A Scalable Resource Provisioning Scheme for the Cloud using Peer to Peer Resource Discovery and Multi Attribute Utility Theory

Rayhanur Rahman*

IIT,
University of Dhaka,
Dhaka, Bangladesh
E-mail: rayhan@du.ac.bd
*Corresponding author

Kazi Sakib

IIT,
University of Dhaka,
Dhaka, Bangladesh
E-mail: sakib@iit.du.ac.bd

Abstract: Serving large number of users without compromising service availability and performance is key to the success of the Cloud. A fundamental challenge in building such services is incorporating scalability and fail safe techniques for discovering and provisioning of resources. As Peer to Peer (P2P) architectures are invincible to these setbacks, the work proposes a P2P based resource discovery and provisioning method for the Cloud. It first addresses the multi attribute data publishing and the range querying inability of existing Distributed Hash Table (DHT) based P2P schemes and proposes an attribute hub based discovery of provisioning information. Focused on that, a decentralized resource provisioning model is proposed using Multi Attribute Utility Theory methods. The simulation shows that the proposed approach is 44.24% and 45.81% faster than the centralized and DHT based approaches respectively in case of multidimensional range querying. It also shows the lesser number of Service Level Objective (SLO) violations and migrations which are about 24.11% and 33.43% respectively.

Keywords: Cloud, P2P, MAUT, Resource Provisioning, Resource Discovery

Reference to this paper should be made as follows: Rahman, R. and Sakib, K. (2017) 'A Scalable Resource Provisioning Scheme for the Cloud using Peer to Peer Resource Discovery and Multi Attribute Utility Theory', *Int. J. of Cloud Computing*, Vol. x, No. y, pp.xxx–xxx.

Biographical notes: Md. Rayhanur Rahman is working as a Lecturer at the Institute of Information Technology (IIT), University of Dhaka, Bangladesh. He received his Master of Science in Software Engineering (MSSE) and Bachelor of Information Technology, major in Software Engineering (BIT(SE)) from the same institution. He has the experiences of development in enterprise level software industry as well. Currently, core areas of his research interest are

cloud computing, software performance engineering and mobile application testing. He has received several grants and fellowship from University Grants Commission and Ministry of Science and Technology, Bangladesh for conducting his researches.

Kazi Sakib is currently working as a Professor at the Institute of Information Technology (IIT), University of Dhaka, Bangladesh. He received his Master of Computer Science and Bachelor of Computer Science from the Department of Computer Science and Engineering, University of Dhaka. He completed his PhD from RMIT University, Melbourne, Australia where he did research on organization of sensor nodes. Currently he is conducting researches on Distributed System and Cloud Computing, Software Engineering and Testing etc. He leads the Distributed Systems and Software Engineering research Lab at IIT, University of Dhaka where he conducts researches in aforementioned topics and supervise his research students.

1 Introduction

In the Cloud [28, 19, 29], efficient resource provisioning and management is a big challenge due to its dynamic nature and heterogeneous resource requirements. It refers to initializing, monitoring and controlling cloud resources. Besides, scaling, SLO optimization, forecasting resource demand and resource utilization are also important [14]. As the Cloud [25, 14] has become ready for mainstream acceptance, scalability will come under severe scrutiny due to the increasing number of online services in the Cloud and massive number of global users [44]. Such steep uphill task to manage resources in the cloud implies the requirements of a scalable resource provisioning strategies [28]. 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. In addition to single point vulnerability issues, to overcome the scalability challenges, resource discovery should also be decentralized by nature to adaptively maintain the desired system wide connectivity and behavior.

Popular cloud infrastructures used in data center such as Eucalyptus [5], Openstack [10], CloudStack [4], Amazon EC2 [2] are all based on centralized control. A datacenter facilitated with these mentioned stacks either consists of only one node controller 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. This single point dependency also creates bottleneck in overall system performance. Introduction of decentralization is urgent to address those issues 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) along with proposed frameworks found in the literatures [37, 47, 38, 54, 43, 59]; all of 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 aspects 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. 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.

Considering above mentioned issues, the objective of this work 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 DHT based approaches 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 [22, 31, 41, 58, 15, 16]. 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 [55]. Apart from that, routing of provisioning data stored inside the nodes, their indexing and retrieval techniques and fail safe policies for node join as well as leave need to be addressed [35, 26, 45, 46].

2. Based on aforementioned resource discovery technique, how an effective resource provisioning scheme can be developed where provisioning decisions are made in decentralized manner [20]. As major cloud frameworks are master-slave 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 can nodes procure provisioning data with aforementioned proposed discovery scheme.

In answering these research questions, the contributions of this work are summarized as follows. First, unlike traditional approaches, a decentralized resource discovery scheme has been proposed based on structured P2P network. The scheme does not depend upon any global or hierarchical resource arbiter but allows publishing and querying provisioning data stored in a decentralized manner. It adopts P2P approach for routing data tuples and publishing those via storing inside nodes, searching appropriate provisioning data stored in the nodes from the datacenter and fail-safe policies in case of physical machine failure. Result obtained from the simulation implies, the proposed discovery method features 44.24% and 45.81% faster response time in the case of 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 [18]. 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 and delegates each node its own provisioning responsibility. It also uses Multi Attribute Utility Theory (MAUT) methods [23] 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 24.11% and 33.43% lesser than that of the lone resource arbiter based provisioning scheme causing slightly lower resource utilization.

The rest of the article is organized as follows. First, existing discovery and provisioning model of cloud resources are discussed in Related Work section. Then the proposed system architecture is presented followed by simulation and result analysis. Finally, discussion and conclusion of the work is presented.

