Active & Idle Virtual Machine Migration Algorithm for VM Consolidation

Md. Kaviul Hossain
Computer Science and Engineering
BRAC Univsersity
Dhaka, Bangladesh
kaviuln@gmail.com

Mutasimur Rahman
Computer Science and Engineering
BRAC Univsersity
Dhaka, Bangladesh
mutasimur@gmail.com

Azrin Hossain
Computer Science and Engineering
BRAC Univsersity
Dhaka, Bangladesh
azrinhossain16@gmail.com

Md. Motaharul Islam
Computer Science and Engineering
United International University
Dhaka, Bangladesh
motaharul@cse.uiu.ac.bd

***Abstract*—Cloud computing is an internet-based computing system enabling convenient, flexible, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interactions. Although cloud is a virtual, its implementation includes physical and wired connectivity among servers, hard-drives and other network devices resulting in large number of data centers. As such power and energy consumption increase total carbon footprint which is very hazardous for the environment. Green Cloud Computing is a concept of consuming less physical power and saving energy by making the infrastructure more virtual. With the help of the Ant Colony System algorithm, it is possible to consolidate the Virtual Machines and make Cloud Computing more virtual than it already is. This algorithm describes a mechanism of indirect coordination, through the environment, between agents and their actions. Thus, Green Cloud Computing can be easily globalized**

***Index Terms*—Virtual machine, hyper-visor, state-transition rule, physical machine, meta-heuristic, virtualization.**

I. Introduction

Cloud computing is a modern-day necessity that gives the privilege of storing, accessing and controlling day-to-day important documents and other resources. It is making significant changes in our revolutionized world by bringing it to our fingertip. The demand for cloud infrastructure is increasing day by day. Many technological companies such as Amazon, Google and Microsoft are building large-scale cloud data centers around the world which is leading to high energy consumption, high operating cost, and higher carbon emissions [1]. The main objective of our research is to consolidate virtual machine and globalize green cloud computing. Virtualization is a powerful technology that facilitates better use of the available data center resources using a technique called Virtual Machine (VM) consolidation which involves gathering of several virtual machines into a single physical server. To address the problem of high energy usage, it is necessary to eliminate inefficiencies and waste in the way electricity is delivered to computing resources, and in the way these resources are utilized to serve application workloads. This can be done by improving the physical infrastructure of data centers as well as resource allocation and management algorithms.

Recent studies have shown that the power consumed by servers can be assumed to be linear with CPU utilization. An active but idle server can burn about 50% to 70% of the power consumed by the server working at full load [2]. This scenario can be sketched using the following power model:

$$P(u) = k.P_{max} + (1 - k).P_{max}.u \tag{1}$$

Here, P_{max} is the maximum power consumed by a fully loaded server. k is the fraction of power consumption in idle state. u is the CPU utilization of the server.

Virtual Machine consolidation involves live migration, which is the capability of transferring a VM between physical servers with a close to zero down time is an effective way to improve the utilization of resources and energy efficiency in cloud data centers. The computer's ability to store, retrieve and manipulate large amounts of data rapidly and cheaply has led to its widespread use. But, at each stage of computer's life throughout its use, and into its disposal, it exhibits environmental problems [3]. So, in this paper, we are focusing on globalizing green cloud computing. Green computing is the environmentally responsible use of computers and related resources. Such practices include the implementation of energy-efficient central processing units (CPUs), servers and peripherals as well as reduced resource consumption and proper disposal of electronic waste (e-waste). Our key contributions in this work can be summarized as follows:

- We have maintain integrity of the cloud architecture and consolidating the virtual machine placement problem in collaboration of a meta-heuristic algorithm called the Order Exchange Migration Ant Colony System (OEMACS).

- We have designed an algorithm that will solve the VM Placement problem to a great extent with the objective of consolidating the cloud architecture.
- We have ensured Green Cloud Computing by reducing high thermal throttle in physical machines like servers that host the Virtual Machines.
- We have migrated idle or zombie VMs from one server to another containing maximum idle VMs, which has ensure no unwanted lag of power supply for the actively working VMs.

So, in order to make cloud computing safer, more efficient and environment friendlier, use of stigmergic algorithm is inevitable. With the use of an auto catalytic, multi-greedy heuristic algorithm, it is possible to keep the energy emission rate of servers bounded within a considerable range and thus we can ensure green computing.

This paper consists of nine sections. Section II shows the Cloud Modeling and its internal networking system. The concept of idle virtual machine is referred in Section III. Our parent algorithm- Order Exchange Migration Ant Colony System (OEMACS) has been depicted in Section IV. The VM Scheduling Policy is described in Section V. Our multi-objective approach is described in section VI whereas Section VII focuses on our proposed model AIVMM. The proposed algorithm with its mechanism and working system has been discussed in this elaborate section. The evaluation of our proposed algorithm is in Section VIII. Lastly, we ended the paper with relevant suggestion and future possibility of the system's enhancement in Section IX.

