Diversity in automatic cloud computing resource selection

Vinicius Vielmo Cogo
Advised by Prof. Dr. Marcelo Pasin
Outline

• Introduction
• Diversity analysis
• The DiversityAgent
• Integration and evaluation
• Conclusions and future work
A replicated service ...

... is composed by N replicas ...
... requested by M clients ...

... and can tolerate $f$ faults.
Introduction
Towards on a diversity definition

If an attacker discovers an exploitable vulnerability in a replica ...

... then he can compromise that replica.
Introduction
Towards on a diversity definition

But if all replicas are equal ...

... then he can compromise all replicas ($f+1$ is enough).

Common vulnerabilities are a known limitation of replication.
Introduction
Towards on a diversity definition

Then the **diversity** comes. It is a fault and intrusion tolerance mechanism that ...

... consists in **making replicas diversified** ...
... to **improve the probability of** obtaining **vulnerability independence** ...

... and make the **attackers' life harder**.
1) Diversity analysis

- Analyse diversity occurrence in cloud computing scenario

2) The DiversityAgent

- Develop a tool to automatically obtain diversity of any kind during cloud computing resource selection
- Integrate and evaluate this tool
Diversity analysis
Diversity analysis
Overview

This diversity analysis try to answer the following questions:

1) Which are the existing **diversity classifications**?

2) Which are the **scenarios** where they are important?

3) Which is the main **goal** of each diversity group regarding vulnerability independence?

4) Which are the **properties** that can be or already are obtained with **cloud computing**?

**Properties** are information used to specify requirements on resource selection.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>image.name</td>
<td>img_01</td>
</tr>
<tr>
<td>image.os_type</td>
<td>GNU/Linux</td>
</tr>
<tr>
<td>image.os_name</td>
<td>Ubuntu</td>
</tr>
<tr>
<td>image.software</td>
<td>Tomcat Web server</td>
</tr>
<tr>
<td>image.prog_lang</td>
<td>Java</td>
</tr>
</tbody>
</table>
Diversity analysis

Taxonomy

Diversity groups [Deswarte98, Obelheiro06]:

- **Application**
- **Administrative**
- **Location**
- **Support software**
- **Hardware**
- **Security**
Diversity analysis
Application diversity

More than one implementation of the same software specification.

Scenarios:
- N-version programming [Avizienis77]
- Transformation techniques:
  - Rearranging memory
  - Randomizing system calls
  - Randomizing instruction set
  - Randomizing protocol parameters

Goal:
Increase the probability of creating software whose vulnerabilities (if exist) are completely independent.

Properties:
4 identified
Diversity analysis
Administrative diversity

More than one administrative entity to run a service or store data.

Scenarios:
- Vendor lock-in [Bessani11]
- Reliability of cloud providers
- Electrical disturbances and accidents internally to some cloud provider
- Reduce network latencies between service instances and end users

Goal:
Prevent an entire service or data set to be affected by any local administrative event.

Properties:
14 identified
Geographically distributed resources to run a service or store data.

Scenarios:
- Natural disasters
- Political events:
  - Unstable governments
  - Diplomatic positioning
- Legal events:
  - Personal data prosecutions
  - Copyrights prosecutions
- Reduce the latency between globally distributed service instances and clients

Goal:
Prevent an entire service or data set to be affected by any geographically local event.

Properties:
10 identified
Diversity analysis
Support software diversity

Diverse versions and implementations of any software that can provide a basis to run a service.

Scenarios:
- Operating systems vulnerabilities [Henriques11]
- Database management systems vulnerabilities [Gashi07]
- Vulnerabilities on any other support software (depend on service scope):
  - Middleware
  - Virtual machines for bytecode
  - Compilers
  - Libraries

Goal:
Prevent an entire service or data set to be affected by any common software vulnerability shared between replicas.

Properties:
9 identified
Diversity analysis
Hardware diversity

Physical hosts with different hardware components to run a service or store data.

Scenarios:
- Vulnerabilities in processors [Collins98]
- Vulnerabilities in any other hardware component:
  - Network cards
  - Video cards
  - Hard disks
  - Trusted Platform Modules

Goal:
Prevent an entire service or data set to be affected by one common hardware vulnerability.

