A PEER TO PEER RESOURCE DISCOVERY SCHEME FOR PROVISIONING IN CLOUD

Rayhanur Rahman

Institute of Information Technology,
University of Dhaka,
Ramna, Dhaka, Bangladesh
E-mail: rayhanur.rahman@iit.du.ac.bd

Kazi Sakib

Institute of Information Technology,
University of Dhaka,
Ramna, Dhaka, Bangladesh
E-mail: sakib@univdhaka.edu

Abstract: The ability to cater to large number of user demands without service unavailability has become the key to the success of Cloud computing. A fundamental challenge in building such large scale cloud based services is the scalability and fail safe techniques for discovering and provisioning of resources to meet expected Quality of Service (QoS). In the Cloud, existing resource provisioning models rely on centralized information services for resource discovery which are prone to single point failure as well as lack of scalability and fault-tolerance abilities. In order to tackle these challenges, this work first proposes a resource discovery scheme based on structured Peer to Peer (P2P) architecture. As existing Distributed Hash Table (DHT) based P2P schemes inherently do not support multi attribute data publishing and range querying which are fundamental parts of obtaining information while making provisioning decision, the work also addresses this inability by proposing an attribute hub based data tuple indexing. It also proposes routing and storing data tuples inside the nodes along with decentralized P2P oriented model for publishing of provisioning information, retrieval and fail safe policies for node join and leave. The work then presents a decentralized resource provisioning model using the aforementioned resource discovery scheme and utilizing Multi Attribute Utility Theory methods to make provisioning decisions in trade-off situations. Simulation shows that the proposed approach is 44.24% faster on average and 45.81% faster in the case of maximum number of VMs than the centralized and DHT based approaches respectively in case of multidimensional range querying. The proposed system also shows less number of Service Level Objective (SLO) violations and migrations which are about 60.27% and 83.58% respectively. Thus the work demonstrates the effectiveness of the proposed decentralized approach for resource discovery and provisioning in addressing scalability and resilience to single point failure.

Keywords: Cloud Computing, Peer to Peer, Resource Provisioning, Resource Discovery, Distributed Hash Table, Multi Attribute Utility Theory

Reference to this paper should be made as follows: Rahman, Rayhanur and Sakib, Kazi 'A PEER TO PEER RESOURCE DISCOVERY SCHEME FOR PROVISIONING IN CLOUD', *International Journal of Cloud Computing*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Rayhanur Rahman ...
Kazi Sakib ...
Both authors have published ...

1 Introduction

In the Cloud, efficient resource provisioning and management is a big challenge due to its dynamic nature and heterogeneous resource requirements. Here, requests from the end users come in and come out at any time and the magnitude of the workload and resource needs are unknown. Consequently, Cloud Service Providers (CSP) have to guarantee that there is no Service Level Objective (SLO) violation as well as ensure the maximum utilization of available resources, precise decision making regarding scaling up and down as well as enhancing overall responsiveness of the cloud services. In order to keep both CSP and consumers satisfied, effective resource provisioning is mandatory.

1.1 Motivation

Resource provisioning refers to the initialization, monitoring and controlling cloud resources. However, it is not a straightforward task to perform as there are some major aspects pertinent to the Cloud such as initial static launch of virtual machine (VM), VM allocation, migration, replication, monitoring and performance profiling etc. Not only that, horizontal and vertical scaling of VMs, nodes and memory, SLO optimization, forecasting potential resource demand, cost minimization by utilizing maximum resources are also important. Thus all of these aspects are quite interrelated, for example, satisfying SLO directly depends on VM allocation and migrations.

As the Cloud has become ready for mainstream acceptance, scalability [1] of services will come under more severe scrutiny due to the increasing number of online services in the Cloud and massive numbers of global users. Thus it is becoming exponentially difficult to manage resources in dynamic real time environment making scalable resource provisioning strategies urgent these days. The extent of scaling capability of a provisioning scheme heavily depends on the underlying resource discovery technique which refers to the procurement of datacenter metadata such as machine *id*s, VM usages, allocation information, status, availability etc. To overcome the aforementioned scalability challenges in addition to single point vulnerability issues, resource discovery should also be decentralized by nature to adaptively maintain and achieve the desired system wide connectivity and behavior.

Moreover, popular cloud infrastructures used in data center such as Eucalyptus [2], Openstack [3], CloudStack [4], Amazon EC2 [5] are all based on centralized control. That means, in a datacenter facilitated with these mentioned stacks either consists of only one node controller with numerous slave nodes or hierarchical clustering of node controllers.

However, those schemes invite single point failure issues in the case of unexpected burst of workload or physical damage. Not only that, this single point dependency creates bottleneck in overall system performance. In order to avoid these problems, introduction of decentralization and parallelism is urgent in resource discovery and provisioning for cloud computing.

In response to these challenges, researchers have put a lot of efforts in this field. As a result, there are several dynamic and task specific provisioning algorithms in practical use (for example, Eucalyptus, OpenStack, CloudStack, Amazon EC2) which are based on single point resource discovery and decision making methods. These frameworks perform well in small to medium dimensioned datacenter but struggles in the performance and reliability section in case of large scale datacenter specific operations. Meanwhile, current enterprise and consumer market segments are constantly being attracted by the cloud services for the ability to deploy the application services without worrying about computational resource utilization and deployment stack. So, in order to ensure maintaining service level agreements and availability of services, resource provisioning along with underlying resource discovery scheme must be invincible to scalability bottleneck and single point failure.

1.2 Research Question

Considering those issues mentioned above, the objective of this research is to develop a scalable resource provisioning scheme. In this research, this following research question will be answered,
How can a decentralized and scalable resource discovery scheme be developed for provisioning in Cloud so that cloud services can scale for a large datacenter configuration without the issue of single point vulnerability. More specifically,

1. How Peer to Peer (P2P) scheme can be utilized in resource discovery for managing services in large cloud environments. As existing P2P approach are tailored to the storing and locating of contents indexed by their hash value, those approaches must be used in such a manner suitable for resource discovery. Hence, instead of locating the data indexed by a single string literal, those must provide a mechanism so that multi dimensional search, for example, *find a node which has the maximum CPU and IO usage*, can be answered. Apart from that, routing of provisioning data stored inside the nodes, their indexing and retrieval techniques and fail safe policies node join as well as leave need to be addressed as well.

2. Based on aforementioned resource discovery technique, how an effective resource provisioning scheme can be developed where provisioning decisions are made in decentralized manner. As major cloud frameworks are master-salve architecture oriented and global cloud controller based, how P2P can replace this centralized architecture is the main question that needs to be answered. Moreover, how can every node in the datacenter maintain awareness of neighborhood nodes considering the lack of a global resource arbiter; how nodes can procure provisioning data with aforementioned proposed discovery scheme. Apart from these questions, decentralized provisioning decision making approach also needs to addressed.

1.3 Contribution of this Research

In answering these above research questions, this work contributes to scalable resource discovery scheme for provisioning in Cloud. These contributions are summarized as follows.