II. Cloud Modeling and its Federated Networks

A. *Federated Networks in Cloud Architecture*

Current Cloud Computing providers have several data centers at different geographical locations over the Internet in order to optimally serve customers' needs around the world. However, existing systems does not support mechanisms and policies for dynamically coordinating load-shredding among different data centers in order to determine optimal location for hosting application services to achieve reasonable service satisfaction levels. Further, the Cloud service providers are unable to predict geographic distribution of users consuming their services, hence the load coordination must happen automatically, and distribution of services must change in response to changes in the load behaviour. Figure 2 depicts such a service-oriented Cloud computing architecture consisting of service consumer's brokering and provider's coordinator services that support utility-driven inter networking of clouds[12]: application scheduling, resource allocation, and workload migration.

The Cloud coordinator component is instantiated by each data center that: (i) exports the Cloud services, both infrastructure and platform-level, to the federation; (ii) keeps track of load on the data center and undertakes negotiation with other Cloud providers for dynamic scaling of services across multiple data centers for handling the peak in demands; and (iii) monitors the application execution and oversees that agreed SLAs are delivered.

The Cloud Exchange (CEx) acts as a market maker for bringing together service providers and consumers [?]. It aggregates the infrastructure demands from the Cloud brokers and evaluates them against the available supply currently published by the Cloud Coordinators.

B. *Flow of data in cloud data centers*

Figure 3 shows the data flow system in a cloud data center. How data is managed and simulated in a cloud data center can be easily sketched and comprehended from this figure.

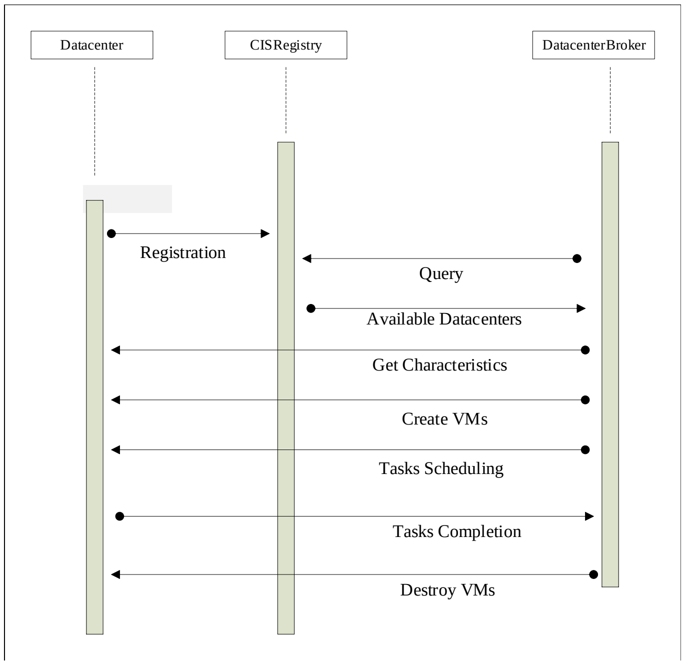


Fig. 1. Data-flow in a Cloud Data-center

III. Idle or 'Zombie' Virtual Machines

Idle VMs are virtual machines that remain running even though they are no longer used or the time time being to complete a project or task. Whenever a scheduled task or a project ends the working process terminates and the VM becomes idle. In such a state, the machine stays active and runs but no task gets executed until another task is appointed to it. These Idle (aka 'Zombie') VMs consume CPU, memory, storage resources and bandwidths that could be used by other active machines. As such, other actively engaged virtual machines in the same server lags and do not get required resources in time which hinders their (active VMs) ongoing projects and tasks.

IV. Order Exchange Migration Ant Colony System Algorithm (OEMACS)

OEMACS is a virtual machine migration algorithm that helps in migrating VMs from one server to another using Order Exchange Migration (OEM) algorithm along with Ant

Colony System Algorithm (ACS). The collaboration of these two renowned algorithms makes the migration of VMs a lot easier and more efficient. In this paper, we are designing an ACS-with OEM operations to decrease the number of active servers and then reduce energy usage for cloud computing. The resulting OEMACS algorithm looks for a result that has less numbers of servers to host all VMs. We identify the minimal number as M_{min} in the possible globally best solution S_{gb}. At the beginning $M_{min} = n$ where N is the number of VMs. It implies that S_{gb}'s original globally best possible approach is to install the N VMs on N servers with one VM corresponding to the one server. We add a pheromone value $\tau(k, j)$ for every pair of VM_k and $VM_j (j \neq k)$ then assign it as $\tau_0 = (N^{-1})$. The pheromone value shows the choice of two VMs to be allocated to the same server. With the help of the state transition rule, the ACS i.e. Ant Colony System algorithm, an agent (ant) selects a suitable server I from server set I_j for the VM in transition.

