OpenNebula Cloud Design & Operations
Technical Workshop
OpenNebula: Cloud Design and Operations

**Goal of the session**
- Discuss some aspects of the three main pillars of a cloud:
  - Compute (hypervisors and Virtual Machines)
  - Storage
  - Networking
- Explore some of the commons operations of an OpenNebula cloud

**Not covered in detail in the session**
- Multi-tenancy: users, groups, ACLs and VDCs
- Advance VM topics: NUMA and CPU pinning, PCI passthrough, SR-IOV, backups
- Advance Networking: Open vSwitch, DPDK
- Services (multi-VM applications), elasticity and OpenNebula Gate
- Automatic cluster provisioning
- Prometheus integration
Hosts & Clusters

OpenNebula Administration

All of the material in this Tutorial is copyright-protected and may not be published in other works without express written permission from OpenNebula Systems. All trademarks are property of their respective owners.
Host & Clusters: Agenda

- Host Subsystem:
  - Overview
  - States
  - Operations
  - Monitoring
  - Monitor Probes
  - Overcommitment

- Cluster Subsystem:
  - Overview
  - Virtual Machine allocation
What is a Host?

- A Linux server able run VM based on KVM
- Host are interfaced with through two drivers
  - **Information**: Obtain monitoring info about the host and VMs
  - **Virtualization**: Perform VM related operations

Configuration Requirements

- oneadmin account (passwordless ssh access)
- oneadmin is used to perform any operation in the host (sudo rules installed as part of node packages)
- known by every other host (DNS or shared /etc/hosts) and frontend
**Host Subsystem: States**

**Operation**
- **Error** state signal connectivity or runtime (probe execution) errors
- **Offline** state is used to perform maintenance operations on the host (e.g. upgrades)
- **Disable** state is used to not allocate more VMs on that host but keep it operational
Host Subsystem: Operations

Hands on!

Flush Hosts [Frontend][oneadmin]

- Useful to perform maintenance operations on a host

  $ onetemplate instantiate “Alpine Linux 3.15”
  $ onevm list

  # pick the host with the most RVMS
  $ onehost flush lab-X-nodeY
  $ onevm list # notice the ‘*’
  $ onehost list # the host is now disabled

  # wait until the VM migrates to node2
  $ onehost list
  # Re-enable the host
  $ onehost enable lab-X-nodeY
Host Subsystem: Monitoring

Monitor Process
- Monitoring agents started in the hypervisors
- Uses TCP/UDP transport for monitoring messages
- Messages includes: Host + VM + Datastore information

Some Configuration Attributes
- **ADDRESS / MONITOR_ADDRESS**: network address to listen/send messages to.
- **PORT**: listening port (4124).
- ***_INTERVAL**: Probe intervals per category (See /etc/one/monitord.conf)

OpenNebula front-end

- OpenNebula demon (oned)
- Monitor demon (monitord)
- drivers
- database

Monitor and state information (TCP/UDP)

Probes

start/stop probe agents

Host
Hands on!

**Monitoring Events** [*Frontend*][oneadmin]

- Look **monitoring events**
  
  $ less /var/log/one/monitor.log

  ...  
  Wed Nov 25 17:06:18 2020 [Z0][HMM][I]: Successfully monitored host: 0
  ...

- Force a **monitor reset** of a host

  $ onehost forceupdate 0
  $ less /var/log/one/monitor.log
  ...
  Wed Nov 25 17:10:21 2020 [Z0][HMM][D]: Monitoring host lab-x-node1(0)
  ...
Host Subsystem: Monitoring Probes (I)

**Location and structure**

- Transfer to the hypervisors and placed in `/var/tmp/one/im/qemu-probes.d`
- Executable that returns `KEY1="VALUE1" KEY2="VALUE2"` ...

```
|-- host
 |  |-- beacon
 |      |-- date.sh
 |      `-- monitord-client-shepherd.sh
 |  |-- monitor
 |      |-- linux_usage.rb
 |      `-- numa_usage.rb
 |  `-- system
 |      |-- architecture.sh
 |      `-- cpu.sh
 |          `-- ... wild_vm.rb
`-- vm
     |-- monitor
     |-- monitor_ds_vm.rb
     `-- poll.rb
|-- status
 |  `-- poll.rb
 |     `-- state.rb
```
Probe versioning

- Mechanism to keep host probes up-to-date and in sync
- Probes are versioned, see /var/lib/one/remotes/VERSION
- Hosts report its current version (version.sh)
- Updates only if version is outdated.
Host Subsystem: Monitoring Probes (III)

Hands on!

