An Availability-aware Approach to Resource Placement of Dynamic Scaling in Clouds

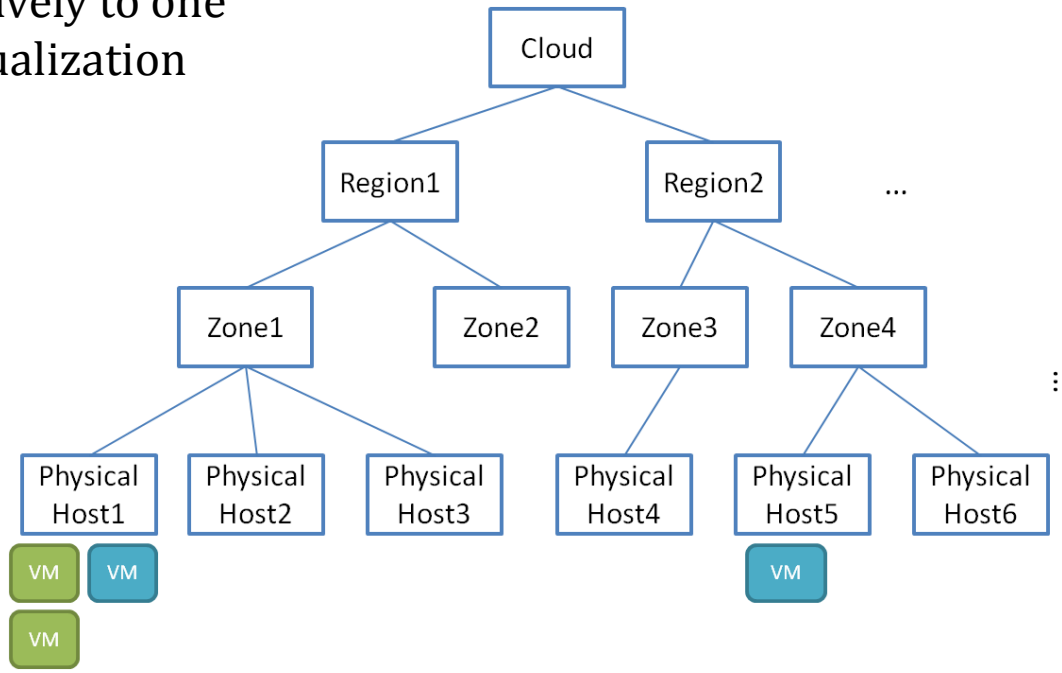
Wenting Wang, Haopeng Chen, Xi Chen
Reins Group, http://reins.se.sjtu.edu.cn
wwtvanessa@gmail.com
Shanghai Jiao Tong University,
China
2012-09-05

Agenda

- Problem Statement
- Availability Modeling
- Availability-aware Approach
- Evaluation
- Future Work
- Summary

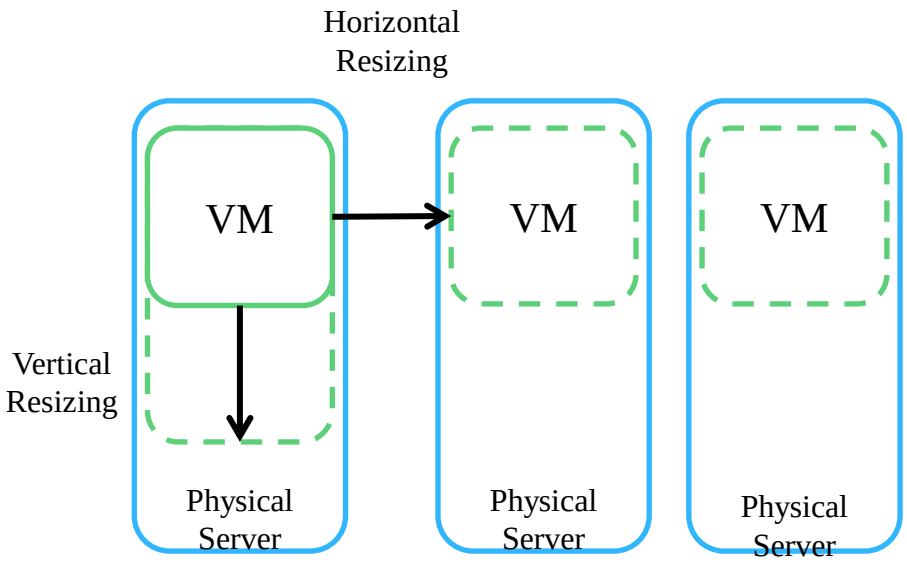
Problem Statement

- Three Factors Affect the Availability
  1) Cloud Environment
     - Regions
     - Availability Zones
  2) Relative Locations
     - How components (i.e. VM instances) of the application are placed relatively to one another in distributed virtualization environment