First, unlike traditional approaches, a decentralized resource discovery scheme has been proposed which is based on structured P2P network. The discovery scheme does not depend upon any global or hierarchal resource arbiter but allows publishing and querying provisioning data stored in a decentralized manner inside all the physical machines in the datacenter. The discovery scheme adopts P2P approach for routing data tuples and publishing them via storing those inside nodes, searching appropriate provisioning data stored in the nodes from the datacenter and fail-safe policies in case of physical machine failure. This technique answers the first question stated above. Result obtained from the simulation implies, the proposed discovery method features 44.24% faster on average and 45.81% faster response time in the case of largest datacenter setup than centralized and DHT based approaches respectively for discovering provisioning resources in the datacenter.

Second, a decentralized resource provisioning scheme has been proposed which is based on structured multi attribute range query P2P network [6]. Provisioning information from peer nodes are achieved via proposed resource discovery method which supports multi dimensional range queries. The provisioning scheme does not depend upon any global provisioning decision maker but delegates each node its own provisioning responsibility. It also uses Multi Attribute utility Theory (MAUT) methods [7] for allocating VMs into suitable physical machines (PMs) and VM migrations. Procured result demonstrates that the proposed system has shown less number of SLO violation and migration which is about 60.27% and 83.58% than the lone resource arbiter based provisioning scheme causing slightly lower resource utilization.

2 Related Work

In this section, provisioning in current state of the art key players in cloud computing domain such as Amazon EC2 [8], Microsoft Azure [9], Google App Engine [10], Eucalyptus [2] and GoGrid [11] will be focused. A diversified number of built in services for monitoring, managing and provisioning resources are incorporated with those. Although the way these techniques are implemented in each of these clouds vary significantly, all of those are centralized in nature. Finally, several resource provisioning schemes have been proposed in the literature are highlighted. Most of those approaches perform centralized provisioning and scaling system through a single decision maker. First centralized and then decentralized schemes will be discussed here.

At present, Amazon Web Services (AWS) uses three centralized services for overall provisioning. Those are Elastic Load Balancer [12] for load balancing, Amazon CloudWatch [5] for monitoring and Amazon Auto-Scaling [13] for auto scaling purposes. Both Elastic Load Balancer and Auto-Scaling services depend on the information regarding resource status reported by the CloudWatch service. Elastic Load Balancer service automatically makes provisioning decisions involving incoming service workload across available Amazon EC2 instances. On the other hand, the Auto Scaling service is used to dynamically scale in or out Amazon EC2 VM instances in order to handle changes in service demand. However, CloudWatch can only monitor the status information at VM level. In reality, a VM instance can host more than one services such as web server, database backend, image server, etc. Therefore, there is a critical requirement of monitoring, maintaining and searching the status of individual services in a scalable and decentralized manner.

Eucalyptus [2] is an open source cloud computing environment which is composed of three controllers namely Node, Cluster and Cloud Controller. Management of VMs on physical resources, coordinating load balancing decisions across nodes in same availability zone and handling connections from external clients as well as administrators are the responsibilities of these controllers. According to the current hierarchical design, Cloud Controller works at the root level, Cluster Controllers are the intermediate nodes and Node Controllers operate at the leaf level. However, the hierarchical design pattern might invoke performance bottleneck with the increase in service workload and system size from the network management perspective.

Windows Azure Fabric [9] is designed based on an architecture similar to an interconnected structure composed of servers, VMs and load balancers known as nodes and power units, Ethernet as well as serial communications known as edges. Applications created and deployed by the developers, hosted in the Azure Cloud are monitored, maintained and provisioned by a centralized service named Azure Fabric Controller (AFC). Management of all the software and hardware components in a Azure powered datacenter is the responsibility of AFC. These components include servers, hardware based load balancers, switches, routers, power-on automation devices etc. Multiple replicas of AFC are used for fail safe purposes and those are constantly synchronized so that information stored in all of the AFCs are consistent and integral. This redundancy design tecnique performs well for fail safe strategies in case of small size datacenters but is inherently unable to scale in large datacenter configuration because, with the number of replication degree rising, it will be infeasible to maintain consistency among replicas of AFC.

GoGrid [11] Cloud uses centralized load balancers for managing application service traffic across servers. Round Robin algorithm and Least Connect algorithm are used for routing application service requests. The load balancer can also identify server crashes, which is tackled by rerouting future requests for the failed server to other available ones. However, the centralized architecture of the load balancer is vulnerable to single point failure and bottleneck in case of large number of provisioning requests.

Unlike other cloud platforms which provide a direct access to VMs to the developers, Google App Engine [10] offers a scalable but closed platform where applications can be run. Therefore, access to the underlying operating system is restricted from the developers in Google App Engine. The platform manages load balancing strategies, service provisioning and auto scaling automatically under the hood. At this moment, very little or no documentation has been found about inner details regarding architecture of this platform.

In [14], a service discovery and load balancing scheme for cloud provisioning has been proposed which uses DHT based structured P2P scheme named Chord. As DHT implementation does not support multi dimensional query lookup inherently, it utilizes MX-CIF region tree for indexing queries via control points which are distributed all over the nodes. As the scheme uses hashing techniques, it guarantees uniform load balancing leading to suffering from high response time in query lookup, update and publish due to hashing and control point managing overheads.

In [15], 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 Global Resource Arbiter (GRA).

Reinforcement learning (RL) based provisioning scheme has been proposed in [16]. 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 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 [17]. By utilizing statistical analysis, the system manages to decrease resource wastage in both peak and off peak hour by a significant margin. However, the system is prone to miscalculate the forecast and suffer from single point failure due to dependency on a GRA.

A decentralized decision making based resource provisioning scheme is proposed by Chieu et al. aiming to eliminate the threat of centralized provisioning schemes [18]. 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 PM and VM 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 [19]. 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 that source proves to be yet an another centralized implementation.

In [14], 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 a decentralized manner.

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 so that it issues less migration [20, 21, 22]. Those are similar to the previously discussed systems because of the presence of a Global Resource Arbiter. 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 aforementioned policies, most of which are closed source and proprietary as well, utilized single or hierarchal policies for resource discovery in provisioning. Hence, those are unable to offer a scalable and fail safe approach to obtain information for provisioning in cloud. In addition, these systems recalculate the whole datacenter configurations periodically through centralized process. 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 threat for large datacenters is also questionable.

3 Proposed System Architecture of Resource Discovery

The focus of this section is to provide comprehensive details on proposed resource discovery scheme for provisioning in the Cloud. First the system and application models considered are described. Then the proposed P2P model that delivers reliable and scalable resource provisioning in cloud environment is presented based on these models. There are three key aspects of the scheme which will be discussed in the following sections. These are data model, query routing and node management. Finally, this proposed resource discovery architecture which is utilized in resource provisioning will be discussed.

3.1 Data Model

In the proposed scheme, provisioning status is represented as attribute-value pair which is similar to tuple of relational database management system. More specifically, each field is a tuple of this form: <*type, attribute, value*>. The model features four data types and these are *int, char, float* and *string*. Data queries are conjunction of predicates which are tuple of the form <*type, attribute, operator, value*>. The following predicates supported in the scheme are: $<, >, <=, =>, ==, !=$.

