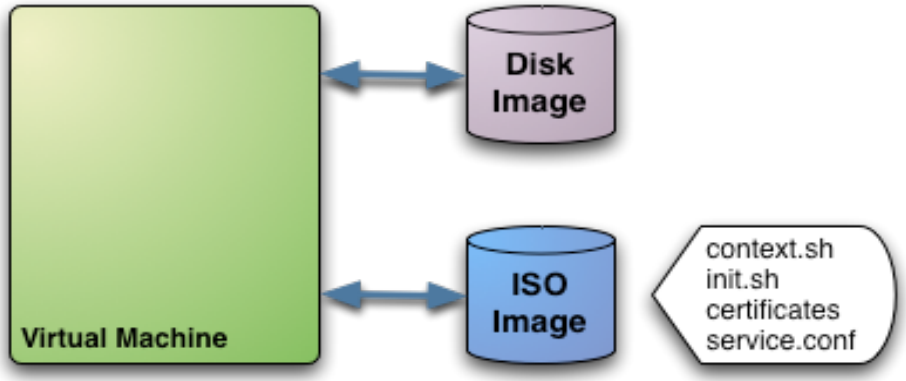
OpenNebula Technology Days, 20-21 July 2010

Session 6
Advanced Usage

Javier Fontán
jfontan@fdi.ucm.es

Using the Private Cloud: Virtual Machines

- Context contains data to be passed to the VM at boot time




```
#-----------------------------------
#  Context for the VM
#    values can be:
#    $<template_variable>
#    $<template_variable>[<attribute>]
#    $<template_variable>[<attribute>, <attribute2>=<value2>]
#    $<vm_id>.<context_var>
#-----------------------------------

CONTEXT = [
  var_1 = "value_1",#Will be in context.sh as var_1="val_1" (sh syntax)
  var_n = "value_n",#Will be in context.sh as var_N="val_N" (sh syntax)
  files = "space-separated list of paths to include in context device",
  target= "device to attach the context device" ]
```

Using the Private Cloud: Virtual Machines

- Hands on... Add custom ssh keys the VM image
  - Check boot process of the ttylinux VM (systemrc.local) it will
    - mount iso (do it yourself and see the ISO layout...)
    - Source context.sh
    - In this example it will execute init.sh so you can try anything

```
CONTEXT = [
    files      = "/srv/cloud/one/ttylinux-xen/init.sh /srv/cloud/
one/.ssh/id_rsa.pub",
    target      = "hdc",
    root_pubkey = "id_rsa.pub"
]
```

```
$ more init.sh
#!/bin/bash
if [ -f /mnt/context/context.sh ]
then
  . /mnt/context/context.sh
fi
if [ -f /mnt/context/$ROOT_PUBKEY ]; then
        cat /mnt/context/$ROOT_PUBKEY >> /root/.ssh/authorized_keys
fi
```

Using the Private Cloud: Virtual Machines

- Tunning the placement of VMs with the Match-making scheduler
  - First those hosts that do not meet the VM requirements are filtered out (REQUIREMENTS)
  - RANK is evaluated for the remaining hosts
  - That with the highest RANK is used for the VM
- Placement policies are specified per VM

```
#------------------------------------
#           Scheduler
#------------------------------------
# Use Host Monitor attributes
REQUIREMENTS = "Bool_expression_for_reqs"
RANK         = "Arith_expression_to_rank_hosts"
```

