A Peer to Peer Resource Provisioning Scheme For Cloud Computing Environment Using Multi Attribute Utility Theory

Rayhanur Rahman*, Asif Imran*, Alim Ul Gias* and Kazi Sakib†
Institute of Information Technology, University of Dhaka
Ramna, Dhaka–1000
*{bit0101, bit0119, bit0103}@iit.du.ac.bd
†sakib@univdhaka.edu

***Abstract*—Resource provisioning is critical for cloud computing because it manages the virtual machines (VM) and allocated resources. Traditional resource provisioning schemes make decisions based on centralized configuration and global calculation of resource allocation. However these provisioning frameworks become hindered in large deployment followed by Service Level Objectives (SLO) violation and those are exposed to single point failure as well. The proposed provisioning scheme is based on Peer to Peer (P2P) architecture with no lone decision maker. Each node in the data center makes its own provisioning decision regarding VM allocation and migration which are resolved with Multi Attribute Utility Theory (MAUT) methods. Simulation experiments demonstrate that in comparison with centralized schemes, the proposed scheme generates 60.27% less SLO violations and 83.58% less VM migrations on average. Such outcome proves that the system can manage its resources better compared to centralized scheme.**

***Index Terms*—Cloud Computing, Resource Provisioning, Peer to Peer, Multi Attribute Utility Theory**

I. Introduction

From technical point of view, virtualization holds the key to deploying dynamic number of applications in cloud computing environment. In datacenters physical machines (PM) host VMs depending on its capacity. VMs are configured with proper Application Environments in order to host the applications. VMs are also loosely coupled with PMs as those can be reconfigured or migrated from one PM to another. Configuration and utilization of these computational resources in the datacenter with dynamic adjustment is called resource provisioning [1], [2]. It also scales applications horizontally and vertically according to their resource need and consumer provided constraints. This process is critical to the effectiveness of cloud computing because it manages all deployed applications intelligently to keep consumers satisfied through fulfilling SLO.

Traditionally, a resource provisioning scheme consists of two phases [3]. These are (i) initial static provisioning involving deploying applications in VMs and mapping VMs to PMs as well as (ii) runtime provisioning including reconfiguration and migration of VMs across the datacenters in time variant workloads. Phase (i) is performed at initial startup of datacenter as well as maintenance purpose whereas, phase (ii) is performed continuously in the runtime. In Phase (i), the issue of mapping between static number of VMs and PMs can be resolved with optimization techniques such as vector bin packing [4], [5]. On the other hand Phase (ii) is the most critical phase of resource provisioning because, datacenter linger on large portion of their operation span here. As research has been done extensively regarding the Phase (i) [3], this paper solely focuses on Phase (ii).

Most of the provisioning schemes found in the literature feature statistical analysis [6], utility computing [7], reinforcement learning [8] and combinatorial optimization techniques [4]. Although those work well in fulfilling SLO, with the increase of datacenter workload, centralized architecture becomes impractical and inefficient [9] due to scalability issues. Moreover computing a global or sub-optimal configuration on the runtime of a large datacenter poses a serious impediment to achieve SLO. Thus scalability issue and single point failure threat stand in the way of provisioning effectively.

The contribution of this paper is a decentralized resource provisioning scheme which utilizes P2P [10], [11] configuration of PMs in the datacenter. In P2P architecture, there is no dedicated client or server nodes in the network resulting in making the provisioning scheme inherently decentralized and scalable. For the sake of avoiding network and computational bottlenecks, instead of maintaining global awareness the system considers local awareness from the neighborhood peers. In addition, each node makes its own provisioning decision regarding VM allocation and migration. Those decisions are affected by numerous criteria such as computational resource availability, network bandwidth, migration cost and SLO constraints. This is why Multi Attribute Utility Theory (MAUT) [12], [13] methods are used to produce the best decision in such trade-off situations. Thus decentralized organization of nodes with each node doing its own job ensures that the system is scalable and resilient to single point failure.

