OGF25/EGEE User Forum
Catania, Italy
2 March 2009

Elastic Management of a Grid Computing Service with OpenNebula and Amazon EC2

Constantino Vázquez Blanco
Javier Fontán Muiños
Raúl Sampedro

dsa-research.org

Distributed Systems Architecture Research Group
Universidad Complutense de Madrid

dsa-research.org

Outline

- **Part I – Open Nebula Overview**
  - What is OpenNebula
  - Benefits
  - Features
  - Architecture
- **Part II – Deploying and using OpenNebula on a simple scenario**
  - Physical architecture
  - VM description
  - Image Management
  - Network Management
  - VM lifecycle
- **Part III – Building a scalable SGE cluster**

Part I

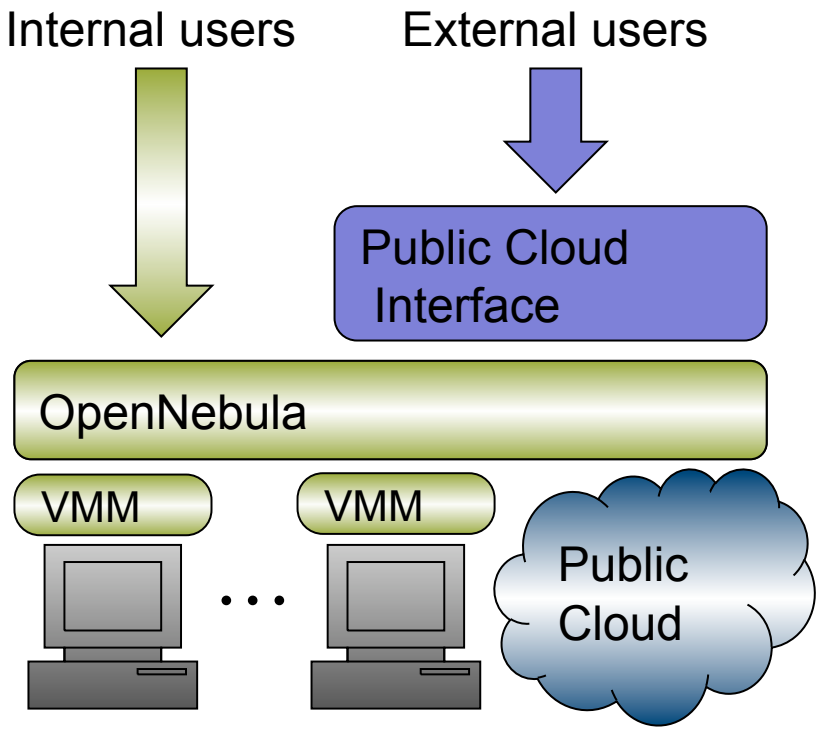
OpenNebula Overview

What is OpenNebula?

The OpenNebula Virtual Infrastructure Engine

Extending the Benefits of Virtualization to Clusters

- Dynamic deployment and re-placement of virtual machines on a pool of physical resources
- Transform a rigid distributed physical infrastructure into a flexible and agile virtual infrastructure



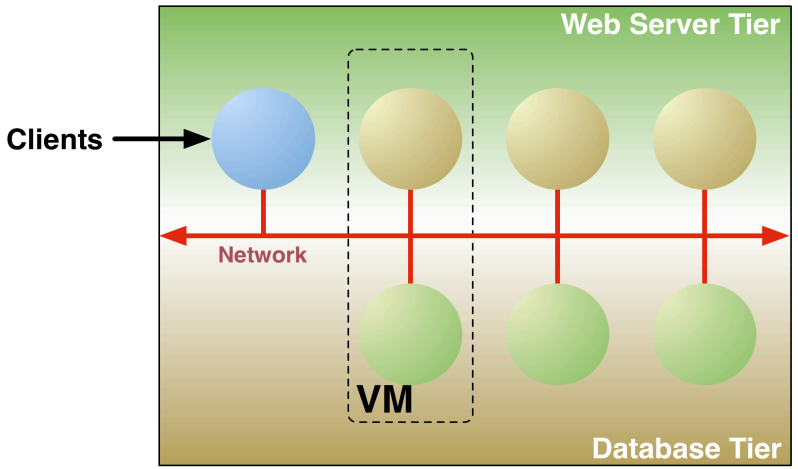
- Backend of Public Cloud: Internal management of the infrastructure
- Private Cloud: Virtualization of cluster or data-center for internal users
- Cloud Interoperation: On-demand access to public clouds

What is OpenNebula?

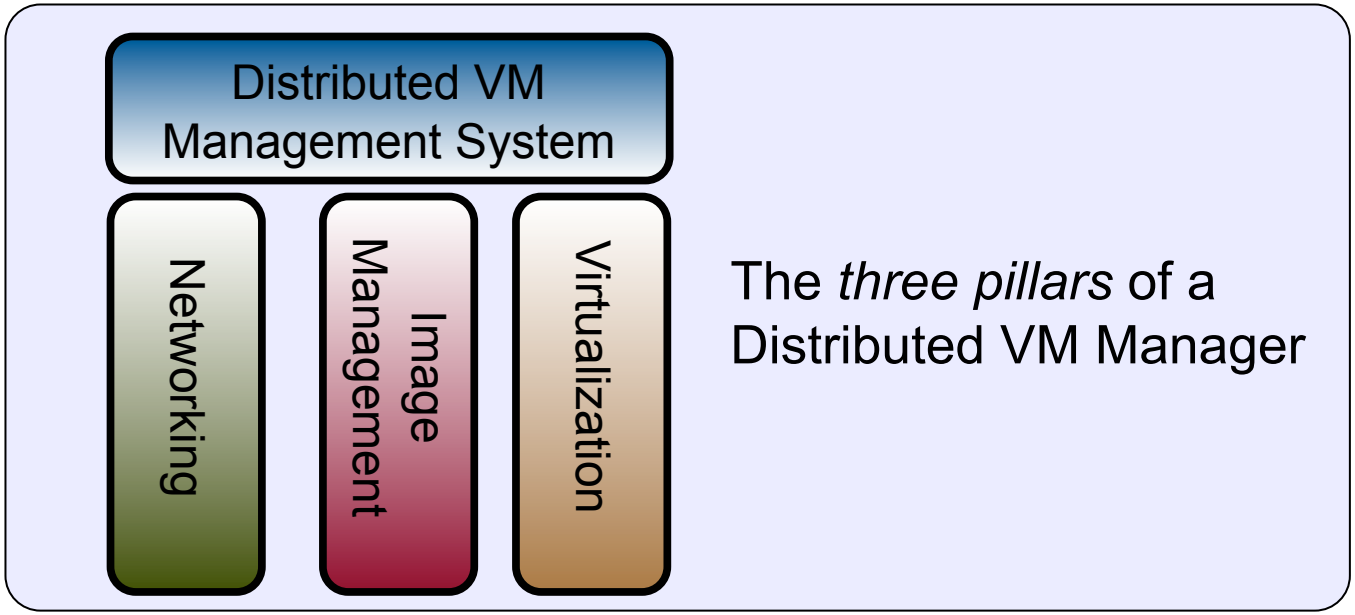
Virtual Machine Management Model

Service as Management Entity

- Service structure
  - Service components run in VMs
  - Inter-connection relationship
  - Placement constraints
- The VM Manager is service agnostic
- Provide infrastructure context