Problem Statement

- 3) Scaling Policy
  - Vertical scaling:
    - change the partition of resources (e.g. CPU, memory, storage, etc.) inside a VM
  - Horizontal scaling:
    - adjust the amount of VM instances



Problem Statement

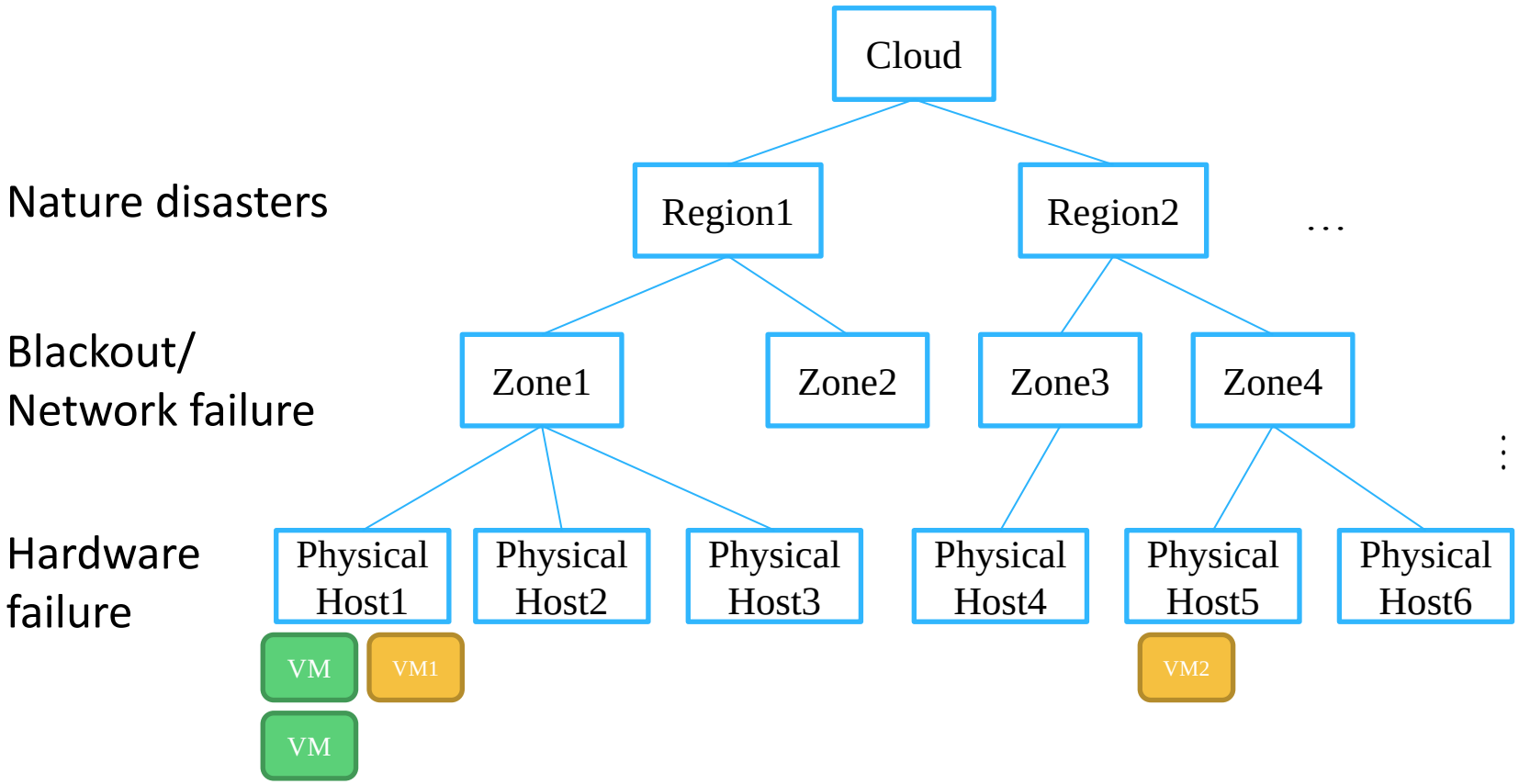
- Objectives
  - Modeling the availability of IaaS cloud
  - Modeling the availability of the dedicated infrastructure of a cloud application
  - Resource placement of dynamic scaling in an availability-aware and cost-efficient way integrated with vertical and horizontal resizing policies