For example, a physical machine has been operating on 65% CPU workload. This can be represented in the following manner, $< int, node - id, 1 >< float, cpu - usage, 0.65 >$

A query looking for a node having 30% CPU and 50% memory can be represented like this following, $< float, cpu - usage, 0.30 >$ && $< float, memory - usage, 0.50 >$. More example can be similar to this, *Cloud Service Type = âŁœweb hostingâŁž* && *Host CPU Utilization* $< 50\%$ && *Instance OS Type = âŁœWinSrv2003âŁž* && *Host Processor Cores* > 1 && *Host Processors Speed* $> 1.5GHz$ && *Host Cloud Location = âŁœEuropeâŁž*

Cloud Service Type = âŁœweb hostingâŁž && *Host CPU Utilization* $= 30\%$ && *Instance OS Type = âŁœWinSrv2003âŁž* && *Host Processor Cores* $= 2$ && *Host Processors Speed* $= 1.5$ *GHz* && *Host Cloud Location = âŁœEuropeâŁž*

3.2 Query Routing

The nodes in the datacenter are partitioned into attribute hubs. This partition is logical only and that means a physical node can be part of multiple attribute hubs. Each hub is accountable for a specific data attribute in overall schema. For example, an attribute hub can be responsible for *node-id* attributes. Another attribute hub is responsible for *memory-usage* attribute. Thus, hubs can be imagined as orthogonal dimensions of a multi dimensional Cartesian space. The decision regarding which dimension to route through is determined by the first node in the neighborhood. The rest of the routing is one-dimensional and is based on the values of that particular attribute of the data item.

Let A denote the set of attributes in overall schema, A_q denotes the set of attributes existing in a provisioning query, A_d denotes the set of attributes in a provisioning information record and H_a denotes the attribute hub for attribute a. In the datacenter, PMs within an attribute hub are arranged in a circular overlay where each PM is responsible for a continuous range of that particular attribute. Thus a node is responsible for all the queries where the particular attribute is a member of A_q and query attribute range predicate is true for the range of that PM. The PM also stores all the data records where the particular attribute is a member of A_d. In addition to having a link to the predecessor and successor within its own hub, each node must also maintain a link to each of the other hubs.

Queries are sent to exactly one of the hubs that manages attributes matched with the query attributes. A query Q is routed to the attribute hub H_a where a is any attribute member of A_q. Inside that particular hub, the query is routed to all the nodes that could potentially have matching values. In order to ensure that the queries find all the matching data-records, during insertion, a data-record D is sent to all H_b where b is a member of A_d. This is done as the set of queries matching D can arrive in any of these attribute hubs. Inside the hub, the data record is routed to the responsible node for the value of the record.

Fig. 1 demonstrates the routing of queries and data-records. It denotes two attribute hubs namely H_{cpu} and H_{ram} which correspond to current CPU and RAM usage of a

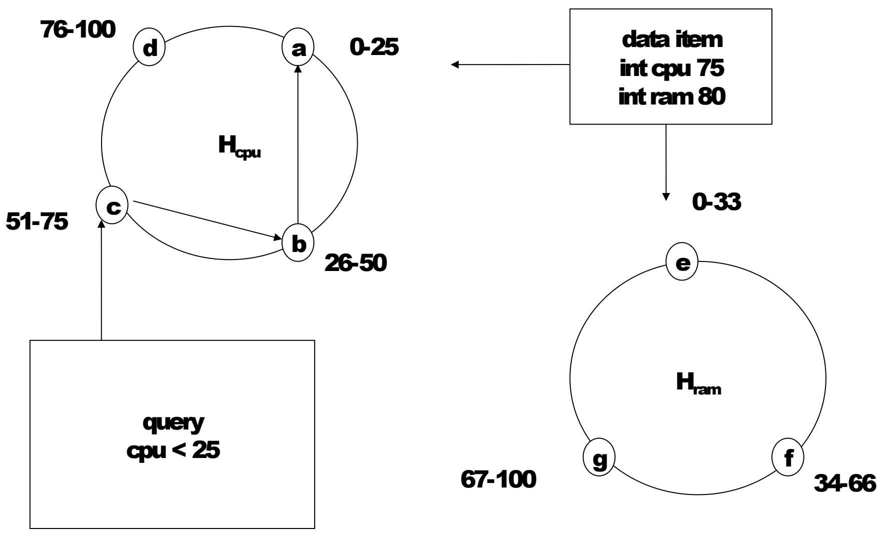

Figure 1 Data and Query Routing

physical node. The minimum and maximum values for the cpu and ram attributes are 0 and 100 respectively. The figure denotes that the ranges are being distributed to nodes. A provisioning data which contains the information of CPU usage of 75% and memory usage of 80% is sent to both H_{cpu} and H_{ram} and it is stored at both of the nodes c and g. The query which looks for data where CPU usage is below 25% enters H_{cpu} at node c and is routed at nodes b and a (destination).

3.3 Node Management

As datacenter environment is highly dynamic, physical machines in the datacenter might join and leave the datacenter network frequently. This is why the proposed scheme has to handle both the node join and leave scenarios without abrupting the datacenter operation. This section describes the detailed protocol used by nodes during join and departure.

Each node in the datacenter needs to construct and maintain the following set of links: a) successor and predecessor links within the attribute hub and b) one cross-hub link per hub for connecting to other hubs. The crosshub link implies that each node knows about at least one representative for every hub in the system. In order to recover during node departures, nodes keep a small number (instead of one) of successor/predecessor and cross-hub links.

Like most other distributed overlays, an incoming node needs information about at least one (or at most a few) node(s) that are already part of the routing system. This information can be obtained via a match-making server or any other out-of-band means. The incoming node then queries an existing node and obtains state about the hubs along with a list of representatives for each hub in the system. Then, it randomly chooses a hub to join and contacts a member m of that hub. The incoming node then installs itself as a predecessor of m, takes charge of half of m's range of values and becomes a part of the hub circle.

To start with, the new node then copies the routing state of its successor m, including its links to nodes in the other hubs. At this point, it starts random-walks on each of the other hubs to obtain new cross-hub neighbors distinct from its successor's. It is noteworthy that these processes are not essential for correctness and only affect the efficiency of the routing protocol.

When nodes depart, the successor/predecessor links and the inter-hub links must be repaired. To repair successor/predecessor links, each node maintains a short list of contiguous nodes further clockwise on the ring than its immediate successor. When a node's successor departs, that node is responsible for finding the next node along the ring and creating a new successor link.

Finally, to repair a broken cross-hub link, a node considers the following three choices: a) it uses a backup cross-hub link for that hub to generate a new cross-hub neighbor, or b) if such a backup is not available, it queries its successor and predecessor for their links to the desired hub, or c) in the worst case, the node contacts the bootstrap server to query the address of a node participating in the desired hub.

