Karlsruhe Institute of Technology

Steinbuch Centre for Computing

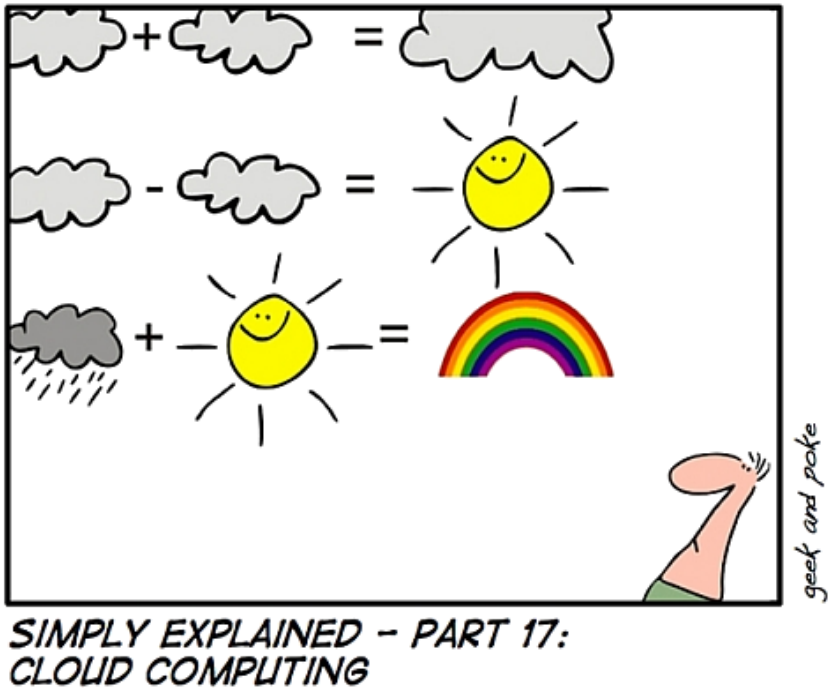
GridKa School 2010
Cloud Computing Workshop

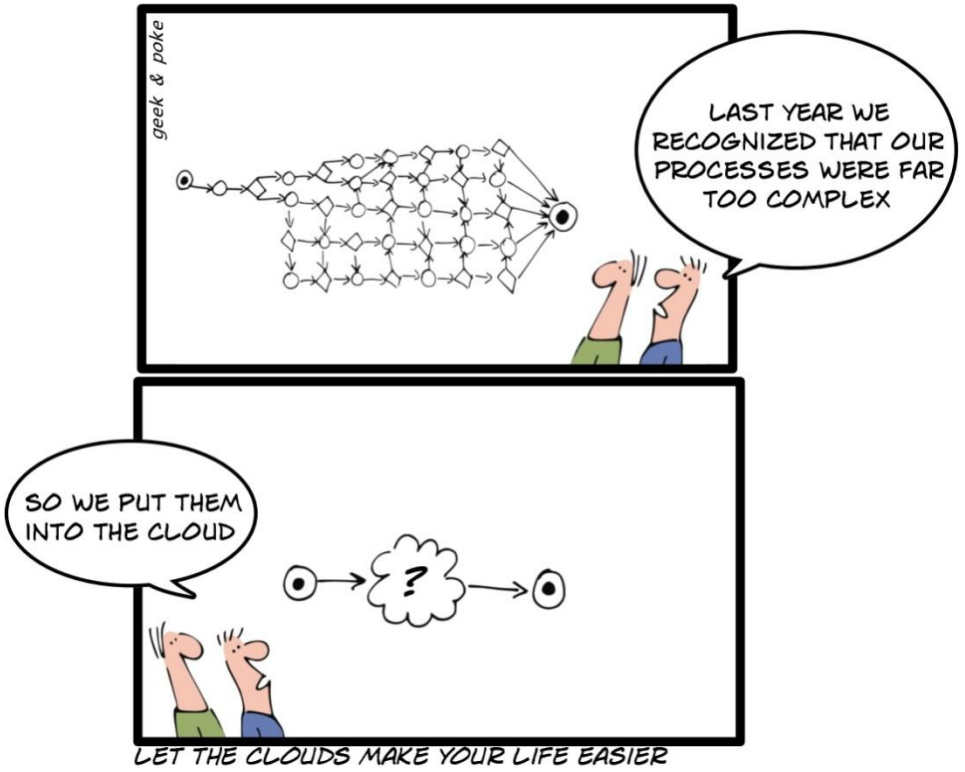
Christian Baun, Matthias Bonn, Thomas Hauth, Marcel Kunze, Tobias Kurze, Viktor Mauch

Steinbuch Centre for Computing (SCC), Research Group Cloud Computing

Contents

- Cloud Computing
  - Concept
  - Everything as a Service (XaaS)
  - Commercial Cloud Providers





What's behind all this??