2 Related Work

In this section, provisioning in current state of the art key players in cloud computing domain such as Amazon EC2 [53], Microsoft Azure [13], Google App Engine [7], Eucalyptus [5] and GoGrid [6] 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 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 [3] for load balancing, Amazon CloudWatch [2] for monitoring and Amazon Auto-Scaling [1] for 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 [5] 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, The Cloud Controller works at the root level, Cluster Controllers reside at 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 [13] is designed based on an architecture similar to an interconnected graph like structure which is composed of servers, VMs, power units and load balancers known as nodes along with ethernet and 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 an 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 technique 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 [6] 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 prone to bottleneck.

Unlike other cloud platforms which provide a direct access to VMs to the developers, Google App Engine [7] 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 under the hood. At this moment, very little or no documentation has been found about inner details regarding architecture of this platform.

Ranjan et al proposed a service discovery and load balancing scheme for cloud provisioning which uses DHT based structured P2P scheme named Chord [42]. As DHT implementation does not support multi dimensional query lookup, it utilizes region tree for indexing queries via control points which are distributed all over the nodes. Hence 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.

An autonomous computing agent for datacenter is proposed by Kephart et al [33]. 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 [50]. Instead of having any predefined model, it uses decompositional RL method to allocate resources dynamically. However, centralized architecture of this framework makes the provisioning scheme unable to scale 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 [59]. 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 GRA.

A decentralized decision making based resource provisioning scheme is proposed by Chieu et al. aiming at eliminating the threat of centralized provisioning schemes [21]. 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 [56]. 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 is yet another centralized implementation.

In addition, various optimization techniques such as Constraint Satisfaction Problem, Vector Bin Packing and Integer Programming have been implemented to minimize server sprawling and SLO violation to host a fixed number of VMs into minimum number of PMs so that it issues less migration [34, 37, 39, 49]. 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 policies, most of which are closed source and proprietary as well, utilized single or hierarchical policies for resource discovery and 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 are 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. The scheme uses structured P2P architecture to achieve decentralization in resource discovery. Thus data model, data storing, query routing and node management are fundamental aspects of the proposed work described in this section.

3.1 Data Model

The proposed resource discovery scheme makes data available to all the nodes in the datacenter in decentralized manner and hence data modeling of resource provisioning information is designed keeping their ease of routing and indexing via P2P architecture in consideration. Such provisioning information usually contains current state of physical and virtual machines, their resource usage, Service Level Objective (SLO) parameters and violation information. In a datacenter, each of the physical machine has their own set of provisioning information as well. These information are necessary for any provisioning decision making, for example, virtual machine allocation and migration. To represent in the proposed scheme, those resources are considered as attribute-value pair (also known as key-value pair or dictionary) data item or tuple. This schema-less design is chosen because of the convenience regarding data storage, distributed organization and query efficiency with key-value pair based data items. More specifically, each field is a tuple of this form: <*type, attribute, value*>. The model features only primitive 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*>. These following binary predicates are used for the proposed provisioning context $<, >, <=, =>, ==, != $ as these mentioned operations are required for the attribute search in primitive data types.

For example, a physical machine has been operating on 65% CPU workload. This can be represented in the following manner, $< int, node - id, 1 >$ and $< float, cpu - usage, 0.65 >$. A query looking for a node having 30% CPU and 50% memory can be represented like the following, $< float, cpu - usage, 0.30 > \&\& < float, memory -$

$usage, 0.50 >$. More examples can be similar to this, *Service Deploy Type = "web hosting"* && *Native CPU Utilization* $<= 50\%$ && *Virtualized OS Type = "WinSrv2003"* && *Native Processor Cores* $>= 1$ && *Native Processors Speed* $>= 1.5GHz$ && *Native Cloud Region = "Asia"*. *Service Deploy Type = "web hosting"* && *Native CPU Utilization* $= 30\%$ && *Virtualized OS Type = "WinSrv2003"* && *Native Processor Cores* $= 2$ && *Native Processors Speed* $= 1.5$ *GHz* && *Native Cloud Region = "Europe"*

3.2 Query Routing

The required provisioning resources are needed to be stored in the datacenter to make provisioning decision. Such decision making process involves obtaining knowledge about other physical machines in the datacenter. The proposed scheme utilizes each node of the datacenter for storing and publishing those provisioning information resources of all the nodes. However, a particular node will only store and publish data items having a value in between a specific range and attribute. For example, a node *n* will store only the data items which have an attribute *n* (this very data item might also have other attributes as well) with the value between $50 - 75$. However, all these information stored in the nodes must be discoverable from searching the exact value or value ranges of any of the attributes. Thus multi attribute range query must be supported in the scheme and to facilitate this, the nodes in the datacenter are split up into attribute hubs organized as structured P2P nodes mentioned in [18]. This segregation 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 as the data item is now discoverable in that particular attribute hub and is based on the values of that particular attribute of the data item.

Let A denotes 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 directed to precisely 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.

Figure 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 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

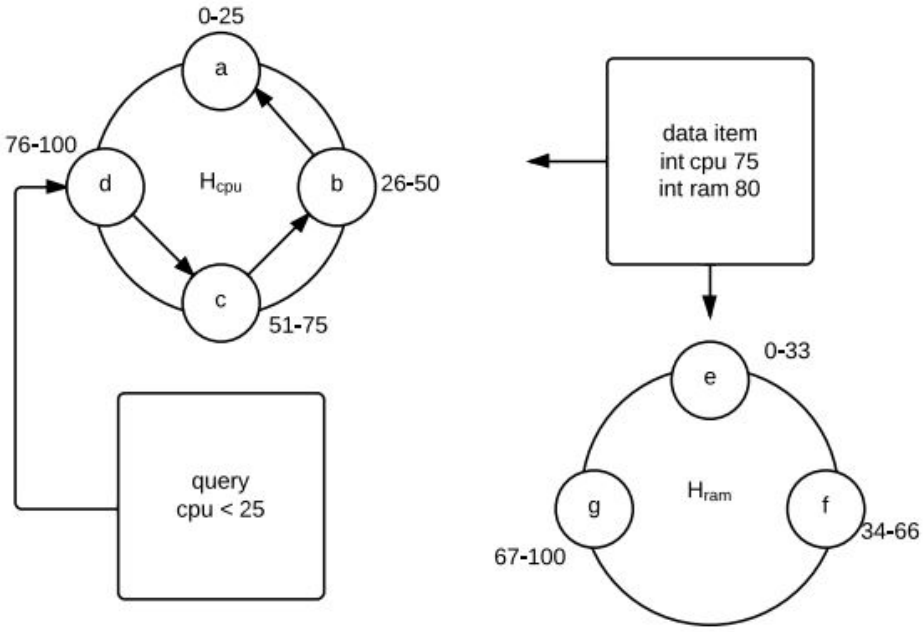

Figure 1 Data and Query Routing

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).