3.4 Provisioning Resource Discovery

Resource provisioning information usually contains current state of physical and virtual machines, their resource usage, Service Level Objective parameters and violation information. In a datacenter, each of the physical machine has their own set of provisioning information. These information are necessary for any provisioning decision making, for example, virtual machine migration. Such decision making process involves obtaining knowledge about other physical machines in the datacenter. This is why, provisioning information of all the nodes in the datacenter should be published so that they can be discovered whenever needed.

Provisioning information can be considered as data item D in the proposed scheme. These items are stored in the physical machine which matches the attribute hub and range of the attribute in the given data item. Meanwhile, for routing queries, it is routed to the first value appearing in the range and then the contiguity of range values is used to pass the query along the circle.

As cloud environment is highly dynamic, provisioning information of each physical machine changes with the passage of time. Hence, those information are needed to be frequently updated while phased out information should be invalidated. In order to do so, each physical machine publishes its provisioning information with a timestamp and epoch count in each epoch. On the other hand, each physical machine will delete the outdated provisioning information stored inside those. Each query will also have a timestamp and epoch number in order to prevent returning query matched data which are outdated.

4 Overview of the Proposed Resource Provisioning

In order to cope with a large number of VMs deployed in the datacenter, scalability issue and single point failure threat must be addressed. Hence the proposed system features no Global Resource Arbiter(GRA). Each node will make the decision regarding its own provisioning such as application profiling, resource utilization, allocation and migration. Moreover each node 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 will maintain global awareness in the datacenter, the system will be exposed to single point failure and higher data retrieval latency. To tackle such threats, nodes in the proposed system are organized based on structured P2P architecture proposed in the previous section. Such architecture is used in order to form a large datacenter setup where centralized configuration is undesired for its inability to tackle scalability and single point fault issues.

Each node will pull out information from the neighborhood but not necessarily from all the nodes in the datacenter. As maintaining global awareness in the datacenter is nearly infeasible due to huge computational overheads, the proposed provisioning scheme uses local awareness. Each node will pull 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 [23]. This overlay structure and interconnectivity of the nodes are formed according to the multi attribute range query policy described in the previous section. Upon this structure, each node can pull out important provisioning information from other nodes using multi dimensional range queries facilitated by aforementioned proposed resource discovery method in the previous section. Finally, the system proposes a policy that uses Multi Attribute Utility Theory (MAUT) for migration of VMs that minimizes the undesired re-migration of VMs as well as provides opportunity of tweaking and tuning migration criteria.

5 Proposed System Architecture for Decentralized Resource Provisioning

The system architecture of the proposed resource provisioning is composed of three major modules. These are Application Agent (AA), Node Agent (NA) and a 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 is always subject to change, responsibility of an 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 will host the same application. Therefore, VMs are considered as application or task unit and assigned to PMs which have the ample resource to host and run those.

Every PM in the datacenter hosts exactly one NA. Each NA monitors the resource usage of the VMs hosted by the same PM. 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 mapped to a VM and stored in this pool. During the whole life cycle of a VM, it maintains certain status for its corresponding phase. Inside the JP, status of each VM is *unassigned*. NAs will look for unassigned VMs here and allocate it to a suitable PM.

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

Apart from executing VMs, each NA also performs several actions in order to maintain the resource utilization of its host to a desired level which is demonstrated in Figure 2.

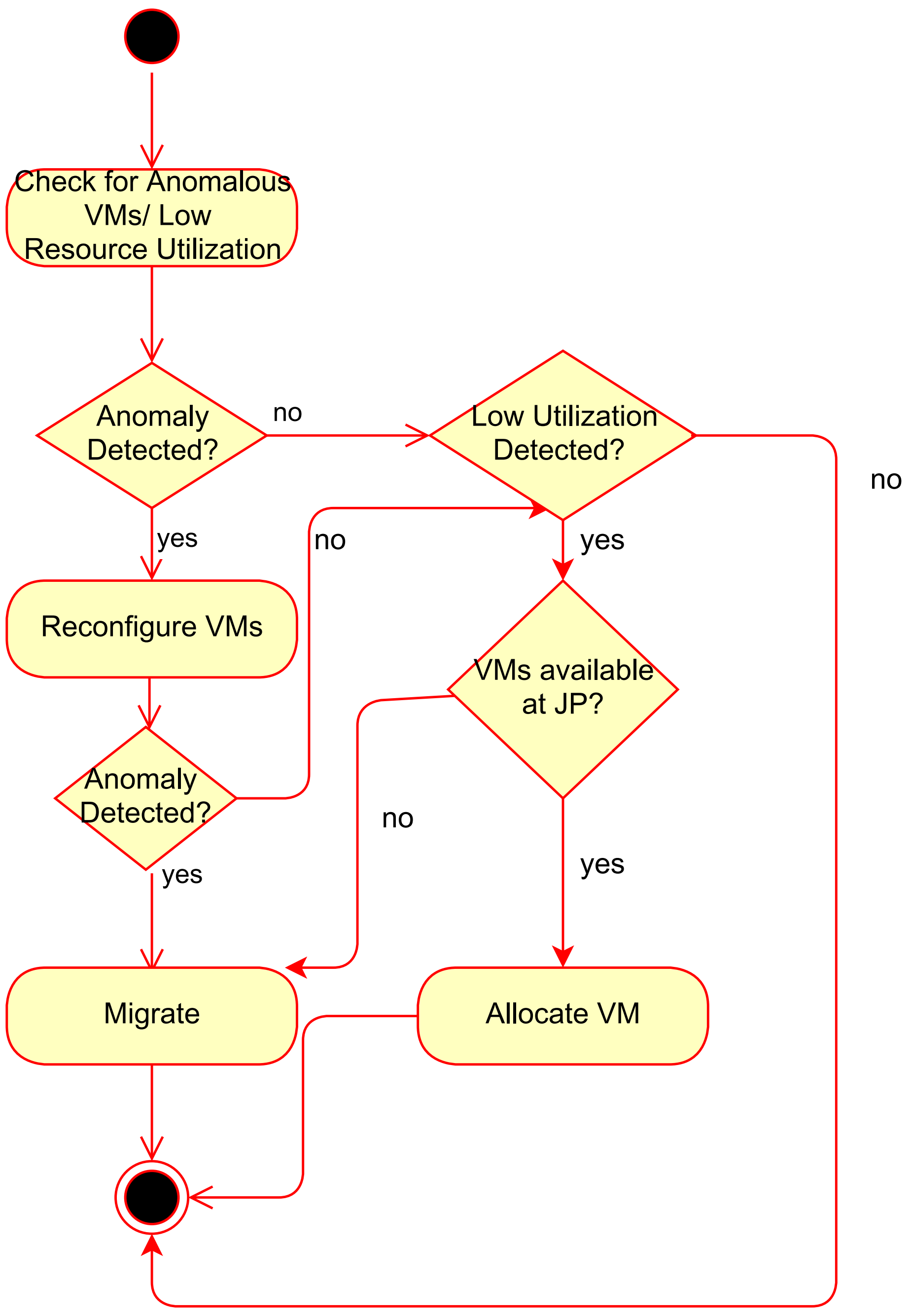

Figure 2 NA Activity Flow in Each Loop

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 will look for an unassigned VM in the JP and if it finds out a VM which can be accommodated by the host PM, NA will assign that VM to itself. Otherwise, it will look for a suitable PM in the neighborhood which can host it.