Agenda

- Problem Statement
- Availability Modeling
  - *Datacenter Modeling*
  - *Availability Modeling*
  - *Communication Cost Modeling*
- Availability-aware Approach
- Evaluation
- Future Work
- Summary

Availability Modeling

- Datacenter Modeling

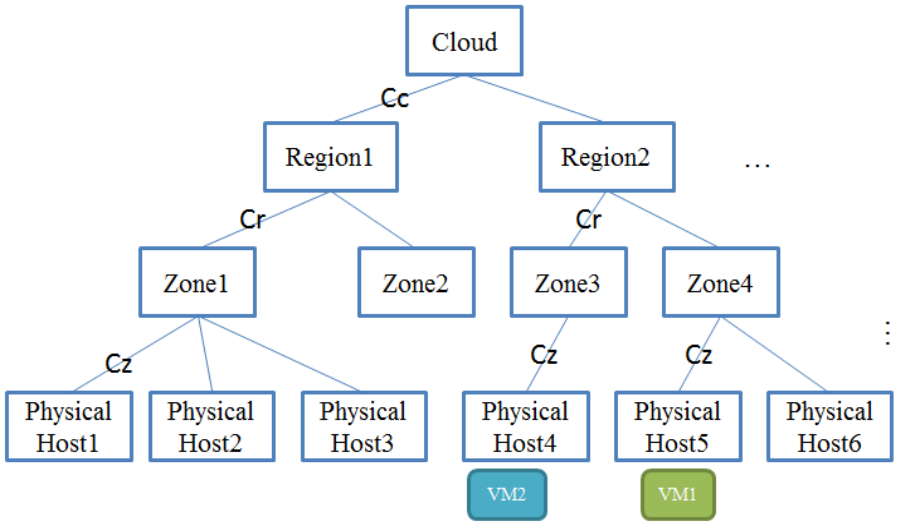


Availability Modeling

REliable, INtelligent & Scalable Systems

- The Availability of One VM

$$P_i(VM) = P_i \prod_{j \in PP(i)} P_j$$



- The Failure Probability of Two VMs

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

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

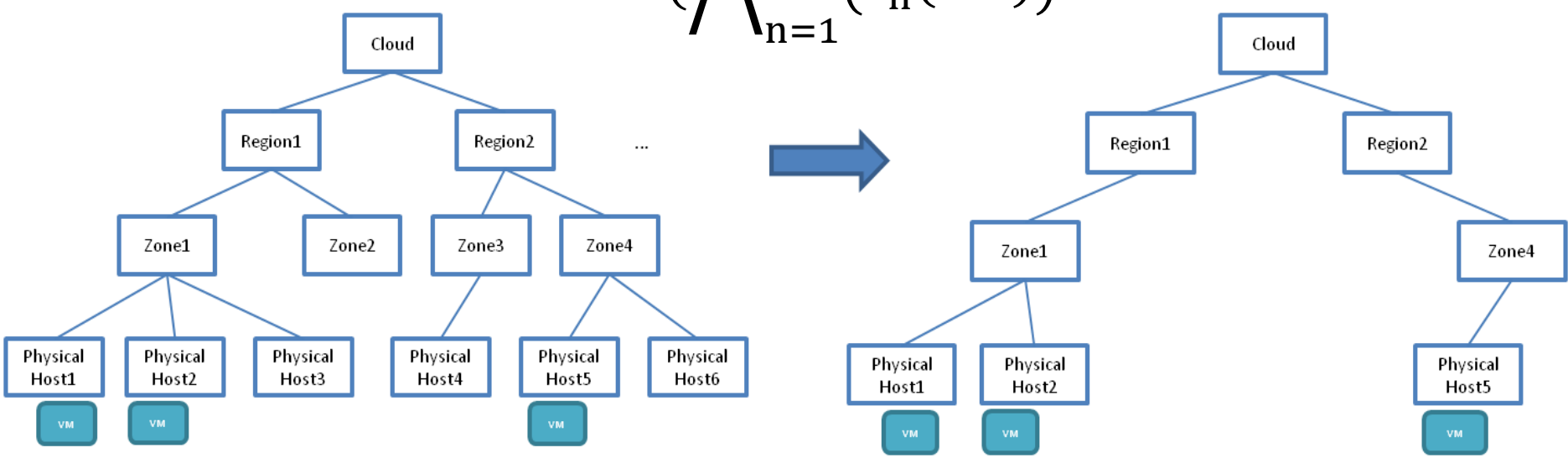
Availability Modeling

- The Failure Probability of multiple VMs

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

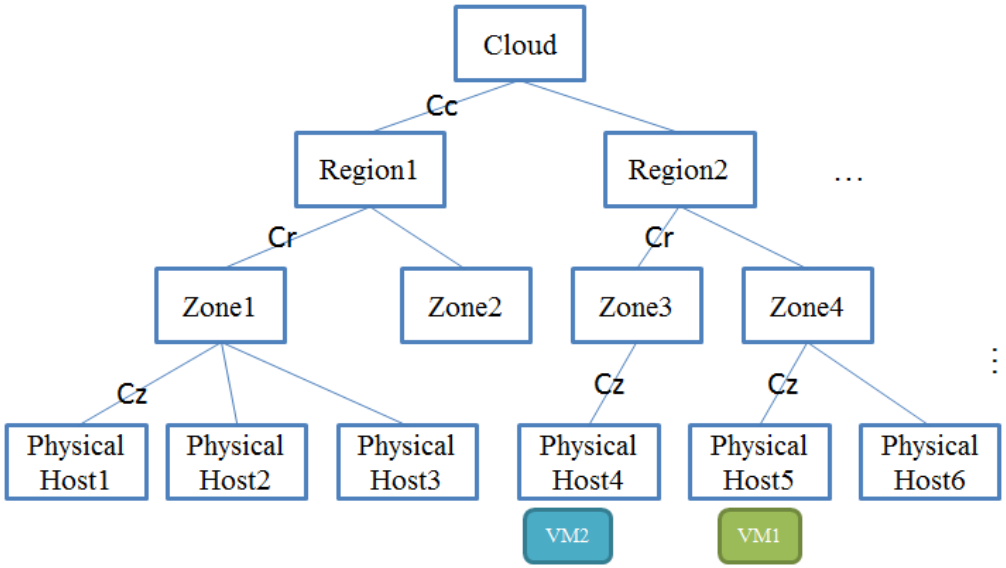
➢ The application availability(denoted as A) is based on the sub-tree generated by multiple VMs:

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



Availability Modeling

- Communication Cost Modeling
  - Let $cc(v_1,v_2)$ donates the communication cost between VM_1 and VM_2.
  - 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 $cc(v, S - \{v\}) = \sum_{x!=v} cc(v, x)$



Agenda