As cloud environment is highly dynamic, provisioning information of each physical machine changes with the passage of time. Hence, those information is needed to be frequently updated, (usually several times in a minute) while phased out information should be invalidated. 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 deletes the outdated provisioning information stored inside those. Each query also has a timestamp and epoch number in order to prevent returning query matched data which are outdated.

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, demonstrated in Figure 2.

Each node in the datacenter needs to construct and maintain the following set of links:

- successor and predecessor links within the attribute hub.

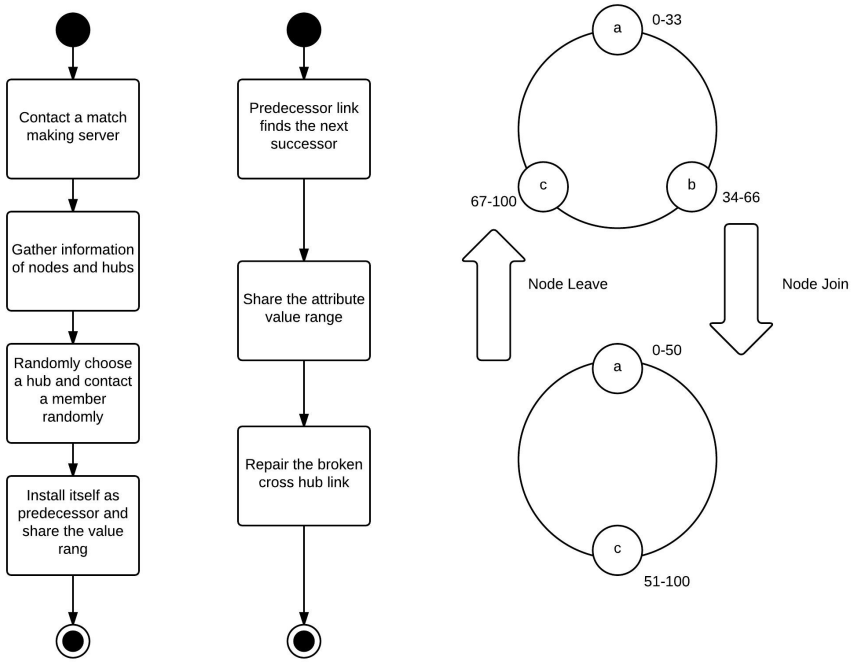

Figure 2 Node join and leave policy of proposed resource discovery

- one crosshub link for each hub to connect with other hubs. The crosshub link denotes that each node knows about at least one representative for every hub in the system.

In order to recover during node departures, nodes keep multiple numbers of successor/predecessor and cross-hub links. In addition, similar to most other decentralized overlay network, a joining node requires information about at least one (or at most a few) node(s) that are already part of the overlay system. This information can be collected via a match-making bootstrap server. The incoming node then queries an existing node and gathers status of the hubs along with a list of representatives for each hub in the overlay. Then, it chooses a hub randomly to join and contacts a member m of that hub. The incoming node then establish itself as a predecessor of m, takes responsibility of half of m's range of values and becomes a part of the hub circle. After that the new node then copies the routing state of its successor m, including its links to nodes in the other hubs.

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 a node chooses among the following three options to repair a broken cross-hub link:

1. it uses a contingency cross-hub link for that hub to generate a new cross-hub neighbor

2. if such crosshub link is not found, it asks its successor and predecessor for their links routed to the desired hub

3. in the worst case, the node connects to the bootstrap server to ask the address of a node participating in the desired hub

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 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 instead of from all the nodes in the datacenter. This local awareness is preferred as maintaining global awareness in the datacenter is nearly infeasible due to huge computational overheads. Each node will retrieve 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 [51]. This overlay structure and interconnectivity of the nodes are formed according to the multi attribute range query policy described in section 3.2. 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. 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 as represented in Figure 3. These are Application Agent (AA), Node Agent (NA) and a Job Pool (JP). AA is an entity 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

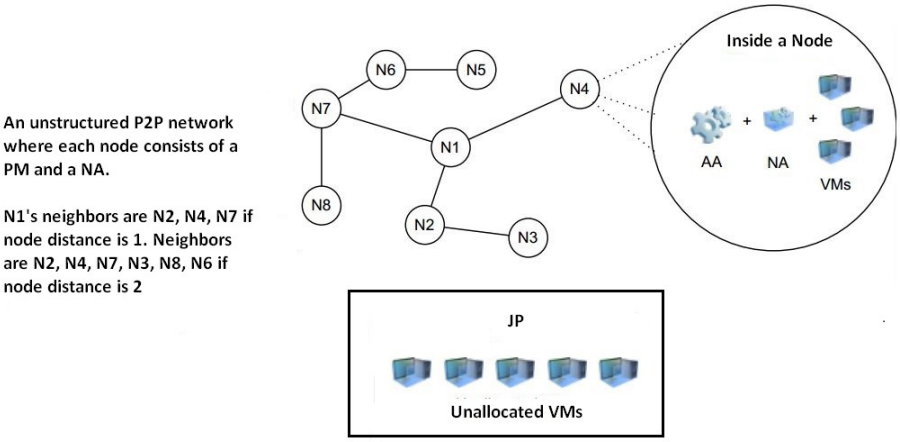

Figure 3 P2P Architecture of the Proposed Provisioning Scheme

just reached the datacenter. When the corresponding PM is incapable of hosting VMs, NA reconfigures or migrates 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, the 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 4. These tasks are executed sequentially in a loop. Each NA maintains its utilization index which is bounded by an upper and a lower value, 0.95 and 0.05 respectively for the sake of simplicity in simulation purposes. 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 the 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 a structured P2P network, it might be noticed that a certain region is more overloaded than the rest of the network. Any

