_i) and *MC* (V_i). At step-13, If this one is the highest till now, at step-14, $VM_{highest}$ will be updated. At step-15, VM VM_m will be updated if CPU_{mc} is smaller than $CPU_{threshold}$. If all VM is greater than the threshold and the highest fuzzy output VM is selected for migration at step-18 and finally step-19 returns the VM to be migrated.

Table 2 Fuzzy inference rule

Input			Output
RAM	*Correlation*	*Stddev*	*VM Selection*
Low	High	High	Very_High
Low	High	Medium	Very_High
Low	High	Low	High
Low	Medium	High	Very_High
Low	Medium	Medium	High
Low	Medium	Low	Medium
Low	Low	High	High
Low	Low	Medium	Medium
Low	Low	Low	Low
Medium	High	High	Very_High
Medium	High	Medium	High
Medium	High	Low	Medium
Medium	Medium	High	High
Medium	Medium	Medium	Medium
Medium	Medium	Low	Low
Medium	Low	High	Medium
Medium	Low	Medium	Low
Medium	Low	Low	Low
High	High	High	High
High	High	Medium	Medium
High	High	Low	Low
High	Medium	High	Medium
High	Medium	Medium	Low
High	Medium	Low	Low
High	Low	High	Low
High	Low	Medium	Low
High	Low	Low	Low

C. Mean, Median and Standard deviation based Overload Detection(MMSD)

Algorithm 3. MMSD based overload detection (MMSD)

***Input**: host h*
***Output:** True or False indicating overloaded or not*
1. *Input Overloaded host h ;*
2. *VM_h = GetVm(h);*
3. *for each VM V_i of VM_h*
4. *TotalRequestedMIPS+=GetCurrentMIPS(V_i);*
5. *Stddev=GetStddevOfUtilization(V_i);*
6. *Mean= GetMeanOfUtilization(V_i);*
7. *Median= GetMedianOfUtilization(V_i);*
8. *$Stddev_{median=}$ GetStddevUsingMedian(V_i);*
9. *If Stddev < StdDevThreshold*
10. *TotalPredictedMIPS+= Mean + Stddev;*
11. *Else TotalPredictedMIPS+= Median + $Stddev_{median}$;*
12. *End;*
13. *Utilization= TotalRequestedMIPS /totalMIPSofHost;*
14. *Prediction= TotalPredictedMIPS/ totalMIPSofHost;*
15. *If Utilization>1 or Prediction>1*
16. *Return True;*
17. *Else Return False*

Overload detection algorithm ensures that when a host is overloaded, then the algorithm will find it. Moreover, it will provide intelligent measure so that the datacenter does not get overloaded. There are several overload detection algorithms proposed in [1, 2, 4], 1) A Static CPU Utilization Threshold (THR): where overload decision is based on a static threshold; 2) Adaptive Median Absolute Deviation (MAD): the overload threshold is calculated dynamically using median absolute deviation; 3) Adaptive Interquartile Range (IQR): overload threshold is calculated dynamically using interquartile range method; 4) Local Regression(LR); and 5) Robust local Regression(LRR). LR and LRR are predication methods which will predict whether the host is going to be overloaded or not.

In this work, a new overload detection algorithm has been devised. When overloaded, a host incurs SLA violation. To be precise, a host incurs SLA violation when the required CPU utilization is greater than the actual utilization capacity. To avoid SLA violation, we have to design an overload detection mechanism which will predict this scenario. Host utilization is calculated from the VM utilization. If the summation of mean (μ) and standard deviation (δ) of last *n* iteration is greater than the maximum capacity of the VM then it can be inferred that this VM will request more utilization than allocated in future [15]. We apply that idea in our research.

$$\mu + \delta > 1 \tag{9}$$

Equation 9 means that if the summation of mean utilization of a VM for last *n* cycles and standard deviation of utilization of that VM for last *n* cycles is higher than the allocated CPU, then in next iteration this VM can go beyond the maximum capacity of that VM. As our objective is to keep SLA violation at the lowest level, we can calculate predicted utilization of all VM of a corresponding host using Eq. (9) and check whether the total predicted utilization of a host is greater than the capacity or not. If the predicted value is greater, then the host is at risk of being overloaded and SLA violation. This technique we are going to apply in our overload detection algorithms. However, mean and standard deviation is very much influenced by terminal values that refer the outlier values or values that are too large or too small. So, when the standard deviation is high (i.e. the value falls in the high range of standard deviation membership function of fuzzy VM selection method) meaning that there is a possibility of high values present in the last *n* cycles. From the fuzzy membership variable *Stddev*, high range is considered. To avoid terminal values, when standard deviation is high, we replace mean with median and standard deviation formula is changed by replacing mean with median by Eq. (10). Hence, the prediction formula can be represented by Eq. (11). So it ensures, if in last n cycles any sudden fluctuation, i.e., very low

or very high CPU utilization is found, this will not impact on the overall decision. δ_{Median} is the standard deviation calculated from median instead of mean. Like Eq. 9, Eq. 11 provides the prediction of a host. If the summation of Median and δ_{Median} is greater than 1 i.e. more than the total CPU then the host is considered to be overloaded.