Properties:
9 identified
Diversity analysis
Security diversity

More than one security method or policy within the group of cloud providers or service instances.

Scenarios:
- Attacks to authentication methods
- Attacks to access control protocol
- Attacks to data integrity methods
- Attacks to confidentiality methods
- Physical attacks to some specific cloud provider

Goal:
Prevent an entire service or date set to be affected by one common security vulnerability or security flaw.

Properties:
6 identified
Diversity analysis
Identified diversity properties

**Application:**
- VM Image name
- Application name
- Application version
- Transformation method

**Administrative:**
- Cloud provider name
- Available APIs
- Physical host name
- Rack name
- Cluster name
- Number of failures
- Number of power outages
- Number of VM failures
- Cloud uptime
- Host uptime
- Autonomous system (AS)
- Network latency to X
- Round trip time to X
- Number of hops to X

**Location:**
- GPS coordinates
- Location based on IP
- City
- State
- Region
- Country
- Continent
- Economic group
- Political union
- Geographic distance to X

**Security:**
- Cloud provider name
- Authentication methods
- Access control methods
- Data integrity methods
- Data confidentiality methods
- Security policies

**Support software:**
- Image name
- OS type
- OS name
- OS architecture
- Kernel
- Virtualization type
- Application and service
- Supported programming languages
- Compatible hypervisors

**Hardware:**
- CPU model
- CPU architecture
- CPU Speed
- Network card model
- Network card speed
- Video card model
- Hard disk model
- Hard disk speed
- Hypervisor
Diversity analysis
Identified diversity properties

52 properties were identified in this work

8 properties are completely supported by OpenNebula

13 properties are completely supported by Amazon Web Services

18 properties are partially supported by Amazon Web Services

0 tool was found to automatically obtain diversity considering more than one diversity group
Diversity analysis
Existent solutions

If a client needs to use more than one cloud, then a broker service can be used.

Broker service allows clients to use more than one cloud.

But brokers normally do not provide automatic cloud selection.

Then clients have to inform which cloud should be selected to allocate each server.
Diversity analysis
Existent solutions

There is an algorithm for automatic OS selection [Henriques11].

But it does not provide selection considering multiple diversity groups.
The DiversityAgent
The DiversityAgent
Overview

Generic solution to select resources considering more than one diversity group

**Cloud computing** resource selection

**Hierarchy** between diversity algorithms

Diversity of cloud providers, physical hosts and operating systems were implemented

**Extensible** for new diversity algorithms

Implemented in Java
The DiversityAgent
Registering available resources

**Instantiate the DiversityAgent**

```java
DiversityAgent myAgent = new DiversityAgent();
```

**Add cloud providers**

```java
myAgent.createCloud("cloud_01", "OPEN_NEBULA", "Private", "194.117.20.195", "myUser", "myPassword");
```

**Add VM images**

```java
myAgent.createImage("image_name");
```

**Register diversity algorithms**

```java
myAgent.createDiversity("OPERATING_SYSTEM", "CLOUD", "PHYSICAL_HOST");
```
Requesting a new VM
The DiversityAgent
Creating a virtual machine

1. Requesting a new VM
2. New property set
The DiversityAgent
Creating a virtual machine

1. Requesting a new VM
2. New property set
3. Inserting an ID

Diagram:
- **ResourceManager**
  - **DiversityAgent**
  - **myProperties**
    - vm.id.agent: 1 3
The DiversityAgent
Creating a virtual machine

1. Requesting a new VM
2. New property set
3. Inserting an ID
4. Requesting diversity contributions to the property set
The DiversityAgent
Creating a virtual machine

1. Requesting a new VM
2. New property set
3. Inserting an ID
4. Requesting diversity contributions to the property set
5. *CloudProvider* contribution (*cloud.name*)
The DiversityAgent
Creating a virtual machine

1. Requesting a new VM
2. New property set
3. Inserting an ID
4. Requesting diversity contributions to the property set
5. *CloudProvider* contribution (*cloud.name*)
6. *Hostname* contribution (*host.name.differ*)
The DiversityAgent
Creating a virtual machine

1. Requesting a new VM
2. New property set
3. Inserting an ID
4. Requesting diversity contributions to the property set
5. CloudProvider contribution (cloud.name)
6. Hostname contribution (host.name.differ)
7. OperatingSystem contribution (disk.image.name)
The DiversityAgent
Creating a virtual machine