V. VM Scheduling Policy

Scheduling means to establish policies to regulate the sequence of task to be executed using a processor system. There have been different forms of scheduling algorithm available in the dispersed computing system, and work scheduling is part on them. The key benefit of job scheduling algorithm is to attain a great execution computing and the most excellent system all through. Scheduling similarly manages the existing computer memory and better scheduling plan provides utmost application of resource [9]

In a cloud architecture, there are mostly 2 types of VM Scheduling policies:

- Space-shared policy
- Time-shared policy

In space-shared policy, the VMs complete their tasks by turn. For example: If there are 2 VMs in a host and both the VMs are allocated 2 tasks each; in a space-shared scheduling policy VM1 will complete its task by using the full CPU core all by itself while VM2 will wait for its turn.

On the other hand, in time-shared policy, the VMs will complete there tasks parallelly by sharing the CPU core. Both the VMs will share the core will finishing their first Cloudlet and once done they will move on to the second one. Figure 3 depicts the concept.

VI. A multi-objective approach towards VMM

Virtual Machine Migration (VMM) is a common phenomenon in cloud architecture. For the assurance of continuous connectivity and better performance, migration of virtual machine from one server to another is a must. In order to migrate a VM from one server to another and to accommodate the VM with required resource and enough power in the destination server, some parameters are needed to be satisfied. These parameters are:

- Service Legal Agreement (SLA)
- Resource Utilization
- Power or energy utilization

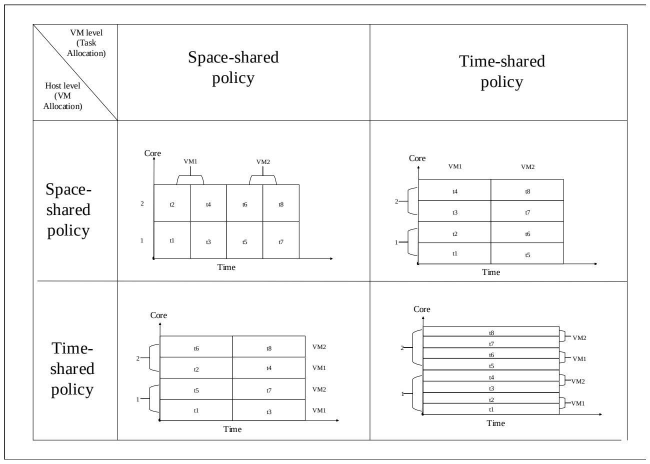


Fig. 2. Scheduling Policy In Cloud

A. Service Level Agreement (SLA)

Maintaining the Quality of Service (QoS) is an important requirement in cloud computing. QoS requirements are commonly formalized as SLAs, which specify enterprise service-level requirements for data center in terms of minimum latency or maximum response time. A workload independent metric called the SLA Violations (SLAV) is used to measure the SLA delivered in an IaaS cloud. It measures both violations due to Over-utilization (SLAV(O)) and violations due to Migrations (SLAV(M)). Both these violations characterize the level of SLA Violations by the infrastructure independently and with equal importance.

$$SLA = \frac{\Sigma Req - AReq}{\Sigma Req} \tag{2}$$

Here, AR_{eq} denotes allocated requests

$$SLAV = SLAV(O) \times SLAV(M) \tag{3}$$

SLAV(O) represents the percentage of time during which active servers have experienced the resource utilization of 100%. It is defined as:

$$SLAV(O) = (M^{-1}) \sum_{i=1}^{M} \frac{T_{u_i}}{T_{a_i}} \tag{4}$$

Here, M is the number of servers, T_{u_i} represents total time i has experienced the utilization of 100% leading to a SLAV; T_{a_i} is the total duration of server i being in the active state.

SLAV(M) represents the overall performance degradation by VMs due to migrations. It is computed as:

$$SLAV(M) = (N^{-1}) \sum_{j=1}^{N} \frac{C_{d_j}}{C_{r_j}} \tag{5}$$

Here, N is the total number of virtual machines. C_{d_j} is the performance degradation estimation of VM_j by migration, C_{r_j} is the total CPU capacity requested by VM_j in its lifetime.

B. Resource Utilization

To make full use of the resources, we have considered the CPU and the RAM to define the resource wastage in Eq.(6) Here, W_R represents resource wastage, UC and UM denotes CPU and RAM utilization respectively. When W_R is large, more energy gets wasted.

$$W_R = \frac{|UC - UM|}{\sqrt{(UC)^2 + (UM)^2}} \tag{6}$$

C. Energy Utilization

Our focus is to minimize and utilize proper energy consumption as high use of energy creates high level radiation and increases pollution. To ensure green cloud computing our foremost target is to prevent idle virtual machines from disrupting the necessary power consumption for the actively working machines. Past researches have shown that idle virtual machines consume nearly 50% to 70% of the total server power and as such the neighboring virtual machines working with full load cannot get the salient power proportionately in time. Thus, there remains a lag in service.