Definition: Cloud-Computing

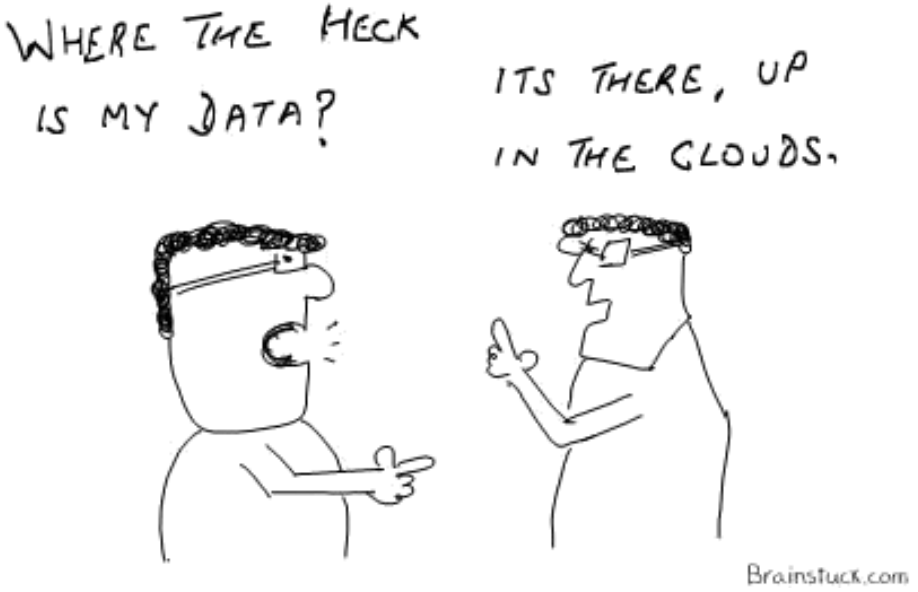
- Definition
  - Building on compute and storage virtualization, and leveraging the modern Web, Cloud Computing provides scalable, network-centric, abstracted IT infrastructure, platforms, and applications as on-demand services that are billed by consumption.

- Transition of IT into the era of industrialization
  - One or few data centers with heterogeneous or homogenous resources under central control
  - Virtualized resources
  - Pay-as-you-go
  - Ease of use
  - Transition of data centers to IT service centers
    - *„Old IT“*: services are created and managed manually
    - *„New IT“*: fully automated services