For the simulation, the proposed provisioning scheme and underlying datacenter environment have been developed in C# programming language. In addition, a centralized provisioning scheme has also been implemented which uses a single global decision maker (GDM) and computes provisioning decision based on first fit and first fit decreasing bin packing schemes

[14] referred as FF and FFD in this paper. Those were then tested to observe their SLO violation, migration and CPU utilization for a large number of VMs. The outcome concluded with the proposed provisioning scheme generating a significantly less number of SLO violations and migrations which is about 60.27% and 83.58% respectively paying the penalty of slightly lower CPU utilization. Such outcome proves the proposed system provisions resources better than centralized architectures.

II. Related Work

In the literature several resource provisioning schemes have been proposed. Most of those approaches perform centralized provisioning and scaling system through a GDM. In this section, first centralized and then decentralized schemes will be discussed.

In [7] an autonomous computing agent for datacenter is proposed. It transforms physical machines to a set of rational agents to manage their own behavior without human intervention. Although the system behaves adaptively without any explicitly defined decision model, it lacks scalability due to the dependency upon a single GDM.

Reinforcement learning (RL) based provisioning scheme has been proposed in [8]. Instead of having any predefined model, it uses decompositional RL method to allocate resource dynamically. However, centralized architecture of this framework and utility calculation of all the nodes in the datacenter make the provisioning scheme unable to scale with the number of VMs increasing and avoid single point failure threat.

Zhang et al. proposed a resource management policy to perform load balancing based on VM performance and resource demand forecasts [6]. By utilizing statistical analysis, the system manages to decrease resource wastage in both peak and off peak hours by a significant margin. However, the system is prone to miscalculate the forecast and suffer from single point failure due to dependency on a GDM.

A decentralized decision making based resource provisioning scheme is proposed by Chieu et al. aiming to eliminate the threat of centralized provisioning [3]. A distributed lightweight resource management system called Distributed Capacity Agent Manager (DCAM) is used in this framework. However, this decision making framework uses a central database for storing all PMs and VMs information that makes the system inherently centralized and thus difficult to scale.

Onat Yazir et al. proposed a decentralized resource provisioning scheme using PROMETHEE II, ELECTRE III and PAMSSEM II outranking methods [15]. This framework considers each PM as an independent decision making module. Provisioning information of other PMs are supplied from a global information source. Scalability is in question as the source proves to be yet another centralized implementation.

In [16], a distributed hash table based cloud framework named *cloud peer* is proposed. It deals with how real life P2P architectures can handle cloud monitoring, routing architecture, service discovery, load balancing and message passing schemes. However, the work focused little on VM allocation and migration policies.

In addition, various optimization techniques have been implemented to minimize server sprawling and SLO violation. Constraint Satisfaction Problem and NP Hard optimization such as bin packing and integer programming are utilized to host a fixed number of VMs into minimum number of PMs in such a manner so that those issue less migration [4], [17], [5]. Those are similar to the previously discussed systems because of the presence of a GDM. Although being effective in small to medium sized datacenter, performance of these schemes in extremely large datacenters is still a major concern.

In principle, these systems recalculates the whole datacenter configurations periodically through centralized procedure. Consequently suffering from system lag and outdated outputs are imminent regardless of the goodness of the optimization techniques. Moreover, scaling and resilience to single point failure vulnerability for large datacenters is also questionable.

III. Overview of Proposed Resource Provisioning

In order to cope with a large number of VMs deployed in the datacenter, scalability issue and single point vulnerability must be addressed. Hence the proposed system features no GDM. Each node makes the decision regarding its own provisioning such as application profiling, resource utilization, allocation and migration. Moreover it needs to know about similar provisioning information of other nodes. As a result, all the nodes need a source for obtaining such knowledge. However if the system has a global information provider which maintains global awareness in the datacenter, the system will be exposed to single point failure and data retrieval latency. To tackle such threat, nodes are organized based on unstructured P2P network. Such architecture is used in order to form a large datacenter where centralized setup is undesired for its scalability and single point vulnerability issues.