D. Proposed multi-objective approach

Based on the objectives presented above [5] [6] [**?**], we designed a model that tends to optimize the utilization of resources used and energy consumed. These models try to minimize the resource wastage, SLA violations and excess energy consumption in the cloud infrastructure. The models are as follows:

$$Min \sum_{i=1}^{n} S_i.SLA \tag{7}$$

$$Min \sum_{i=1}^{n} W_R = \sum_{i=1}^{n} [S_i \times \frac{|UC - UM|}{\sqrt{(UC)^2 + (UM)^2}}] \tag{8}$$

$$Min \sum_{i=1}^{n} P(u) = \sum_{i=1}^{n} [S_i \times (k \times P_{max} + (1-k) \times P_{max} \times u)] \tag{9}$$

$$u = UC_i = \frac{\sum_{i=1}^{n} V_i.UC_j}{\sum_{i=1}^{n} V_i}; 0 \leq V_i \leq 1; 0 \leq S_i \leq 1 \tag{10}$$

For S_i, possible constraints are

$$\sum_{i=1}^{n} S_i.UC \leq R_{cpu} \tag{11}$$

$$\sum_{i=1}^{n} S_i.UM \leq R_{mem} \tag{12}$$

VII. Proposed optimized model of AIVMM

The algorithm with the help of which actively working virtual machines and idle VMs can be migrated from one overloaded or non-overloaded server to another non-overloaded server with the view to make power utilization efficient can be called Active Idle Virtual Machine Migration (AIVMM). Since our main problem regarding the Cloud Infrastructure and Virtual Machine Architecture is the massive energy consumption of the idle virtual machines which is around 50% - 70% of the total server input power, we plan to solve this problem by separating the idle VMs from the actively working VMs in a server. In order to do so, we need to design a model that can detect the number of actively working and idle VMs in a server at a given time.

Virtual Machines act as independent, enclosed machines that use the resources they are allocated with during the process of virtualization. A VM is comprised of Processor (CPU), Memory (RAM), Space (Hard drive) and Network Bandwidth of its own. Based on the usage of CPU and RAM, we can easily detect which VM is actively busy and which VM is sitting idle. On using that information, it becomes quite simple to migrate all idle or active VMs from one server to another.

A. AIVMM Model

Our idea is to exchange all idle virtual machines of a single server with the actively working fully loaded ones of a non-overloaded server in order to ensure maximum number of active VMs in a single server at most of the times. As such, the actively working VMs will face no interruption in their power consumption since idle VMs tend to consume 50% - 70% of the total power of host server.

Judging from the CPU and RAM utilization of a VM, for an instantaneous time t, the following scenarios can co-exist in a cloud environment:

- $U_{m_j} > U - c_j[RAM > CPU]$
- $U_{m_j} < U - c_j[RAM < CPU]$
- $U_{m_j} \approx U - c_j[RAM \approx CPU]$

Here, U_{m_j} represents the memory utilization of a VM, U_{c_j} represents the processor or CPU utilization of a VM and $0 \leq U_{m_j} \leq 1, 0 \leq U_{c_j} \leq 1$ i.e U_{m_j} and U_{c_j} represents the percentage of RAM CPU utilization.

B. Mathematical Formulation, Constraints & Explanation

In the process of calculating total number of active and idle VMs of a single server at an instant t, we will use the resource utilization percentage of every single virtual machine. As a result, it will become easier to decide whether to migrate active VMs or idle ones. The possible scenarios of our applied method would be:

a) When $RAM > CPU$

- $U_{m_j} >> U_{c_j}$
- $U_{m_j} \approx U_{c_j}$

In (i), the RAM utilization is much greater than the CPU utilization which denotes that the VM is idle as it is using RAM or memory to keep itself on but not using much CPU compared to its RAM usage.

In (ii), the RAM utilization is almost or exactly equal to the CPU utilization which denotes that the virtual machine is actively working with full load. Although the scenario of RAM usage being exactly equal to the CPU usage can be very rare.

b) When $RAM < CPU$

- $U_{m_j} << U_{c_j}$
- $U_{m_j} \approx U_{c_j}$

When the CPU of a VM is working with full load but it is saving the data in cache or temporary memory rather than the RAM, then the CPU usage of the machine seems to be greater than the volatile memory i.e. RAM usage.

Here, in (i) the CPU utilization is much greater than the RAM utilization which denotes that the VM is active, but it is saving its data in the temporary memory or cache memory. A such the virtual machine is online and working with full load.