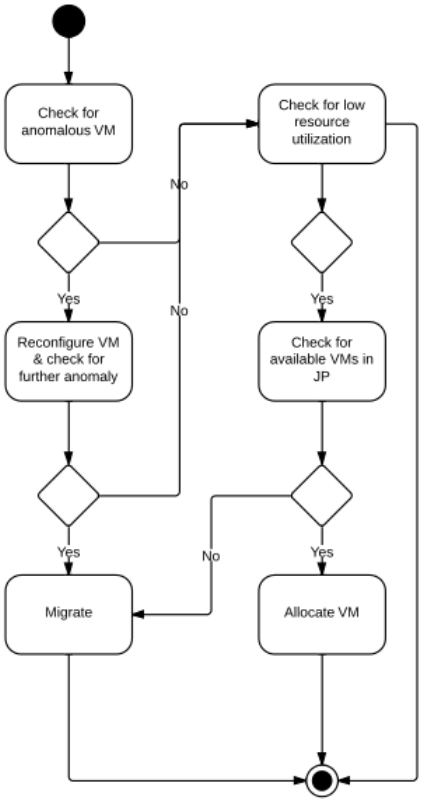

Figure 4 NA Activity Flow in Each Loop

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 Multi Attribute Utility Theory (MAUT) methods. The computation can be based on various criteria such as resource availability, peer distance etc. However the most important criterion 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 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 to solve the anomalous behavior caused by that VM.

The computation regarding first step is not simple because of several critical issues. These are maximizing the resource utilization and minimizing the migration cost, associated SLO violation as well as 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. The (-) 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 potentially 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 PMs. 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 the context switching followed by SLO violations. 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 in terms of degrading the priority.

It is obvious that migrating a VM that is rendering most numbers of abnormal resource demands or choosing a PM that is most suitable for accommodating a particular VM is a multi criteria decision making problem which has several criteria to consider and more than one alternative to choose. Let us consider a decision making problem with m criteria and n alternatives. Let $C_1, \ldots, C_m$ and $A_1, \ldots, A_n$ denote the criteria and alternatives, respectively. In Table 1 each row belongs to a criterion and each column describes the performance of an alternative. The score a_{ij} describes the performance of alternative A_j against criteria C_i. It is assumed that a higher score value means that the corresponding alternative will be a better choice.

Weight	Criteria	1^{st} Alternative Score	...	n^{th} Alternative Score
		x_1	...	x_n
		A_1	...	A_m
w_1	C_1	a_{11}	...	a_{1n}
...	...	...	...	...
w_m	C_m	A_{m1}	...	A_{mn}

Table 1 Multiple Criteria with weights

As shown in Table 1, weights $w_1, \ldots, w_m$ are assigned to the criteria. Weight w_i reflects the relative importance of criteria C_i to the decision and is assumed to be positive. In this context, the choices of a suitable VM among several VMs inside a node are the alternatives. Besides, migration cost, SLO violation penalty are criteria to be considered regarding choosing a VM from many alternatives. Similarly, choosing a PM from several PMs in the datacenter are alternatives and CPU, I/O bandwidth are consider-worthy criteria.

Multi Attribute Utility Theory (MAUT) methods are designed to solve decision making problems where criteria and relative importance have an impact on final outcome [36, 32, 24]. Hence, the aforementioned decision making problem can be solved with MAUT. The values $x_1, \ldots, x_n$ associated to the alternatives in the Table 1 are used in the MAUT methods and are the final ranking values of the alternatives. Usually, a higher ranking value means a better performance of the alternative, so the alternative with the highest ranking value is the best among the alternatives. In this proposed system three MAUT methods are used. Those are two Simple Multi Attribute Ranking Techniques (SMART) which are SMART Arithmetic and SMART Geometric followed by Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [52]. 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. The ranking value x_j of j^{th} alternative is obtained simply as the weighted mean of the score associated to it.

The method SMART Arithmetic is based on equation (1) and SMART Geometric is based on equation (2).

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

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

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 [27].

In principle, the proposed system is based on structured multi attribute range query P2P network to avoid centralization. In addition, it makes multi attribute based decision locally via multi dimensional range query resource discovery method. This policy ensures scalability no matter how many VMs are deployed. P2P architecture also ensures resilience to single point failure because there is no Resource Arbiter that can shut down the whole datacenter upon failure. Flexibility is also achieved from Cloud Service Provider's viewpoint 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 against the centralized and Distributed Hash Table based scheme. Then how the proposed resource provisioning scheme performs in comparison with the centralized scheme will be observed. Before going into the details, the simulation environment of the datacenter are highlighted. Then the simulation goal are 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 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 for executing VMs, it can host a limited number of VMs without violating the 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 the chosen Gaussian, Poisson, uniform and burst statistical distributions as they closely represent the cpu intensive, real time and web application deployed in the datacenter [48].

As the production level datacenters deploy about 1 million physical machines, in this simulated environment, the VM counts were kept around 1×10^5 to emulate the scenario of a large datacenter [11, 8, 9]. However, for the procurement of the result, around 3×10^4 VMs were 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 the 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 discrete event 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, the nodes in the simulated environment published their provisioning status and query on provisioning information. Each node in the datacenter published its *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 implemented in the aforementioned simulated environment. A single global resource arbiter based centralized resource discovery scheme was also implemented in order to compare the proposed P2P based scheme. Finally, DHT based Cloud Peer, proposed in [42], 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 provisioning queries was compared for centralized, proposed and DHT based approach. In addition, resource publishing or mapping time was observed for all of those P2P approaches as well.

