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
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

```
Cloud
  ├ Region1
  │   ├ Zone1
  │   │   └ Physical Host1 (VM1, VM)
  │   │   └ Physical Host2
  │   └ Physical Host3
  └ Region2
      └ Zone2
          └ Physical Host4 (VM2)
          └ Physical Host5
          └ Physical Host6
```

Nature disasters

Blackout/Network failure

Hardware failure
Availability Modeling

- The Availability of One VM

\[ P_i(\text{VM}) = P_i \prod_{j \in \text{PP}(i)} P_j \]

- The Failure Probability of Two VMs

\[ \overline{(P_u(\text{VM})) \cap (P_v(\text{VM}))} = \prod_{n \in \mathcal{C}(u,v)} P_n \]

\[ + \prod_{n \in \mathcal{C}(u,v)} P_n \times \left( \prod_{x \in \mathcal{N}(u), x \notin \mathcal{C}(u,v)} P_x \right) \left( \prod_{y \in \mathcal{N}(v), y \notin \mathcal{C}(u,v)} P_y \right) \]
• The Failure Probability of multiple VMs

\[ P'_m = P_m + P_m \prod_{n \in \text{children}(m)} P'_n \]

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

\[ A = 1 - P\left(\bigwedge_{n=1}^{k} (P_n(VM))\right) \]
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
    $$ cc(v, S - \{v\}) = \sum_{x \neq v} cc(v, x) $$
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:**
  - \( S=\{V_1, V_2, \ldots, V_n\} \) 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
Availability-aware Approach

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:  } Else{ // scale down
14:    VerticalResizeDown(S, 1);
15:  }
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: }

VerticalResizeUp

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);
Availability-aware Approach

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:          } Else{
9:              // Ac<Ar
10:             HorizontalResizeUp(S, 1, t);
11:             t++;
12:          }
13:  } Else{
14:      // Ac<Ar
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:      }

HorizontalResizeUp
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:  }
Availability-aware Approach

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: }
19: while(Ac<Ar && relocatedTimes >0){
20:    // rebalance overall application
21:    Relocate(S);
22:    Ac=calculateAvailability();
23:    relocatedTimes --;
24:}

```
VerticalResizeDown
1: For every VM in S{
2:    if Vj.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)
```
Availability-aware Approach

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: }

Relocate{
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);
}

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

Figure 3: Availability enhancement and performance change when scaling up.

Figure 4: Availability decline 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
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