Furthermore, in (ii) the RAM utilization is almost or exactly equal to the CPU utilization which denotes that the virtual machine is actively working with full load. Although the scenario of RAM usage being exactly equal to the CPU usage can be very rare.

In order to calculate the total number of idle virtual machines in a server for an instant t, the following equations can be used:

$$V_0 = U_{m_j} - U_{c_j} \tag{13}$$

$$V_0` = \begin{cases} 0, & 0 \leq V_0 \leq 0.3 \\ 1, & 0.4 \leq V_0 \leq (0.9 \approx 1) \end{cases} \tag{14}$$

$$V_idle = \begin{cases} \sum_{0=1}^{N1} V_0`, & [V_0`N1] \\ 0, & otherwise \end{cases} \tag{15}$$

Here, N1 is a set of virtual machines where $U_{m_j} > U_{c_j}$, V_0 represents the difference of RAM and CPU utilization while Vo‘ represents a piecewise function of V_0. Vidle denotes the number of idle virtual machines in a server. Again, in order to calculate the total number of working virtual machines in a server for an instant t, the following equations can be used:

$$V_a = U_{c_j} - U_{m_j} \tag{16}$$

$$V_a` = 1 if 0 \leq V_a \leq (0.9 \approx 1) \tag{17}$$

$$V_{active} = \begin{cases} \sum_{a=1}^{N2+N3} V_a` + c, & [V_a`(N2 + N3)] \\ c, & otherwise \end{cases} \tag{18}$$

$$c = \sum V_a \tag{19}$$

Here, N2 is a set of virtual machines where $U_{c_j} > U_{m_j}$, Va represents the difference of CPU and RAM utilization of VM while $V`_a$ represents a function of Va. V_{active}denotes the number of actively working virtual machines in a server, c represents the number of active VMs from Eq.(13)

When the number of actively working VMs and idle VMs at any time t, in any server i turns out to be equal, then the following equations can be used to calculate the total CPU and memory utilization of the VMs:

$$UC_{va} = \sum_{j=1}^{V_{active}} U_{c_j} \times c_j \tag{20}$$

$$UM_{va} = \sum_{j=1}^{V_{active}} U_{m_j} \times m_j \tag{21}$$

Here, UC_{va} and UM_{va} denotes the CPU and Memory utilization of actively working virtual machines respectively, while U_{c_j} and U_{m_j} represents that of a VM. Also, c_j and m_j represents the total CPU and memory capacity of a VM.

$$UC_{vi} = \sum_{j=1}^{V_{idle}} U_{c_j} \times c_j \tag{22}$$

$$UM_{vi} = \sum_{j=1}^{V_{idle}} U_{m_j} \times m_j \tag{23}$$

The equations to calculate the total CPU and RAM usage of both an actively working and an idle virtual machine are respectively:

$$TU_{va} = UC_{va} + UM_{va} \tag{24}$$

$$TU_{vi} = UC_{vi} + UM_{vi} \tag{25}$$

Here TU_{va} and TU_{vi} represent the total utilization of both CPU and RAM by the active and idle VMs respectively, UC_{va} and UMva represents the CPU and memory utilization of the active VMs while UC_{vi} and UM_{vi} represents those of the idle VMs.

C. Architectural Design

The Active & Idle Virtual Machine Migration algorithm can consolidate the energy consumption error in a cloud infrastructure in the form of random and efficient VM Migration (VMM). The following design depicts the scenario:

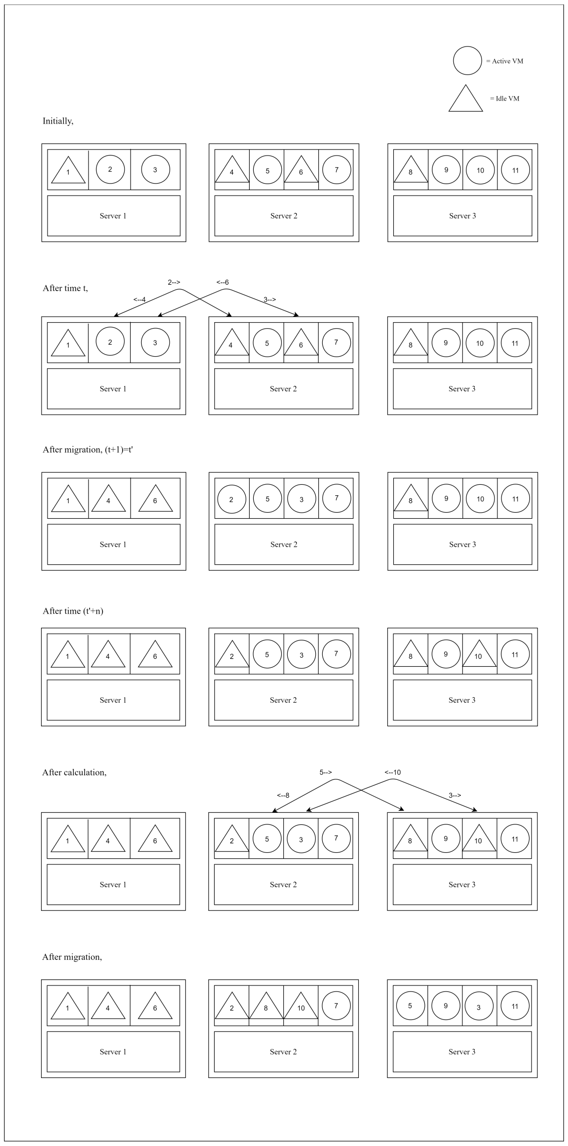


Fig. 3. The architectural design of Active & Idle Virtual Machine Migration (AIVMM)

D. The complete AIVMM algorithm

The algorithm AIVMM is composed of ACS (Ant Colony System Algorithm) [4] [7] and OEM (Order and Exchange Migration) along with VMM (Virtual Machine Migration). The complete algorithm is sketched below:

Step 1: Initialization. Compute the number of active and idle VMs in a single host server. Take the number of active VMs as Vactive and the number of idle VMs as Vidle. Compare Vactive and Vidle. [Three possible scenarios can occur. Such as- i) $V_{idle} > V_{active}$ii) $V_{idle} < V_{active}$ and iii) $V_{idle} = V_{active}$ Step 2: The parameter for ACS is set to o. The feasible globally best solution is set as S_gb for placing N VMs on N servers. Thus, the number of minimum servers is set to $M_{min} = N$. Set iteration t=1 and maximum iteration as T.

Step 3: Set $M_t = M_{min} - 1$. In each iteration, m ants construct m solutions and perform local pheromone updating on each solution. Step 4: The fitness function f(S) is applied in order to evaluate the fitness value of the constructed solution.

Step 5: The best solution S_b of the current iteration is set after evaluating the fitness value of the constructed solution. If S_b is feasible, S_{gb} is updated as Sb and $M_{min} = f1(S_b)$ is set. Otherwise OEM local search is performed on Sb. S_{gb} and M_{min} are updated respectively if local search succeeds.

Step 6: After S_{gb} and M_{min} are locally updated, global pheromone update is eventually done on S_b and S_{gb}.

Step 7: Check if t is less than or equal to T. If not equal, then set $t = t + 1$ and go to Step 3. Otherwise terminate the algorithm.

Step 8: After t terminates, calculate $V`_{active}$ and $V`_{idle}$ of the host server. If $V`_{active} = V_{active}$ and $V`_{idle} = V_{idle}$, then move forward to step 9. Otherwise update V'active and $V`_idle$ and then move to step 9.

Step 9 (a) If $V_{idle} > V_{active}$, then migrate all the actively working VMs from host server to a nearby (ACS) non-overloaded (OEM) server with the opposite scenario i.e. the server in which $V_{idle} < V_{active}$. The idle VMs of the destination server will be exchanged with the actively working ones from the host.

(b) If $V_{idle} < V_{active}$, then there will be no migration.

(c) If $V_{idle} = V_{active}$, calculate the CPU utilization $(UC_{va} \& UM_{va})$ and memory utilization $(UC_{vi} \& UM_{vi})$ of the VMs (both actively working and idle) and compare the total utilization of both type VMs using Eqn. (24) & (25)

E. Limitations of AIVMM

Although AIVMM algorithm helps to exchange active and idle virtual machines among servers and decrease unnecessary power consumption, this model in under development and in its testing phase. There are some limitations viable in the current model. They are:

- Time complexity of the algorithm is an important factor. The run-time complexity of AIVMM is quadratic and greater than that of OEMACS.
- Since this algorithm helps to migrate idle and actively working VMs from one server to other, it is a continuous process. An idle VM can start working anytime in the process, even when its being migrated while an actively working VM can cease to work and become idle any time in the process.

- Migration of all zombie (idle) virtual machines to a different server, will leave the source server behind with all actively working VMs in it. As such, usage of CPU will be to the fullest and it will lead to thermal throttle of the server at some point.

VIII. Evaluation Results

We have successfully evaluated our algorithm- AIVMM in an advanced simulator CloudSim that helps to simulate cloud environments with necessary resources and infrastructure. The hardware device that we used in this purpose had the following configurations:

- Hardware - Intel Core i5-7200U 2.5 GHz Processor, 64-bit CPU, 1TB Hard Drive, 4GB DDR4 Memory
- Software - Eclips IDE
- Environment - CloudSim 3.0.3
- Language - Java
- OS - Windows 10 Education
- Network Bandwidth - 40 Mbps

We first formulized the AIVMM algorithm using the Eclipse IDE in Java. Then we collaborated the algorithm with OEMACS as AIVMM is a dependent algorithm.