Regarding the migration of VMs to suitable physical machines which have the enough computational resources to host it, all the criteria were fed into three MAUT methods specified in section 6. However, for choosing an anomalous VM, two out of four identified criteria were fed into the methods for the sake of simplicity in simulation purpose. These criteria were Priority and Cumulative Migration factor. All the criteria were assigned equal weights for simplicity as well. 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 reallocated 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 Global Resource Arbiter (GRA) or simply known as Resource Arbiter (RA). When migration was needed, each PM requested the Resource Arbiter 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 [57]. 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 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 cases, 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

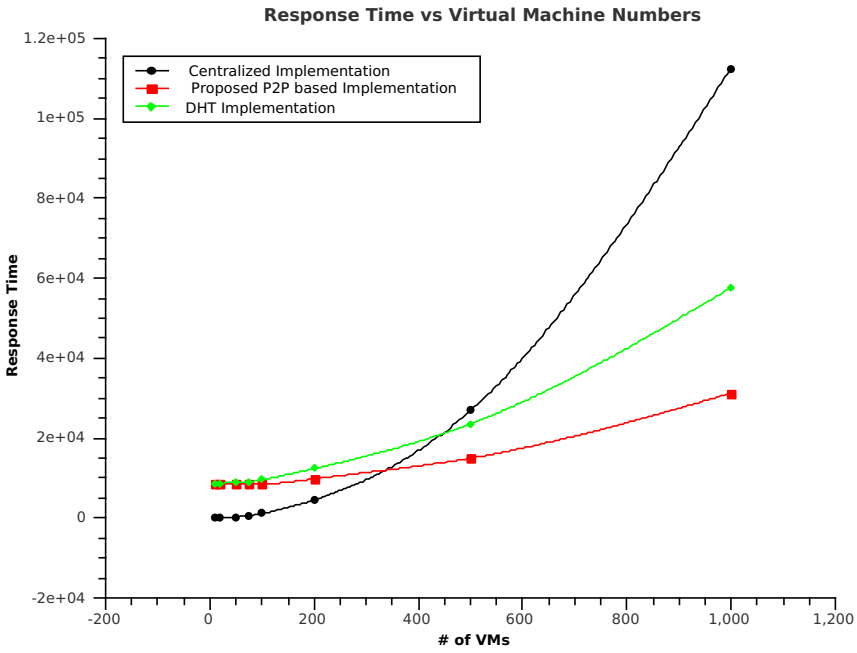

Figure 5 Response Time vs. Number of VMs

and TOPSIS methods. In the third test, server sprawling was focused for both centralized and proposed systems similarly to the first and second tests.

Response Time vs. Number of VMs

Figure 5 demonstrates the comparison of query response time among the proposed, centralized and DHT based schemes. From the figure, both of the decentralized resource discovery schemes show almost a linear growth in query response time with respect to the increase of VM size. On the other hand, the centralized scheme shows an exponential growth in respect to VM size which is much higher than both of the decentralized schemes. As the number of VMs increased upto 1×10^3, the lone resource arbiter simply could not cater to all provisioning requests at the same time. However in the proposed decentralized scheme, such bottleneck situations never occurred. This is why the proposed system generates 44.24% less response time in case of 5×10^2 in comparison with the centralized scheme. It is noteworthy that, the maximum number of VM in this particular experiment is 1×10^3, the aforementioned contrast of response time could have been even bigger in case of maximum number of VMs. In case of 5×10^2 number of VMs, the proposed method is slightly ahead of DHT based implementation in generating less response time which is around 15%. However, the proposed method also generates 45.81% less response time 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. It is also obvious to notice that, these three mechanism show an

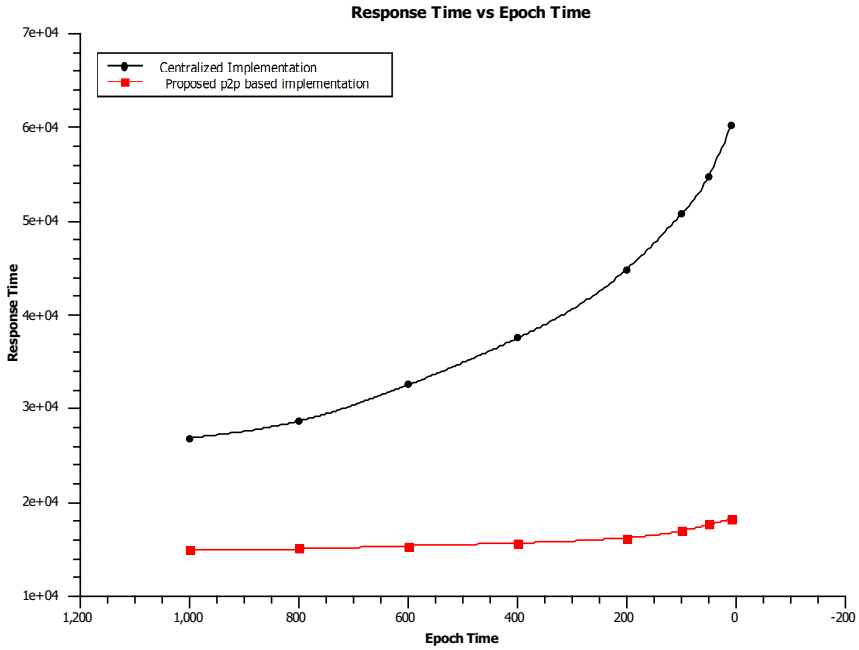

Figure 6 Response Time vs. Epoch for a Fixed Number of VMs

exponential growth of response time with the increase of VMs and performance difference is significant enough. Hence, the number of VMs has been kept within 1×10^3 in this particular experiment.