Distributed VM Management Model



The *three pillars* of a Distributed VM Manager

dsa-research.org

Benefits

System Manager

- Centralized management of VM workload and distributed infrastructures
- Support for VM placement policies: balance of workload, server consolidation…
- Dynamic resizing of the infrastructure
- Dynamic partition and isolation of clusters
- Support for heterogeneous workload
- Dynamic scaling of private infrastructure to meet fluctuating demands

Service Manager

- On-demand provision of virtual machines

System Integrators

- Open and flexible architecture and interfaces, open source software
- Integration with any component in the virtualization/cloud ecosystem, such as cloud providers, hypervisors, cloud-like interfaces, virtual image managers, service managers, schedulers…

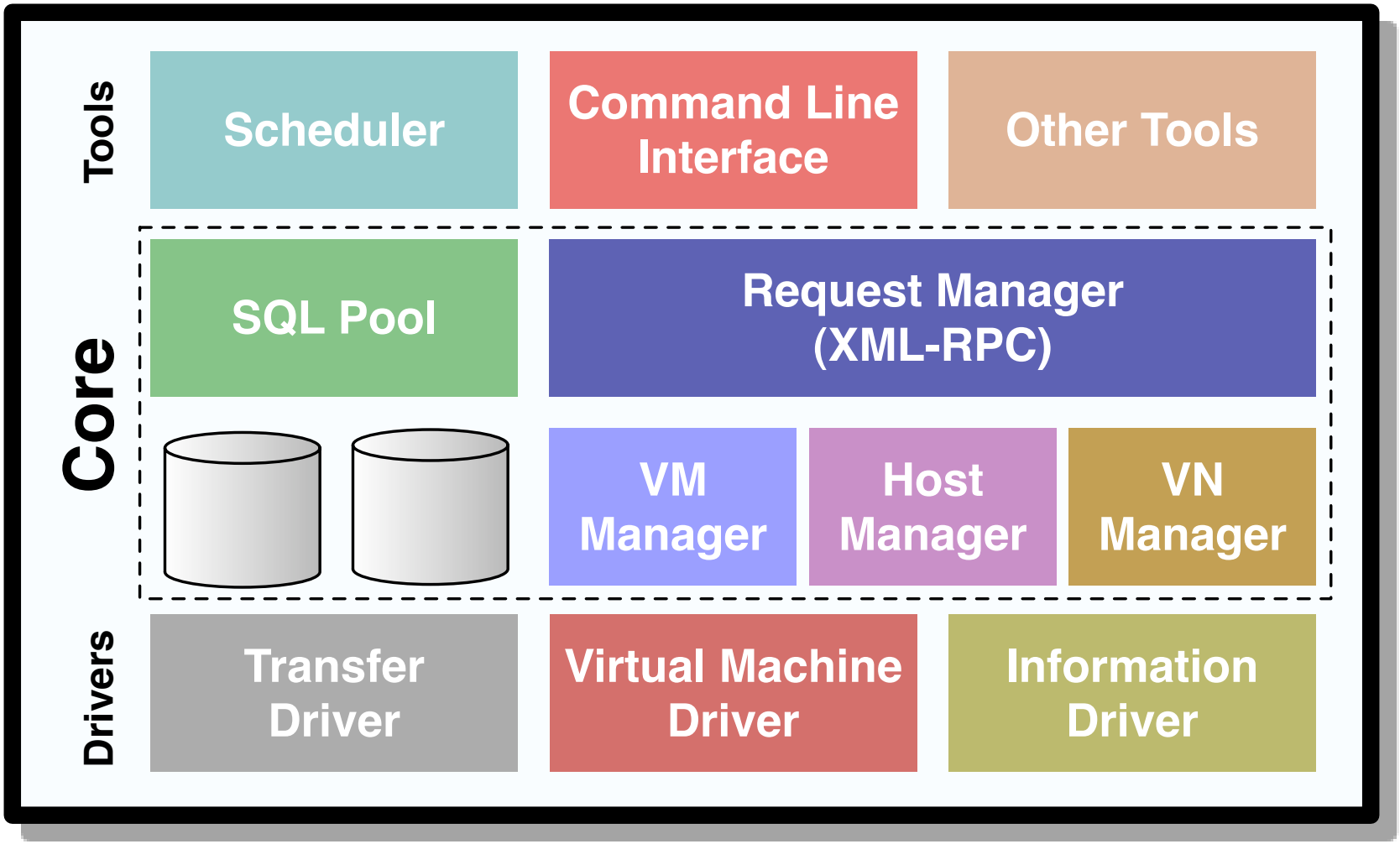
dsa-research.org

Features

Feature	Function
User Interface	• Unix-like CLI to manage VM life-cycle and physical boxes • XML-RPC API and libvirt interface
Scheduler	• Requirement/rank matchmaker • Generic framework to build any scheduler
Virtualization Management	• Xen, KVM and libvirt connectors • Amazon EC2
Image Management	• General mechanisms to transfer and clone VM images
Network Management	• Definition of virtual networks to interconnect VMs
Fault Tolerance	• Persistent database backend to store host and VM information
Scalability	• Tested in the management of hundreds of VMs
Installation	• Installation on a UNIX cluster front-end without requiring new services in the remote resources • Distributed in Ubuntu 9.04 (Jaunty Jackalope), due to be released in April 2009

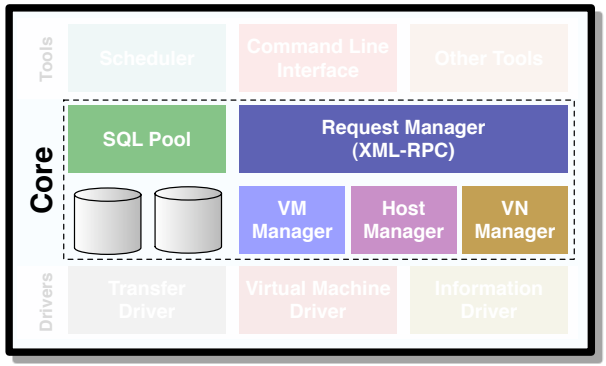
Architecture

OpenNebula Architecture



Architecture

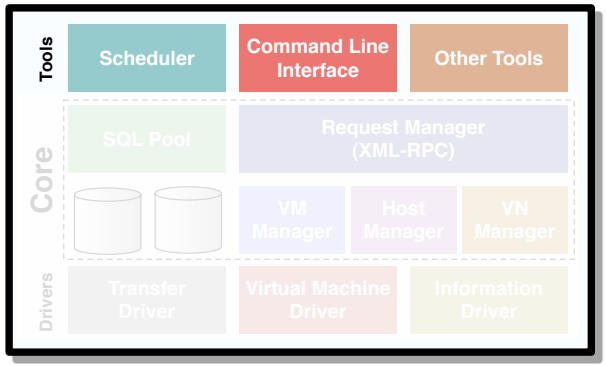
Core



- Request manager: Provides a XML-RPC interface to manage and get information about ONE entities.
- SQL Pool: Database that holds the state of ONE entities.
- VM Manager (virtual machine): Takes care of the VM life cycle.
- Host Manager: Holds the information about hosts and how to interact with them.
- VN Manager (virtual network): This component is in charge of generating MAC and IP addresses.