- Problem Statement
- Availability Modeling
- Availability-aware Approach
- Evaluation
- Future Work
- Summary

Availability-aware Approach

- Assumptions:
  - Demanded resource quantity need to be resized up or down is known from other related studies.
  - All the required VMs are standardized in unitized size.
- Input of the algorithm:
  - **S={V_1,V_2,…,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

Availability-aware Approach

REliable, INtelligent & Scalable Systems

```
Ac=calculateAvailability();
t =1;
For(k=1;k<=Quantity;k++){
 If(scale == up){
   //the current availability is met
    If(Ac>=Ar){
      VerticalResizeUp(S, 1);
    }
    Else{// Ac<Ar
      HorizontalResizeUp(S, 1, t);
      t++;
    }
 }
 Else{ //scale down
    VerticalResizeDown(S, 1);
 }
 Ac=calculateAvailability();
} //end for
while(Ac<Ar && relocatedTimes >0){
  //rebalance overall application
  Relocate(S);
  Ac=calculateAvailability();
  relocatedTimes --;
}
```

```
VerticalResizeUp

For every VM in S{
If(SpareQuant =VM.host().spareRes()>1){
    VM.ResizeUp(min(Quant,SpareQuant));
    Quant -= min(Quant,SpareQuant)
  }
   If(Quant == 0) return;
}
HorizontalResizeUp(S,Quant,1);
```

