2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing

An Availability-aware Virtual Machine Placement Approach for Dynamic Scaling of Cloud Applications

Wenting Wang, Haopeng Chen, Xi Chen
REINS Group, School of Software
Shanghai Jiao Tong University
Shanghai, P.R.China
{wwtvanessa, chen-hp, chenxi_sjtu}@sjtu.edu.cn

***Abstract*- Cloud computing promises customers the on-demand ability to dynamically provision virtualization resources in face of workload variations. Most existing scaling approaches addressed this problem by allocating application to a certain amount of cloud resources. However, the problem of the availability of application influenced by VM-based physical locations during resource scaling process is a serious challenge due to dynamic complex workload and has not been widely discussed yet. In this paper, we present a novel availability-based computing model to describe availability attribute of one application in the hierarchical structured cloud. Moreover, we propose an availability-aware policy by performing both vertical and horizontal scaling to explore how and where to allocate computing resource. Simulation results indicate that our model captured the availability of cloud applications properly and proposed scaling approach achieves the objectives of meeting availability demands and minimizing the total communication cost.**

Keywords-Cloud-computing; VMs placement; Scaling up and down; Availability; Resizing

I. INTRODUCTION

Virtualization is one of the most prominent technologies that make cloud computing, a large-scale distributed computing paradigm [1] providing on-demand and reliable computational resources, possible. Generally, as shown in Fig. 1, cloud applications are provisioned with a set of Virtual Machines (VMs) sharing physical resources. The responsiveness and availability requirements satisfaction [2][14], and commercial profits [29] of one application greatly depend on dynamic allocation strategy applied in scaling. Compared with intensively discussed issues of the resource allocation [4][18][25][29], such as how many VMs should be added or removed and when to perform scaling in datacenter, some essential principles in such problem, i.e. the methods of VMs-based scaling have not been widely discussed yet. To be specific, VMs-based scalability can be implemented by either changing the partition of resources (e.g. CPU, memory, storage, etc.) inside a VM or adjusting the amount of VM instances. These two kinds of scalability lead to different impacts on other resource management decisions, such as VM migration and consolidation. Moreover, they behave differently on affecting satisfactions for customer's QoS requirements expressed in Service Level Agreements (SLAs), such as availability and performance. In this paper, we focus on the comparison of these two scaling methods, which are referred as vertical and horizontal resizing respectively in our study.

When scaling resource in cloud applications, availability of applications is equivalently crucial as SLA-driven and profits-driven requirements [4][15][29] especially for enterprise-level services. Scaling resolution need to consider the locations of VMs and the availability features of cloud to satisfy the availability demand of applications. To be specific, the availability of the application is affected by how its components (i.e. VMs) are placed relatively to one another in the cloud [14]. It is also significantly influenced by the specific structure of clouds. For instance, Amazon provides a relatively transparent structure of EC2, namely regions and availability zones, in which allows customers to launch instances to protect their applications from a failure of a single location [7]. How to locate VM instances in such cloud environment is another unsettled issue for customers. Therefore, this paper addressed the problems of: 1) modeling the availability of cloud and the cloud application without loss of generality; 2) resource placement of dynamic scaling in an availability-aware and cost-efficient way integrated with vertical and horizontal resizing policies.

The remainder of this paper is organized as follows: Section 2 presents the architecture of the proposed system. Section 3 discusses the comparison of two different policies of dynamic scaling. In Section 4, the availability of hierarchical datacenters and applications are modeled, and the availability-aware approach by enabling both VM-based

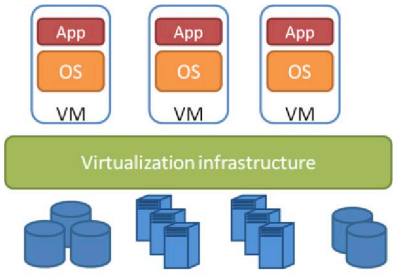


Figure1 Applications in the cloud

978-0-7695-4843-2/12 $26.00 © 2012 IEEE
DOI 10.1109/UIC-ATC.2012.31

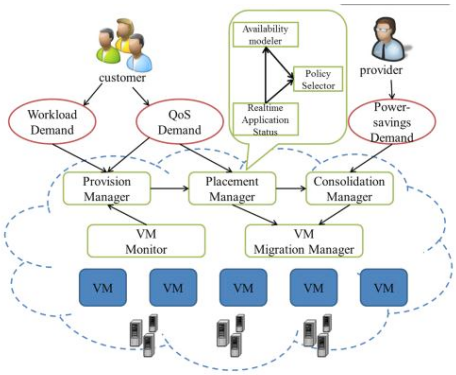


Figure2 Placement Manager in Resource Management

scalability methods is proposed. The simulation results are presented in Section 5, related work and conclusion are summarized in Section 6 and Section 7 respectively.