Moreover, NA will continuously monitor the performance of the hosted 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 will first reconfigure 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 will stop the VM and invoke a migration. This is identical to the scenario of looking for a suitable PM of an unassigned VM for allocation. In both cases, NA will issue an inquiry message to the neighborhood for a PM suitable enough to host that problematic VM. In an structured 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 that neighborhood. However, such scenario is highly unlikely as P2P network is dynamic and its overlay network configuration is subject to change periodically.

Upon acquiring information from peers, the NA will compute the best suitable PM for this particular VM using MAUT methods. The computation can be based on various criteria. However 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 which monitors the receiving PM, will check for the resource whether it is still available when it is accepting the lock request. If resource availability is found, the lock request will be accepted. In addition, the receiving NA will maintain a timeout value until which it maintains the lock. If timeout expires, it will cancel the lock and release the resources. NA also keeps a timeout value until which, it will await the reply from neighborhood. If no such reply is received within that time window, the VM status will be changed to unassigned and sent back to the pool for processing later.

6 Provisioning Decision Making Model

The overall process regarding allocation of an unassigned VM or migrating a problematic VM to other suitable node 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 further probability to migrate the VM. The most important criteria of choosing a suitable PM are mentioned here. The (+) emblem denotes that the criterion impacts the decision positively via maximization. For example the PM that has higher available computation resource is a better choice. (-) emblem signifies exactly the opposite. These criteria are:

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

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 an anomalous VM, critical factors are:

- **Migration Cost** (-): Cost of migrating a VM to other PM. This cost depends on type of migration such as live or offline, distance, network bandwidth, application type etc.
- **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.
- **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.
- **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 migrations.

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 such as neighborhood PMs that can host migration-worthy VMs and VMs that are causing undesired behaviors in this proposed system scenario. 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. Among these 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 furthest from the negative ideal solution [24].

In principle, the proposed system is based on structured multi attribute range query P2P network with making decision locally via multi dimensional range query resource discovery method. 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 Resource Arbiter 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 criteria fed into MAUT methods.

7 Simulation Setup and Experimental Result Analysis

In this section, first the focus will be put on how the proposed resource discovery scheme performs in comparison with the centralized and DHT based scheme. Then, the focus will be put on how the proposed resource provisioning scheme performs in comparison with the centralized scheme. First, the simulation environment of the datacenter will be highlighted. Then simulation goal will be presented followed by the discussion on obtained results.

7.1 Simulation Setup

The simulation platform was built using C# programming language and common language runtime. It focused on emulating a datacenter including numerous PMs forming a structured P2P overlay. The overlay network was built according to the policy described in the proposed resource discovery and provisioning scheme. As each PM contained a fixed amount of CPU, Memory and IO bandwidth resources required for executing VMs, it can host a limited number of VMs without violating SLO. VMs were also characterized by their change of resource demand in CPU, memory and IO bandwidth with respect to time. These changes in the resource demands were generated following Gaussian, Poisson, uniform and burst statistical distributions [25].

As the production level datacenters deploy about 1 million physical machines in the datacenter, in this simulated environment, the VM counts are kept around 1×10^5 to emulate the scenario of a large datacenter[26, 27, 28]. However, for the procurement of the result, around 3×10^4 VMs are considered sufficient to understand the characteristics of the plotted graphs as well as the performance comparisons among the schemes under observation.

When the simulation started, VMs demanded 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 a 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 if 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 run were used, datacenter configuration with the same input would have culminated in different outcomes.

In order to do provisioning jobs, the nodes in the simulated environment published their provisioning status and query on provisioning information that were already published. Each node in the datacenter published their *id*, allocated virtual machine *ids*, total cpu, memory, IO, bandwidth and resource usage statistics of each of the virtual machine running inside the host. On the other hand, each of the node periodically updated their provisioning information in each epoch. They also made queries on provisioning information using the proposed resource discovery scheme. The queries included both multi dimensional value and range matching such as *equal, between, greater than, less* than predicates as well, .

The proposed resource discovery scheme was then utilized in the aforementioned simulated environment. A single global resource arbiter based centralized resource discovery scheme was also implemented in order to compare to the proposed P2P based scheme. Finally, DHT based Cloud Peer, proposed in [14], has also been implemented in the proposed environment and then utilized to obtain resources needed for the provisioning. In the centralized scheme, all the nodes in the datacenter communicated to a single resource arbiter for querying information from neighborhood for making provisioning decision. An example for such queries can be *find the nodes where cpu usage is below* 10% *and memory usage is below* 50%. The response time of such queries was compared for centralized, proposed and DHT based approach. In addition, resource publish or mapping time was observed for those P2P approaches as well.

Regarding the migration of VMs to suitable physical machines which have the enough computational resource to host it, 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 methods. These 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 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 centralized scheme has to be defined. Hence, datacenter incorporated a GRA. When migration was needed, each PM requested the RA for the migration. RA 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 [29]. 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, 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 number of virtual machines were 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.

7.2 *Experimental Result Analysis*

The experiment first focused on the comparison among the response time of the queries invoked by the resource discovery schemes against total number of virtual machines in the datacenter. Then response time was observed against the epoch time in the case of a fixed number of virtual machines. After that, resource mapping time was observed for the decentralized resource discovery schemes followed by the data distribution among the nodes. Time is considered as discrete steps in this simulation context. Finally, there were three types of comparisons done in this experiment to observe how proposed resource provisioning scheme performs against the centralized ones. In the first and second case, 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.

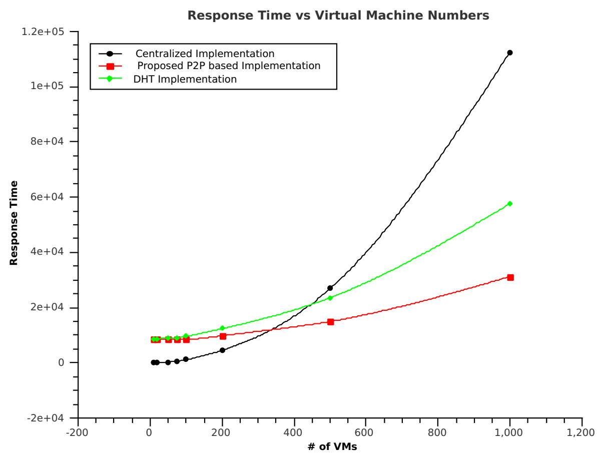

Figure 3 Response Time vs. Number of VMs

Response Time vs. Number of VMs:

Figure 3 demonstrates the comparison of query response time among the proposed scheme, the centralized scheme and the DHT based scheme. From the figure, both of the decentralized resource discovery schemes show almost a linear growth in response time of the queries with respect to the increase of VM size. On the other hand, The centralized scheme showed an exponential growth in respect of VM size which is much higher than both of the decentralized schemes. As the number of VMs increased, the lone resource arbiter simply could not cater to all provisioning request at the same time. However in the proposed decentralized scheme, such bottleneck situation never occurred. This is why the proposed system generates 44.24% less response time on an average. However, the proposed method also generates 45.81% less response time as well in comparison with proposed DHT based implementation in the case of maximum number of VMs. The justification of this improvement from proposed implementation is: as DHT implementation requires determining the address around the overlay of each resource information via hierarchical region tree organization and hashing, a time penalty is invoked because of computational processing invoked by centralized nature of the overlay for each information addressing.