Availability-aware Approach

REliable, INtelligent & Scalable Systems

```
Ac=calculateAvailability();
t =1;
For(k=1;k<=Quantity;k++){
 If(scale == up){
   //the current availability is met
    If(Ac>=Ar){
      VerticalResizeUp(S, 1);
    }
    Else{// Ac<Ar
      HorizontalResizeUp(S, 1, t);
      t++;
    }
 }
 Else{ //scale down
    VerticalResizeDown(S, 1);
 }
 Ac=calculateAvailability();
} //end for
while(Ac<Ar && relocatedTimes >0){
  //rebalance overall application
  Relocate(S);
  Ac=calculateAvailability();
  relocatedTimes --;
}
```

```
HorizontalResizeUp
While(Quant>0){
 Find a host where(cc(VM,S)>=t&&
   SpareQuant =VM.host().spareRes()>1)
 {
  host.newVM(min(Quant,SpareQuant));
  Quant -= min(Quant,SpareQuant);
 }
}
```

Availability-aware Approach

REliable, INtelligent & Scalable Systems

```
Ac=calculateAvailability();
t =1;
For(k=1;k<=Quantity;k++){
 If(scale == up){
   //the current availability is met
    If(Ac>=Ar){
       VerticalResizeUp(S, 1);
    }
    Else{// Ac<Ar
       HorizontalResizeUp(S, 1, t);
       t++;
    }
 }
 Else{ //scale down
    VerticalResizeDown(S, 1);
 }
 Ac=calculateAvailability();
} //end for
while(Ac<Ar && relocatedTimes >0){
  //rebalance overall application
  Relocate(S);
  Ac=calculateAvailability();
  relocatedTimes --;
}
```

```
VerticalResizeDown
For every VM in S{
 if Vj.size() > 1 {
    //Atomic operation
   VM.ReizeDown(1);
     Quant--;
 }
 If(Quant == 0) return;
}//end for
HorizontalResizeDown(S,Quant,1)
```