II. ARCHITECTURE

In this section, the architecture of resource management in clouds is presented. As shown in Fig. 2, the high-level architecture is composed by the following components:

- VM Monitor collects and tracks the data of resource utilization on various metrics of VMs for Provision Manager.
- Provision Manager evaluates the amount of virtual resources needs to be scaled up or down.
- Placement Manager locates VMs onto physical hosts and into different areas after receiving evaluation from Provision Manager, which is composed of following components:
  a) *Real-time Application Status* is composed of the locations, size and other detailed information of customer's applications.
  b) *Availability modeler* uses the data offered by *Real-time Application Status* to evaluate the availability of applications.
  c) *Policy selector* makes decisions about which policies should be chosen and where to locate the resized resource based on the data provided by the other two components.
- Consolidation Manager performs VM consolidation by migrating VMs to a smaller number of physical servers to achieve energy savings.
- VM Migration Manager manages the process of VM migration triggered by adjusting VM's logical metrics or consolidation.

III. COMPARISON OF VERTICAL RESIZING AND HORIZONTAL RESIZING

Vertical resizing and horizontal resizing are widely adopted resolutions in implementation of VM-based

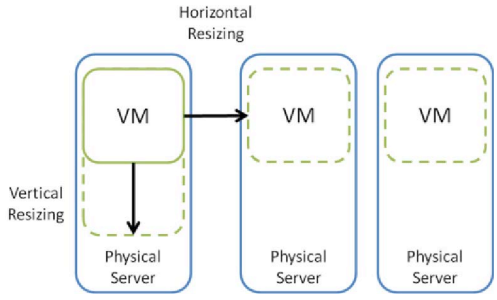


Figure3. Vertical Resizing and Horizontal Resizing

dynamic resource allocation (see Fig. 3). Specifically, Vertical Resizing is a basic feature of VM which adjusts logical partition of multiple metrics (e.g. CPU, Memory, Bandwidth, I/O etc.) in a VM. Dynamic Logical Partition Resizing (DLPAR) allows users to logically attach and detach resource to and from a logical partition's division without rebooting [5]. On the other hand, Horizontal Resizing changes the number of VM instances, which involves running applications on two or more separate VMs hosted on different servers. As illustrated in Table1, the main different aspects that Vertical Resizing and Horizontal Resizing have impacts on are characterized as follows:

3.1 Physical Limitation:

The physical upper limitation of Vertical Resizing is the spare computational resources of the VM's current host. When computational resource a VM demands exceeds this limitation, Vertical Resizing up will fail and has to turn to VM migration. While for Horizontal Resizing, within fine-grained VMs, the upper physical limitation is the overall spare resources inside clouds. Theoretically, cloud provider can offer as many VMs as possible in response to customers' demands. If the resources in clouds exhausts, cloud federation [8] could be put into use.

3.2 Cost

The cost of resizing mainly includes three parts: the time of reconfiguration, the cost of migration and the spending on software license.

First, regarding to the time of reconfiguration, Vertical Resizing is obviously time-efficient. DLPAR technology used in Vertical Resizing has been experimented in [6] that the duration time of Vertical Resizing on pHyp hypervisor and VMWare ESX3 was less than 1 second. However, the duration of Horizontal reconfiguration consists of new instance copying time and initializing time. This process involves VM Migration and time cost of migration is generally far more than the time cost of metrics adjustment.

Table1. THE COMPARISON BETWEEN VERTICAL RESIZING AND HORIZONTAL RESIZING

		Vertical Resizing	Horizontal Resizing
Physical Limitation		Resource of a single host	Resource of a Cluster
Cost	***Time of reconfiguration***	Short	Long
	Cost of Migration	High	Low
	Additional software license	Low	High
Availability effect		Weak	Strong
Other concerns		No coordination overhead	Need load balance and gateway

Secondly, resource fragments after a long term scaling requires cloud providers to perform server consolidation. Consolidation [28][9] is an effective approach involving a large scale migration[3][13] to a smaller number of servers in order to minimize energy consumption. Some researchers such as Y.Tamura [12] presented the method that enhanced the efficiency of VM migration by Network File System (NFS) such as KEMARI [11]. However, the extra support of NFS increases the complexity of system. Furthermore, the consolidation after Vertical Resizing scaling is much more complicated than since the fragments caused by it have are probably irregular. Hence, the cost of migration caused by Vertical Resizing is higher than Horizontal Resizing.

