Towards FIT-aware scheduling policies for Cloud Computing

Vinicius Vielmo Cogo
Marcelo Pasin
The Motivation:

Cloud Computing managers deal with large amounts of globally distributed resources.
The Problem:

Scheduling policies are mainly focused on performance and economy.
Main Goal:

Choose the best **resource** for a Fault and Intrusion Tolerant system.
Background:

- There are basically 3 types of resources in Cloud Computing (IaaS)

![Diagram showing storage, network, and processing resources connected in a cloud-like structure]
Background:

- This work is focused on Processing resources
Outline:

1 - CloudFIT Project
2 - Intrusion-Tolerance Mechanisms
3 - Resource Allocation Algorithm
4 - Requirements for Scheduling
5 - Future Work
6 - Open Questions
CloudFIT: Fault and Intrusion Tolerance for Clouds

- **Goal:**
  Create an infrastructure for FIT services in a cloud environment

- **Components:**
  - Cloud Resource Manager (OpenNebula)
  - Hosts with Virtual Machine Monitors (e.g.: Xen)
  - Service that uses the bft-SMaRt library for State Machine Replication
  - And others ...
2 - Intrusion-Tolerance Mechanisms:

State Machine Replication ... 

Proactive Recovery ... 

Diversity ...
2 - Intrusion-Tolerance Mechanisms:

State Machine Replication ...  
... to increase integrity and availability

Proactive Recovery ...  
... to tolerate any number of intrusions over the system's lifetime

Diversity ...  
... to increase the difficulty of attacks and independence between faults
2 - Intrusion-Tolerance Mechanisms:

Clients

OpenNebula

Physical Host 01

Physical Host 02

Physical Host 03

Physical Host 04

Physical Host 05

Physical Host 06

Physical Host N

Other Clouds
2 - Intrusion-Tolerance Mechanisms:

Clients

OpenNebula

Physical Host 01

Physical Host 02

Physical Host 03

Physical Host 04

Physical Host 05

Physical Host 06

Physical Host N

Other Clouds

1

2

3

4
2 - Intrusion-Tolerance Mechanisms:

Clients

OpenNebula

Physical Host 01

Physical Host 02

Physical Host 03

Physical Host 04

Physical Host 05

Physical Host 06

Physical Host N

Other Clouds
2 - Intrusion-Tolerance Mechanisms:
2 - Intrusion-Tolerance Mechanisms:

Clients

OpenNebula

Physical Host 01
Physical Host 02
Physical Host 03
Physical Host 04
Physical Host 05
Physical Host 06
Physical Host N

Other Clouds
2 - Intrusion-Tolerance Mechanisms:

Clients

OpenNebula

Physical Host 01

Physical Host 02

Physical Host 03

Physical Host 04

Physical Host 05

Physical Host 06

Physical Host N

Other Clouds
3 - Resource Allocation Algorithm:

- Choose one host to allocate a Virtual Machine

- Algorithms are normally based on two steps, or a combination of them:
  - REQUIREMENTS (boolean expression)
  - RANK (numeric expression)

- OpenNebula uses a matchmaking algorithm:
  1. Filter out the hosts based on REQUIREMENTS
  2. Rank the remaining hosts based on RANK
  3. Choose the highest ranked host to allocate the VM
### Resource Allocation Algorithm:

**OpenNebula**

**Requirement:** Hypervisor = Xen

**Rank:** Total of Memory

<table>
<thead>
<tr>
<th>HOSTNAME</th>
<th>HYPERVERSOR</th>
<th>RUNNING_VMS</th>
<th>CPUSPEED</th>
<th>TOTALMEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>host01</td>
<td>Xen</td>
<td>5</td>
<td>3.2</td>
<td>32</td>
</tr>
<tr>
<td>host02</td>
<td>Xen</td>
<td>2</td>
<td>3.2</td>
<td>16</td>
</tr>
<tr>
<td>host03</td>
<td>KVM</td>
<td>0</td>
<td>2.3</td>
<td>8</td>
</tr>
<tr>
<td>host04</td>
<td>VMWare</td>
<td>0</td>
<td>1.8</td>
<td>8</td>
</tr>
<tr>
<td>host05</td>
<td>VMWare</td>
<td>3</td>
<td>2.5</td>
<td>8</td>
</tr>
<tr>
<td>host06</td>
<td>KVM</td>
<td>6</td>
<td>2.7</td>
<td>16</td>
</tr>
<tr>
<td>host07</td>
<td>KVM</td>
<td>1</td>
<td>3.0</td>
<td>32</td>
</tr>
<tr>
<td>host08</td>
<td>VMWare</td>
<td>2</td>
<td>1.8</td>
<td>32</td>
</tr>
<tr>
<td>host09</td>
<td>VMWare</td>
<td>3</td>
<td>2.7</td>
<td>16</td>
</tr>
<tr>
<td>host10</td>
<td>Xen</td>
<td>4</td>
<td>2.3</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>hostN</td>
<td>Xen</td>
<td>6</td>
<td>3.2</td>
<td>8</td>
</tr>
</tbody>
</table>
4 - Requirements for Scheduling:

4.1 - Diversity Requirements

4.2 - Proactive Recovery Requirements

4.3 - State Machine Replication Requirements
4.1 - The Various Diversities:

Diversity of ...

... Application
    N-Version programming

... Supporting Software
    Manifold Operating Systems, Server or Daemon implementation

... Hardware
    Distinct Architecture, CPU Model or Speed

... Administrative Domain
    Different Hostname, Cluster or Cloud Provider

... Location
    Varied GPS coordinates, Geopolitical structures
### 4.1 - Diversity of Administrative Domain:

- **Replicas should not be placed in the same physical host, rack or cluster**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOSTNAME != &quot;some-host&quot;</td>
<td>✔️</td>
</tr>
<tr>
<td>RACK != &quot;some-rack-id&quot;</td>
<td>✗</td>
</tr>
<tr>
<td>CLUSTER != &quot;some-cluster&quot;</td>
<td>✔️</td>
</tr>
</tbody>
</table>

- **Replicas should not be placed in the same cloud provider**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOUD_PROVIDER != &quot;some-cloud-provider&quot;</td>
<td>✗</td>
</tr>
</tbody>
</table>

- **Other measures:**

<table>
<thead>
<tr>
<th>Measure</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;- ROUND_TRIP_TIME to X.X.X.X&quot;</td>
<td>✗</td>
</tr>
<tr>
<td>&quot;- NETWORK_LATENCY to X.X.X.X&quot;</td>
<td>✗</td>
</tr>
<tr>
<td>&quot;- NUM_HOPS to X.X.X.X&quot;</td>
<td>✗</td>
</tr>
</tbody>
</table>

* Cloud Providers: Private, Amazon, Rackspace, Salesforce, ElasticHosts, GoGrid, SliceHost, etc.
### 4.1 - Diversity of Physical Hardware:

- **Replicas should be placed in distinct architectures**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ARCH != &quot;some-arch&quot;</code></td>
<td>✓</td>
</tr>
</tbody>
</table>

- **Replicas should use different processors**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MODELNAME != &quot;some-processor-model&quot;</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>&quot;CPUSPEED&quot;</code></td>
<td>✓</td>
</tr>
</tbody>
</table>

- **Replicas should use different hardware components**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TPM != &quot;some-tpm-implementation&quot;</code></td>
<td>✗</td>
</tr>
<tr>
<td><code>NETWORK_CARD != &quot;some-network-card&quot;</code></td>
<td>✗</td>
</tr>
<tr>
<td><code>VIDEO_CARD != &quot;some-video-card&quot;</code></td>
<td>✗</td>
</tr>
</tbody>
</table>

* Trusted Platform Module: Atmel, Broadcom, Infineon, Intel, Sinosun, STMicroelectronics, Winbond, Toshiba, etc.
### 4.1 - Diversity of Supporting Software:

- **Replicas should be placed in different hypervisors**
  
<table>
<thead>
<tr>
<th>HYPERVERSOR != “some-hypervisor”</th>
<th>REQUIREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>

- **Replicas must use different Wormholes**
  
<table>
<thead>
<tr>
<th>WORMHOLE != “some-wormhole”</th>
<th>REQUIREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td></td>
</tr>
</tbody>
</table>

- **Replicas should use different Operating System**
  
<table>
<thead>
<tr>
<th>Service Notion or Service Description Language</th>
<th>PLUG-IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Notion or Service Description Language</td>
<td>!</td>
</tr>
</tbody>
</table>

- **Replicas should use different Server implementation**
  
<table>
<thead>
<tr>
<th>Service Notion or Service Description Language</th>
<th>PLUG-IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Notion or Service Description Language</td>
<td>!</td>
</tr>
</tbody>
</table>
4.1 - Diversity of Application:

- Replicas should use different application versions/implementations

The service notion consists in deploy any service's replica instead of a specific image, on each replica replacement.