- Hands on... try a simple VM pinning

```
REQUIREMENTS = "HOSTNAME=\"...\""
```

- Hands on... try a simple load-aware policy

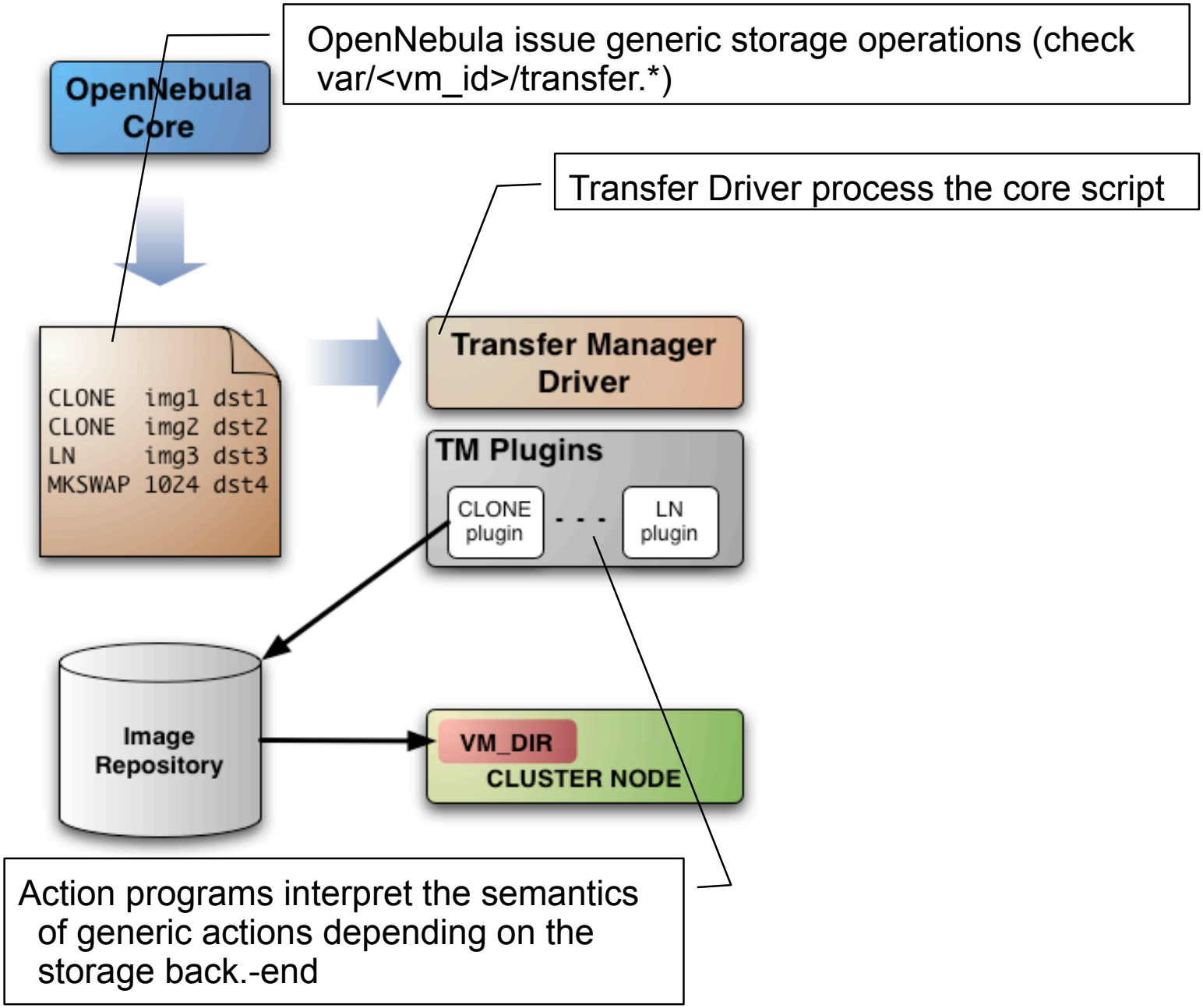
```
RANK = FREECPU
```

Customizing and Extending your Cloud

- You can customize your cloud by:
  - Tunning or adapting the transfer operations to your **storage back-end**
  - Adding new **monitorization** probes to improve the VM placement
  - Adjusting VM operations to your hypervisor installation
  - Trigger **custom actions** on specific VM events (e.g. "on VM creation update the accounting DB" or "on VM shutdown send an email")
- You can extend your cloud by:
  - Developing new drivers for other hypervisors
  - Developing new drivers for other storage back-ends
  - Developing Cloud applications using the OpenNebula API or the Cloud APIs

OpenNebula is very scripting friendly, drivers can be written in any language. You can modify the current ones or use them as templates for new ones.