$$\delta_{Median} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\text{CPU}_\text{i} - \text{CPU}_{\text{Median}})^2} \quad (10)$$

$$\text{Median} + \delta_{\text{Median}} > 1 \quad (11)$$

Algorithm 3 describes how MMSD works. The input of the algorithm is the host of interest and the output of the algorithm is to determine whether the host is overloaded or not. At second step the VMs are identified which are currently active on the host. For every VM a loop is started at step 3. Total requested MIPS (Million Instructions Per Second) is accumulated to get the total requested MIPS of the host. Then Mean, Standard deviation, Median and Standard deviation from median are calculated. Now the predicted MIPS is calculated by the standard deviation value. If the standard deviation is greater than the threshold (this threshold is taken from the fuzzy member ship variable Stddev's High value which starts from 8.5) then Eq. (11) is followed else Eq. (9) is followed. Then utilization of the host and predicted utilization of the host is calculated. If any of these are beyond 1(meaning 100%) then the host is marked as overloaded by returning true otherwise returning false.

D. Underload detection and handling

There is an underload detection and handling algorithm in CloudSim. The algorithm is simple; it sorts the hosts according to their utilization and starts with the lowest utilized host. If all VMs of a particular low utilized host can be placed to any/some of active hosts using VM placement method then the host is put to sleep mode by migrating all VMs to other hosts. VM placement will be discussed later section. It is worthwhile to mention that before migrating VM to other active hosts, the destination host is checked whether it will be overloaded by the new assignment with our newly designed overload detection algorithm.

E. VM placement Algorithm

In CloudSim toolkit power aware BFD (Best Fit Decreasing) algorithm is used for VM placement. When overload detection or underload detection finds VMs to migrate, VM placement algorithm assigns the VM in such way that power consumption is increased minimally [4]. VM is placed in the host with decreasing utilization order. In this work it has been ensured that if a new VM placement is considered, then our newly overload detection algorithm certifies that the destination host will not be overloaded in next iteration.

Experimental setup

In this experiment, we have implemented our algorithms in CloudSim 3.0.3 and analyze the performance of our proposed method. We have considered 800 heterogeneous physical nodes, half of which are HP ProLiant G4 and the rest are HP ProLiant G5 servers. Energy consumption is calculated based on HP ProLiant G4 and HP ProLiant G5 CPU usage and power consumption that is represented in Table 3 [4]. These servers are assigned with 1860MIPS (Million instruction per second) and 2660 MIPS for each core of G4 and G5 servers respectively. Network bandwidth is considered as 1GB/s. The VMs which were created were single core. VM were of 4 types, for example, 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, 613 MB). Fuzzy rules are defined and integrated to CloudSim using JFuzzyLogic Tool [13].

In this work we have used real world work load data that is provided from CoMon project, a monitoring infrastructure for PlanetLab [25]. This data is collected from more than thousand VMs of different servers that are located in 500 different locations. The workload is representative of an IaaS cloud environment such as Amazon EC2, where VMs are created and managed by several independent users. Table 4 presents the day wise VM number for this data. These real world traces contain VM utilization records in every 5-min interval. Ten days' data of year 2011 have been used in this experiment. Each VM contains 288 (=24*(5/60)) data of CPU utilization. The simulation checks CPU data every 5 min interval and those trace data is plugged into dynamic VM consolidation.

The main target of VM consolidation is to reduce energy consumption and at the same time the QoS should be at an acceptable level. The energy consumption metric is discussed below and for QoS parameter, several metrics are stated which are used in several researches [2, 4]. The main QoS is SLA violation. In VM consolidation SLA violation occurs due to host overload and VM

Table 3 Power consumption for diferent level of utilization

Machine type	Power consumption based on CPU utilization					
	0 %	*20 %*	*40 %*	*60 %*	*80 %*	*100 %*
HP G4 (Watt)	86	92.6	99	106	112	117
HP G5 (Watt)	93.7	101	110	121	129	135

Table 4 Day wise planet lab data

Date	*Number of VMs*
3 March	1052
6 March	898
9 March	1061
22 March	1516
25 March	1078
3 April	1463
9 April	1358
11 April	1233
12 April	1054
20 April	1033

migration. To quantify SLA violation for overloaded host, the metric Overload time fraction (OTF) is used and on the other hand to quantify the SLA violation due to VM migration, the PDM (performance degradation due to migration) is defined. SLAV (SLA violation) is the product of OTF and PDM. Moreover, the number of VM migration indicates the efficiency of the consolidation method which is also described as a metric. But the main objective of our research is to obtain Energy-QoS trade off and that is defined by the metric ESV which is the product of energy consumption and SLA violation (SLAV). So the method providing the lowest ESV and at the same, the lowest energy consumption and the lowest SLA violation, is undoubtedly the best method. Based on these six metrics proposed method will be verified and they are described in more detail and mathematically below.