Each node pulls information from the neighborhood but not necessarily from all the nodes. As maintaining global awareness in the datacenter is nearly infeasible due to huge computational overheads, the proposed provisioning scheme uses local awareness. Each node gathers information from its neighbor with a constant node distance with the value of $\log_d N$ where N is the total number of nodes and d is the average degree of a node [18]. In addition, this overlay structure is refreshed periodically with each node randomly linked with other nodes to avoid any locality of interest. Finally, the system proposes a policy that uses MAUT for migration of VMs that minimizes the undesired re-migration as well as provides opportunity of configuring migration criteria.

IV. System Architecture

The system architecture is composed of three major modules as demonstrated in Fig. 1. These are Application Agent (AA), Node Agent (NA) and Job Pool (JP). AA is an entity that is tightly coupled with each application that is submitted to the datacenter. As the resource requirements in real time application are always subject to change, responsibility of an

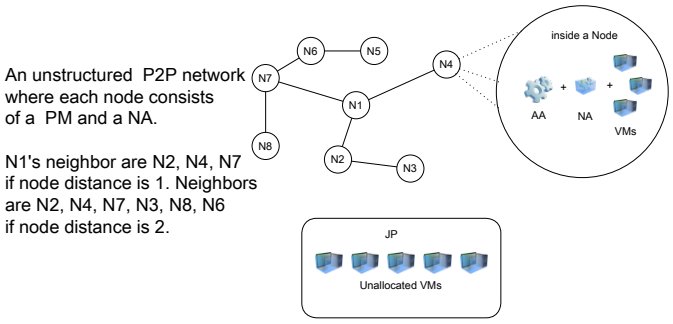

Fig. 1. P2P Architecture of the Proposed Provisioning Scheme

AA is to demand latest computational resource from its host. In the proposed system, each application is considered to be deployed in a VM and no more than one VM hosts the same application. Therefore, VM is considered as application or task unit and assigned to PMs which have the ample resource to host those.

Every PM in the datacenter hosts exactly one NA. Each NA monitors the resource usage of the hosted VMs. It also performs allocation of VMs which has just reached the datacenter. When the corresponding PM is incapable of hosting VMs, NA reconfigure or migrate those VMs.

The system also maintains a pool named Job Pool (JP) where unallocated VMs are stored. When an application is submitted to the datacenter for execution, it is deployed in a VM and stored in this pool. During the whole life cycle of a VM, it maintains certain statuses for its corresponding phases. Inside the JP, status of each VM is *unassigned*. NAs look for unassigned VM here and allocate it to a suitable PM.

Once a VM is assigned to a PM, the status changes to *assigned*. When the PM begins the execution of the VM, the status becomes *allocated*. During migration process, its status is called *migrating*. Finally, once the VM finishes its task, its status becomes *terminated*.

Apart from executing VMs, each NA performs several actions in order to maintain the resource utilization of its host to a desired level which is demonstrated in Fig. 2. These following tasks are executed sequentially in a loop. Each NA maintains its utilization index which is bounded by an upper and a lower value. Utilization index value denotes how much system resource is being used by the host. In each loop, NA checks whether its utilization value is below the lower bound threshold. If so, it looks for an unassigned VM in the JP and if it finds out a VM which can be accommodated by the host PM, NA assigns that VM to itself. Otherwise, it looks for a suitable PM in the neighborhood which can host it.

Moreover, NA continuously monitors the performance of the VMs. If NA detects one or several VMs behaving anomalously (this scenario can be characterized as resource usage beyond the upper bound of utilization index), it first reconfigures these problematic VMs. Reconfiguration simply means allocating more computational resources or withdrawing some. However, if the reconfiguration does not work or seems to be impossible due to system resource constraints, NA stops the VM and invoke a migration. This is identical to the scenario of looking for a suitable PM for an unassigned VM to allocate. In both cases, NA issues an inquiry message to the neighborhood for a PM capable enough to host that problematic VM. In an unstructured P2P network, it might be noticed that a certain region is more overloaded than the rest of the network. Any node in that region might not find a suitable option for migration inside the neighborhood. However, such scenario is highly unlikely as P2P network is dynamic and its overlay network configuration is subject to change periodically.