**Monitoring Probes** `[lab-x-node2/3][oneadmin]`

- Browse probes and inspect their content
- Try some probes directly

```bash
$ ls -lh /var/tmp/one/im/qemu-probes.d/
$ cat /var/tmp/one/im/qemu-probes.d/host/system/cpu.sh
$ /var/tmp/one/im/qemu-probes.d/host/system/cpu.sh
```
Hands on!

Monitoring Probes \texttt{[lab-x-node1][oneadmin]}

- Add a probe to report hypervisor uptime

```bash
$ vi /var/lib/one/remotes/im/qemu-probes.d/host/system/uptime.sh
#!/bin/bash
echo UPTIME=""$(uptime -p)"
$ chmod +x /var/lib/one/remotes/im/qemu-probes.d/host/system/uptime.sh
$ /var/lib/one/remotes/im/qemu-probes.d/host/system/uptime.sh
```

- Increase version number

- Propagate changes (this will update host based on VERSION)

```bash
$ onehost sync
```

- Force a monitor update (or wait system probes interval 10 minutes by default)

```bash
$ onehost forceupdate
```
Over and Under commitment

- It is possible to reserve CPU or Memory:
  - **RESERVED_CPU**: percentage, subtracted from TOTAL_CPU
  - **RESERVED_MEM**: in KB, subtracted from TOTAL_MEM
- Can be applied to the whole Cluster or to a single Host
- Accepts negative values, to increase capacity (overcommitment)
Clusters

- Logical set of physical resources that can work together (resemble the layout of the DC):
  - Hosts - **Important** a Host can only belong to a single Cluster
  - Datastores - **Important** must have a system datastore (at-least)
  - Virtual Networks

- Default cluster groups all resources by default

**Applications:**

- Ensure that VMs will work (scheduling)
- QoS planning: "Gold" Cluster, "Silver" Cluster
- Access control in Virtual Datacenters (VDCs)

**To add remove resources:**

- `onecluster addhost/delhost`
- `onecluster addvnet/delvnet`
- `onecluster adddatastore/deldatastore`
Cluster Subsystem: Virtual Machine Allocation

VM compatibility

- OpenNebula core, checks resources consistency at creation (A VM cannot have resources in different Clusters)
- OpenNebula adds a directive so it's deployed in Host belonging to that Cluster:
  - AUTOMATIC_REQUIREMENTS="CLUSTER_ID=100 | CLUSTER_ID=101"

Capacity Planning

- Custom variables defined in the Cluster template can be used as placing conditions (note that QOS is a custom attribute):
  
  ```bash
  $ onecluster show the_cluster
  CLUSTER TEMPLATE
  QOS="GOLD"
  
  In the VM you can add the following:
  ```
  - SCHED_REQUIREMENTS = "QOS = GOLD"
Cluster Subsystem: Virtual Machine Allocation (II)

**Hands on!**

**Cluster requirements**

- Check the resources on the default cluster (CLI `onecluster show` or Sunstone)
- Add a QoS label to the cluster
- Create a VM and set QoS requirements (e.g. `QOS="training"`)
- Check the generated requirements and verify allocation

```
$ onevm show 2 | grep -i 'SCHED_REQU\|AUTOMATIC_REQU'
SCHED_REQUIREMENTS="QOS="training""
AUTOMATIC_REQUIREMENTS="(CLUSTER_ID = 0) & !(PUBLIC_CLOUD = YES) & !(PIN_POLICY = PINNED)"
```

- Terminate hard the VM
Datastores & Images

OpenNebula Administration
Datastore & Images: Agenda

- Overview
- Images Operations
- Datastores Drivers
Datastores & Images: Overview

Datastores
- Any storage medium used to store disk images for VMs or files
- OpenNebula supports multiple simultaneous Datastores

The **Image Datastore** stores the image repository

The **Files & Kernels Datastore** stores plain files (not supported in vCenter)

The **System Datastore** holds images of running VMs

Datastores can be local to the hypervisor
## Features & Characteristics

- Performance should be assessed per workload and deployment

<table>
<thead>
<tr>
<th>Feature</th>
<th>NFS/NAS</th>
<th>SSH</th>
<th>Ceph</th>
<th>SAN / LVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk snapshots</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>VM snapshots</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Live migration</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>VM FT</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Disk type</td>
<td>File (raw and qcow2)</td>
<td>Block (rbd / logical volume)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Images: Overview

Images
- Represents a virtual disk image for a VM
- Decoupled from the VM definition

Types (based on data life-cycle)
- Non-persistent: changes are discarded, can be used by several VMs (*install once instantiate many*)
- Persistent: changes are preserved, only 1 VM at a time
- Volatile: Created on-the-fly in the system datastore. Cannot be saved