1) Energy Consumption(kWh)
   This is the main metric as the target of VM consolidation is to reduce energy consumption. Energy consumption is computed by taking into account all hosts throughout the simulation by mapping of CPU and energy consumption from Table 3. At each iterations the CPU utilization is measured and power consumption is calculated from Table 3 and at the end of the simulation energy consumption is measured by accumulating all hosts' energy consumption.
2) Number of VM migration
   This metric counts the number of VMs migrated during the siumaltion. VM migration is an important factor because unnecessary migration causes SLA violation and network traffic.
3) OTF
   Overload time fraction [4], OTF is a measure of SLA violatoin. it provides a measure of the fraction of time a host experienced 100 % CPU utilization leading to SLA violation. In Eq. (12), if N is the number of hosts, T_{si} is the total time when host i experienced 100 % utilizaiton leading SLA Violation, T_{ai} is the total active time of host i, then OTF is defined by:

$$OTF = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{si}}{T_{ai}} \qquad PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{dj}}{C_{rj}} \tag{12}$$

4) PDM
   Performance degradation due to migration(PDM) [4] is a measure of SLA violation. It measures the total SLA violation due to VM migration. When a host is overloaded, VMs are migrated from that host to non-overloaded host. At the time of migration, that VM is not capable of serving user needs, hence, it causes SLA violation. This metric calculates the SLA violation caused by migration. From Eq. (12), if M is the number of total VMs, C_{dj} stands for the CPU request at the time of migration of VM j and C_{rj} stands for total CPU requested by VM j, then PDM is defined by the Eq. (12).
5) SLAV
   Service level agreement violation, SLAV, is combined impact of OTF and PDM. It provides a SLA violation measure for the simmulation which is a product of OTF and PDM i. e., SLAV = OTF*PDM.
6) ESV
   Energy consumption and SLA is already defined. It is perceivable that if we try to reduce too much energy consumption the SLA violation will be increased, because consolidating many VMs in a host increase the probability of overload. So it is desirable to obtain a method which will consume less power and still incur less SLA violation. To measure this, ESV is introduced. It is the combination of Energy consumption and SLA violation, i.e., ESV = Energy*SLAV. So this can be treated as one metric to make an overall measurement. If the product of energy consumption and SLA violation is lower, it means that the approach reduces energy consumption and making less SLA violation.

Experimental result

In our experiment, using CloudSim, we have experimented with five Overload detection algorithms (IQR, LR, LRR, MAD and THR) and three VM selection (MC, MMT, RS) methods. So in combination there are 15 methods (IQR_MC, IQR_MMT, IQR_RS, LR_MC, LR_MMT, LR_RS, LRR_MC, LRR_MMT, LRR_RS, MAD_MC, MAD_MMT, MAD_RS, THR_MC, THR_MMT, THR_RS)

which will be compared against our proposed MMSD_FS method based on aforementioned performance metrics. Based on the result for 10 days Box grpahs have been prepared to compare the results. It is represented in Figs. 3, 4, 5, 6, 7 and 8. We discuss the performance with respect to each metric are given below.

A. Energy Consumption

Main objective of this research is to design a VM consolidation algorithm so that the energy consumption is reduced. By comparing the proposed and existing methods in the Fig. 3, it is found that energy consumption is significantly reduced in proposed (MMSD_FS) method. Minimum energy consumption by the proposed method is 102 Kwh where the minimum of all other methods is 112 Kwh, therefore we got 8.5 % reduction. If we consider average value, MMSD_FS consumed 136.5 Kwh and all other methods consumed 169 Kwh on average resulting 19 % energy saving. Therefore, we can infer that the basic objective of this research is achieved by saving energy consumption.

B. SLA Violation

SLA violation is one of the key indicators of QoS. SLA Violation is calculated by keeping two scenarios into consideration, i) if any VM got overloaded, and ii) The SLA violation incurred while migration. A method having low SLA violation ensures the desired QoS. From Fig. 4, SLA violation is decreased significantly which is clearly visible. Minimum SLAV by proposed method is 0.0004 % whereas the minimum of all other method is 0.00279 %, resulting 84 % reduction. If we consider average value, MMSD_FS incurred 0.0005 % SLAV and all other methods incurred 0.00617 % on average, resulting 91 % reduction in SLA violation. This is main achievement of this research. It means that the overload detection method we have used, predicted the overloaded host efficiently and as an outcome, SLA violation was dropped significantly. If host overload is predicted successfully then there will be less number of migration which will reduce SLA violation as well.

C. ESV

As energy consumption has been successfully reduced by the proposed method, now energy-QoS trade off needs to be checked. ESV is the metric which is a product of Energy consumption and SLA violation; hence, provides a tradeoff picture of the proposed method with other existing methods. From previous two sub-sections, we have observed that both energy and SLA violation reduced, so it is inevitable that ESV will also be reduced significantly. From the Fig. 5, ESV is found to be reduced significantly which is clearly visible. If ESV reduces it means that this approach saves energy and at the same time SLA violation is controlled. As ESV is reduced significantly, it means that Energy and SLA tradeoff has been achieved. Minimum ESV by proposed method is 0.04 whereas the minimum all other method is 0.49, resulting 91 % reduction. If we consider average value, MMSD_FS incurred 0.07 ESV and all other method incurred 1.09 on average, resulting 93 % reduction in ESV.