Lastly, Horizontal Resizing incurs additional cost due to the increased number of licenses while Vertical Resizing does not add the software payments, which is also a main concern for cloud customers.

3.3 Availability

Vertical Resizing does not affect the quantity of VM instances; therefore it does not change the distribution of applications, consequently maintaining the availability. Horizontal Resizing, on the other hand, can affect the overall availability of cloud application by increasing the number and relocate VMs, and vice versa.

3.4 Other Concerns

There are some other concerns remaining in resizing activities. Vertical method allows users to beef up a VM with no coordination or latency among multiple instances. For example, when scaling up a VM where a Database is hosted in, Vertical Resizing is prior to Horizontal Resizing because multiple instances of Database would cause synchronization. Moreover, Horizontal Resizing needs a gateway and workload balance that dispatches requests to multiple instances which makes it resource-consuming, which provides more throughout at expense of complexity.

IV. MODELING AND ALGORITHM

We have designed an availability-aware scaling approach to address the problem of resources placement in clouds. Our objectives is to improve overall system availability while maintain communication costs. The main steps of our approach involve three phases: First, regarding to the different geographical network topology and logical groupings of cloud datacenters from multiple providers, modeling the cloud structure and the availability of applications must be done in effective and scalable way to support heterogeneous environment. Second, the performance in the application is represented in measuring communication cost among VMs composing the application. Third, the placement plan which performs both vertical and horizontal methods will be made according to the availability gap between the existing application and customer's demand by single step.

4.1 Modeling

The modeling consists of three parts- namely the structural cloud datacenter, the availability and the communication costs of the application hosted on cloud.

A. Datacenter Modeling

We model the cloud infrastructure as a tree structure with arbitrary depth (see Fig. 4). In the tree structure of the cloud, top levels which have been already taken into commercial providers' consideration such as Amazon [7] are Regions and Availability Zones: The Regions which are physically dispersed in the geography incur high availability, while the Availability Zones provides isolation of failure due to independent networking, power, and cooling systems of each other. The nodes of bottom level are Physical Hosts where VMs are hosted. Without loss of generality, the structure of middle levels (e.g. server racks), such as the number of levels can be defined in a scalable way as needed. We implicitly assume that all the nodes in the same level are at equal height and all VMs should be placed in leaves of the tree namely physical hosts.

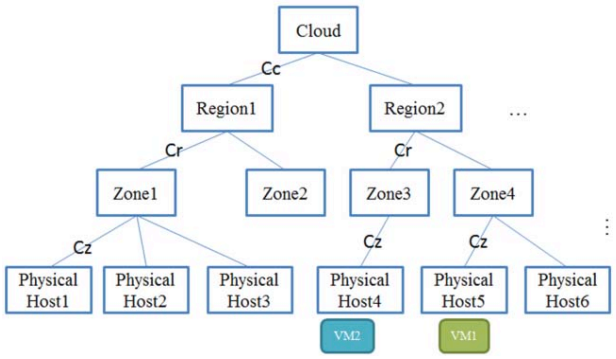


Figure4. Availability Model in clouds

B. Availability Modeling

The failures in nodes of different levels are caused by various reasons. For instance, a region failure may be caused by natural disasters such as earthquake or flood whereas the failure in availability zone may be caused by blackout and the reason of a physical server breakdown could be hardware failure. Thus, the availability of a VM on a physical server depends on the availability of physical host as well as its parent nodes such as availability zones and regions. Specifically, the availability of a VM on physical leaf node v denoted as $P_v(VM)$ is evaluated as follows (where set $N(v)=\{n_1,n_2,\ldots,n_n\}$ specifies all parent nodes of node v, P_j is the availability of one node in the tree):