1. Requesting a new VM
2. New property set
3. Inserting an ID
4. Requesting diversity contributions to the property set
5. CloudProvider contribution (cloud.name)
6. Hostname contribution (host.name.differ)
7. OperatingSystem contribution (disk.image.name)
8. Sending the property set to CloudDriver
The DiversityAgent
Creating a virtual machine

1. Requesting a new VM
2. New property set
3. Inserting an ID
4. Requesting diversity contributions to the property set
5. CloudProvider contribution (cloud.name)
6. Hostname contribution (host.name.differ)
7. OperatingSystem contribution (disk.image.name)
8. Sending the property set to CloudDriver
9. Parsing properties and requesting the VM creation
The DiversityAgent
Creating a virtual machine

1. Requesting a new VM
2. New property set
3. Inserting an ID
4. Requesting diversity contributions to the property set
   - CloudProvider contribution (cloud.name)
   - Hostname contribution (host.name.differ)
   - OperatingSystem contribution (disk.image.name)
5. Sending the property set to CloudDriver
6. Parsing properties and requesting the VM creation
7. Adding the new VM in the VM list
The DiversityAgent
Creating a virtual machine

1. Requesting a new VM
2. New property set
3. Inserting an ID
4. Requesting diversity contributions to the property set
5. CloudProvider contribution (cloud.name)
6. Hostname contribution (host.name.differ)
7. OperatingSystem contribution (disk.image.name)
8. Sending the property set to CloudDriver
9. Parsing properties and requesting the VM creation
10. Adding the new VM in the VM list
11. Returning the ID of the new VM
Integration and evaluation
Integration and evaluation
Integration with CloudFIT use cases

Proactive and reactive recovery

Dynamic adaption

DiversityAgent is responsible for obtaining diversity on each deployment

http://cloudfit.di.fc.ul.pt
Correct = provide the largest number of combinations without repetitions

Experiment with 3 cloud providers, 5 physical hosts and 3 VM images (15 different combinations):

<table>
<thead>
<tr>
<th>ID on agent</th>
<th>Cloud provider</th>
<th>ID on cloud</th>
<th>Physical host</th>
<th>VM IP address</th>
<th>OS name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>cloud_01</td>
<td>270</td>
<td>s4</td>
<td>192.168.2.33</td>
<td>Ubuntu Dapper</td>
</tr>
<tr>
<td>1</td>
<td>cloud_02</td>
<td>0</td>
<td>s6</td>
<td>192.168.2.39</td>
<td>Ubuntu Intrepid</td>
</tr>
<tr>
<td>2</td>
<td>cloud_03</td>
<td>1935</td>
<td>s7</td>
<td>192.168.2.46</td>
<td>Ubuntu Oneiric</td>
</tr>
<tr>
<td>3</td>
<td>cloud_02</td>
<td>1</td>
<td>s5</td>
<td>192.168.2.40</td>
<td>Ubuntu Dapper</td>
</tr>
<tr>
<td>4</td>
<td>cloud_03</td>
<td>1936</td>
<td>s7</td>
<td>192.168.2.47</td>
<td>Ubuntu Dapper</td>
</tr>
<tr>
<td>5</td>
<td>cloud_01</td>
<td>271</td>
<td>s3</td>
<td>192.168.2.34</td>
<td>Ubuntu Intrepid</td>
</tr>
<tr>
<td>6</td>
<td>cloud_03</td>
<td>1937</td>
<td>s7</td>
<td>192.168.2.48</td>
<td>Ubuntu Intrepid</td>
</tr>
<tr>
<td>7</td>
<td>cloud_01</td>
<td>272</td>
<td>s4</td>
<td>192.168.2.35</td>
<td>Ubuntu Oneiric</td>
</tr>
<tr>
<td>8</td>
<td>cloud_02</td>
<td>2</td>
<td>s6</td>
<td>192.168.2.41</td>
<td>Ubuntu Oneiric</td>
</tr>
<tr>
<td>9</td>
<td>cloud_02</td>
<td>3</td>
<td>s6</td>
<td>192.168.2.42</td>
<td>Ubuntu Dapper</td>
</tr>
<tr>
<td>10</td>
<td>cloud_01</td>
<td>273</td>
<td>s3</td>
<td>192.168.2.36</td>
<td>Ubuntu Dapper</td>
</tr>
<tr>
<td>11</td>
<td>cloud_01</td>
<td>274</td>
<td>s4</td>
<td>192.168.2.37</td>
<td>Ubuntu Intrepid</td>
</tr>
<tr>
<td>12</td>
<td>cloud_02</td>
<td>4</td>
<td>s5</td>
<td>192.168.2.43</td>
<td>Ubuntu Intrepid</td>
</tr>
<tr>
<td>13</td>
<td>cloud_02</td>
<td>5</td>
<td>s5</td>
<td>192.168.2.44</td>
<td>Ubuntu Oneiric</td>
</tr>
<tr>
<td>14</td>
<td>cloud_01</td>
<td>275</td>
<td>s3</td>
<td>192.168.2.38</td>
<td>Ubuntu Oneiric</td>
</tr>
<tr>
<td>15</td>
<td>cloud_01</td>
<td>276</td>
<td>s4</td>
<td>192.168.2.32</td>
<td>Ubuntu Intrepid</td>
</tr>
</tbody>
</table>
Integration and evaluation
Performance evaluation