D. Number of VM migration

Less number of VM migrations means efficient consolidation, less traffic in cloud network and less SLA violation for VM migration. Reduction in Number of VM migration is also clearly visible

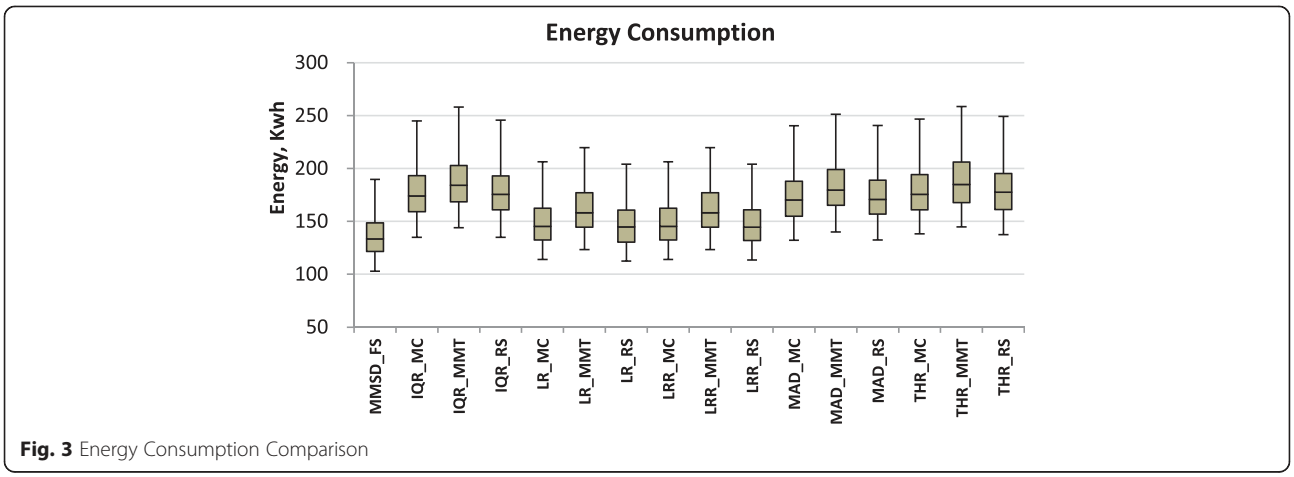

Fig. 3 Energy Consumption Comparison

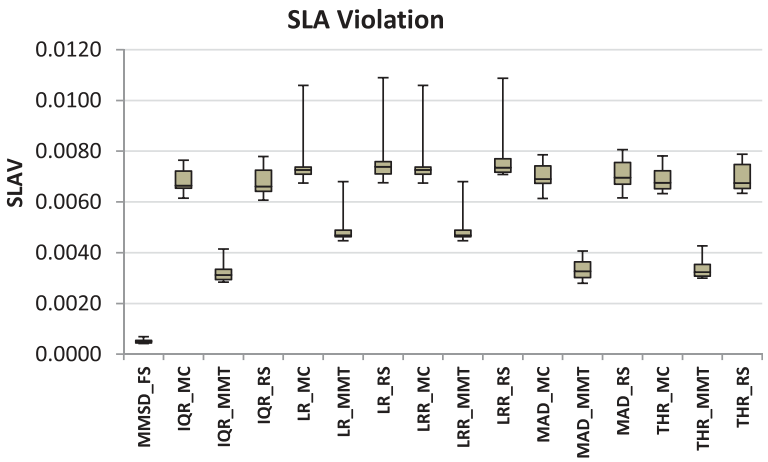

Fig. 4 SLA violation

from Fig. 6. To quantify, minimum number of VM migration caused by MMSD_FS is 5185 whereas the minimum all other method is 16,317, resulting 68 % reduction. If we consider average value, MMSD_FS incurred 7943 migrations on an average and average of all other methods is 24,929, resulting 68 % reduction in migration. From this percentage it is evident that the proposed method provides most optimum VM consolidation compared to the existing VM consolidation approaches.

E. OTF and PDM

From Fig. 7 to Fig. 8, it can be inferred that OTF and PDM is significantly reduced. The proposed method reduced both SLA violation due to overload and SLA violation due to VM migration. Minimum OTF is reduced up to 60 % by the proposed method and Minimum PDM is reduced up to 64 %. On an average OTF is decreased by 67 % and PDM is decreased by 74 % compared to the existing methods.

Finally, we have performed a statistical test namely two-tailed students' *t*-test on the performance of the proposed method MMSD_FS and IQR_MMT method (the best method in CloudSim as per the ESV, Fig. 4). Our null hypothesis is: "There is no significant difference in the performance between two techniques". Table 5 reports *p*-values for six performance metrics between MMSD_FS and IQR_MMT generated from 10 days' experimental data. If the *p*-value is greater than 0.05, then we must accept the null hypothesis, otherwise we must reject the null hypothesis. From Table 5 we find that the *p*-value is significantly smaller than 0.05 for every performance metric. Therefore, we must reject the null hypothesis and we could conclude that there is significantly difference in performance found.