$$P_v(VM) = \prod_{j \in N(v)} P_j \quad (1)$$

However, the failure probability in two VMs host on u_{th} and v_{th} physical nodes specified as $(\overline{P_u(VM)} \wedge \overline{P_v(VM)})$ cannot be simply calculated as $\overline{P_u(VM)} * \overline{P_v(VM)}$ because the hierarchical paths of u_{th} and v_{th} node might be not completely independent, which is to say they may share common parent nodes. Hence, calculating the failure probability in two VMs involves two parts: (i) probability of the failure happens in any of their common parents and (ii) the probability of failure happens at the same time in their own private hierarchical path under their first common parent when all their common parents function well. The failure probability in two nodes is given by:

$$\left(\overline{P_u(VM)}\right) \bigwedge \left(\overline{P_v(VM)}\right) = \overline{\prod_{n \in C(u,v)} P_n} \quad (2)$$

$$+ \prod_{n \in C(u,v)} P_n * \left(\overline{\prod_{x \in N(u), x \notin C(u,v)} P_x}\right)\left(\overline{\prod_{y \in N(v), y \notin C(u,v)} P_y}\right)$$

where $C(u, v)=\{ n_1,n_2,\ldots,n_n\}$ which are common parents for node u and v. The first part on the right side of equation is the failure probability of common parents of u and v; the second part is the availability of common parents multiples the failure probability in private path of u and v.

Algorithm1: AvailabilityAwarePlacement
Input: **S={$V_1,V_2,\ldots,V_n$}** denotes the set of all virtual machines in the application; **Quantity** denotes demanded unitized resource quantity need to be resized up or down; **Scale**=up/down shows scaling flag; **Ac** is the current availability of the application; **Ar** is the demanded availability; **relocatedTimes**: the max times of relocation
Output: scaling solution in physical nodes

```
1: Ac=calculateAvailability();
2: t =1;
3 :For(k=1;k<=Quantity;k++){
4:   If(scale == up){
5:      //the current availability is met
6:       If(Ac>=Ar){
7:           VerticalResizeUp(S, 1);
8:       }
9:       Else{// Ac<Ar
10:          HorizontalResizeUp(S, 1, t);
11           t++;
12:       }
13:  }
14:  Else{//scale down
15:       VerticalResizeDown(S, 1);
16:  }
17:  Ac=calculateAvailability();
18: }//end for
19: while(Ac<Ar && relocatedTimes >0){
20:    //rebalance overall application
21:    Relocate(S);
22:    Ac=calculateAvailability();
23:    relocatedTimes --;
24:}
```

Figure5. Algorithm1: AvailabilityAwarePlacement

More generally, we generalize the calculation approach from two VMs to multiple ones. First, we denote $\overline{P'_m}$ as the probability of failure for the sub-tree with root m, which is caused by (i) the failure due to the node m itself($\overline{P_m}$) and (ii).the failure due to its children nodes when the root functions well($P_m \prod_{n \in children(m)} \overline{P'_n}$). The failure probability of sub-tree is shown as follows in a recursion:

$$\overline{P'_m} = \overline{P_m} + P_m \prod_{n \in children(m)} \overline{P'_n} \quad (3)$$

The evaluation of the failure probability with a cluster of VMs as $P(\bigwedge_{n=1}^{k}(\overline{P_n(VM)}))$ is the same as $\overline{P'_m}$ where m is the root of the sub-tree generated by the multiple VMs. Assumingly the number of VMs in an application is k, the calculation of the application availability(denoted as A) is based on the sub-tree generated by multiple VMs:

$$A = 1 - P(\bigwedge_{n=1}^{k}(\overline{P_n(VM)})) \quad (4)$$

```
Alogrithm2:VerticalResizeUp
Input: S denotes the set of all virtual machines in the
application; Quant: the quantity of demanded unit VMs

1:For every VM in S{
2:If(SpareQuant =VM.host().spareRes()>1){
3:        VM.ResizeUp(min(Quant,SpareQuant));
4:        Quant -= min(Quant,SpareQuant)
5:    }
6:      If(Quant == 0) return;
7: }
8: HorizontalResizeUp(S,Quant,1);
```

Figure6. Algorithm2:VerticalResizingUp

```
Alogrithm3: HorizontalResizeUp
Input: S denotes the set of all virtual machines in the
application; Quant: the quantity of demanded unit VMs;
t: the threshold for availability improvement
1:While(Quant>0){
2:   Find a host where(cc(VM,S)>=t&&
3:      SpareQuant =VM.host().spareRes()>1)
4:   {
5:    host.newVM(min(Quant,SpareQuant));
6:    Quant -= min(Quant,SpareQuant);
7:   }
8:}
```