Architecture

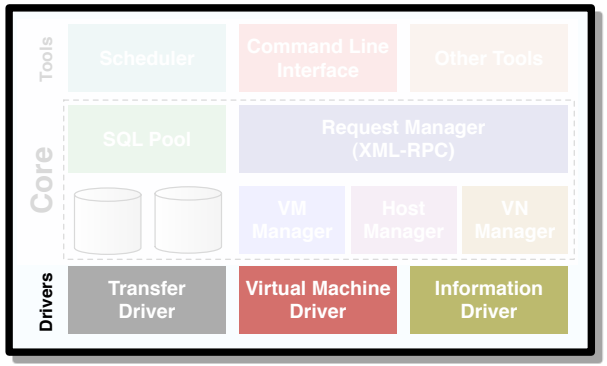
Tools



- Scheduler: This component searches for physical hosts to deploy newly defined VMs
- Command Line Interface: Commands used to manage OpenNebula entities.
  - onevm: Virtual Machines
    - *create, list, migrate…*
  - onehost: Hosts
    - *create, list, disable…*
  - onevnet: Virtual Networks
    - *create, list, delete…*

Architecture

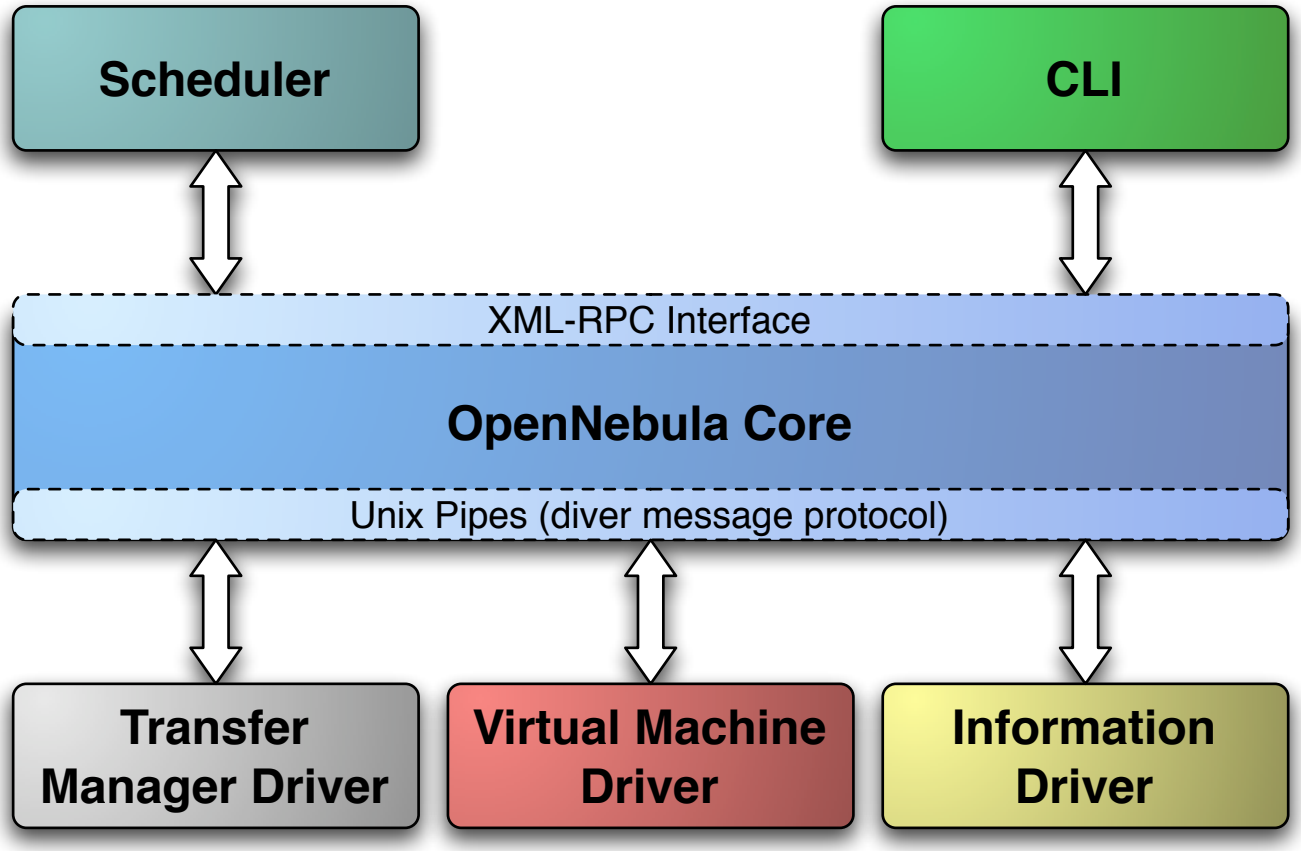
OpenNebula Drivers



- Transfer Driver: Takes care of the images.
  - *cloning, deleting, creating swap image…*
- Virtual Machine Driver: Manager of the lifecycle of a virtual machine
  - *deploy, shutdown, poll, migrate…*
- Information Driver: Executes scripts in physical hosts to gather information about them
  - *total memory, free memory, total cpus, cpu consumed…*