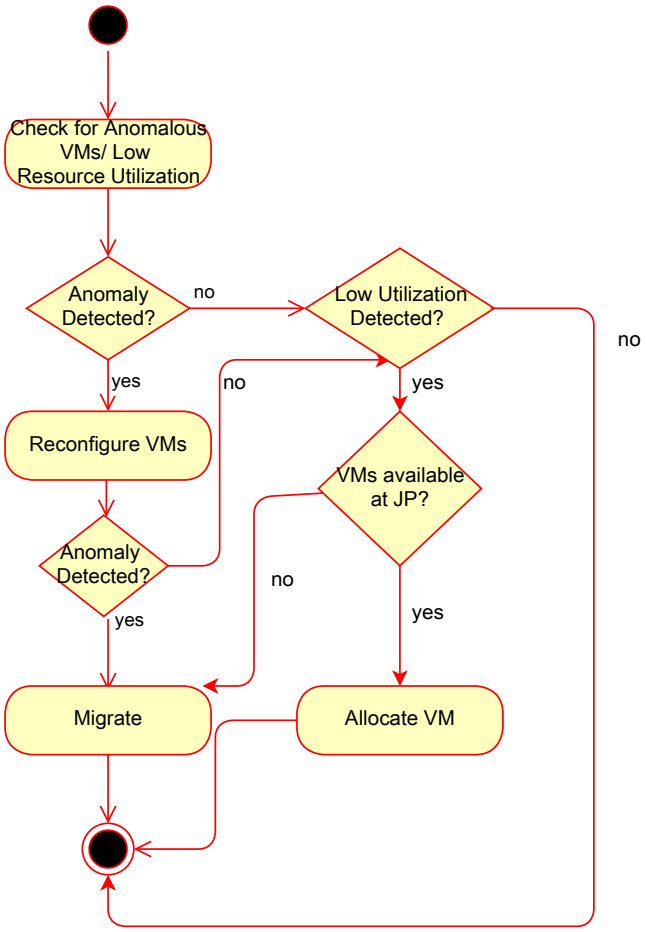

Fig. 2. NA Activity Flow in each loop

Upon acquiring information from peers, the NA computes the best suitable PM for this particular VM using MAUT methods. The computation is based on various criteria. But the most important criteria is the availability of the resources demanded by the VM. Other criteria will be discussed later. If such a PM is found, NA sends a resource lock request to the target node before delegating the task.

The NA monitoring the receiving PM, checks for the resource whether it is still available when it is accepting the lock request. If resource availability is found, the lock request is accepted. In addition, the receiving NA maintains a timeout value until which it maintains the lock. If timeout expires, it

cancels the lock and releases the resources. The sender NA also keeps a timeout value until which, it awaits the reply from neighborhood. If no such reply is received within that time window, the VM status is changed to unassigned and sent back to the pool for processing later.

V. Provisioning Decision Making Model

The overall process regarding allocation of an unassigned VM or migrating a problematic VM to other suitable nodes can be simplified into two steps. First it needs to determine which VM is the root of anomalous behavior and the second one is to decide in which PM the VM will be migrated.

The computation regarding first step is not simple because of several critical issues. These are maximization of resource utilization and minimization of migration cost, SLO violation and probability to re-migrate the VM. The most important criteria of choosing an anomalous VM are mentioned here. The (+) emblem denotes that the criterion impacts the decision positively via maximization. For example the PM which has higher available computation resources is a better choice. (-) emblem signifies exactly the opposite. These criteria are:

i **Migration Cost** (-): Cost of migrating a VM to other PMs. This cost depends on type of migrations such as live or offline, distance, network bandwidth, application type etc.
ii **Priority** (+): The priority of the application executing inside the VM. The priority depends on the type of the application. For real time application, the priority is higher than the batch processing applications.
iii **SLO Violation Penalty** (-): Application performance will be degraded during the migration because of context switching followed by SLO violation. CSPs might have to pay monetary penalty for such violations.
iv **Cumulative Migration Factor** (-): One single VM cannot be allowed to cause consequent migration requests. The more a VM invokes migration, the less it will be given preference for future migration.

