Deploy a Highly-dynamic Virtual Cluster Based on OpenNebula and Xen in Grid'5000

Rodrigue Chakode
(INRIA/Mescal - LIG Laboratory - Grenoble University)
Ph.D Student
Rodrigue.Chakode@{gmail.com,imag.fr}

Grid'5000 Challenge
Reims, April 2011
Key points

- Automatic deployment and customization of an OpenNebula cloud on Grid'5000
- Introducing SVMSched, a tool designed to enable the set-up of custom VMs on-the-fly onto such a cloud
- Easily reproducible experiment
  - Script-based deployment and configuration of the virtual cluster's nodes
  - Custom environments + Kadeploy3
  - ...
OpenNebula

- Open-source toolkit to build private, public and hybrid clouds
  - Orchestrates storage, network, virtualization, monitoring, and security technologies
  - Unix-like command line interfaces and cloud interfaces (REST, OCCI, Amazon EC2, etc.)
  - Modular with an XML-RPC API to access its core functionality

Rodrigue Chakode (LIG/INRIA/Mescal)
SVMSched (Smart Virtual Machine Scheduler)

- Originally designed to enable and ease the set-up of on-demand SaaS clouds
  - Drop-in replacement for the OpenNebula's default scheduler
  - **Dynamic VM provisioning** → according to requests
  - **Advanced scheduling policies** → e.g. resource sharing, best-effort + preemption, etc.
  - **Proper interfaces** to deal with requests

Rodrigue Chakode (LIG/INRIA/Mescal)
Deployment Architecture

Physical node 1
- VMM
- HCA
- Eth

Physical node 2
- VMM
- HCA
- Eth

Physical node 3
- VMM
- HCA
- Eth

... Physical node n
- VMM
- HCA
- Eth

Network-Attached Storage (Application and Data Repository)

Fabric Infiniband

Ethernet switch

Cloud Backend (OpenNebula Server)

NFS-based VM Image Repository

Cloud

Cloud Frontend (SVMSched Scheduler)

User Terminal (SVMSched Client)
## Grid'5000 Deployment

### Deployment data

```
nancy@g5k:/home/rchakode/g5kss11challenge/
```

### Single site deployment

- Edit `oar.conf` to reserve X nodes for Y hours for example
- Edit `nets.conf` to be compliant with the use of network addressing in Grid'5000

### Custom kadeploy3 environments

```
# kaenv3 -l
```

### Running the deployment

```
# cd /tmp

# $SVMSCHED_DIR/g5kss11challenge.sh 2> /tmp/svmsched_debug.msg
```

### Get debugging details about the deployment processing

```
# tail -f /tmp/svmsched_debug.msg
```
Cloud Configuration

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Config SYSTEM "svmsched.dtd">
<Config>
  <!-- CLOUD CONFIGURATION SECTION -->
  <Cloud>
    <RpcUrl>http://opennebula@server:2633/RPC2</RpcUrl>  <!-- Url to bind to the OpenNebula's XML-RPC listener -->
    <OneAuth>oneadmin:7bc8559a8fe589e686562b85c337f170956fcb06</OneAuth>
    <RootDir>/opt/cloud</RootDir>  <!-- Default data-repository to mount automatically to each VM -->
    <AppPrefix>/opt/cloud/install</AppPrefix>
    <TempDir>/tmp</TempDir>
  </Cloud>
  <!-- VM CONFIGURATION SECTION -->
  <SvmDescription>
    <ServerAddress>svmsched@server</ServerAddress>
    <DefaultXenVmTemplate> /opt/cloud/svmsched/etc/xenvm.tpl </DefaultXenVmTemplate>
    <ContextFiles>
      <!-- Generic data use to customize each VM machine -->
      <ContextFile> /opt/cloud/svmsched/etc/init.sh </ContextFile>  <!-- Script to initialize VM at startup -->
      <ContextFile> /opt/cloud/svmsched/etc/svmschedclient </ContextFile>  <!-- Generic data use to customize each VM machine -->
    </ContextFiles>
  </SvmDescription>
  <!-- SERVICE CONFIGURATION SECTION -->
  <AppServices>
    <AppService id="defaultapp">
      <Software executable="/opt/cloud/install/jivarod" userid="jivarod" type="sequential"> Jivarod </Software>
      <DataServer>opennebula@server</DataServer>  <!-- Required a NFS-based OpenNebula deployment -->
      <DataRepository>/opt/cloud</DataRepository>  <!-- If the service requires a specific data-repository -->
      <Description>The default App is Jivarod</Description>
      <ResourceAllocation weight="1">  <!-- Set the level of accessing ressources by a service-->
        <DefaultVmMemory>256</DefaultVmMemory>
        <DefaultVmMemoryCpu>1</DefaultVmMemoryCpu>
        <ParallelismLevel>1</ParallelismLevel>
      </ResourceAllocation>
    </AppService>
  </AppServices>
</Config>
```
SVMSched's Client Interfaces

- Unix-like command line client
- System Workload Format-compatible load injector
After the deployment

- Log files generated from kadeploy are located in `./tmp`

- From a new terminal, log on to the svmsched and check the core log file
  
  ```
  # ssh svmsched@cloud.frontend 'tail -f var/svmsched-core.log'
  ```

- From a new terminal, log on to the OpenNebula node and check the pool of physical nodes and the virtual network
  
  ```
  # ssh oneadmin@cloud.backend 'onehost list ; onevnet list ; onevnet show 0'
  ```

- From a new terminal, log on to the svmsched node and check the monitor log file
  
  ```
  # ssh svmsched@cloud.frontend 'tail -f var/svmsched-monitor.log'
  ```
After the deployment... Goto Test

• From the Grid'5000 site frontend create a VM that will run during 10 seconds
  
  # ./bin/svmschedclient -H svmsched@server -r defaultapp – a 10

• Create a campaign of jobs from a SWF file
  
  # ./bin/svmsched-swf-injector -H <swf file> \n  
  [svmsched_server=localhost] [max_job_duration=600]
Appendix

- Service configuration
  - **Simple program enforcing a sleep according to the parameter**
    ```bash
    #!/bin/bash
    sleep $1
    ```

- Materials and scripts
  - [nancy@g5k:/home/rchakode/g5kss11challenge/](nancy@g5k:/home/rchakode/g5kss11challenge/)

- Papers about SVMSched
  - [http://mescal.imag.fr/membres/rodrigue.chakode/pubs.html](http://mescal.imag.fr/membres/rodrigue.chakode/pubs.html)

- SVMSched is open source and available for downloading
  - [https://gforge.inria.fr/projects/svmsched/](https://gforge.inria.fr/projects/svmsched/)
Conclusion

- Automatic deployment and customization of an OpenNebula cloud
- Dynamic/On-demand provisioning of custom Vms through SVMSched
  - Cloud service-oriented approach
  - High-level abstraction of VMs
  - Transparent customization of VMs
  - Efficient way to set up SaaS (PaaS?) clouds
- Documented and easily reproducible experiment
Thanks for your attention
Design to enable custom integrations

- XML-RPC API to access the core functionality
- Template-based VMs
- Support for automatic configuration of VMs
- Support for contextualizing VMs