We conducted our experiments in CloudSim 3.0.3 that contain all the necessary classes required to simulate a cloud environment. A brief description of CloudSim and how it operates as given below.

A. CloudSim & its architecture

CloudSim 3.0.3 is a very useful simulation tool [8].

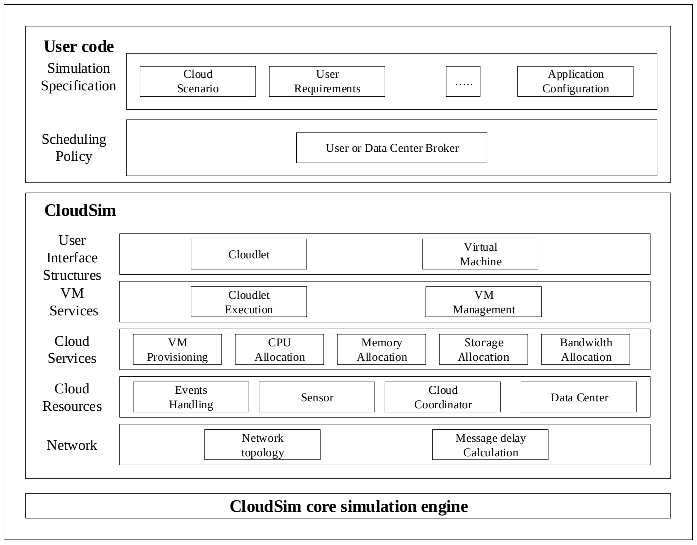


Fig. 4. Layered Cloud-sim Architecture

B. Evaluation

Figure 7 and 8 represents the amount of time and memory required respectively for the simulation instantiation when the number of virtual machines increases in a host server. The amount of time required is exponential to the number of VMs while the amount of memory required has a linear relationship.

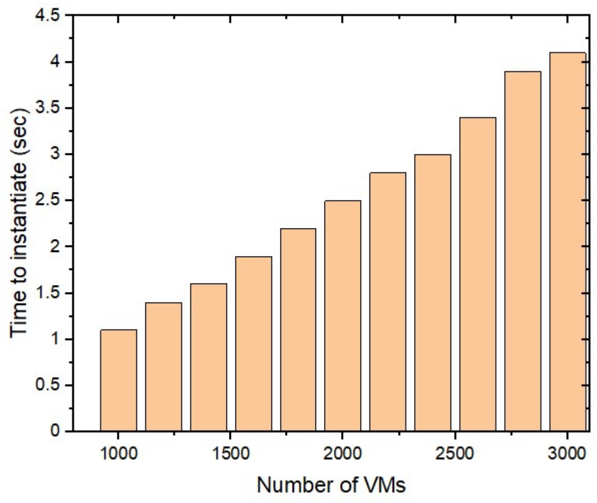


Fig. 5. Time to Simulation instantiation

Regarding time overhead related to simulation instantiation, the growth in terms of time increases exponentially with the number of virtual machines. Here, the time required to 3000 virtual machines is approximately 4 seconds, which is reasonable and standard for an experiment.

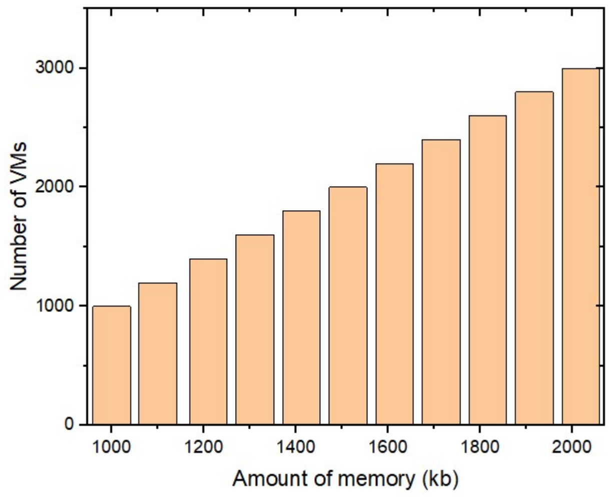


Fig. 6. Memory to Simulation instantiation

The growth in memory allocation is linear, with an experiment with 3000 virtual machines demanding approximately 2.1 MB RAM. It makes our simulation moderately suitable to run in desktop computers with standard processing powers.

Figure 9 denotes the comparison between our parent algorithm (OEMACS) and collaboration of all three algorithms mentioned in this paper (OEMACS and AIVMM). The graph is sketched with time to migrate VMs against 15 virtual machines.