The issues regarding the second step are finding the nodes having available resources, maximizing the resource utilization and minimizing the possibility of re-migration. For choosing a suitable PM, critical factors are:

i **CPU** (+): Availability of CPU resource of the PM
ii **Memory** (+): Availability of primary memory of the PM
iii **I/O bandwidth** (+): Availability of read/write bandwidth of the PM
iv **Peer Distance** (-): Node distance between the current PM and potential suitable PM

As the criticality of these criteria depends upon the nature of application, workload and datacenter configuration, the problem can be solved with Multi Attribute Utility Theory (MAUT) Methods. According to MAUT methods, there are several alternatives and criteria. In this proposed system scenario, alternatives are neighborhood PMs that can host migration-worthy VMs and VMs that are causing undesired behaviors. MAUT methods will be fed the alternatives and criteria with weights. These weights will signify how important the contribution of a single criteria in choosing the alternative. In this proposed system three MAUT methods are used. Those are two Simple Multi Attribute Ranking Techniques (SMART) and Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS). According to MAUT methods, there are m criteria with weights and n alternatives. a_{ij} denotes the score of j^{th} alternative on i^{th} criteria. This method output the best alternative. Among the three MAUT methods, SMART is the simplest. The ranking value x_j of j^{th} alternative is obtained simply as the weighted mean of the score associated with it.

$$x_j = \frac{\sum_{i=1}^{m} w_i a_{ij}}{\sum_{i=1}^{m} w_i}; j = 1, ..., n \tag{1}$$

$$x_j = \prod_{i=1}^{m} a_{ij}^{\frac{w_i}{w}}; j = 1, ..., n \tag{2}$$

The method SMART Arithmetic is based on equation 1 and SMART Geometric is based on equation 2. TOPSIS is the third MAUT method used in this proposed system which focuses on the best decision closest to the ideal solution and farthest from the negative ideal solution [19].

In principle, the proposed system is based on unstructured P2P network with making decision locally. This policy ensures scalability no matter how many VMs are deployed. P2P architecture ensures no single point failure threat is imminent because there is no GDM that can shut down the whole datacenter if it fails. Flexibility is also achieved as VM allocation and migration decisions can be taken with arbitrary number of reconfigurable weighted criteria fed into MAUT methods.

VI. Simulation Setup and Experimental Result Analysis

In this section, simulation environment setup of the proposed system will be discussed followed by simulation goal and result analysis.

The simulation platform was built using C# programming language. It focused on emulating a datacenter including numerous PMs forming a P2P structure. The overlay network was fully randomized and dynamic with time. As each PM contained a fixed amount of CPU, Memory and IO bandwidth resources required for executing VMs, it hosted a limited number of VMs without violating SLO. VMs were also characterized by their change of resource demands in CPU, memory and IO bandwidth with respect to time. These changes in the resource demands were generated following Gaussian, exponential and uniform statistical distributions [20].

When the simulation started, VMs asked for computational resources according to predefined resource demand based on statistical distributions. The passage of time in the datacenter was simulated as steps. Each step corresponded to an unit of time. Thus the flow of simulation was executed step by step where in each step, PMs executed the VMs according to their resource demand and then performed reconfiguration and migration processes when necessary. VMs also demanded resources in stepwise manner. This step based simulation

facilitated measuring the state of every PMs precisely the same point in each simulation run. If an asynchronous parallel simulation was used, datacenter configuration with the same input would have culminated in different outcomes.