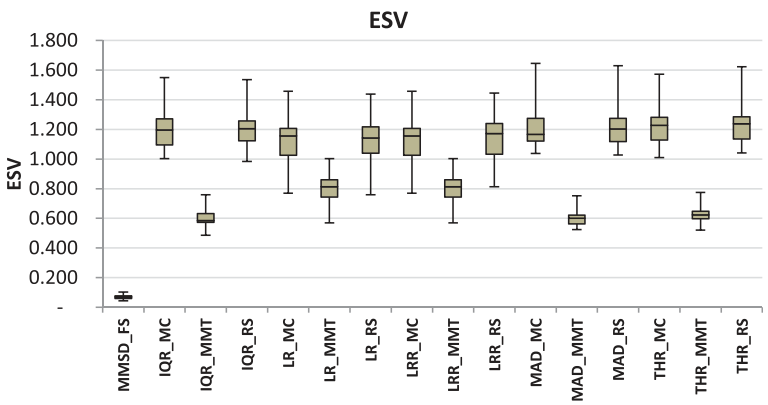

Fig. 5 ESV

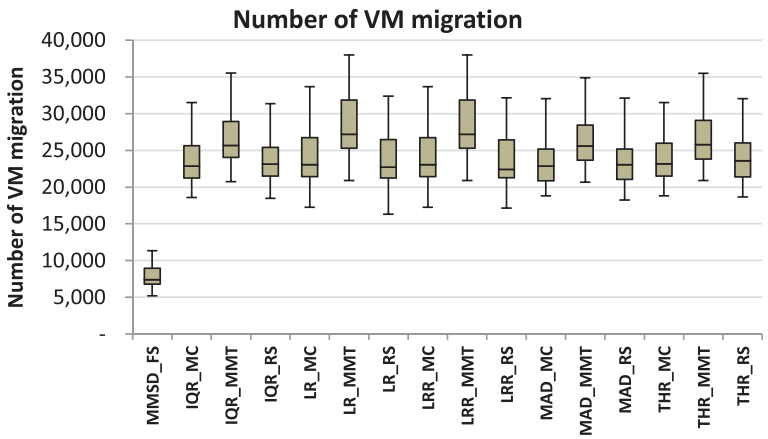

Fig. 6 Number of VM Migration

From all the performance metrics it can be inferred that the proposed method outperforms all other methods.

F. A Deep dive

From the above result analysis, we have found that the proposed method improved significantly. Most of the improvement came from the SLA violation part. This phenomenon indicates that the proposed method identifies the host overload efficiently. To visualize the performance in easier way we have generated two heat maps of MMSD_FS method and another is IQR_MMT method which are given in Fig. 9 and Fig. 10 respectively.

For this experiment, we have used 50 hosts and 50 VMs and random load. In the heat map, if a host is in sleeping mode i.e., 0 % utilization then it is marked by blue color and red color for high utilization. In the X-axis the time is portrayed. As iteration duration is 5 min, so there are total 288(starting from 0 to 8600 s) iterations as the simulation is done for 1-day data. Y-axis represents 50 hosts. From Fig. 9, it is visible that hosts are experiencing ON-OFF frequently and the map seems scattered and the total number of overload occurs 685 times. So this method will invoke VM migration at least 685 times. On the other hand, Fig. 10 shows the heat map for MMSD_FS where we can observe less fluctuation (ON-OFF) of the host. It is easily perceivable that the host is put to sleeping mode and stays in sleeping mode for long. Total number of overload incident is 92, which indicates the efficiency of the algorithm. The main reason behind the performance of MMSD_FS is the prediction done by this algorithm helped to reduce the number overloaded hosts.

Related works

Considerable number of researches has been conducted for VM consolidation using various methods based on

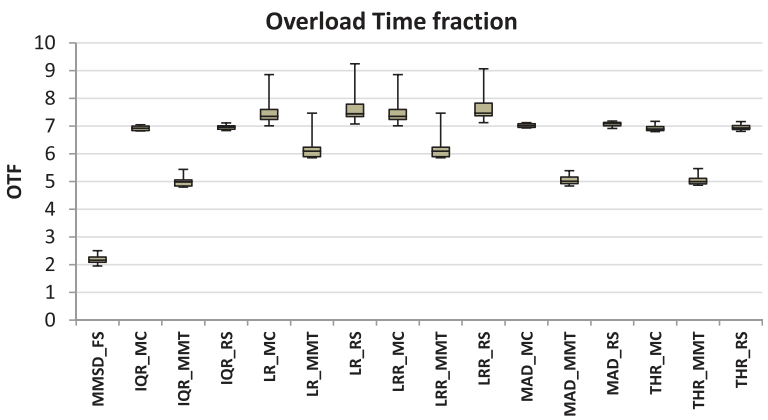

Fig. 7 Overload Time Fraction

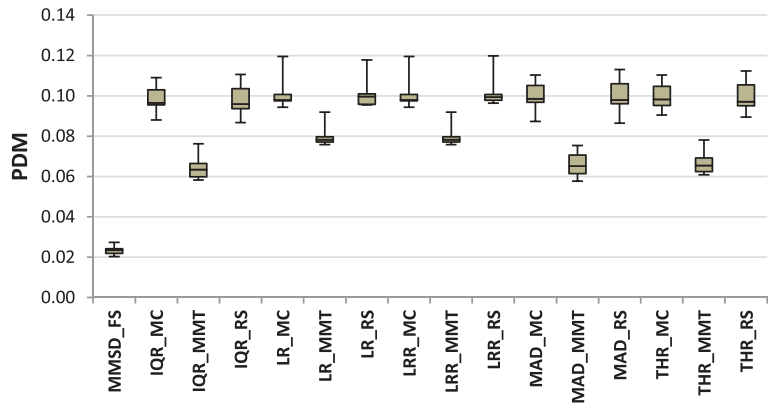

Fig. 8 Performance Degradation for migration

heuristics. In this research, we have worked on two problems, 1) VM selection method, and 2) Overload detection method. Here, we will discuss about various VM selection methods. For Overload detection methods, there are two types of algorithms, a) Threshold based methods, and b) Prediction based method. In threshold based method, researchers apply different statistical or heuristic methods to calculate threshold for a host to be identified as overloaded. On the other hand, there are some predictive methods where researchers use approaches/techniques to predict the future load of a host.