Response Time vs. Epoch:

Figure 4 refers to the outcome of the experiment where resource discovery response time of both the centralized and the proposed scheme were compared in the context of

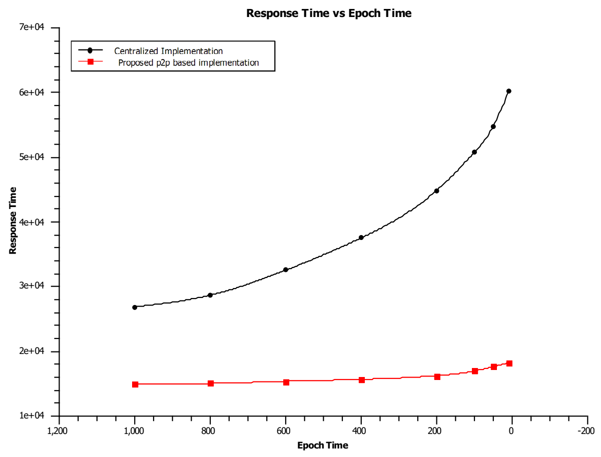

Figure 4 Response Time vs. Epoch for a Fixed Number of VMs

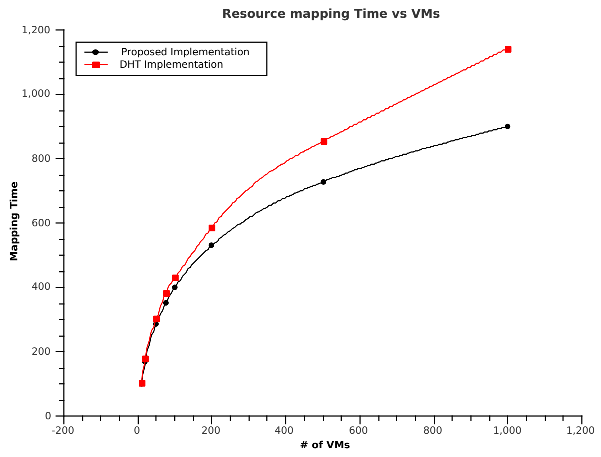

Figure 5 Resource Mapping Delay Time vs. VM

epoch time. Epoch time refers to the time interval between two subsequent updates of provisioning information of the same physical machine. With the increase of epoch time frequency, the resource discovery scheme has to cope with more numbers of queries in a fixed amount of time. As the figure displays, the centralized scheme reveals exponential growth in query response time while the proposed scheme shows a linear growth in that occasion. Thus, it is obvious that response time in centralized discovery scheme is much higher than that of the proposed scheme with the decrease of epoch time window. The more frequent provisioning information were updated/queried, the more centralized scheme suffered bottleneck situation in serving all the requests because, the single resource arbiter had to deal with all the increased amount of discovery requests but its processing power and I/O bandwidth were fixed. The proposed scheme did not suffer from such lagging situation because, the increased amount of resources were distributed all over the attribute hub space in the overlay network and hence, none of them had to face the bottlenecked situation.

Resource Mapping Delay Time vs. VM:

Figure 5 shows the measurement of average resource mapping delay for each provisioning data with respect to increase in the virtual machine size in the datacenter. The results shows that at higher number of virtual machines, the resource mapping or publishing task experienced increased delay logarithmically, proving the scalability of the proposed scheme. This happens due to the fact that the mapping tasks had to await longer period of time before getting matched against an update query. However, mapping delay is slightly higher, about 14.75% for the DHT based implementation than the proposed implementation due to time penalty obtained from hashing computation of data items to the node location mapping.

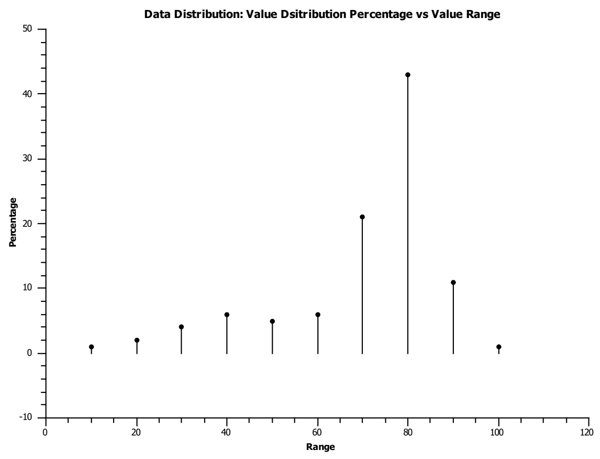

Figure 6 Distribution Percentage of CPU Attribute vs. Value Range

Distribution Percentage of CPU Attribute vs. Value Range:

Figure 6 shows the provisioning resource distribution over the physical machines in the datacenter. In the simulated datacenter environment, most of the physical machines CPU usage were kept around 80% and thus most of the provisioning resource contained CPU usage data attribute between the range of 0.7 and 0.9. Consequently, the physical machines which were assigned to the range of $[0.7, 0.9]$ in CPU usage attribute hub, held most of the resources or pointers to those resources. On the other hand, DHT based implementation ensured uniform distribution of resources around the overlay network because hashing ensures that resources will be distributed uniformly and randomly over the nodes.

From the above experiments, the proposed resource discovery scheme shows significantly less query response time. It also reveals that, with the increase of virtual machines in the datacenter, the resource publishing time does not rise significantly. Although large number of resources are queried successfully by the proposed scheme, the resource distribution of the scheme is slightly uneven. Thus, it can be claimed that the proposed resource discovery scheme offers both decentralized and scalable approach for provisioning in cloud with paying the penalty of nonuniform data distribution across the nodes.

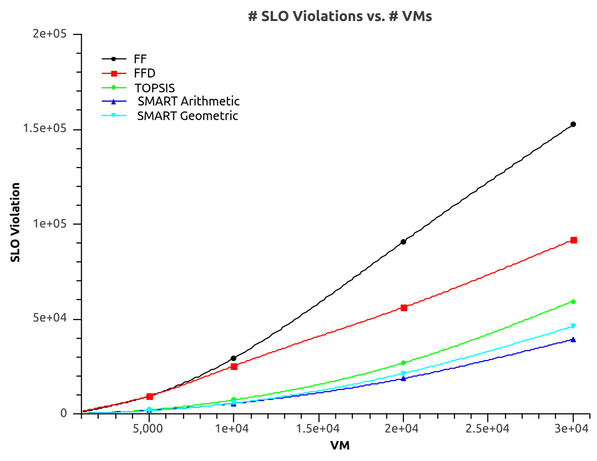

Figure 7 SLO Violation vs. VM

SLO Violation vs. VM:

The first experimental result shown in Figure 7 demonstrates how the proposed scheme performs to tackle SLO violation in comparison with centralized schemes. 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 RA simply could not cater to all provisioning request. This is why the proposed system generates 60.27% less number of SLO violations on average. Between two centralized schemes FFD shows better result than that of 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 show similar outputs with TOPSIS falling behind for very high number of VMs.