Regarding the migration of VMs to a suitable node, all the criteria were fed into three MAUT methods specified in the previous section. However, for choosing an anomalous VM, two out of four identified criteria were fed into the MAUT methods. Those criteria were Priority and Cumulative Migration factor. All the criteria were assigned equal weights. In addition, the situation when no suitable PM was found for a VM that was needed to be migrated, simulation environment sent the VM to JP and reallocate it to a new PM from JP.

In order to compare the proposed system with the centralize scheme, simulation of a centralized scheme has also to be defined. Hence, datacenter incorporated a GDM. When migration was needed, each PM requested the GDM for the migration. GDM used two schemes named First Fit (FF) and First Fit Decreasing (FFD) in order to migrate VMs. FF scheme denotes migrating the VM into the very first available PM which can host that VM while FFD scheme means migrating the VM into the very first PM sorted in descending order on the basis of criteria. It is noteworthy that allocating a static number of VMs in as less PMs as possible is a vector bin packing problem. This problem is a NP-Hard problem and applying greedy as well as approximation scheme FF and FFD ensures $(11/9)OP+1$ solution where OP denotes the optimal solution [21]. Hence, the outputs of FF and FFD schemes can be considered as the output of any VM migration algorithm based on centralized provisioning acquiring global information.

The simulation targeted the number of total SLO violations, number of total migrations and server sprawling in the datacenter. In the simulation, when a VM did not manage to get desired computational resource in each step of application lifetime, it was assumed that a SLO violation had occurred. In addition, let v as the number of virtual machines allocated to p number of physical machines. However with the passage of time, for avoiding SLO violations those v number of VMs used some arbitrary additional number of PMs besides p number of PMs. This scenario is called server sprawling. Higher number of server sprawling denotes relatively lower CPU utilization.

There were three types of comparison done in this experiment. In the first and second one, SLO violation and migration count were compared for both the centralized with FF and FFD methods as well as proposed system with SMART Arithmetic, SMART Geometric and TOPSIS method. In the third test, server sprawling was focused for both centralized and proposed systems similarly to the first and second test.

The first experimental result shown in Fig. 3 demonstrates how the proposed scheme performs to tackle SLO violation in comparison with centralized scheme. Both FF and FFD schemes generate a higher number of SLO violations compared with SMART Arithmetic, SMART Geometric and TOPSIS. As number of VMs increased, the lone GDM simply could not cater to all provisioning request. This is why the proposed system generates 60.27% less number of SLO