Architecture

Process separation



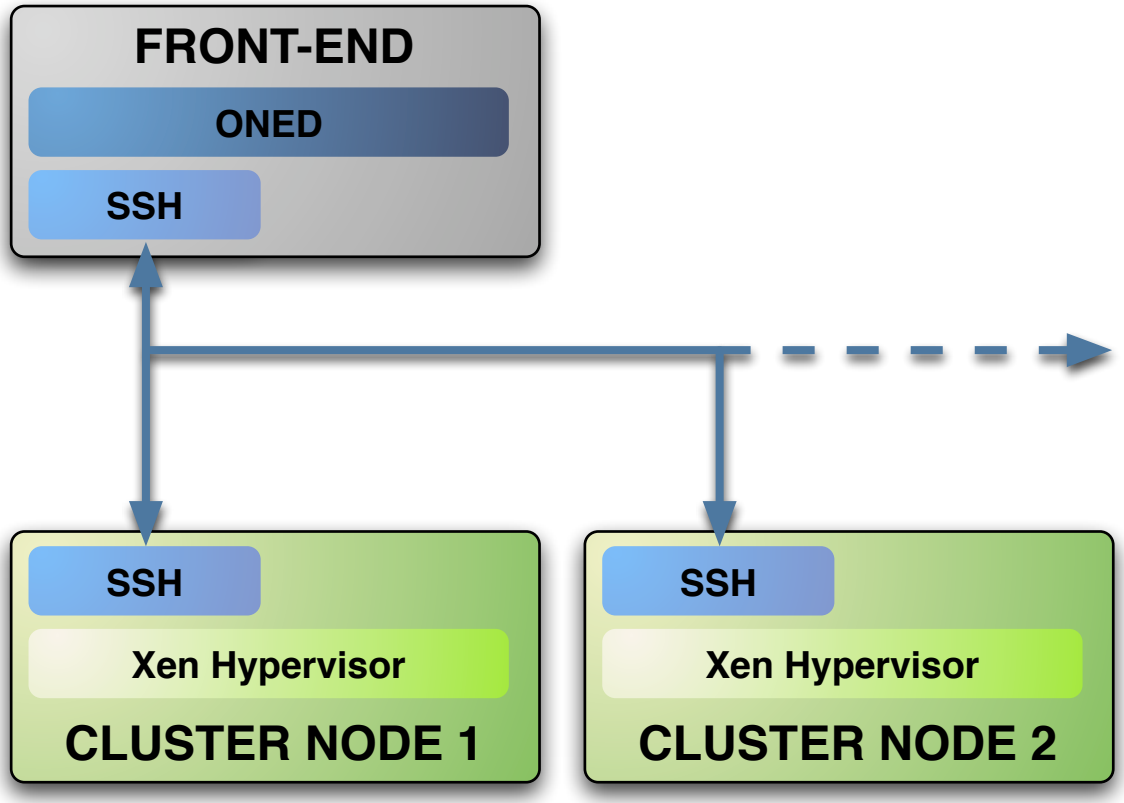
- Scheduler is a separated process, just like command line interface.
- Drivers are also separated processes using a simple text messaging protocol to communicate with OpenNebula Core Daemon (oned)

Part II

Deploying and using OpenNebula on a simple scenario

Physical Architecture

OpenNebula engine from the physical point of view



dsa-research.org

VM Description

Common template options

Option	Description
NAME	• Name that the VM will get for description purposes.
CPU	• Percentage of CPU divided by 100 required for the Virtual Machine.
OS (KERNEL, INITRD)	• Path of the kernel and initrd files to boot from.
DISK (SOURCE, TARGET, CLONE, TYPE)	• Description of a disk image to attach to the VM.
NIC (NETWORK)	• Definition of a virtual network the VM will be attached to.

- Multiple disk an network interfaces can be specified just adding more disk/nic statements.
- To create swap images you can specify TYPE=swap, SIZE=<*size in MB*>.
- By default disk images are cloned, if you do not want that to happen CLONE=no can be specified and the VM will attach the original image.

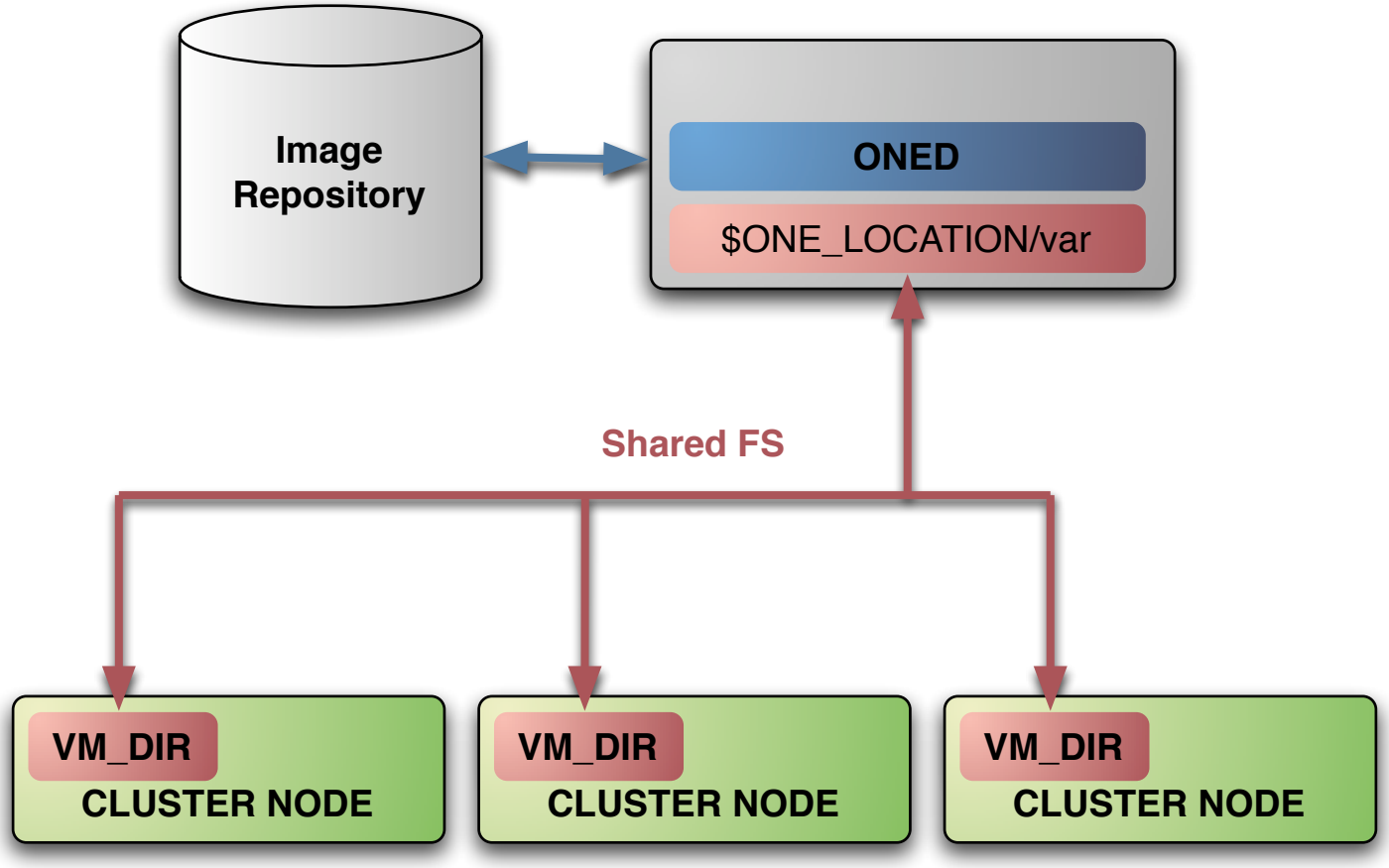
VM Description

Example

```
NAME = vm-example
CPU    = 1
MEMORY = 512

# --- kernel & boot device ---
OS = [
  kernel   = “/vmlinuz”,
  initrd   = “/initrd.img”,
  root     = “sda” ]

# --- 2 disks ---
DISK = [
  source   = “/images/etch/disk.img”,
  target   = “sda” ]

DISK = [
  type     = swap,
  size     = 1024,
  target   = “sdb” ]

# --- 1 NIC ---
NIC = [ network=“public” ]
```

Image Management

Architecture for a cluster configured with shared directory.