Customizing the Storage of your Cloud



Customizing the Storage of your Cloud

- OpenNebula requests the following abstract operations over a VM image
  - `CLONE`: This action will basically make a copy of the image from ORIGIN to DESTINATION.
  - `LN`: Creates a symbolic link in DESTINATION that points to ORIGIN
  - `MKSWAP`: Generates a swap image in DESTINATION. The size is given in ORIGIN in MB.
  - `MKIMAGE`: Creates a disk image in DESTINATION and populates it with the files inside ORIGIN directory.
  - `DELETE`: Deletes ORIGIN file or directory.
  - `MV`: Moves ORIGIN to DESTINATION.

Customizing the Storage of your Cloud

- Actions are defined in

`$ONE_LOCATION/etc/tm_<storage>/tm_<storage>.conf`

```
$ more /srv/cloud/one/etc/tm_ssh/tm_ssh.conf
CLONE   = ssh/tm_clone.sh
LN      = ssh/tm_ln.sh
MKSWAP  = ssh/tm_mkswap.sh
MKIMAGE = ssh/tm_mkimage.sh
DELETE  = ssh/tm_delete.sh
MV      = ssh/tm_mv.sh
```

- Actions scripts are placed in

`$ONE_LOCATION/lib/tm_commands/<storage>/`

```
$ ls /srv/cloud/one/lib/tm_commands/ssh/
tm_clone.sh    tm_delete.sh  tm_mkimage.sh  tm_mv.sh
tm_context.sh  tm_ln.sh      tm_mkswap.sh
```

Customizing the Storage of your Cloud

- Hands on... Take a look to the tm_clone.ssh

```
. $TMCOMMON
...
log "Creating directory $DST_DIR"
exec_and_log "ssh $DST_HOST mkdir -p $DST_DIR"
...
case $SRC in
http://*)
    log "Downloading $SRC"
    exec_and_log "ssh $DST_HOST wget -O $DST_PATH $SRC"
    ;;

*)
    log "Cloning $SRC"
    exec_and_log "scp $SRC $DST"
    ;;
esac

exec_and_log "ssh $DST_HOST chmod a+w $DST_PATH"
```

- Hands on... Check the semantics of other operations for the ssh storage, e.g. tm_ln.ssh

Storage Customization Examples

- Make swap images local to the physical node executing the VM
  - The script that generates swap images is called MKSWAP
  - Swap images are usually generated in VM directory
  - Link the newly create swap image to the VM directory
- Make OpenNebula aware of compressed images
  - Images are cloned by CLONE script

Customizing the Information System

- OpenNebula gets host information by executing an arbitrary number of probes
- A probe is a program that returns the monitorization metric in the form

```
METRIC_NAME = VALUE
```

- Probes are configured in

```
$ONE_LOCATION/etc/im_<hypervisor>/im_<hypervisor>.conf
```