Response Time vs. Epoch

Figure 6 refers to the outcome of the experiment where resource discovery response time of both the centralized and the proposed schemes are compared in the context of 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 the 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. This is caused by the single resource arbiter which 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 those had to face the bottlenecked situation.

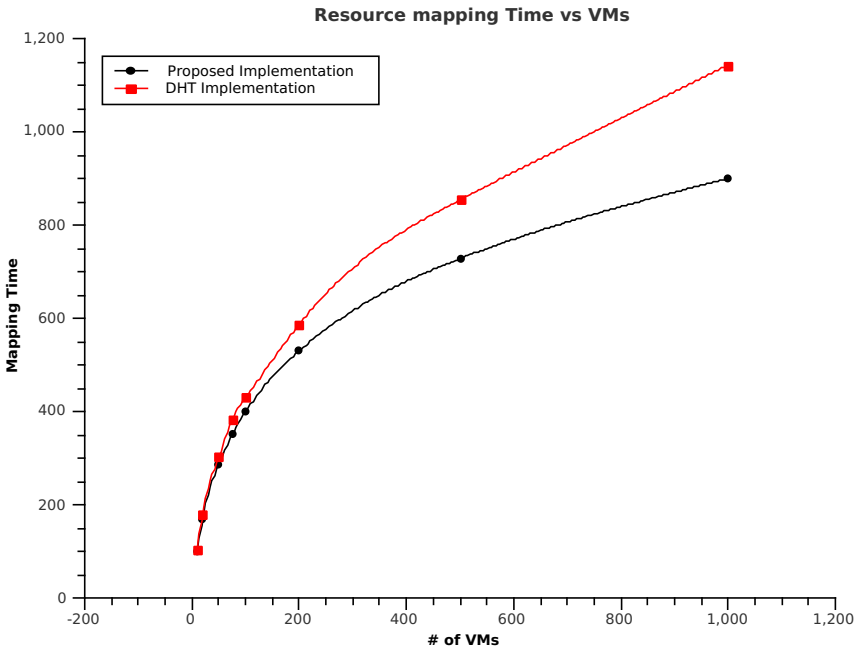

Figure 7 Resource Mapping Delay Time vs. VM

Resource Mapping Delay Time vs. VM

Figure 7 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 show 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 one due to time penalty obtained from hashing computation of data items to the node location mapping.

Distribution Percentage of CPU Attribute vs. Value Range

Figure 8 shows the provisioning resource distribution over the physical machines in the datacenter. In the simulated datacenter environment, CPU resources of most of the physical machines were 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

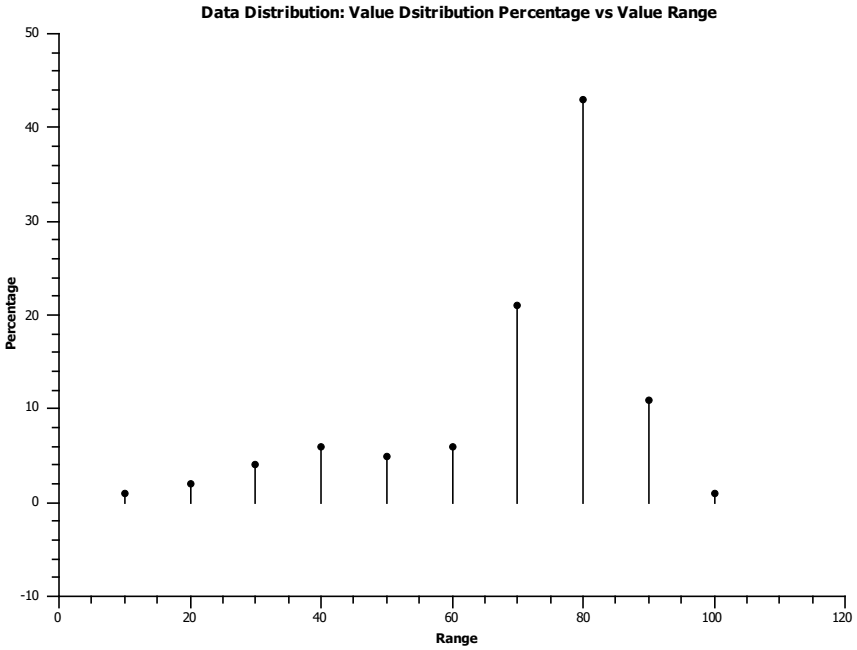

Figure 8 Distribution Percentage of CPU Attribute vs. Value Range

large number of resources were 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.

SLO Violation vs. VM

The experimental result shown in Figure 9 demonstrates how the proposed scheme performs to tackle SLO violation in comparison with centralized schemes. Both FF and FFD schemes generated a higher number of SLO violations compared to 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 24.11% 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.

Migration vs. VM

Similar behavior like the first experiment has been observed from the second one where number of migrations is observed. The proposed decentralized schemes generated significantly less number of migration requests than the centralized ones. Overall, the proposed system generates 33.43% less migrations on average as displayed in Figure 10. Compared to the first experiment, centralized schemes perform worst in case of overall