Types (based on purpose)
- OS: contains a bootable operating system
- CDROM: Read only data in ISO format
- DATABLOCK: Generic storage disk, *in-guest* format needed
- CONTEXT: A plain file to be included in the Context CD
Important Attributes

- Datastore to create the image
- Prefix
  - Determines the BUS type: hd - IDE, sd - SCSI/SATA, vd - virtio
- Format
  - Based on the storage backend it determines the hypervisor interface
- Type: OS, CDROM etc..
- Persistent status

Sources

- A path in the frontend or URL (PATH, --path)
- Another image (oneimage clone)
- A disk of a running VM (onevm disk-saveas)
- An app in a MarketPlace
- An empty image
### Image Operations

- **Alpine Linux** (note use `oneimage show` command)
  - Look for the Image and check attributes: `DEV_PREFIX`, `FORMAT`...
- **Clone** the image (e.g. `oneimage clone` or Sunstone)
- Create a new image **saving the disk of an existing VM in POWEROFF** *(onevm disk-saveas or Sunstone)*
- Change the persistent attribute of an image (check with `oneimage list`)

*Using KVM instead of qemu allows to perform disk-saveas in RUNNING state*

### Datastore Contents

- List the contents of the datastore.
  - Where is the datastore located? (Use the `SOURCE` attribute of alpine image)
  - List the contents
  - Use `qemu-img info` command to list image attributes.
**Datastores: Drivers**

**MarketPlace Drivers**
- **http**: uploads to a folder in a web-server.
- **S3**: uploads to an S3 service

**Datastore Drivers (DS_MAD)**
- **fs**: images in file form.
- **ceph**: images as RBDs
- **RDM**: existing block devices

**Transfer Drivers (TM_MAD)**
- **Filesystem Datastore**: shared, ssh, lvm transfer modes
- **Ceph Datastore**: ceph, rbd based
- **RDM Datastore**: dev, symlinks to block devices
- **OpenNebula > 5.6**: mixed modes (e.g. Ceph + ssh)
Driver actions scripts

- Operations are performed through action scripts (Folder: /var/lib/one/remotes/tm/ssh)
  - Check the clone script ssh and describe how the image is moved to the hypervisor

Image Types and Virtual Machines

- Create a VM template (clone an existing one). Use Suntone.
  - Create a persistent DATABLOCK disk (use 10GB, qcow2 and virtio bus)
  - Add the data image to the template
  - Add a Volatile disk to the template for swap (use 4GB)
  - Instantiate the template and check the files on the hypervisor (/var/lib/one/datastores/0/<vm_id>)
  - Login in the VM and check the devices (e.g. lsblk, swapon)
Virtual Networks
OpenNebula Administration

All of the material in this Tutorial is copyright-protected and may not be published in other works without express written permission from OpenNebula Systems. All trademarks are property of their respective owners.
Virtual Networks: Agenda

- Virtual Networks:
  - Overview
  - Definition
  - Example: VXLAN networks
  - Security Groups
  - Operations & Usage
  - Self-provisioning
Virtual Networks: Overview

- Leaf-Spine switched or routed backbone
- Under-control of SDN
- Most deployments just few TOR switches

Datacenter Network

Virtual switches

SDN

OpenNebula

Virtual Network

VM

VM

VM

VM
Virtual Networks: Definition

Addresses Space
- Logical super-net with multiple Address Ranges
- IPv4, IPv6 (Global & ULA), Dual stack and Ethernet
- Can override virtual network settings

Security Groups
- Managed as a separate entity
- Inbound & Outbound, TCP/UDP/ICMP

Configuration & QoS
- Guest: IP and network address, routes, DNS, Gateway
- Average, peak and burst bandwidth per virtual NIC

VNET drivers & Configuration (e.g. bridge VLAN_ID)

Three-phase setup
- pre, post and clean
- live-migrations
- live updates (QoS & SG)
Virtual Networks: VXLAN Networks

**Bridged Networks**

- Virtual machine traffic is directly bridged through an existing Linux bridge
- VXLAN networks will create associated interfaces (bridges, tagged interfaces)
- VXLAN networks can operate in two modes
  - multicast
  - BGP EVPN
Virtual Networks: Bridged Networks (II)

Hands on!