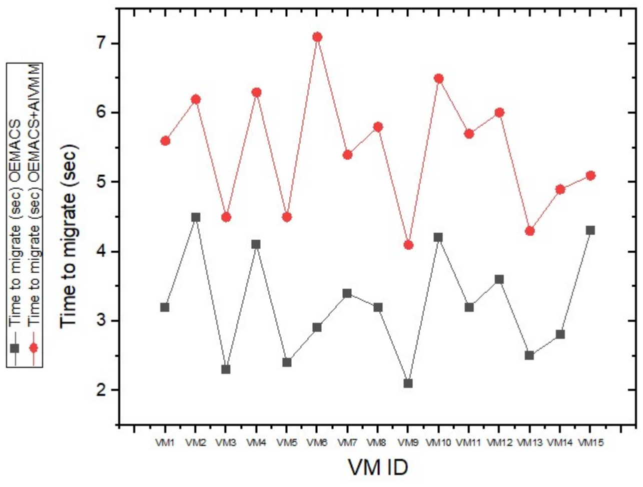


Fig. 7. OEMACS Vs (OEMACS+AIVMM)

IX. Conclusions & future works

Virtual Machine Consolidation is a very salient feature of today's cloud infrastructure. With the help of continuous VM migration, cloud has become more resilient and lenient. But because of thermal throttle and excessive heat emission, the process of virtual machine migration has become a major issue. We have designed Active Idle Virtual Machine Migration algorithm which has not only solved the problem but has also consolidated the process to a great extent keeping the collateral parameters intact. Although there are limitations regarding run-time and migration threshold, in near future we plan to solve these issues by collaborating Migration of Idle Machine via Parking Server algorithm along with Active Idle Virtual Machine Migration algorithm.

References

[1] X. Liu, J. Zhang, "An Energy Efficient Ant Colony System for Virtual Machine Placement in Cloud Computing," Evolutionary Computation, IEEE Transactions on, vol. 22, no. 1, pp.30-65, 2018.

[2] A. Ashraf and I. Porres, "Using ant colony system to consolidate multiple web applications in a cloud environment," in 22nd Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2014, pp. 482–489.

[3] M. Dorigo, V. Maniezzo, and A.Colorni, "The ant system: optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics–Part B, vol. 26, No. 2, pp. 29–41, 1996.

[4] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," Evolutionary Computation, IEEE Transactions on, vol. 1, no. 1, pp. 53–66, 1997.

[5] C. Yuan, and X. Sun, "Server Consolidation Based on Culture Multiple-Ant-Colony Algorithm in Cloud Computing," Sensors 2019

[6] L.M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," Proceedings of IEEE International Conference on Evolutionary Computation, IEEE-EC 96, IEEE Press, 1996, pp. 622–627.

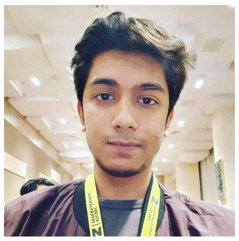
[7] D. Fogel, "Applying evolutionary programming to selected traveling salesman problems," Cybernetics and Systems: An International Journal, vol. 24, pp. 27–36, 1993.

[8] Y. Gao, H. Guan, L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," Journal of Computer and System Sciences 79 (2013) 1230-1242

[9] J. Kommeri, T. Niemi, J. Nurminen, "Energy efficiency of dynamic management of virtual cluster with heterogenous hardware," (2015)

[10] R. Buyya, R. Ranjan, R. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities."

[11] Y. Tohidirad, S. Abdezadeh, M. Moradi, "Virtual Machine Scheduling in Cloud Computing Environment," International Journal of Managing Public Sector Information and Communication Technologies (IJMPICT), Vol. 6, No. 4, December 2015



Md. Kaviul Hossain has been pursuing his B. Sc in Computer Science Engineering from BRAC University, Bangladesh for the last three years. His field of interest is Cloud Computing, Big Data Management and Block Chain Infrastructure. His research field is based on Cloud Architecture, VM Consolidation and Modification.

Mutasimur Rahman is currently on his 4th year of B. Sc in Computer Science Engineering from BRAC University, Bangladesh. His field of interest is Cloud Computing, Networking and Network Security. His research field is VM consolidation and virtualization.

Azrin Hossain is a last year undergraduate student of Brac University, Bangladesh. She is studying Computer Science and Engineering. She is interested in cloud computing and cyber security. Her research field is VM consolidation and virtualization.

Md. Motaharul Islam has been working as Professor in the Dept. of Computer Science and Engineering at United International University(UIU). He has been awarded PhD in Computer Engineering from Kyung Hee University, South Korea in the field of Smart Internet of Things(IoT). He has een working as an Associate Editor of International Journal of Computers and Applications, Taylor & Francis Group, UK since 2017.