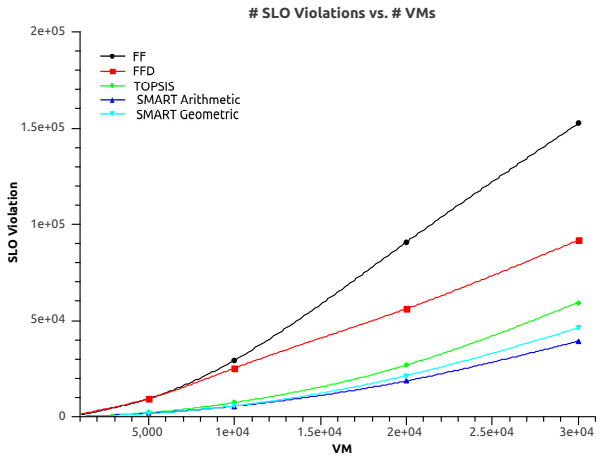

Fig. 3. SLO Violation vs. VM

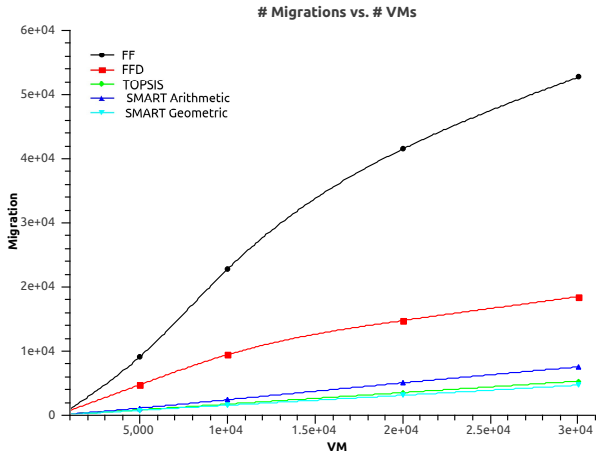

Fig. 4. Migration vs. VM

violations on average. Between two centralized schemes FFD shows better result than FF as, with the increase of VMs, SLO violation count increases linearly and exponentially for FFD and FF respectively. However, among the decentralized schemes, SMART Arithmetic performs the best while SMART Geometric and TOPSIS shows similar outputs with TOPSIS falling behind for very high number of VMs.

Similar behavior has been obtained from the second experiment where number of migrations is observed. As displayed in Fig. 4, proposed decentralized schemes generates significantly less number of migration request than the centralized ones. Compared to the first experiment, centralized schemes perform worse. The hindrance created by the single GDM is mainly responsible for this. Moreover, the proposed system migrates VM in such a manner so that the probability of re-migration of already migrated VMs are kept low. FF performs worse than FFD because it allocated VMs in the first available PM increasing the chance of re-migration. SMART Arithmetic, SMART Geometric and TOPSIS perform almost similar while SMART Arithmetic and SMART Geometric present the most and least migration affinity respectively. Overall, the proposed

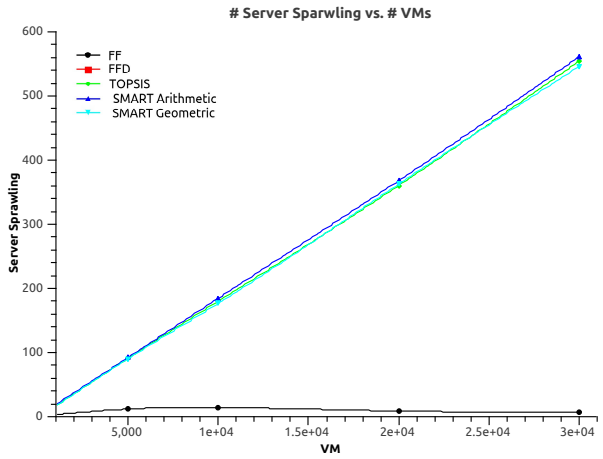

Fig. 5. Sprawling vs. VM

system generates 83.58% less migrations on average.

The third experiment, as shown in Fig. 5, demonstrates that server sprawling is literally zero in FF and FFD scheme while two SMART and TOPSIS schemes show high server sprawling. This happens in decentralized schemes when no suitable PMs were found in the neighborhood in P2P network. In that scenario, a new PM was used to host that VM. Hence server sprawling occurs in the proposed scheme.

From these three experiments, the proposed system violated SLO linearly with respect to increase in total number of VMs. That proves scalability of the system. Besides, migration invoked by the proposed system is far less than centralized schemes which have GDM vulnerable to single point dependency. MAUT methods used in provisioning decision allow VM allocation and migration in flexible manner. In order to achieve this, the scheme has to sacrifice a margin of utilization highlighted by high rate of server sprawling.

In principle, the experiment highlights that proposed scheme issues significantly less number of SLO violations and migrations considering the penalty of using slightly more resources.

VII. Conclusion

Traditional resource provisioning schemes are centralized in nature culminating in scalability and single point failure issues. To tackle this problem, a decentralized resource provisioning scheme has been proposed which is based on unstructured P2P network. The provisioning scheme does not depend upon any global provisioning decision maker but delegates each node its own provisioning responsibility. It also uses MAUT methods for allocating VMs into suitable PMs and migrating VMs.

From the simulation, the proposed system has shown less number of SLO violations and migrations causing slightly lower resource utilization. Such occurrence characterized by server sprawling is a concern that can be addressed in future. Moreover, identification of more provisioning criteria involved in VM allocation and migration can improve this scheme further.

Acknowledgement

This work is partly supported by IIT, University of Dhaka and Panacea Systems LTD, Bangladesh.

References

[1] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 164–177, 2012.