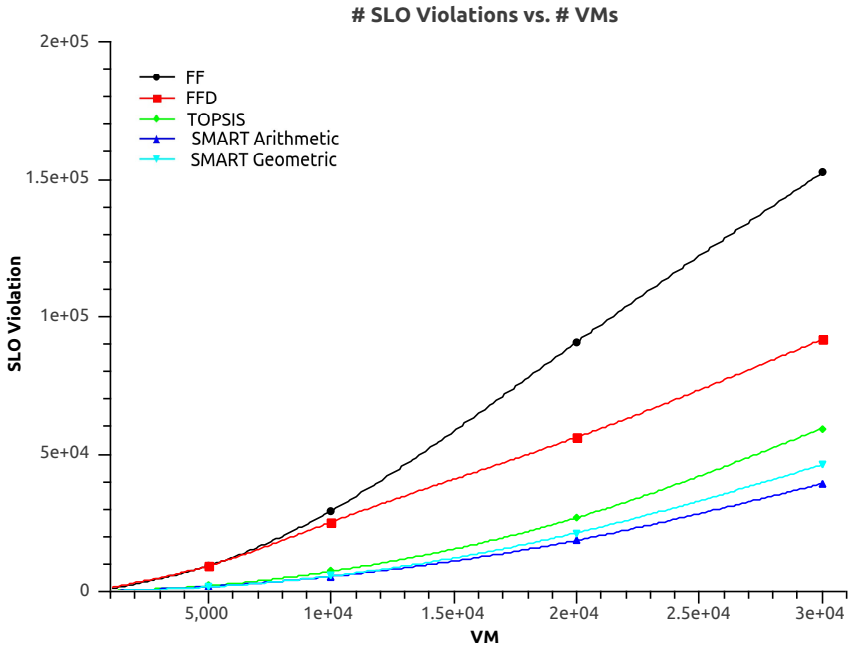

Figure 9 SLO Violation vs. VM

number of migrations. The hindrance created by the single Global Resource Arbiter 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.

Sprawling vs. VM

The third experiment, as shown in Figure 11 demonstrates server sprawling is literally zero in FF and FFD schemes 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.

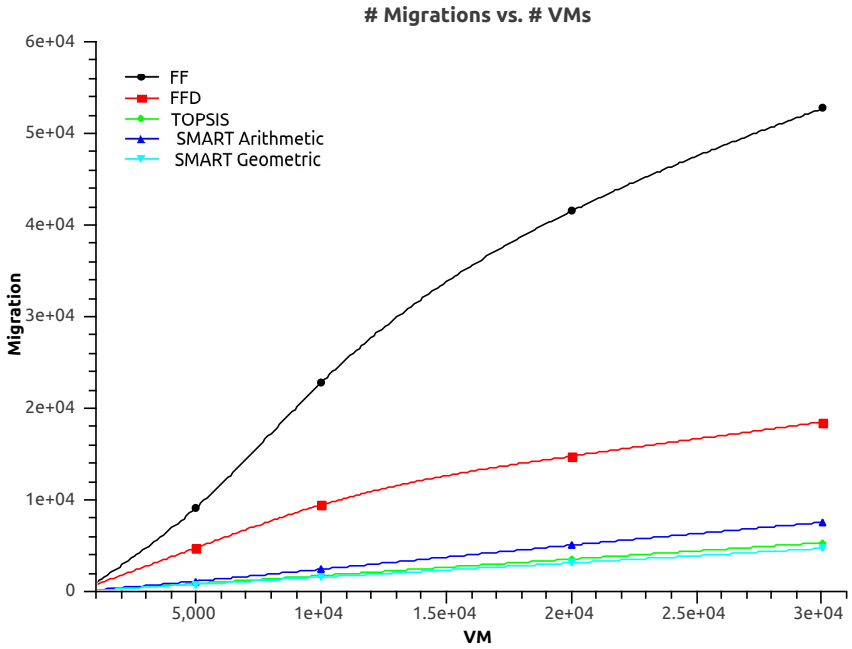

Figure 10 Migration vs. VM

8 Conclusion and Future Research Direction

Developing provisioning techniques that integrate application services in a scalable and fail-safe fashion is critical to the success of Cloud computing platforms. However, current market leaders such as OpenStack, Eucalyptus, Amazon EC2 feature centralized architecture of resource discovery and provisioning [2, 5, 10]. Thus SLO is always in threat of violation caused by the potential of sudden service outage due to scalability and single point failure issues. Hence, 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). In this research, a structured P2P based decentralized resource provisioning with the underlying resource discovery scheme are presented.

The work first proposes decentralized P2P based discovery scheme. It proposes how the inherent inability of answering multi dimensional range queries in structured DHT based P2P network is addressed by introducing an attribute hub based network overlay with data indexing, routing and storing inside the nodes. Resource publishing and querying along with node join and departure fail safe policies are also presented here. The research then proposes a decentralized resource provisioning scheme which is based on structured multi attribute range query P2P overlay network. Provisioning information from peer nodes are achieved via the proposed resource discovery method which supports multidimensional 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.

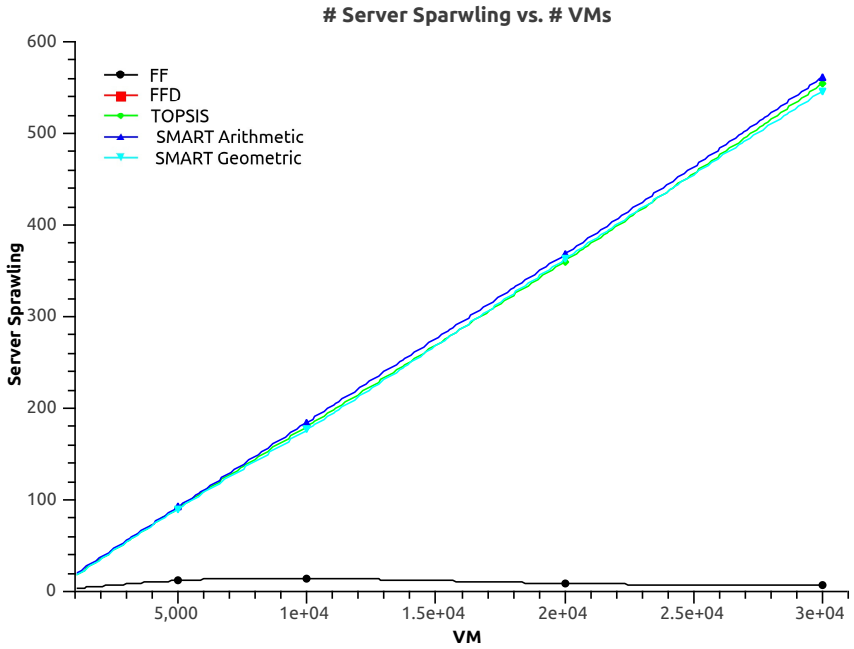

Figure 11 Sprawling vs. VM

In the simulation, the proposed resource discovery scheme is put to comparison with centralized and DHT based approach of resource discovery. From the obtained results, 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 maximum 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 the added secondary memory cost [40, 17, 12]. The proposed decentralized provisioning scheme has also been compared to global resource arbiter based centralized provisioning approach. Procured results demonstrate that the proposed system has shown a less number of SLO violation and migration causing slightly lower resource utilization which is about 24.11% and 33.43%. 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 [40].