dsa-research.org

Image Management

VM description example

Disk description options

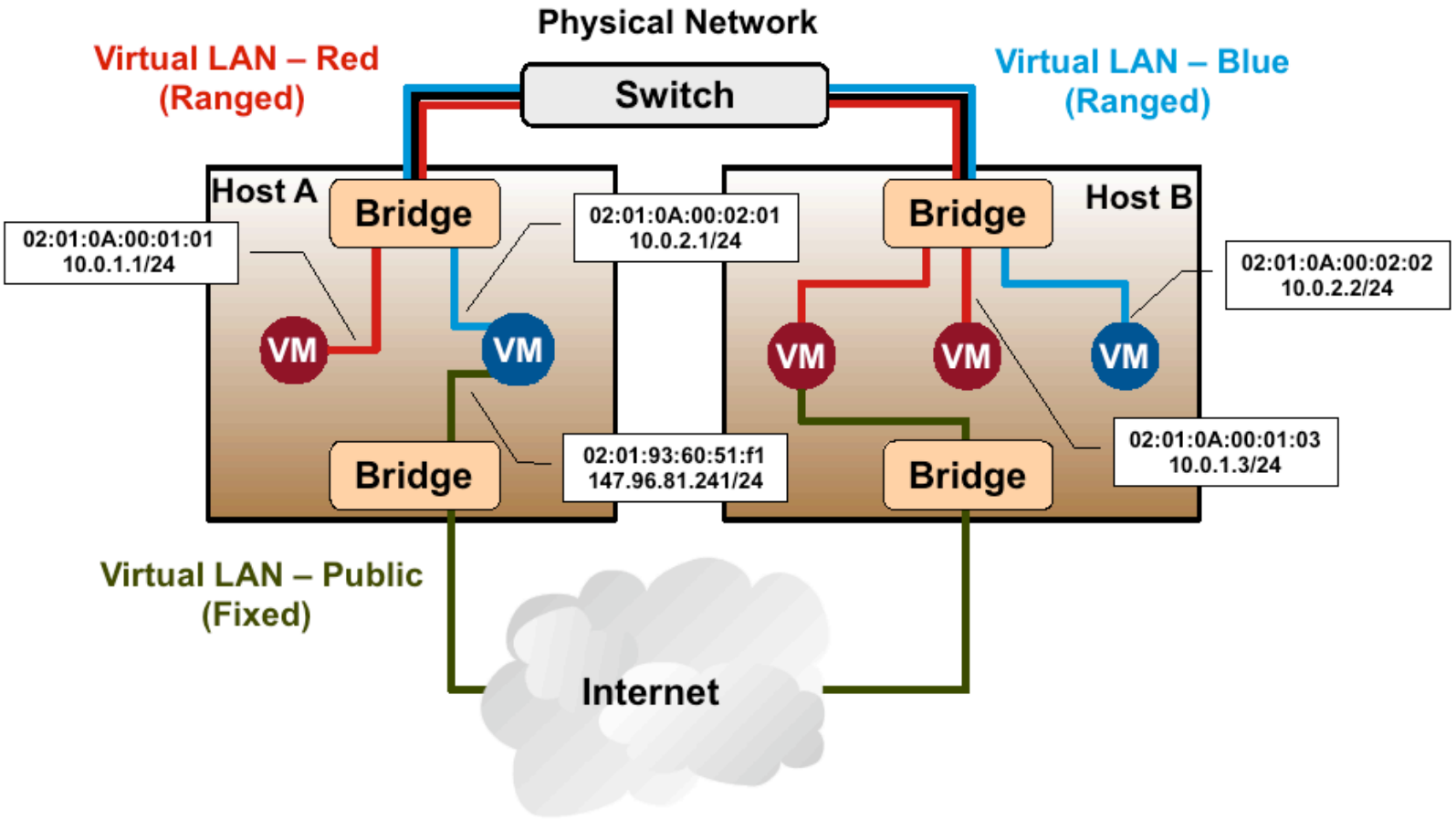
```
DISK = [
  type     = “floppy|disk|cdrom|swap”,
  source   = “path_to_disk_image_file|physical_dev”,
  size     = “size_in_GB”,
  target   = “device_to_map_disk”,
  bus      = “ide|scsi|virtio|xen”,
  readonly = “yes|no”,
  clone    = “yes|no”,
  save     = “path_to_disk_image_file” ]
```

Disk description examples

```
DISK = [
  source   = “/images/etch/disk.img”,
  target   = “sda” ]

DISK = [
  type     = swap,
  size     = 1024,
  target   = “sdb” ]
```

Network Management

Physical configuration



dsa-research.org

Network Management

Ranged network definition

```
NAME            = “Private LAN”
TYPE            = RANGED
BRIDGE          = eth0
NETWORK_SIZE    = 250
NETWORK_ADDRESS= 10.0.0.0
```

Fixed network definition

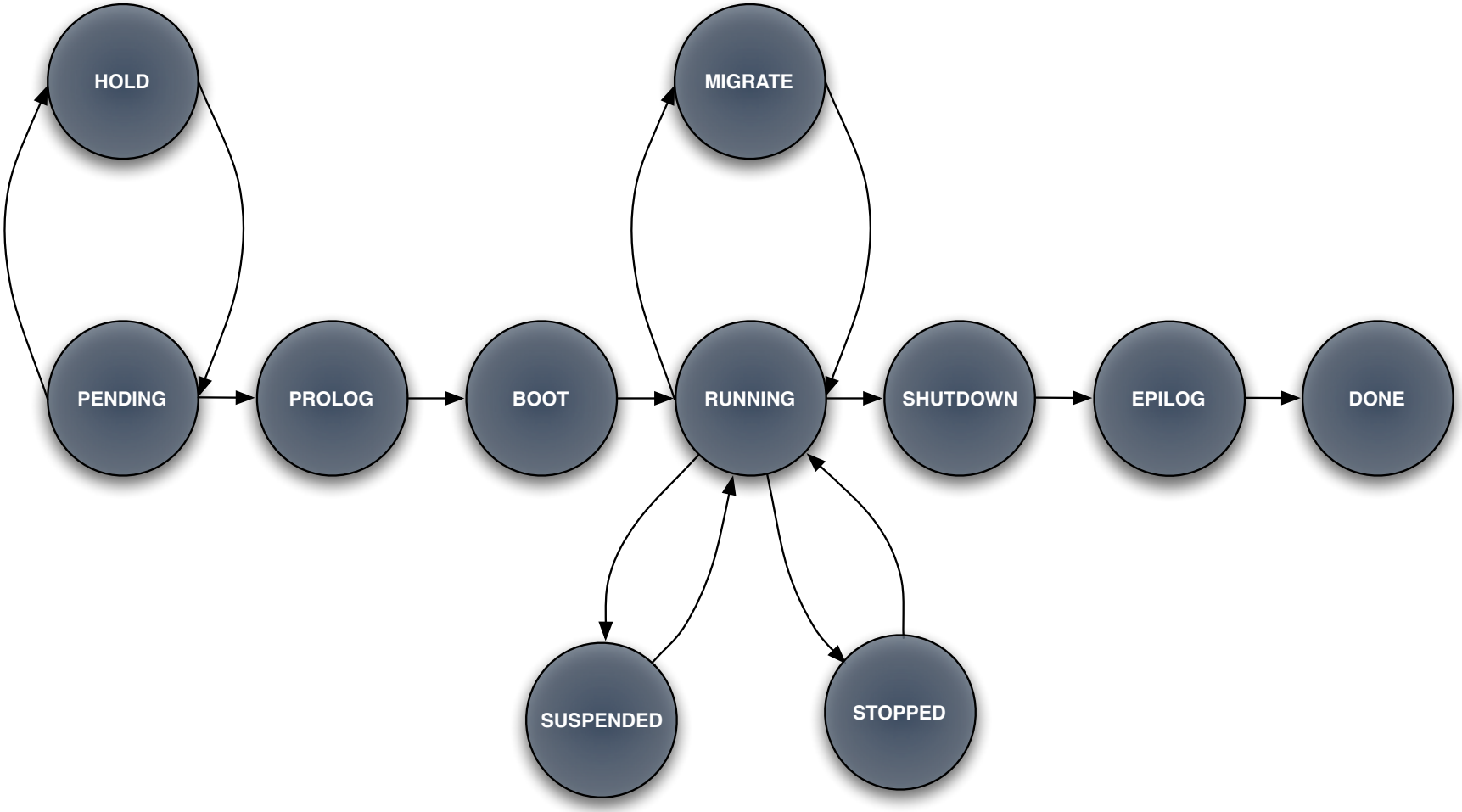
```
NAME  = “Public LAN”
TYPE  = FIXED
BRIDGE= eth1
LEASES= [IP=130.10.0.1,MAC=50:20:20:20:20:20]
LEASES= [IP=130.10.0.2]
```