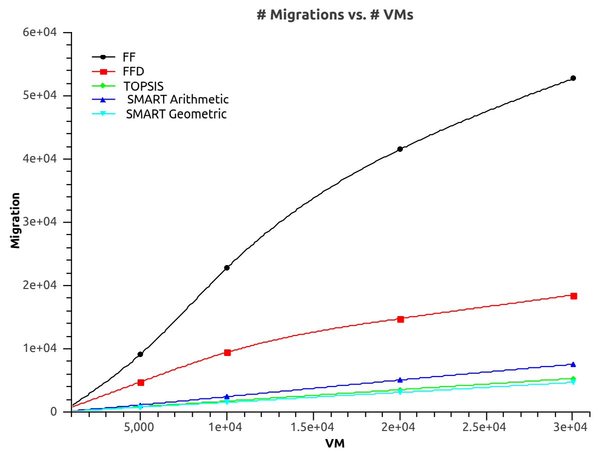

Figure 8 Migration vs. VM

Migration vs. VM

Similar behavior has been obtained from the second experiment where number of migrations is observed. The proposed decentralized schemes generate significantly less number of migration request than the centralized ones. Overall, the proposed system generates 83.58% less migrations on average as displayed in Figure 8.. Compared to the first experiment, centralized schemes perform worse. The hindrance created by the single GRA is mainly responsible for this. Moreover, the proposed system migrates VM in such manner so that the probability of re-migration of already migrated VMs is kept low. FF performs worse than FFD because it allocates 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.

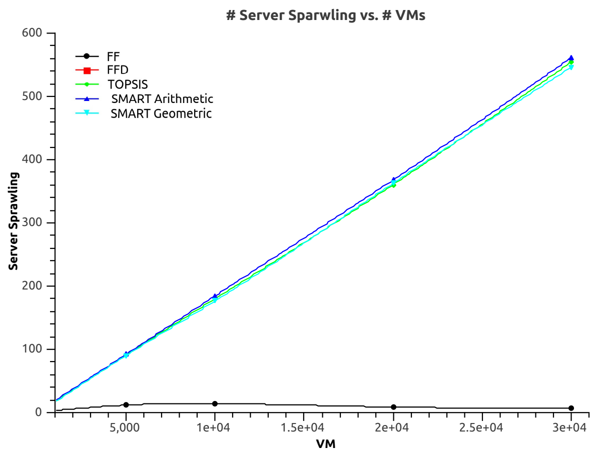

Figure 9 Sprawling vs. VM

Sprawling vs. VM

The third experiment, as shown in Figure 9 demonstrates server sprawling is literally zero in FF and FFD scheme while two SMART and TOPSIS schemes show high rate of server sprawling. This happens in decentralized schemes when no suitable PMs is found in the neighborhood region in P2P network. In that scenario, a new PM was used to host that VM. Hence server sprawling occurs in the proposed scheme.

From the three experiments, the proposed system violated SLO linearly with respect to increase in total number of VMs. That proves scalability of the system. Besides, migrations invoked by the proposed system are far less than centralized schemes which have GRA vulnerable to single point dependency. MAUT methods used in provisioning decision allows 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.

8 Discussion and Conclusion

Developing provisioning techniques that integrate application services in a scalable and fail-safe fashion is critical to the success of Cloud computing platforms. Architecting provisioning techniques based on peer-to-peer network models is significant; since peer-to-peer networks are highly scalable, they can gracefully adapt to the dynamic system expansion (join) or contraction (leave, failure), and are not susceptible to a single point of failure. In this research, a structured P2P based decentralized resource provisioning with underlying resource discovery scheme is presented. In this section, first discussion regarding the proposed resource discovery and provisioning model will be presented followed by the future direction and room for improvements of this research.

8.1 A Peer to Peer Resource Discovery Scheme in Cloud Using Multi Attribute Range Query

Resource discovery is a key process for provisioning in Cloud as fundamental provisioning information such as the status of the nodes, resource usage, virtual machine allocation, deployed applications and SLO updates. Provisioning jobs such as virtual machine allocation and migration require knowledge about neighbor nodes are procured through this discovery method. At present, major cloud stacks perform this process in centralized fashion which leads to suffering from scalability issues.

To tackle scalability and single point vulnerabilities, this research proposes decentralized P2P based discovery scheme . It proposes how inherent inability of answering multi dimensional queries in structured DHT based P2P based network is addressed by proposing an attribute hub based network overlay with data indexing, routing and storing inside nodes. Resource publishing and querying along with node join and departure fail safe policies are also presented here.

In the simulation, the proposed resource discovery scheme is put to comparison with centralized and DHT based approach of resource discovery. From the obtained result, it is observed that the proposed system has shown significantly lower response time in resource query response time which is about 44.24% on average and 45.81% in the case of maixmum number of VMs respectively in comparison with centralized and DHT based implementation. However, the proposed method invokes uneven data distribution which leads to the fact that the datacenter configuration needs more secondary memory to cope with uneven data distribution around the overlay network. However, quicker response time guarantees ensuring SLA as promised by the cloud service provider which is much more cost effective in comparison with added secondary memory cost [30, 31, 32].

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

The management and decision making regarding datacenter operation such as VM and task allocation, VM migration, physical node and application service management is referred to as resource provisioning. From its definition, understandably, effectiveness of cloud based services with huge user base depends on how underlying provisioning can handle large workload having direct impact on SLO. Current market leaders in IaaS providers such as OpenStack, Eucalyptus, Amazon EC2 features master-slave architecture oriented resource discovery and provisioning [5, 2, 3]. Thus SLO is always in threat of violation caused by the potential of sudden service outage due to scalability issues.

To address this problem, a decentralized resource provisioning scheme has been proposed which is based on structured multi attribute range query P2P overlay network. Provisioning information from peer nodes are achieved via proposed resource discovery method which supports multi dimensional range queries. 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 VM migrations.

In the simulated environment, the proposed decentralized provisioning scheme has been compared to global resource arbiter based centralized provisioning approach. Procured result demonstrates that the proposed system has shown less number of SLO violation and migration causing slightly lower resource utilization which is about 60.27% and 83.58%. Although slightly higher rate of server sprawling has also been noticed, the provisioning model succeeds in terms of meeting the SLO demand and saving the CSP from paying monetary penalties [30].