In [1, 2, 4] Beloglazov et al. proposed heuristic based approach to deduce thresholds through different statistical measures. VM Consolidation problem is divided into sub-problems and algorithms for each sub-problem had been designed. Heuristic based algorithms are designed for each sub problems and designed in such a way that they act, adapt and keep their threshold changing based on different scenario in different time so that they can still provide the functionality and consolidation decision in the changed environment. This adaption process allows the system to be dynamic. They designed threshold based (e.g. IQR) and prediction based (e.g. LR) host overload detection mechanisms. We have shown in result section, our proposed algorithms outperformed the algorithms proposed in literature. References [5, 6] describe CloudSim which provides various functionalities of a cloud environment and facilitates in cloud simulation. Reference [1, 2, 4] have also used CloudSim for simulation. The main components of CloudSim are datacenter, Virtual Machine (VM) and cloudlet. Cloudlet can be data from real cloud. The simulator creates datacenter, Virtual Machine and cloudlet based on the defined parameters. When the simulation starts, Virtual Machines are placed in the datacenter for processing. Sub-problems (i–iv) are already developed in CloudSim. To extend it further, one needs to create new class and develop new methods to test it. The VM selection methods and Overload detection methods are compared in this research and the proposed algorithm performs better in all metrics defined in CloudSim. In the previous section we have compared our proposed algorithm with both thresholds based (MAD, IQR and THR) and prediction based approaches (LR and LRR). We have discussed several approaches which are proposed in the literature.

Table 5 *P*-values for performance metrics

Metric	*P-value*
Energy Consumption	0.004
ESV	2E–09
Number of Migration	1.4E–08
SLAV	6.78E–12
OTF	5.38E–19
PDM	1.44E–12

Farahnakian et al. [9] used ant colony system to deduce a near-optimal VM placement solution based on the specified objective function. In [3] VM consolidation with migration control is introduced. Here VMs with steady usage are not migrated and not steady VMs are migrated to ensure better performance. The migrations are triggered and done by heuristic approaches. But this research has not been used the other sub problem rather focused on only the VM placement problem. We have considered all the sub-problems together.

Farahnakian et al. [18] proposed a Reinforcement Learning-based Dynamic Consolidation method (RL-DC) to minimize the number of active host considering the resource requirement. The RL-DC utilizes an agent to learn the optimal policy. The agent uses the past knowledge to take intelligent decision whether to keep

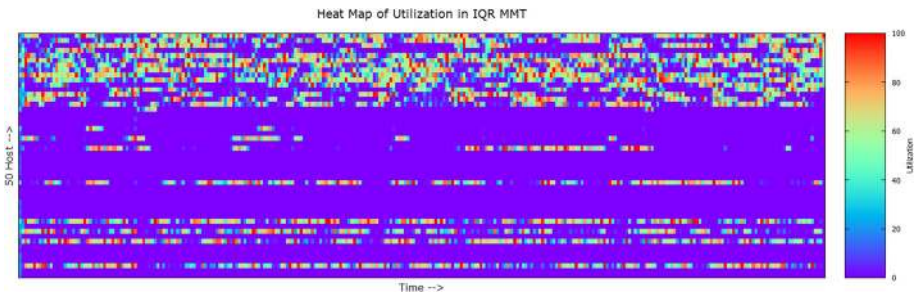

Fig. 9 Heat map of IQR_MMT method

the host in active or sleep mode and improves itself as the workload changes. It also dynamically adapts changes. In [12] linear regression has been used to predict CPU utilization by the same author. These researches are developed in CloudSim and follow the distributed architecture. From result and comparison, it is evident that our proposed algorithm performs better in regression based host overload detection method.

Cao et al. [15] proposed a redesigned energy-aware heuristic framework for VM consolidation to achieve a better energy-performance tradeoff. They designed a Service Level Agreement (SLA) violation decision algorithm which is used to decide whether a host is overloaded with SLA violation or not. SLA violation is determined if the allocated CPU of a particular VM is less than the requested CPU of that VM. In other words, if a host is not capable of assigning CPU resource to all the VMs as per demand, then the SLA violation occurs. There is another type of SLA violation which is SLA violation due to migration. If a particular VM is in migration, at the time of migration, the VM is not capable of serving users need hence it is counted as SLA violation. This research is based on CloudSim and algorithms are developed in CloudSim and they have used mean and standard deviation as the prediction method, whereas in our research we used median and standard deviation derived from median. In the result section we depicted that how our method outperformed. Duy et al. [21] proposed a neural network predictor in a Green scheduling algorithm to predict future resource requirements based on historical data. Based on the prediction, decision is taken to keep unused servers in sleep mode and keep the high utilized servers in active mode. There are also similar works that can be found in [22, 23]. These works provide a host utilization prediction and one sub problem is discussed which is overload detection. Srikantaiah et al. [24] have studied the interrelationships between energy consumption, resource utilization, and performance of consolidated workloads. The study shows the energy performance trade-off for consolidation. That research did not use all the sub problems of VM consolidation, rather considered the whole problem as a single one.