[2] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, and N. Sharma, "Towards autonomic workload provisioning for enterprise grids and clouds," in *Proc. of 10th IEEE/ACM International Conference on Grid Computing*. IEEE, 2009, pp. 50–57.

[3] T. C. Chieu and H. Chan, "Dynamic resource allocation via distributed decisions in cloud environment," in *Proc. of 8th IEEE International Conference on e-Business Engineering*. IEEE, 2011, pp. 125–130.

[4] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application performance management in virtualized server environments," in *Proc. of 10th IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2006, pp. 373–381.

[5] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, A. Monje *et al.*, "On the optimal allocation of virtual resources in cloud computing networks," 2012.

[6] Z. Zhang, H. Wang, L. Xiao, and L. Ruan, "A statistical based resource allocation scheme in cloud," in *Proc. of International Conference on Cloud and Service Computing*. IEEE, 2011, pp. 266–273.

[7] J. O. Kephart and W. E. Walsh, "An artificial intelligence perspective on autonomic computing policies," in *Proc. of 5th IEEE International Workshop on Policies for Distributed Systems and Networks*. IEEE, 2004, pp. 3–12.

[8] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, "A hybrid reinforcement learning approach to autonomic resource allocation," in *Proc. of IEEE International Conference on Autonomic Computing*. IEEE, 2006, pp. 65–73.

[9] Y. O. Yazır, "Multiple criteria decision analysis in autonomous computing: A study on independent and coordinated self-management," Ph.D. dissertation, University of Victoria, 2011.

[10] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," in *Proc. of 1st International Conference on Peer-to-Peer Computing*. IEEE, 2001, pp. 101–102.

[11] M. Yang and Y. Yang, "An efficient hybrid peer-to-peer system for distributed data sharing," *IEEE Transactions on Computers*, vol. 59, no. 9, pp. 1158–1171, 2010.

[12] R. L. Keeney and H. Raiffa, *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge University Press, 1993.

[13] J. S. Dyer, "Mautmultiattribute utility theory," in *Multiple criteria decision analysis: state of the art surveys*. Springer, 2005, pp. 265–292.

[14] D. S. Johnson, "Near-optimal bin packing algorithms," Ph.D. dissertation, Massachusetts Institute of Technology, 1973.

[15] Y. O. Yazir, Y. Akbulut, R. Farahbod, A. Guitouni, S. W. Neville, S. Ganti, and Y. Coady, "Autonomous resource consolidation management in clouds using impromptu extensions," in *Proc. of 5th IEEE International Conference on Cloud Computing*. IEEE, 2012, pp. 614–621.

[16] R. Ranjan and L. Zhao, "Peer-to-peer service provisioning in cloud computing environments," *The Journal of Supercomputing*, pp. 1–31, 2013.

[17] H. Nguyen Van, F. Dang Tran, and J.-M. Menaud, "Autonomic virtual resource management for service hosting platforms," in *ICSE Workshop on Software Engineering Challenges of Cloud Computing*. IEEE Computer Society, 2009, pp. 1–8.

[18] S. Tewari and L. Kleinrock, "Entropy and search distance in peer-to-peer networks," UCLA Computer Science Dept Technical Report UCLACSD-TR050049, Tech. Rep., 2005.

[19] T. Gwo-Hshiung, G. H. Tzeng, and J.-J. Huang, *Multiple attribute decision making: Methods and applications*. Chapman & Hall, 2011.

[20] A. Shingo, M. Murata, and H. Miyahara, "Analysis of network traffic and its application to design of high-speed routers," *IEICE Transactions on Information and Systems*, vol. 83, no. 5, pp. 988–995, 2000.

[21] M. Yue, "A simple proof of the inequality ffd (l) 11/9 opt (l) + 1, l for the ffd bin-packing algorithm," *Acta Mathematicae Applicatae Sinica*, vol. 7, no. 4, pp. 321–331, 1991. [Online]. Available: http://dx.doi.org/10.1007/BF02009683