Overhead caused on a VM deployment

<table>
<thead>
<tr>
<th>Time</th>
<th>cloud_01</th>
<th>cloud_02</th>
<th>cloud_03</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment time with DiversityAgent (in s)</td>
<td>127</td>
<td>111</td>
<td>150</td>
</tr>
<tr>
<td>Deployment time without DiversityAgent (in s)</td>
<td>124</td>
<td>115</td>
<td>150</td>
</tr>
<tr>
<td>Real overhead time (in s)</td>
<td>0.000974</td>
<td>0.000048</td>
<td>0.000053</td>
</tr>
<tr>
<td>Overhead (%)</td>
<td>0.000007</td>
<td>0.000004</td>
<td>0.000003</td>
</tr>
</tbody>
</table>

Average case overhead = 0.00025% per VM
Worst case overhead = 0.0007% per VM on cloud_03

Complexity time analysis:

<table>
<thead>
<tr>
<th>Diversity algorithm</th>
<th>Complexity time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud provider</td>
<td>Linear</td>
</tr>
<tr>
<td>Physical host</td>
<td>Polynomial</td>
</tr>
<tr>
<td>Operating system</td>
<td>Polynomial</td>
</tr>
<tr>
<td>DiversityAgent</td>
<td>Polynomial</td>
</tr>
</tbody>
</table>
Conclusions

Final remarks

Diversity analysis in cloud computing:

- 52 diversity properties were identified.
- 2 interfaces (OpenNebula and Amazon) were analysed.
- There are opportunities for cloud providers to improve diversity management area.

DiversityAgent library:

- First generic solution to automatically select cloud computing resources considering more than one diversity group.
- 3 diversity algorithms were implemented.
- 1 cloud driver was implemented (OpenNebula).
- Code published under LGPL v3.

- Integration: DiversityAgent was integrated with two use cases from CloudFIT.
- Correctness: The diversity algorithms implemented are correct.
- Performance: DiversityAgent does not caused a considerable overhead on CloudFIT use cases, because they consider small amount of resources.
- Complexity: DiversityAgent has polynomial complexity time, which can be improved.
Conclusions

http://code.google.com/p/diversity-agent/

Summary

The DiversityAgent is an efficient and extensible library developed in Java and focused on obtaining diversity automatically from cloud computing resource selection. It allows the creation of algorithms to obtain many kinds of diversity from any supported cloud provider or tool.

Getting started on DiversityAgent

On this page, you can find a starting point in understanding diversity, using, customising and improving DiversityAgent.

Acknowledgements

This component appears in the context of fault and intrusion tolerance (FIT) applied on Cloud Computing, and it was developed within the CloudFIT (Fault-and-Intrusion Tolerance for Clouds) project, on Navigators research group of the Large-Scale Informatics Systems Laboratory (LaSIGE/FCTUL).
Conclusions

Future work

Promote all identified diversity properties and opportunities

Develop more diversity algorithms and cloud drivers


Thank you!