Figure7. Algorithm3: HorizontalResizeUp

C. Communication Cost Modeling

Performance affected by communication cost is another significant issue in virtualized resource scaling and attracted attention recently [23][14]. In our modeled datacenter, the communication cost between two VMs in the same host is assumed in zero, and the communication costs remain equal in the same hierarchy. Regions may be more geographically insulated than Availability Zones, consequently incurring higher communication cost (e.g., lower latency or more bandwidth). The communication cost between two VMs can be formalized in measuring the end-to-end communication cost between two hosts [14]. For example, let $cc(v_1,v_2)$ donates the communication cost between VM_1 and VM_2 in Fig. 4and as 2Cz+Cr.

Then the communication cost from one VM v to the other VMs in an application (where S is the set of all VMs composing the application) is

$$\mathrm{cc}(\mathrm{v}, \mathrm{S} - \{\mathrm{v}\}) = \sum_{x!=v} cc(v,x) \qquad (5)$$

Thus, the problem is to find the correct placement solution of satisfying the availability demand while minimizing the cost from communication to improve the overall performance.

```
Alogrithm4:VerticalResizeDown
Input: S denotes the set of all virtual machines in the
application; Quant: the quantity of demanded unit VMs
1:For every VM in S{
2:   if V_j.size() > 1 {
3:        //Atomic operation
4:      VM.ReizeDown(1);
5:        Quant--;
6:   }
7:   If(Quant == 0) return;
8: }//end for
9: HorizontalResizeDown(S,Quant,1)
```

Figure8. Algorithm4: VerticalResizingDown

```
Alogrithm5: HorizontalResizeDown
Input: S denotes the set of all virtual machines in the
application; Quant: the quantity of demanded unit VMs;
t: the threshold for availability improvement
1:While(Quant>0){
2:   if Find a VM in S(
3:      min{cc(VM,S-{VM})}<=t){
4:      If VM.size()==1
5:        VM.terminate();
6:      else
7:        VM. ReizeDown(1);
8:      Quant--;
9:    }
10:   else t++;
11:}
```

Figure9. Algorithm5: HorizontalResizeDown

4.2 Availability-Aware Placement Approach

Here we present an availability-aware scaling placement algorithm as illustrated in Fig. 5. The increased or decreased amount of VMs predicted provisioning component offered has been studied in the literature [4][17][29]. Therefore, in our work, we assume implicitly all the required VMs are standardized in unitized size. A_c and A_r in our algorithm are calculated using availability computational model in Eq(4) stated in Section 4.1.

Algorithm 1 generally provides the decision between Vertical Resizing and Horizontal Resizing by single step depending on the availability difference of the existing situation and requirement. To be specific, we vertically resize (in Fig. 6) the existing resource to maintain the communication and software cost, when the availability requirement has been already fulfilled. Otherwise, HorizontalResizeUp (in Fig. 7) is prior. We pick a new host by dynamically measuring the communication cost to the current VMs group, namely a threshold. A higher threshold implies the greater availability enhancement to the overall

application. In scaling down case, we vertically scale down the current resource to reduce the influence on overall availability, as shown in Fig. 8. When all VMs in the application stay in unit size, HorizontalResizeDown has to be involved since no VMs can be vertically resized down. Algorithm 5 in Fig. 9 terminates a VM whose communication cost to others is smallest, which, in other words, maintain the availability to the largest extent.

Finally, in Fig. 10, we provide an effective mechanism to ensure the availability satisfied. Specifically, Relocate focus on enabling the application to generate a more distributed placement through relocating one of its VMs on a new host.

In brief, we maximized the availability improvement when scaling up and maintained the existing availability level when the application needs to be resized down. Algorithm 1 presents the brief guidance of how to remove or add VMs and other algorithms give the detailed implementation of vertical and horizontal resizing. Furthermore, we propose a relocation mechanism to ensure the overall availability satisfied to the largest extent.

V. EVALUATION

To evaluate the effectiveness of our proposal, we have created a prototype of the proposed framework in Java; there we have implemented all the algorithms in Section 4. To the best of our knowledge, the hierarchical cloud computing infrastructure in our model has not been fully supported in any other existing tools. Thus, we simulated a cloud in a hypothesized environment consisting of over 200 homogenous physical machines. Moreover, we set up 5 regions, each of which is composed of 5 availability zones with an even distribution of all hosts. Assumingly, regions and availability zones have 95% and 93% availability respectively, and the availability of physical machines is 90%.