<table>
<thead>
<tr>
<th>Network Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Create one VM and study the virtual network at three levels:</td>
</tr>
<tr>
<td>○ Virtual Machine (in-guest)</td>
</tr>
<tr>
<td>■ Check address (ip address)</td>
</tr>
<tr>
<td>○ Host bridges configuration (hypervisor)</td>
</tr>
<tr>
<td>■ Check devices (ip -d link show eth0.15 and ip link show master onebr.15)</td>
</tr>
<tr>
<td>■ Optional (using vtysh execute show bgp evpn route)</td>
</tr>
<tr>
<td>○ OpenNebula (Cloud)</td>
</tr>
<tr>
<td>■ Check network usage (Leases &amp; Addresses tabs)</td>
</tr>
</tbody>
</table>
Virtual Networks: Security Groups

Security Groups
- Defines a set of rules to accept specific traffic to/from a VM
- SG’s can be applied to Virtual Networks or Address Ranges
- A SG update will be propagated to running VMs
- **DEFAULT SG** allows all traffic!

Rules

<table>
<thead>
<tr>
<th>Traffic Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol (TCP, UDP, IPSEC, ICMP)</td>
</tr>
<tr>
<td>Port or ICMP type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Traffic Origin/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>INBOUND/OUTBOUND</td>
</tr>
<tr>
<td>IP (or IP set)</td>
</tr>
<tr>
<td>OpenNebula Virtual Network</td>
</tr>
</tbody>
</table>
**Security Groups: Creation and live update**

- Add a new Security Group that allows access to TCP port 22 only.
- Remove the ‘default’ SG from the private network and add the ‘ssh’ specific one.
- Instantiate the VM
- Log into the host where the VM is running and check iptables (e.g. `iptables-save`)

```bash
-A one-4-0-i -m state --state RELATED,ESTABLISHED -j RETURN
-A one-4-0-i -j RETURN
-A one-4-0-i -p tcp -m multiport --dports 22 -j RETURN
-A one-4-0-i -j DROP
-A one-4-0-o -m state --state RELATED,ESTABLISHED -j RETURN
-A one-4-0-o -j RETURN
-A one-4-0-o -j RETURN
-A one-4-0-o -j DROP
-A opennebula -m physdev --physdev-in one-4-0 --physdev-is-bridged -j one-4-0-o
-A opennebula -m physdev --physdev-out one-4-0 --physdev-is-bridged -j one-4-0-i
```

- Remove the ‘ssh’ rule from the SG and allow all traffic. Check the security group status & update.
## Virtual Networks: Operations & Usage

### Virtual NIC Type
- **Default**
  - Creates a network interface inside the VM
- **Alias**
  - Add a new address to an existing interface. It does not create a vNIC

### Network Scheduling
- **Default**
  - A network must be specified before instantiating the VM
- **Auto**
  - The network is automatically selected by the scheduler using a match-making algorithm

### Lease Management
- Leases used in an address range can be put on hold
- Hold leases won’t be assigned to a VM
Reservations

- A user can take a Virtual Network slice for personal usage
- A reservation looks and behaves as a standard Virtual Network (same interface)
- Reservations are private for each users and inherit physical attributes (drivers, VLAN identifiers...)

Virtual Network Templates

- Equivalent relationship as Virtual Machine Templates and Virtual Machines
- Define the implementation and physical attributes (driver, VLAN identifiers...)
- Allow the final user to define the logical attributes
- Attributes can be restricted by VNET_RESTRICTED_ATTR
Virtual Networks: Operations & Usage (II)

Hands on!

Virtual Network Operations
- Hold a lease and check the network (onevnet hold or Sunstone)
- Make a reservation and update the VM template to use this reservation
- Verify the Virtual Network status

Virtual Network Usage
- Go to instantiate VM (Alpine)
- Add a nic interface (select private network)
- Add a second interface as alias of the first one
- Instantiate the VM
- Check inside the VM that there is only one interface, but 2 ip addresses
Virtual Machines

OpenNebula Administration

All of the material in this Tutorial is copyright-protected and may not be published in other works without express written permission from OpenNebula Systems. All trademarks are property of their respective owners.
Virtual Machines: Agenda

- Virtual Machines:
  - Overview
  - VM Templates
  - Contextualization
  - Scheduling
  - Advance configurations
Virtual Machines: Overview

**Virtualization**
- Communication with the hypervisor
- KVM Hypervisor (other supported LXC, firecracker…)

**VM Operations**
- Create and terminate
- Life-cycle operations (poweroff, stop, pause, resume, migration)
- Capacity resize
- Disk snapshots
- Disk & vNIC hotplug
- System Snapshot
- Scheduling actions
Hands on!

**VM Drivers**

- VM operations as action scripts.
  
  **Hands on!**
  
  Explore `/var/lib/one/remotes/vmm/kvm` (e.g. deploy)

- VM information is obtained through **poll action** and coupled with the monitoring process.
  