Availability-aware Approach

REliable, INtelligent & Scalable Systems

```
Ac=calculateAvailability();
t =1;
For(k=1;k<=Quantity;k++){
 If(scale == up){
   //the current availability is met
    If(Ac>=Ar){
       VerticalResizeUp(S, 1);
    }
    Else{// Ac<Ar
       HorizontalResizeUp(S, 1, t);
       t++;
    }
 }
 Else{ //scale down
    VerticalResizeDown(S, 1);
 }
 Ac=calculateAvailability();
} //end for
while(Ac<Ar && relocatedTimes >0){
  //rebalance overall application
  Relocate(S);
  Ac=calculateAvailability();
  relocatedTimes --;
}
```

```
Relocate{

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

Agenda

- Problem Statement
- Availability Modeling
- Availability-aware Approach
- Evaluation
- Future Work
- Summary

Evaluation

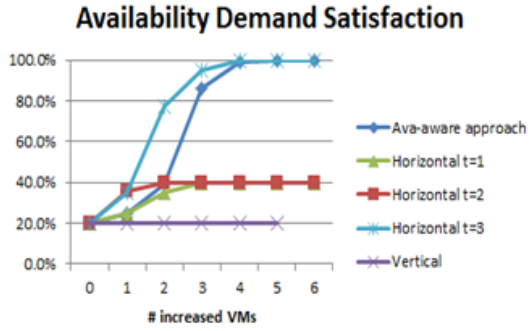
- Experiments Environment
  - a prototype of the proposed framework in Java
  - a hypothesized cloud environment consisting of over 200 homogenous physical machines.
    - 5 regions, each of which is composed of 5 availability zones

- Execution
  - 1) 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.

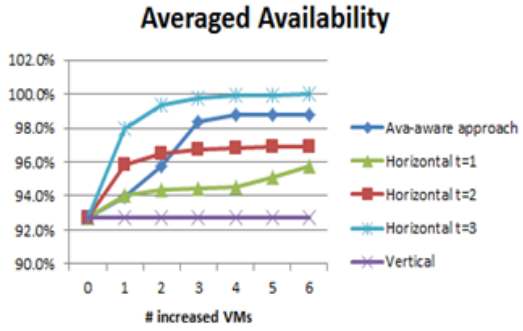
  - 2) simulated two typical scenarios
    - scaling up
      - adapting the application size from 3 VMs to 9 VMs
    - scaling down
      - Adapting the application size from 8VMs to 2 VMs
    - for each of scenarios we presented the averaged results over 300 independent executions.

Evaluation

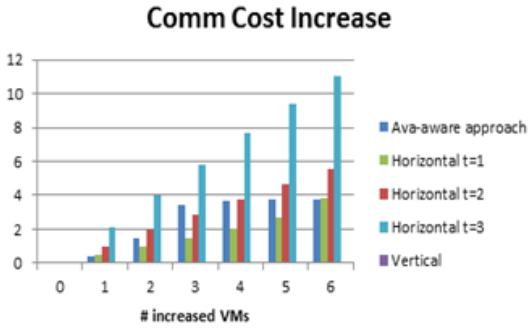
- Evaluation of Availability Model



(a)Demand satisfaction

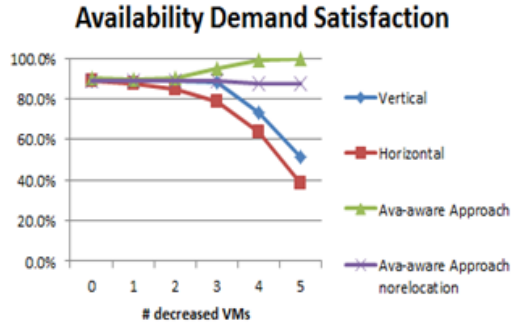


(b) Averaged availability

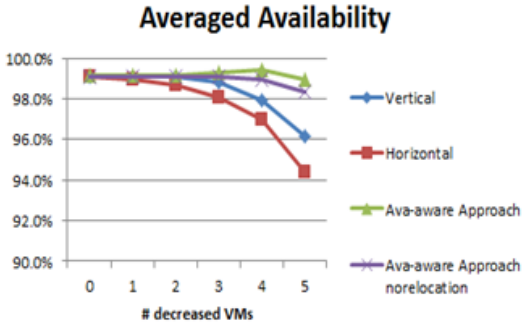


(c)Communication cost Increase ratio

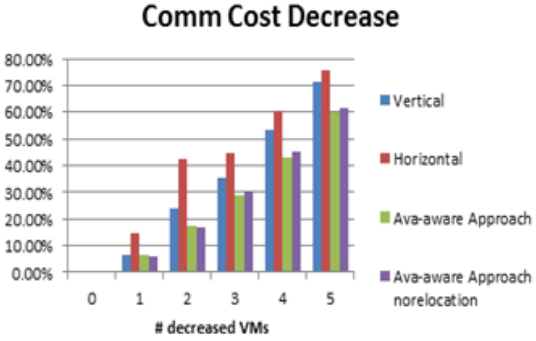
Figure3 availability enhancement and performance change when scaling up



(a)Demand satisfaction



(b) Averaged availability



(c)Communication cost Decrease ratio

Figure4 availability declination and performance change when scaling down

Agenda

- Problem Statement
- Availability Modeling
- Availability-aware Approach
- Evaluation
- Future Work
- Summary

Future Work

- Improvement on the availability model
  - Integrating with real data from industry rather than assumption
- Improvement on experiments
  - in an industrial cloud environment
  - e.g. in Openstack-based environment.
- Take more factors into consideration
  - e.g. in a hybrid cloud or a cloud federation

Agenda

- Problem Statement
- Availability Modeling
- Availability-aware Approach
- Evaluation
- Future Work
- Summary

Summary

- Analyze the effect factors of availability of applications in clouds
- Provide a hierarchical structure of cloud infrastructure
  - without loss of generality and scalability.
- Provide an availability computation model
  - measure the availability attribute of the application.
- Propose an availability -aware placement approach
  - Vertical and Horizontal Resizing
  - Availability-aware and cost-efficient

- Thank you~

- Wenting Wang, Haopeng Chen, Xi Chen
- Reins Group, http://reins.se.sjtu.edu.cn
- wwtvanessa@gmail.com
- Shanghai Jiao Tong University, China