The execution of the simulation consisted of two phases: The first one is to deploy an application comprising a collection of VMs randomly distributed in the cloud. The initial placement of the application makes different availability probability and communication cost at first. The second phase simulated two typical scenarios, namely scaling up and scaling down, in which we evaluated the proposed availability-aware scaling approach via the comparative analysis of improvements among our algorithm and pure vertical and horizontal methods. In the sense that our proposal is provided for the single application in clouds, a simulation of massive scale applications is no necessity in our study. Therefore we assessed availability-aware scaling algorithms by adapting the application size from 3 VMs to 9 VMs when resizing up and from 8VMs to 2 VMs when resizing down, for each of which we presented the averaged results over 300 independent executions.

```
Alogrithm6: Relocate
Input: S denotes the set of all virtual machines in the
application;

1:For(every VM in S){
2:    t=min{cc(VM,S-VM)};
3:}
4: HorizontalResizeDown(S,1,t);
5: HorizontalResizeUp(S,1,t+1);
```

Figure10. Algorithm6: Relocate

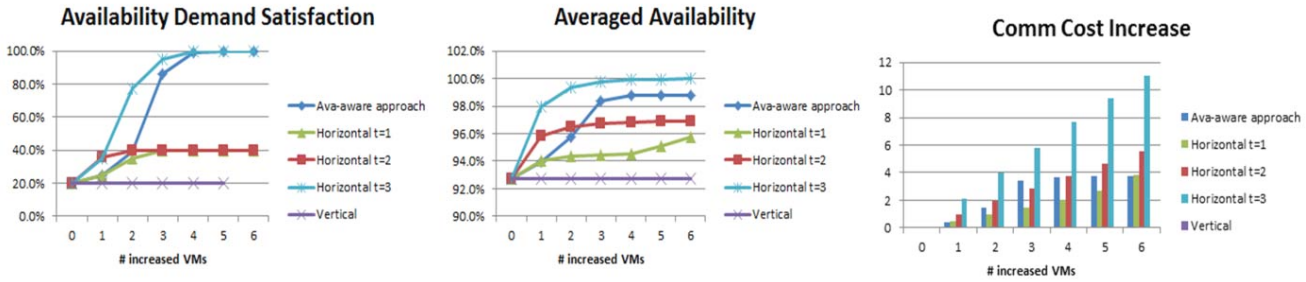
First, we investigated the availability influence by scaling up VMs to the existing application in Fig. 11. Vertical method did not change the number of VMs, thereby showing no impact on average availability or demand satisfaction. Horizontal Resizing method with larger threshold (i.e. t=3) apparently achieved a higher availability satisfaction and averaged level. However, it produced a placement solution with a significantly high communication cost as shown in Fig. 11(c). While our approach produces a more cost-efficient scaling solution of availability-satisfied placement, which is due to the fact that our threshold adjusts dynamically and our approach changes to vertical resizing method once the availability demand is met, providing a placement with less communication cost than horizontal one as the size of resource increases.

Next, we evaluated the scaling down case by the same measurement in Fig. 12. By reducing the amount of VM instances, availability average and satisfaction in horizontal method drops more quickly than the others and the communication cost reduces significantly more as the decrease ratio indicates in Fig. 12(c). Our approach showed similarity with vertical resizing down in the beginning of scaling down since ours resizes resource down as the vertical one did. Moreover, the averaged availability and satisfaction of the relocation mechanism surged a little because it rearranged locations of VMs group.

The results obtained from experiments validate our assumptions, where our algorithm satisfied most availability requirement and minimizing the cost from communication.