<table>
<thead>
<tr>
<th>Service Notion or Service Description Language</th>
<th>PLUG-IN</th>
</tr>
</thead>
</table>

Without Service Notion

With Service Notion
### 4.1 - Diversity of Location:

- **Recoveries should change replica location**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Type</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DISTANCE to +XX.XXXXXX -YY.YYYYYY&quot;</td>
<td>RANK</td>
<td>✗</td>
</tr>
<tr>
<td>&quot;- DISTANCE to +XX.XXXXXX -YY.YYYYYY&quot;</td>
<td>RANK</td>
<td>✗</td>
</tr>
<tr>
<td>COUNTRY != “some-country”</td>
<td>REQUIREMENT</td>
<td>✗</td>
</tr>
</tbody>
</table>

- **Service replicas should not be illegal in the location or should be allocated under a determined legislation (reputation or legal issues)**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Type</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITY != “some-city”</td>
<td>REQUIREMENT</td>
<td>✗</td>
</tr>
<tr>
<td>STATE != “some-state”</td>
<td>REQUIREMENT</td>
<td>✗</td>
</tr>
<tr>
<td>COUNTRY != “some-country”</td>
<td>REQUIREMENT</td>
<td>✗</td>
</tr>
<tr>
<td>CONTINENT != “some-continent”</td>
<td>REQUIREMENT</td>
<td>✗</td>
</tr>
<tr>
<td>GROUP != “some-group”</td>
<td>REQUIREMENT</td>
<td>✗</td>
</tr>
</tbody>
</table>
4.2 - Proactive Recovery Requirements:

- Replicas must be added and removed in a controlled way

- Haizea is a batch scheduler that provides 3 types of lease

<table>
<thead>
<tr>
<th>START_TIME</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;+00:00:30&quot;</td>
<td>Advanced Reservation</td>
</tr>
<tr>
<td>&quot;2008-11-04 11:00:00&quot;</td>
<td>Advanced Reservation</td>
</tr>
<tr>
<td>&quot;best_effort&quot;</td>
<td>Best-effort Provisioning</td>
</tr>
<tr>
<td>&quot;now&quot;</td>
<td>Immediate Provisioning</td>
</tr>
</tbody>
</table>

- The important metrics are:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>START_TIME</td>
<td>&quot;2008-11-04 11:00:00&quot;</td>
<td>✔️</td>
</tr>
<tr>
<td>END_TIME</td>
<td>&quot;2008-11-04 12:00:00&quot;</td>
<td>✔️</td>
</tr>
<tr>
<td>DURATION</td>
<td>&quot;00:00:30&quot;</td>
<td>✔️</td>
</tr>
<tr>
<td>PREEMPTIBLE</td>
<td>&quot;yes&quot;</td>
<td>✔️</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>&quot;10&quot;</td>
<td>✗</td>
</tr>
</tbody>
</table>

- The scheduling can not block the resource allocation properties
4.3 - State Machine Replication Requirements:

- Replicas could reuse the VM image from a local repository

Image Cache on Physical Hosts

ONE Server

central repository

ONE Node 01

ONE Node 02

ONE Node N

cache

cache

cache

VM 01

VM 03

VM 02

VM 04

VM Image A

VM Image B

VM Image C
5 - Future Work:

- Delineating the influence of each metric in scheduling policies
- Specifying which metrics will be implemented
- Implementing and testing
- Organizing the metrics in two-level of scheduling: Cloud-of-clouds and Single Cloud
6 - Open Questions:

- How independent are the faults in diversities cited here?

  For diversity in Software of Support (*Operating Systems*) there is a new paper:

  Miguel Garcia, Alysson Bessani, Ilir Gashi, Nuno Neves, and Rafael Obelheiro.  
  OS Diversity for Intrusion Tolerance: Myth or Reality?  
  In Proc. of the DSN’11: International Conference on Dependable Systems and Networks, Hong Kong, China, July 2011.

- How guarantee that a Virtual Machine will be ready in an Advanced Reservation Lease for proactive recovery?

  A cache for images, placed in hosts, helps but doesn't solve everything.