  **Hands on!**
  
  Start a VM
  
  Terminate it `virsh -c qemu:///system destroy one-<ID>`

  Observe VM progress (`/var/log/one/<ID>.log`) & recover with resume
Virtual Machines: VM Templates

Capacity & Topology

- **MEMORY**: Amount of RAM required for the VM, in Megabytes.
- **CPU**: CPU ratio (e.g. half a physical CPU is 0.5). This value is mapped to libvirt shares value.
  
  **Example**: CPU = 2 will use double physical CPU ticks than CPU = 1
- **VCPU**: Number of virtual CPUs (virtual topology sockets, cores and threads can be defined)
- **CPU model & machine type** can be fine tuned

Storage & Network Attributes

- Most attributes supported by Virtual Networks and Images
- Overrides Image & Virtual Networks attributes

Scheduling Actions

- Periodic actions executed on the VMs (e.g. shutdown at 8 pm.)
- Lease period (e.g. terminate the VM after 1 week)
Virtual Machines: Contextualization

**Contextualization**
- Mechanism to pass information to the VM at boot time
- Usage:
  - Guest OS configuration (hostname, network...)
  - Execute custom startup scripts
  - Include arbitrary files (files & kernels datastore)
- OpenNebula context packages installed in the guest

**User Inputs**
- Procedure to build parameterized images for your users
- User input is passed in CONTEXT section to as input in the boot process
- Can be updated via updateconf (Sunstone or CLI)
Virtual Machines: Contextualization (II)

Hands on!

Contextualization

- Prepare the template to bootstrap a BLOG installation

```
root@ubuntu:~# mount -L CONTEXT /mnt
root@ubuntu:~# cat /mnt/context.sh
# Context variables generated by OpenNebula
...
BLOG_ADMIN_PASS='BLAST!'
BLOG_NAME='OpenNebula Wonders'
...```
Virtual Machines: Scheduling

Scheduler
- In charge of deploying PENDING VMs and reschedules
- Separate process (mm_sched) it uses XML-RPC API

The Matchmaking Scheduler
- Filter out Hosts that don't match the SCHED_REQUIREMENTS or without enough resources
- Evaluate the SCHED_RANK expression using information gathered by the monitoring drivers
- Same process is used to schedule the Datastores and Virtual Network
- Deploy the VM to the highest ranked Host and System Datastore
Virtual Machines: Scheduling (II)

Hands on!

Capacity checking

- Explore the Scheduling section of Sunstone for VM Templates
  - Force the deployment of a VM Template to lab-x-node2
- Create a very big VM (set CPU=4 on instantiate)
- Check VM, SCHED_MESSAGE
- Check /var/log/one/sched.log
- Overcommit the host (Sunstone) so the VM can be scheduled
**Virtual Machines: Advance Configurations**

<table>
<thead>
<tr>
<th>Virtio</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Main platform for IO virtualization in KVM</td>
</tr>
<tr>
<td>● No emulation, integrated with Linux Kernel</td>
</tr>
<tr>
<td>● Consider additional tuning (i.e. queues)</td>
</tr>
<tr>
<td>● For disks: DEV_PREFIX=vd</td>
</tr>
<tr>
<td>● For network cards: MODEL=virtio</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Guest OS Disk Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>● libguestfs to customize the image</td>
</tr>
<tr>
<td>● Install opennebula-context packages</td>
</tr>
<tr>
<td>virt-alignment-scan</td>
</tr>
<tr>
<td>● Use volatile volumes for swap OpenNebula context mounts it.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Performance tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Overcommitment</td>
</tr>
<tr>
<td>● Cgroups used natively by OpenNebula through CPU shares</td>
</tr>
<tr>
<td>● PCI passthrough and SR-IOV</td>
</tr>
<tr>
<td>● Hugepages, NUMA and CPU pinning</td>
</tr>
<tr>
<td>● Disk and vNIC QoS parameters</td>
</tr>
</tbody>
</table>
OpenNebula: Cloud Design and Operations

Goal of the session
● Discuss some aspects of the three main pillars of a cloud:
  ○ Compute (hypervisors and Virtual Machines)
  ○ Storage
  ○ Networking
● Explore some of the commons operations of an OpenNebula cloud

Not covered in detail in the session
● Multi-tenancy: users, groups, ACLs and VDCs
● Advance VM topics: NUMA and CPU pinning, PCI passthrough, SR-IOV, backups
● Advance Networking: Open vSwitch, DPDK
● Services (multi-VM applications), elasticity and OpenNebula Gate
● Automatic cluster provisioning
● Prometheus integration
This work has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union’s Horizon Europe research and innovation programme under Grant Agreement 6G-SANDBOX - 101096328.