8.3 Future Direction

Addressing the uneven data distribution in the proposed discovery method and high rate of server sprawling in the provisioning model are two major concerns that can pave the way for future research. In order to improve the correlation of simulation and real life scenario, network latency, peer co-ordination delay, query patter and routing delay from a production level environment could be observed and characterized into precise statistical workload model. Moreover, only limited numbers of criteria are considered in the simulation however, resource utilization, service level agreement, migration penalty, network bandwidth issues are critical enough to be included in each and every decisions at provisioning schemes.

In addition, MapReduce framework can be used for efficient parallel workflow management of resource identifying queries. The broader vision of this research endeavor is to develop a real life open source cloud component, for example, Openstack Cloud where every entity will be a peer and completely free of centralized provisioning schemes. Other multidimensional data indexing and routing techniques that can achieve close to logarithmic bounds on messages and routing state, balance query (discovery, load-balancing, coordination) and processing load, preserve data locality and minimize the metadata should also be explored.

Another important algorithmic and programming challenge in building robust P2P cloud services is to guarantee consistent routing, look-up and information consistency under concurrent leave, failure, and join operations by application services. To address those issues, robust fault-tolerance strategies should also be based on distributed replication of attribute/query subspaces to achieve a high level of robustness and performance guarantees. Moreover, decentralized P2P based storage management and replication framework can ensure further reliability and fault tolerant capability of the Cloud.

Acknowledgement

This research has been supported by the University Grants Commission, Bangladesh under the Dhaka University Teachers Research Grant and National Science and Technology Fellowship, granted by Ministry of Science and Technology, Government of Bangladesh.

References

[1] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio Martín Llorente, Rubén Montero, Yaron Wolfsthal, Erik Elmroth, Juan Cáceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4–1, 2009.

[2] Eucalyptus systems. http://www.eucalyptus.com/. last accessed on April 20, 2014.

[3] Open source software for building private and public clouds. https://www.openstack.org/. last accessed on April 20, 2014.

[4] Apache cloudstack. http://cloudstack.apache.org/. last accessed on April 20, 2014.

[5] Amazon cloudwatch service. `http://aws.amazon.com/cloudwatch/`. last accessed on April 20, 2014.

[6] Ashwin R Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury: supporting scalable multi-attribute range queries. *ACM SIGCOMM Computer Communication Review*, 34(4):353–366, 2004.

[7] James S Dyer. MautâŁ"multiattribute utility theory. In *Multiple criteria decision analysis: state of the art surveys*, pages 265–292. Springer, 2005.

[8] J Varia. Cloud architecture- amazon web service. Technical report, 2009.

[9] Windows azure platform. www.microsoft.com/azure/. last accessed on April 20, 2014.

[10] Google app engine. http://code.google.com/appengine/. last accessed on April 20, 2014.

[11] Gogrid cloud hosting (2010) (f5) load balancer. gogrid wiki. http://wiki.gogrid.com/wiki/index.php/(F5)-Load-Balancer. last accessed on April 20, 2014.

[12] Amazon load balancer service. http://aws.amazon.com/elasticloadbalancing/. last accessed on April 20, 2014.

[13] Amazon auto scaling service. http://aws.amazon.com/cloudwatch/. last accessed on April 20, 2014.

[14] Rajiv Ranjan and Liang Zhao. Peer-to-peer service provisioning in cloud computing environments. *The Journal of Supercomputing*, 65(1):154–184, 2013.

[15] Jeffrey O Kephart and William E Walsh. An artificial intelligence perspective on autonomic computing policies. In *Proc. of 5th IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 3–12. IEEE, 2004.

[16] Gerald Tesauro, Nicholas K Jong, Rajarshi Das, and Mohamed N Bennani. A hybrid reinforcement learning approach to autonomic resource allocation. In *Proc. of IEEE International Conference on Autonomic Computing*, pages 65–73. IEEE, 2006.

[17] Zhenzhong Zhang, Haiyan Wang, Limin Xiao, and Li Ruan. A statistical based resource allocation scheme in cloud. In *Proc. of International Conference on Cloud and Service Computing*, pages 266–273. IEEE, 2011.

[18] Trieu C Chieu and Hoi Chan. Dynamic resource allocation via distributed decisions in cloud environment. In *Proc. of 8th IEEE International Conference on e-Business Engineering*, pages 125–130. IEEE, 2011.

[19] Yagiz Onat Yazir, Yagmur Akbulut, Roozbeh Farahbod, Adel Guitouni, Stephen W Neville, Sudhakar Ganti, and Yvonne Coady. Autonomous resource consolidation management in clouds using impromptu extensions. In *Proc. of 5th IEEE International Conference on Cloud Computing*, pages 614–621. IEEE, 2012.

[20] Gunjan Khanna, Kirk Beaty, Gautam Kar, and Andrzej Kochut. Application performance management in virtualized server environments. In *Proc. of 10th IEEE/IFIP Network Operations and Management Symposium*, pages 373–381. IEEE, 2006.

[21] Hien Nguyen Van, Frederic Dang Tran, and Jean-Marc Menaud. Autonomic virtual resource management for service hosting platforms. In *ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 1–8. IEEE Computer Society, 2009.

[22] Chrysa Papagianni, Aris Leivadeas, Symeon Papavassiliou, Vassilis Maglaris, A Monje, et al. On the optimal allocation of virtual resources in cloud computing networks. 2012.

[23] Saurabh Tewari and Leonard Kleinrock. Entropy and search distance in peer-to-peer networks. Technical report, UCLA Computer Science Dept Technical Report UCLACSD-TR050049, 2005.

[24] Tzeng Gwo-Hshiung, Gwo Hshiung Tzeng, and Jih-Jeng Huang. *Multiple attribute decision making: Methods and applications*. Chapman & Hall, 2011.

[25] ATA Shingo, Masayuki Murata, and Hideo Miyahara. Analysis of network traffic and its application to design of high-speed routers. *IEICE Transactions on Information and Systems*, 83(5):988–995, 2000.

[26] Report: Google uses about 900,000 servers. `http://www.datacenterknowledge.com/archives/2011/08/01/report-google-uses-about-900000-servers/`. last accessed on May 20, 2014.

[27] Google throws open doors to its top-secret data center. `http://www.wired.com/2012/10/ff-inside-google-data-center/all/`. last accessed on May 20, 2014.

[28] Microsoft now has one million servers âŁ“ less than google, but more than amazon, says ballmer. `http://www.extremetech.com/extreme/161772-microsoft-now-has-one-million-servers-less-than-google-but-more-than` last accessed on May 20, 2014.

[29] Minyi Yue. A simple proof of the inequality ffd (l) â‰¤ 11/9 opt (l) + 1, âˆŁl for the ffd bin-packing algorithm. *Acta Mathematicae Applicatae Sinica*, 7(4):321–331, 1991.

[30] Pankesh Patel, Ajith H Ranabahu, and Amit P Sheth. Service level agreement in cloud computing. 2009.

[31] Salman A Baset. Cloud slas: present and future. *ACM SIGOPS Operating Systems Review*, 46(2):57–66, 2012.

[32] Service level agreement. http://www.gogrid.com/legal/service-level-agreement-sla. last accessed on April 20, 2014.