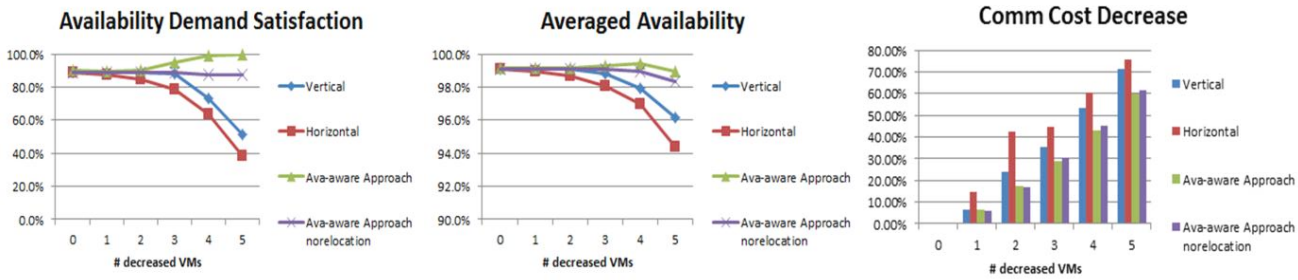
VI. RELATED WORK

In cloud environment, an important issue, which is also the premise of our work, is the provisioning of virtual resource. There is a wide-range of prior research around resource provisioning proposed in virtual infrastructure environment. B. Urgaonkar [4] introduced a horizontal resizing approach to adjust its workload intensity by modeling multi-tier application as a queue network, which had the similarities with [29]. [18] took utilization into consideration and abstracted response time to predict provision of virtualized resources. In [25], Sotomayor



(a)Demand satisfaction (b) Averaged availability (c)Communication cost Increase ratio

Figure11 availability enhancement and performance change when scaling up



(a)Demand satisfaction (b) Averaged availability (c)Communication cost Decrease ratio

Figure12 availability declination and performance change when scaling down

showed a capacity provisioning model based on a variety of lease types provided by OpenNebula and Haizea. However, the difference between policies of scaling, such as vertical and horizontal resizing as we presented, has not been discussed much. [6] gave a brief comparison but mainly focused on the impact of live migration on vertical resizing.

In addition, resource placement has been a significant issue for resource management. There are many works that discuss the power optimization issue of placement in large scale. Generally, they have proposed power model for power analysis and placement algorithm based on power model and migration cost model. A power-aware placement algorithm that minimizes power and migration costs has been proposed in [10] while [19] used similar placement approach on HPC application. Another approach, which was introduced in [20], presented power-performance tradeoffs in modern data centers. [24] addressed dynamic placement issue by an energy-aware heuristic approach.

There are also some works that proposed a placement approach to improve the performance in distributed virtualization environment, for example [26]. Jayasinghe et al. [14] proposed an effective way to reduce communication costs and improve overall system availability. A similar network-aware approach was proposed in [23]. However, they did not give an explicit computation model of availability in clouds. Our work presents the missing piece for placement in resource scaling by providing an availability-aware approach. Besides power-saving and performance enhancement, placement has been investigated based on minimizing the cost. Kimbel [21] focused on minimizing the cost of VM migration in resource reallocation, while Shahabuddin et al. [22] presented a heuristic algorithm in order to minimize the quantity of required hosts.

VII. CONCLUSION AND FUTURE WORK

In cloud environment, the elastic feature of clouds computing requires providers to scale the resource up or down in response to workload intensity. To implement cloud scalability that can efficiently allocate virtual resource to applications, we look the methods namely vertical and horizontal resizing, and compared their influence on resource management and QoS of cloud applications in our study. Although Vertical Resizing has advantages on performance, it creates side effects such as low-availability, low utilization and the cost of consolidation. On the other hand, Horizontal Resizing can enhance the overall availability of applications but brings additional cost of software license and configuration time.

Moreover, we model cloud infrastructure into a hierarchical structure without loss of generality and scalability. For the application in such cloud, we also provide an availability computation model to measure the availability attribute of the application. Aiming at satisfying the availability requirements from customers, an availability -aware placement approach by taking advantages of both

Vertical and Horizontal Resizing to dictate where and how to add or remove resource to the application is proposed next. Furthermore, we design and implement a simulation experiment that employs our proposed methodology to evaluate the feasibility of our approach.

Our next work will focus on the improvement of our approach by taking other factors into account such as multi-tenancy. In addition we will examine the practicability and correctness of our availability model and placement approach in large scale cloud environment, e.g. combing the provision method [28] in Eucalyptus-based environment.

REFERENCE

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. H. Katz etc, "Above the Clouds: A Berkeley View of Cloud Computing," Technical Report No. UCB/EECS-2009-28. Symantec. 2010 State of the Data Center Global Data.