Network information in VM description

```
NIC = [
  network = “name_of_the_virtual_network”,
  ip      = “ip_address”,
  bridge  = “name_of_bridge_to_bind_if”,
  target  = “device_name_to_map_if”,
  mac     = “HW_address”,
  script  = “path_to_script_to_bring_up_if” ]
```

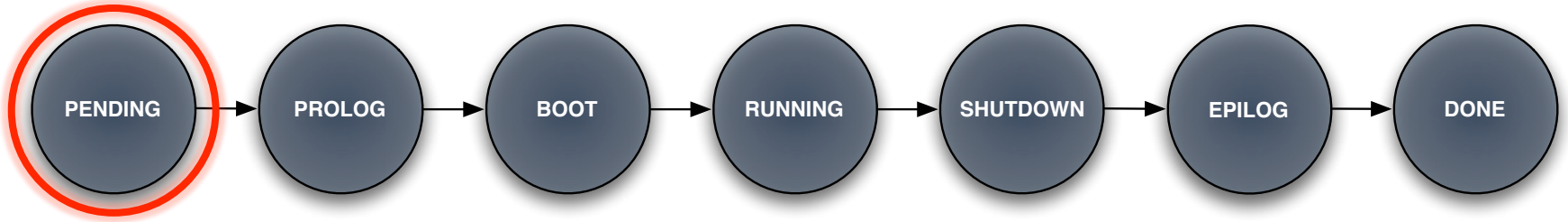
VM Lifecycle

VM lifecycle states



VM Lifecycle

Pending State

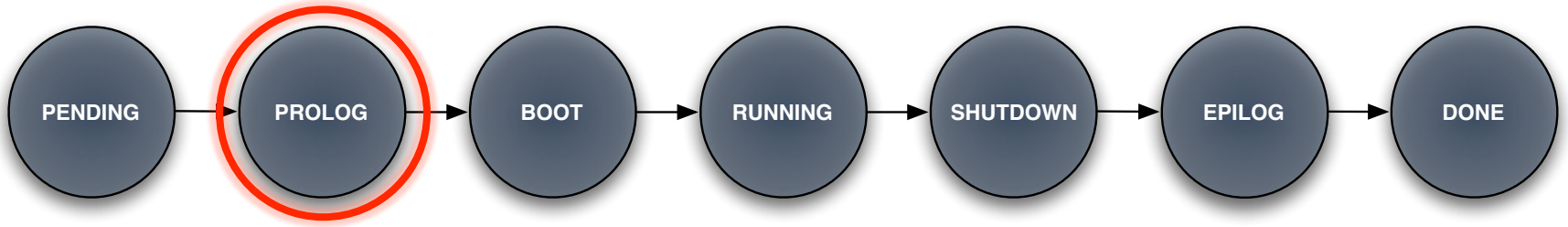


- After submitting a VM description to ONE it is added to the database and its state is set to PENDING.
- In this state IP and MAC addresses are also chosen if they are not explicitly defined.
- The scheduler awakes every 30 seconds and looks for VM descriptions in PENDING state and searches for a physical node that meets its requirements. Then a deploy XML-RPC message is sent to *oned* to make it run in the selected node.
- Deployment can be also made manually using Command Line Interface:

```
⇒onevm deploy <vmid> <hostid>
```

VM Lifecycle

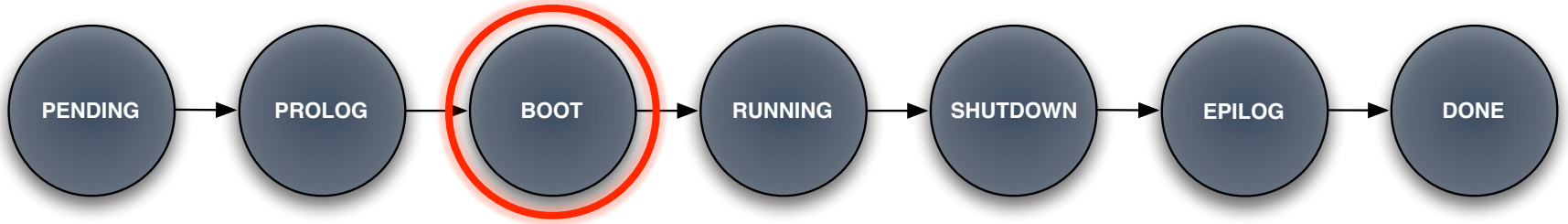
Prolog State



- In PROLOG state the Transfer Driver prepares the images to be used by the VM.
- Transfer actions:
  - **CLONE**: Makes a copy of a disk image file to be used by the VM. If Clone option for that file is set to false and the Transfer Driver is configured for NFS then a symbolic link is created.
  - **MKSWAP**: Creates a swap disk image on the fly to be used by the VM if it is specified in the VM description.

VM Lifecycle

Boot State



- In this state a deployment file specific for the virtualization technology configured for the physical host is generated using the information provided in the VM description file. Then Virtual Machine Driver sends deploy command to the virtual host to start the VM.
- The VM will be in this state until deployment finishes or fails.