On the basis of this proposed work, an 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 [30] 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.

Acknowledgment

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

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

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

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

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

[6] 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.

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

[8] 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.

[9] 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-amazon-says-ballmer. last accessed on May 20, 2014.

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

[11] 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.

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

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

[14] A. Amies, H. Sluiman, Q.G. Tong, and G.N. Liu. *Developing and Hosting Applications on the Cloud*.

[15] Shahram Bakhtiari, Reihaneh Safavi-Naini, Josef Pieprzyk, et al. Cryptographic hash functions: A survey. *Centre for Computer Security Research, Department of Computer Science, University of Wollongong, Australia*, 1995.

[16] Hari Balakrishnan, M Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in p2p systems. *Communications of the ACM*, 46(2):43–48, 2003.

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

[18] 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.

[19] Rajkumar Buyya, James Broberg, and Andrzej M Goscinski. *Cloud computing: Principles and paradigms*, volume 87. John Wiley & Sons, 2010.

[20] Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*, 5(2):164–177, 2012.

[21] 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.

[22] Frank Dabek. *A distributed hash table*. PhD thesis, Massachusetts Institute of Technology, 2005.

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

[24] José Figueira, Salvatore Greco, and Matthias Ehrgott. *Multiple criteria decision analysis: state of the art surveys*, volume 78. Springer, 2005.

[25] Armando Fox, Rean Griffith, A Joseph, R Katz, A Konwinski, G Lee, D Patterson, A Rabkin, and I Stoica. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28:13, 2009.

[26] Prasanna Ganesan, Beverly Yang, and Hector Garcia-Molina. One torus to rule them all: multi-dimensional queries in p2p systems. In *7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004*, pages 19–24. ACM, 2004.

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

[28] Mohammad Hamdaqa, Tassos Livogiannis, and Ladan Tahvildari. A reference model for developing cloud applications. In *CLOSER*, pages 98–103. Citeseer, 2011.

[29] Asif Imran, Alim Ul Gias, Rayhanur Rahman, and Kazi Sakib. Provintsec: a provenance cognition blueprint ensuring integrity and security for real life open source cloud. *International Journal of Information Privacy, Security and Integrity*, 1(4):360–380, 2013.

[30] Bahman Javadi, Jemal Abawajy, and Rajkumar Buyya. Failure-aware resource provisioning for hybrid cloud infrastructure. *Journal of parallel and distributed computing*, 72(10):1318–1331, 2012.

[31] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *29th annual ACM symposium on Theory of computing*, pages 654–663. ACM, 1997.

[32] Ralph L Keeney and Howard Raiffa. *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge University Press, 1993.

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

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

[35] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, Steven Lim, et al. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(1-4):72–93, 2005.

[36] GL Nemhauser, AHG Rinnooy Kan, and MJ Todd. *Handbooks in operations research and management science, vol. 1: optimization*, volume 8. North-Holland, 1989.

[37] 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.

[38] Pradeep Padala, Kang G Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, Arif Merchant, and Kenneth Salem. Adaptive control of virtualized resources in utility computing environments. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 289–302. ACM, 2007.

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

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

[41] Rajiv Ranjan, Aaron Harwood, and Rajkumar Buyya. Peer-to-peer-based resource discovery in global grids: a tutorial. *Communications Surveys & Tutorials, IEEE*, 10(2):6–33, 2008.

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

[43] Jia Rao, Xiangping Bu, Cheng-Zhong Xu, and Kun Wang. A distributed self-learning approach for elastic provisioning of virtualized cloud resources. In *19th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, pages 45–54. IEEE, 2011.

[44] 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.

[45] Hanan Samet. *The design and analysis of spatial data structures*, volume 85. Addison-Wesley Reading, MA, 1990.

[46] Rüdiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *1st International Conference on Peer-to-Peer Computing*, pages 101–102. IEEE, 2001.

[47] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *2nd ACM Symposium on Cloud Computing*, page 5. ACM, 2011.

[48] 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.

[49] Mark Stillwell, David Schanzenbach, Frédéric Vivien, and Henri Casanova. Resource allocation algorithms for virtualized service hosting platforms. *Journal of Parallel and Distributed Computing*, 70(9):962–974, 2010.

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

[51] 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.

[52] Gwo-Hshiung Tzeng and Jih-Jeng Huang. *Multiple attribute decision making: methods and applications*. CRC Press, 2011.

[53] J Varia. Cloud architecture- amazon web service. Technical report, Amazon Inc., 2009.

[54] Xiaoying Wang, Hui Xie, Rui Wang, Zhihui Du, and Li Jin. Design and implementation of adaptive resource co-allocation approaches for cloud service environments. In *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, volume 2, pages V2–484. IEEE, 2010.

[55] Min Yang and Yuanyuan Yang. An efficient hybrid peer-to-peer system for distributed data sharing. *IEEE Transactions on Computers*, 59(9):1158–1171, 2010.

[56] 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 *5th IEEE International Conference on Cloud Computing*, pages 614–621. IEEE, 2012.

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

[58] Xuehai Zhang, Jeffrey L Freschl, and Jennifer M Schopf. A performance study of monitoring and information services for distributed systems. In *12th IEEE International Symposium on High Performance Distributed Computing*, pages 270–281. IEEE, 2003.

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