[2] L. Grit, D. Irwin, A. Yumerefendi, and J. Chase. "Virtual Machine Hosting for Networked Clusters: Building the Foundations for Autonomic Orchestration", In *Proc. VTDC '06*.

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. War_eld. Live Migration of Virtual Machines. In Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI), May 2005

[4] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitizer, A. Tantawi, "An Analytical Model for Multi-tier Internet services and its applications," In Proceedings of ACM SIGMETRICS, 2005 .

[5] http://en.wikipedia.org/wiki/Dynamic_Logical_Partitioning,

[6] A. Verma, G. Kumar, and R. Koller. The cost of reconfiguration in a cloud. In Proc. Middleware (Industrial Track), 2010

[7] Amazon Elastic Computing Cloud, https://aws.amazon.com/ec2/

[8] Buyya R, Ranjan R, Calheiros RN. InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services. Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010) *May* 2010*;* 328–336.

[9] R. Nathuji and K. Schwan. Virtualpower:coordinated power management in virtualized enterprise systems. In Proc. ACM SOSP, 2007.

[10] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," in *ACM 9th,International Middleware Conference*, 2008.

[11] KEMARI, http://www.osrg.net/kemari/

[12] Y. Tamura. Kemari: Virtual machine synchronization for fault tolerance using domt. *Xen Summit*, June 2008

[13] M.R. Hines and K. Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *VEE '09: Proceedings of the 2009 ACM* SIGPLAN/SIGOPS international conference on Virtual execution environments, 2009. ACM,51–60

[14] D. Jayasinghe, ,C. Pu,T. Eilam, M. Steinder, I.Whally, E.Snible, Improving Performance and Availability of Services Hosted on IaaS Clouds with Structural Constraint-aware Virtual Machine Placement, Services Computing (SCC), 2011 IEEE International Conference, July 2011

[15] X. Wang, Z. Du, Y. Chen, and S. Li. Virtualization-based autonomic resource management,for multi-tier web applications in shared data center.Journal of Systems and Software, 2008.

[16] Dai YS, Xie M, Poh KL, Liu GQ. A study of service reliability and availability for distributed systems. Reliab Eng Syst Saf 2003;79:103–12.

[17] Lai CD, Xie M, Poh KL, Dai YS, Yang P. A model for availability analysis of distributed software/hardware systems. Inform Software Technol 2002;44(6):343–50.

[18] Q. Zhang, L. Cherkasova, E. Smirni, "A Regression-Based Analytic model for Dynamic Resource Provisioning of Multi-Tier Application," In Proceedings of the 4th International Conference on Autonomic Computing, IEEE, 2007

[19] A. Verma, P. Ahuja, and A. Neogi. Power-aware dynamic placement of hpc applications. In Proc. ACM ICS, 2008.

[20] M. Cardosa, M. Korupolu, and A. Singh. Shares and utilities based power consolidation in virtualized server environments.In Proceedings of IFIP/IEEE Integrated Network Management (IM), 2009.

[21] T. Kimbrel, M. Steinder, M. Svirdenko, and A. Tantawi. Dynamic Application Placement under Service and Memory Constraints. In Workshop on Efficient and Experimental Algorithms, 2005

[22] J. Shahabuddin, A. Chrungoo, V. Gupta, S. Juneja, S. Kapoor, and A. Kumar. Stream-Packing: Resource Allocation in Web Server Farms with a QoS Guarantee. Lecture Notes in Computer Science, 2001.

[23] X. Meng, V. Pappas, and Li. Zhang. Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. In INFOCOM 2010.

[24] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong. EnaCloud: An Energysaving Application Live Placement Approach for Cloud Computing Environments. In CLOUD 2010.

[25] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster., "Capacity Leasing in Cloud Systems using the OpenNebula Engine," Proc. Cloud Computing and Applications 2008 (CCA 08), 2008; www. cca08.org/papers.php.

[26] D. Ta, S. Zhou, W. Cai, and X. Tang. Network-Aware Server Placement for Highly Interactive Distributed Virtual Environments. In IEEE/ACM DS-RT 2008.

[27] B. Sotomayor, R.S. Montero, I.M. Llorente and I. Foster, Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, **13** 5 (2009), pp. 14–22

[28] Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications,1:7–18

[29] X Chen, H. Chen, Q. Zheng, W. Wang and G. Liu; Characterizing web application performance for maximizing service providers' profits in clouds in: Cloud and Service Computing (CSC), 2011 International Conference,2011