And placed in

```
$ONE_LOCATION/lib/im_probes
```

- Probe information is mainly used for VM placement

Customizing the Information System

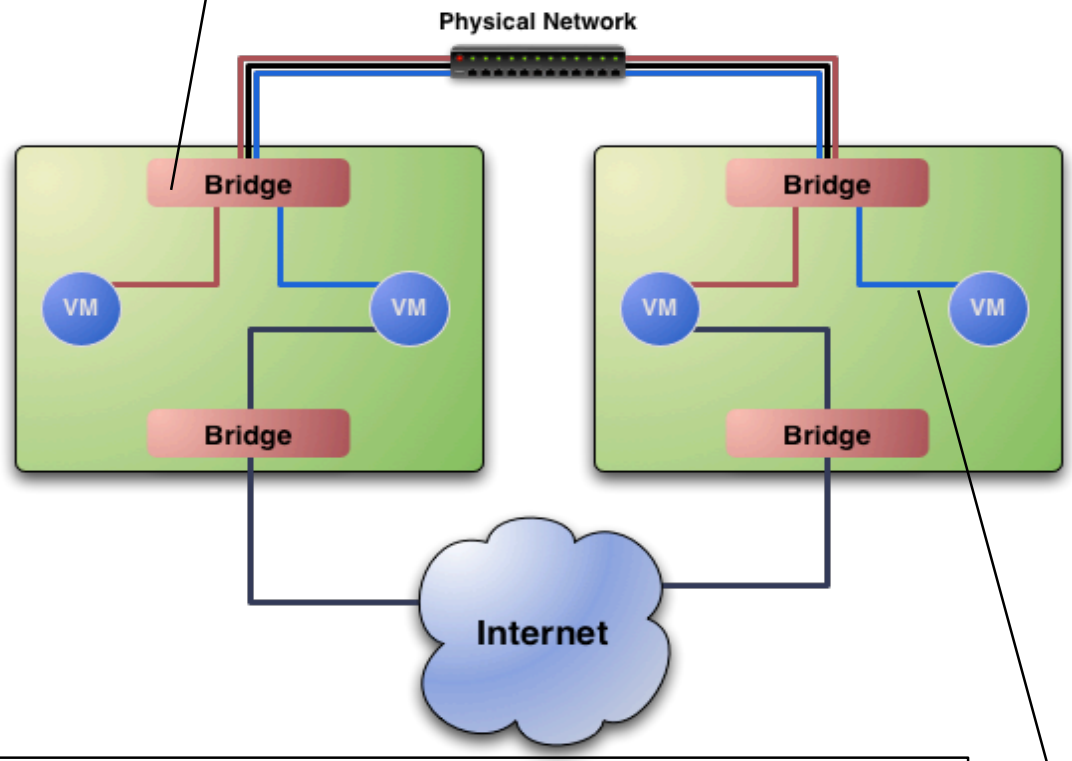
- Hands on... Take a look to the default probes defined for KVM

```
$ more /home/ruben/Virtual/one/etc/im_kvm/im_kvm.conf
cpuarchitecture=architecture.sh
nodename=name.sh
cpu=cpu.sh
kvm=kvm.rb

$ more name.sh
#!/bin/sh

echo HOSTNAME=`uname -n`
```

- Hands on... Create a new probe that returns the number of VMs in `RUNNING_VMS` (e.g. you can use virsh, pgrep kvm...). Use the new metric to pack VMs (`RANK=RUNNING_VMS`).

Customization with Hooks: Network Isolation

- IN: Only Ethernet frames from a MAC in Red LAN
- OUT: Only Ethernet frames from the MAC assigned by OpenNebula



- Networks are isolated at layer 2
- You can put any TCP/IP service as part of the VMs (e.g. DHCP, nagios...)

Customization with Hooks: Network Isolation

- Requirements (this has to be done in all the cluster nodes)
  - Check that ebtables package is installed
  - Allow oneadmin to use the ebtables command through sudo

```
#visudo
...
oneadmin    ALL=(ALL) NOPASSWD: /sbin/ebtables *
...
```

- Configure the hooks for OpenNebula

```
VM_HOOK = [
    name       = "ebtables-start",
    on         = "running",
    command    = "/srv/cloud/one/share/hooks/ebtables-kvm",
    arguments = "one-$VMID",
    remote     = "yes" ]
VM_HOOK = [
    name       = "ebtables-flush",
    on         = "done",
    command    = "/srv/cloud/one/share/hooks/ebtables-flush",
    arguments = "",
    remote     = "yes" ]
```

Customization with Hooks: Network Isolation

- Hands on... Start a couple of VMs in Networks Red and Blue.
  - Check the ebtables rules in the hosts
  - Check connectivity between VMs
  - Change the network mask of the VMs and check connectivity
  - Shutdown and check the ebtables rules