Mastroianni et al. [19] presented ecoCloud, a self-organizing and adaptive approach for the consolidation of VMs on CPU and RAM. Assignment and migration decisions are driven by probabilistic processes and based on local information. Focusing on the VM placement problem, they experimented in real datacenter. However, all the sub-problems are not addressed. Madani et al. [17] focused on an architecture configuration to manage virtual machines in a data center to optimize the consumption of energy and meet SLA by grafting a tracing component of multiple consolidation plans which ensure minimum number of servers is switched on. In this research, the problem is seen as scheduling problem and sub problems are not discussed.

Sheng at al. [11] designed a method based on Bayes model to predict the mean load over a long-term time interval. Prevost et al. [10] introduced a framework

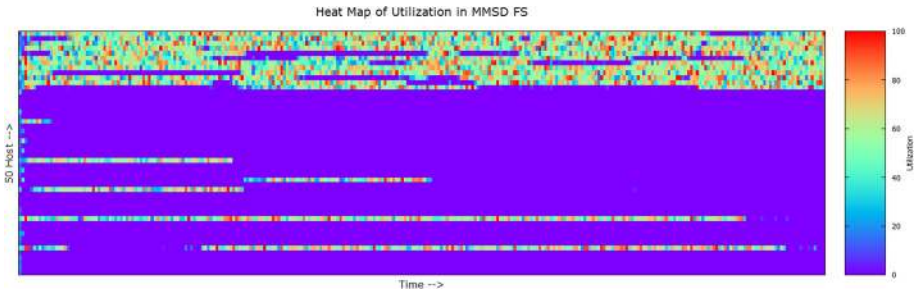

Fig. 10 Heat map of MMSD_FS method

by combining load demand prediction and stochastic state transition models. They used neural network and autoregressive linear prediction algorithms to forecast loads in cloud data center applications. These works used statistical and neural network to predict the host utilization and only focused on overload detection of a host.

In [7] and [8] we worked with basic VM selection algorithm and introduced migration control in the built in CloudSim VM selection methods. In [26] a preliminary study was carried out using fuzzy logic in VM selection. As the initial findings are encouraging, in this research we incorporate all our previous methodologies, e.g., migration control and fuzzy logic together and study the performance on VM selection. Besides, we have introduced a new overload detection algorithm based on mean, median and standard deviation of utilization of VMs. In this research, we study the performance of our proposed VM selection algorithm coupled with the newly designed overload detection algorithm. A comparative study has been also reported to present the performance against previous VM selection algorithms found in CloudSim.

Conclusion

In this research we have devised algorithm for fuzzy VM selection method and introduced migration control in the selection method. Fuzzy VM selection methods take intelligent decision to select a VM to be migrated from one host to the other. Then we designed mean, median and standard deviation based overload detection algorithm. After simulation and making comparison against existing methods, it has been found that the proposed method outperformed other previous methods in both perspectives, i.e., more energy saving and less SLA violation. Therefore, it can be inferred that the objective, energy-SLA tradeoff has been achieved in this work in an efficient manner. As a future work we have plan to improve the default VM placement and underload detection algorithm built in CloudSim to achieve more energy saving and less SLA violation.

Acknowledgement

There is no acknowledgement from authors' side.

Authors' contribution

Monil carried out the survey of literature, the study of energy efficient VM selection algorithms, identified the research problem, proposed the VM selection algorithm and modified the existing overload detection algorithm. Monil carried out the experiments in simulation environment and drafted the manuscript. Rahman discussed and advised to add explanations for few sections. Rahman also provided careful review of the article and helped Monil in answering reviewers' comments. All authors read and approved the final manuscript.

Authors' information

The current research initiative was taken when Mohammad Alaul Haque Monil was doing his Master's program in North South University, Dhaka, Bangladesh. In 2014, he was awarded the M.Sc degree in Computer Science and Engineering from North South University. Currently Monil is a Ph.D. student of University Oregon, Eugene, Oregon, USA. His current research focus is on parallel and distributed computing. Recently he worked on VM placement method for Virtual Machine consolidation. He is also working on applying parallel computing techniques to solve shortest path algorithm in parallel and distributed environment. He received his B.Sc degree on the same major from Khulna University of Engineering and Technology, Bangladesh in 2006. Monil also worked in Grameenphone Bangladesh, a business unit of Telenor for 9 years in different roles including managerial position.

Rashedur M. Rahman is working as an associate professor in Computer and Electrical Engineering department in North South University, Dhaka, Bangladesh. He received his Ph.D. in Computer Science from University of Calgary, Canada and Master's degree from University of Manitoba, Canada in 2007 and 2003 respectively. He has authored more than 100 peer reviewed journals and conference proceedings in the area of parallel, distributed, cloud computing, knowledge and data engineering. He is serving in editorial board of a number of journals. He serves as a program committee member of a number of international conferences organized by IEEE, ACM. His current research interest is in data mining particularly on financial, medical and educational data, VM consolidation in cloud, data replication on grid, computational finance, and image processing.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Computer and Information Science, University of Oregon, Eugene, OR, USA. [2]Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh.