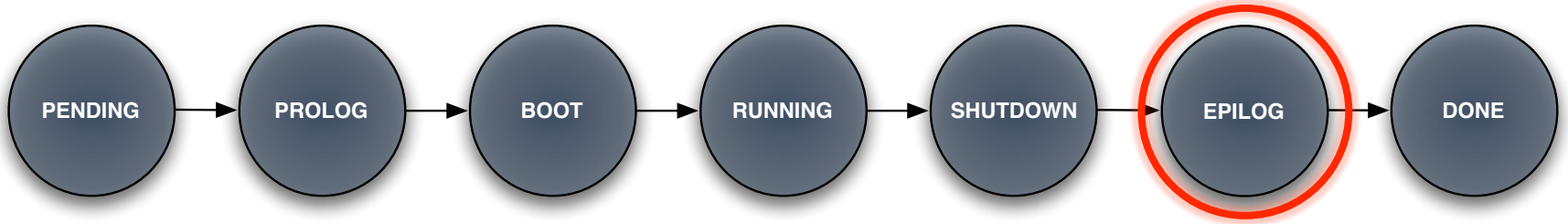
VM Lifecycle

Running and Shutdown States

- While the VM is in RUNNING state it will be periodically polled to get its consumption and state.
- In SHUTDOWN state Virtual Machine Driver will send the shutdown command to the underlying virtual infrastructure.

VM Lifecycle

Epilog State



- In EPILOG state the Transfer Manager Driver is called again to perform this actions:
  - Copy back the images that have ***SAVE=yes*** option.
  - Delete images that were cloned or generated by **MKSWAP**.

Part III

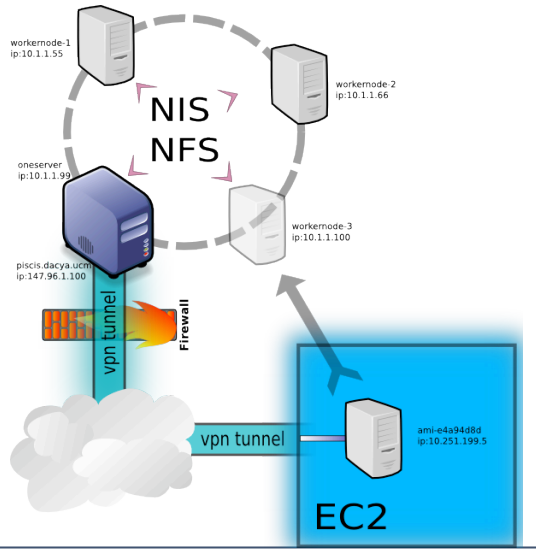
Building a scalable SGE cluster

Use Case Demonstration

Use Cases

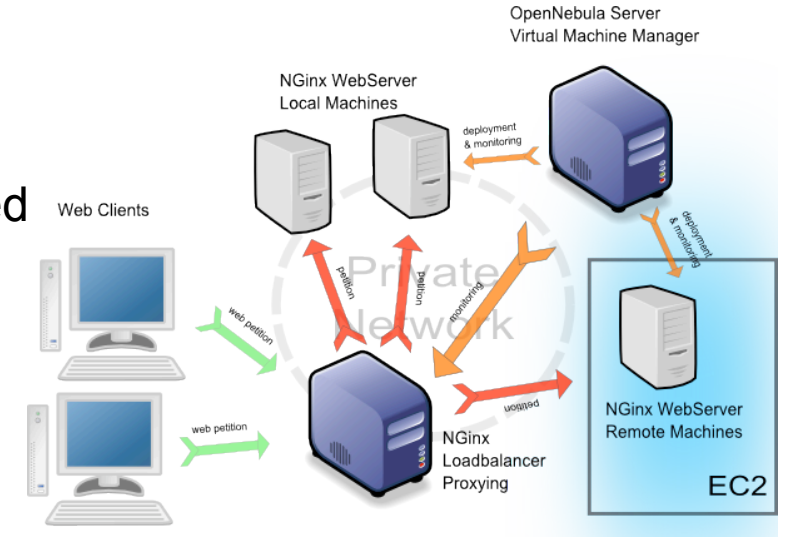
On-demand Scaling of Computing Clusters

- Elastic execution of a SGE computing cluster
- Dynamic growth of the number of worker nodes to meet demands using EC2
- Private network with NIS and NFS
- EC2 worker nodes connect via VPN



On-demand Scaling of Web Servers

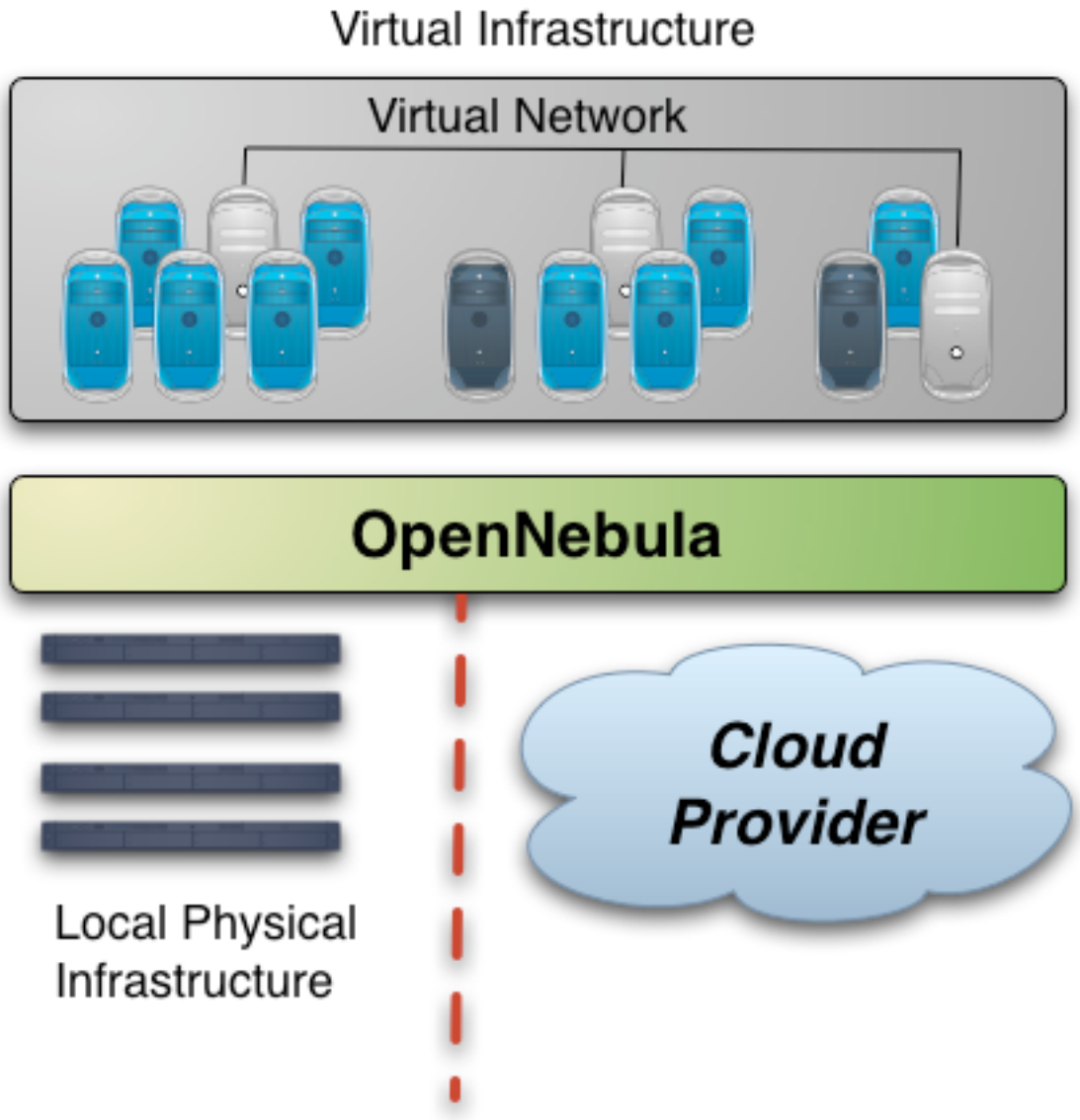
- Elastic execution of the NGinx web server
- The capacity of the elastic web application can be dynamically increased or decreased by adding or removing NGinx instances