Received: 30 November 2015 Accepted: 29 June 2016
Published online: 11 July 2016

References

1. Beloglazov A, Abawajy J, Buyya R (2011) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Futur Gener Comput Syst (FGCS) 28(5):755–768
2. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency and Computation: Practice and Experience (CCPE) 24(13):1397–1420
3. Ferreto TC, Netto MAS, Calheiros RN, De Rose CAF (2011) Server consolidation with migration control for virtualized data centers". Futur Gener Comput Syst 27(8):1027–1034
4. Beloglazov A, (2013) PhD Thesis: "Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing". Link: http://beloglazov.info/thesis.pdf. Accessed 04 Jul 2016
5. Calheiros RN, Ranjan R, Beloglazov A, Rose CAFD, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 41(1):23–50
6. CloudSim link: http://www.cloudbus.org/cloudsim/. Accessed 04 July 2016
7. Monil MAH, Qasim R, Rahman RM (2014) Incorporating migration control in VM selection strategies to enhance performance". Int J Web Appl (IJWA) 6(4):135–151
8. Monil MAH, Qasim R, Rahman RM (2014) "Energy-aware VM consolidation approach using combination of heuristics and migration control", 9th IEEE International Conference on Digital Information Management, pp. 74–79.
9. Farahnakian F, Ashraf A, Liljeberg P, Pahikkala T, Plosila J, Porres I, Tenhunen H "Energy-Aware Dynamic VM Consolidation in Cloud Data Centers Using Ant Colony System" 2014 IEEE 7th International Conference on Cloud Computing (CLOUD), pp. 104–111
10. Prevost J, Nagothu K, Kelley B, Jamshidi M (2011) "Prediction of cloud data center networks loads using stochastic and neural models," in Proc. IEEE System of Systems Engineering (SoSE) Conf., pp. 276–281, 27–30
11. Di S, Kondo D, Cirne W (2012) "Host load prediction in a Google compute cloud with a Bayesian model", in Proc. Of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Salt Lake City, UT, Nov. 10–16

12. Farahnakian F, Liljeberg P, Plosila J (2013) "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers," 2013 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 357–364.
13. Cingolani P, Alcala-Fdez J (2012) "jFuzzyLogic: A robust and flexible Fuzzy-Logic inference system language implementation", Proc. Intâl Conf. Fuzzy Systems, pp. 1–8
14. Verma A, Dasgupta G, Nayak TK, De P, Kothari R (2009) "Server workload analysis for power minimization using consolidation," in Proceedings of the 2009 USENIX Annual Technical Conference, pp. 28–28.
15. Cao Z, Dong S (2013) "Energy-Aware framework for virtual vachine vonsolidation in vloud vomputing" High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference, pp. pp. 1890–1895
16. Akula GS, Potluri A (2014) "Heuristics for migration with consolidation of ensembles of Virtual Machines" Sixth International Conference on Communication Systems and Networks Communication Systems and Networks (COMSNETS), pp. 1–4
17. Madani N, Lebbat A, Tallal S, Medromi H (2013) "New cloud consolidation architecture for electrical energy consumption management", IEEE Africon 2013 Conference, pp. 1–3.
18. Farahnakian F, Liljeberg P, Plosila J (2014) "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning", 22nd Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 500–507
19. Mastroianni C, Meo M, Papuzzo G (2013) "Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data Centers", IEEE Transaction of Cloud Computing, vol.1, pp. 215–228.
20. Barroso LA, Holzle U (2007) The case for energy-proportional computing. Computer 40(12):33–37
21. Duy TVT, Sato Y, Inoguchi Y (2010) "Performance evaluation of a green scheduling algorithm for energy savings in cloud computing", 2010 IEEE international symposium on parallel & distributed processing. Workshops and Phd Forum (IPDPSW), Atlanta, pp. 1–8
22. Feller E, Rilling L, Morin C (2011) "Energy-aware ant colony based workload placement in clouds", Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing, pp. 26–33.
23. Mills K, Filliben J, Dabrowski C (2011) "Comparing VM-placement algorithms for on-demand clouds, IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), Athens, 2011, pp. 91–98.
24. Srikantaiah S, Kansal A, Zhao F (2008) Energy aware consolidation for cloud computing, Proceedings of the 2008 conference on Power aware computing and systems, pp. 10–10.
25. Park KS, Pai VS (2006) CoMon: a mostly-scalable monitoring system for planet- Lab. ACM SIGOPS Operating Syst Rev 40(1):65–74
26. Monil MAH, Rahman RM (2015) "Fuzzy Logic Based Energy Aware VM Consolidation", 8th International Conference on Internet and Distributed Computing Systems (IDCS 2015), Windsor, U.K., September 2–4, pp. 223–227
27. Fuzzy Trapezoidal membership function. Link: http://www.mathworks.com/help/fuzzy/trapmf.html?requestedDomain=www.mathworks.com. Accessed 04 Jul 2016
28. Beloglazov A, Buyya R, Lee YC, Zomaya A (2011) "A taxonomy and survey of energy-efficient data centers and Cloud computing systems," Advances in Computers, M. Zelkowitz (ed.), vol. 82, Elsevier, pp. 47–111