Scaling SGE cluster with OpenNebula and EC2

Use Case

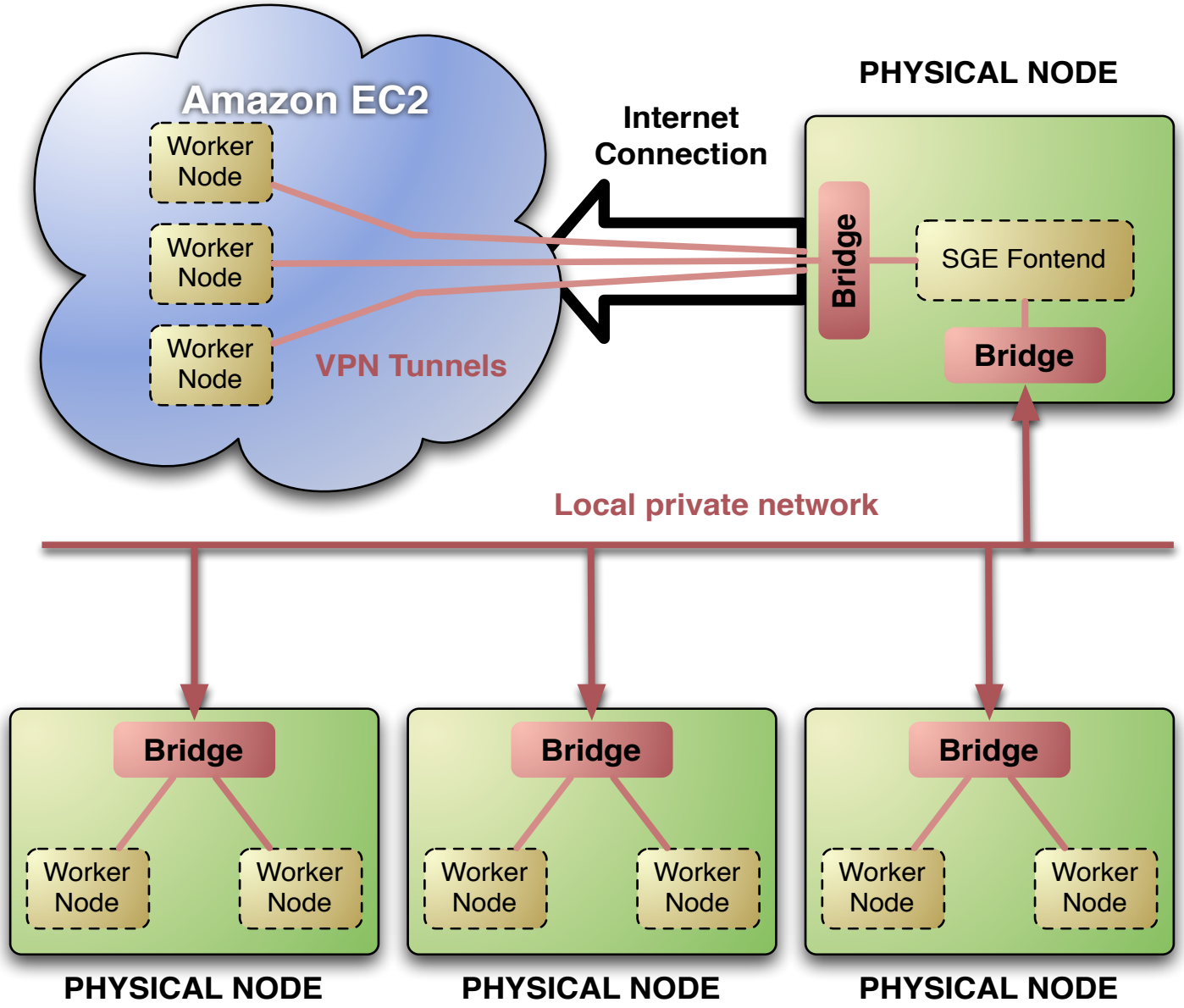
Infrastructure Perspective



Scaling SGE cluster with OpenNebula and EC2

Use Case

Service Perspective

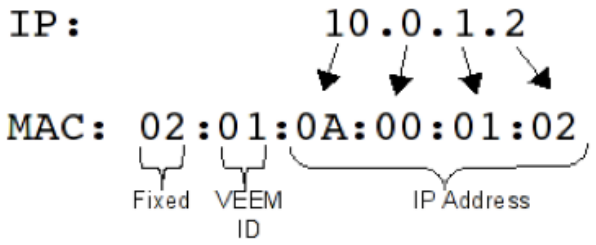


Scaling SGE cluster with OpenNebula and EC2

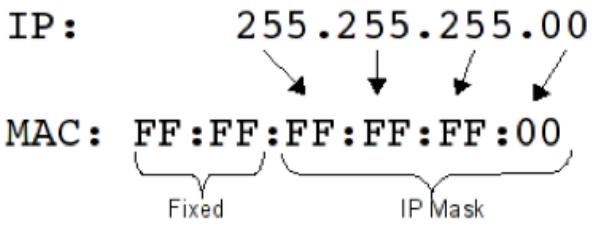
Use Case

Network Scheme

IP-MAC address correspondence



IP-MAC mask correspondence



Contextualization

- VMs need to be configured to execute a script (*vmscript.sh*) at boot time
- Sets the IP based on the MAC of the network interface as it knows the network scheme
- Preliminary version, future OpenNebula versions will support more advanced contextualization
    - Contextualization Virtual Block Devices (VBD)

dsa-research.org

THANK YOU FOR YOUR ATTENTION!!!
More info, downloads, mailing lists at
www.OpenNebula.org

OpenNebula is partially funded by the "RESERVOIR– Resources and Services Virtualization without Barriers" project
EU grant agreement 215605

www.reservoir-fp7.eu/

Acknowledgements

- Ignacio M. Llorente
- Rafael Moreno
- Rubén S. Montero