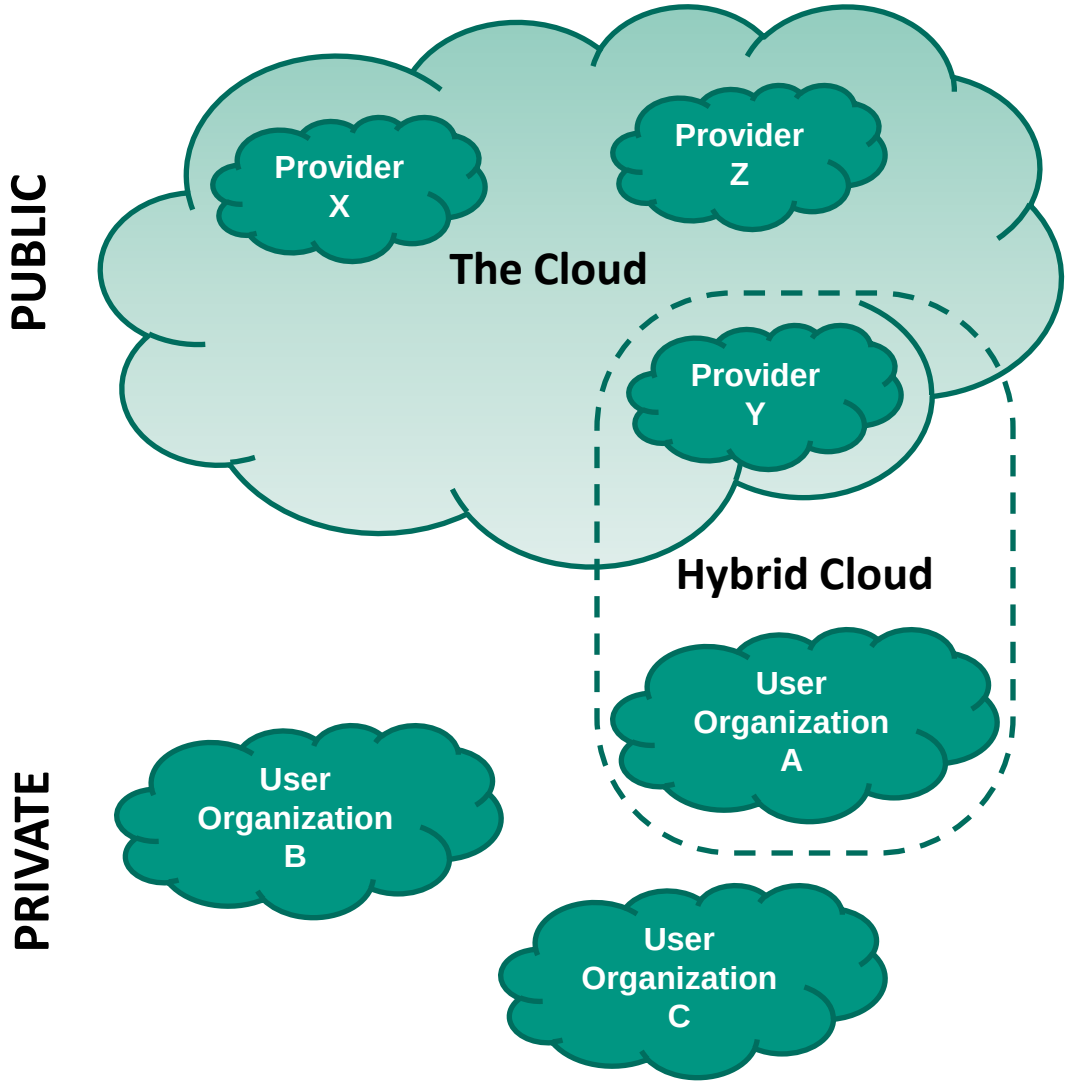
Own hardware?

Reasons to dislike Cloud Computing

- Hardware is somehow *sexy*
  - Lots of hardware looks so important at open house („Tag der offenen Tür“)
  - Loss of hardware means loss of authority
- Users have a bad feeling when important data is stored outside
  - No problem with emails (?!)
- Administrators love their hardware
  - Despite all days and nights full of pain and suffering
  - Stockholm syndrom?!
- Fear for the future
  - In an IT service centre are more management tasks and fewer technical jobs



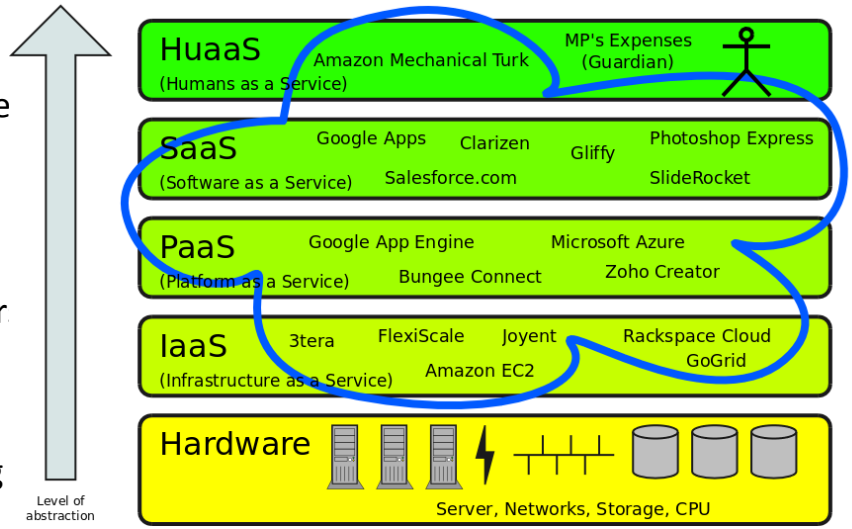
Concept of Cloud Computing – Organisatorical Types



- **Public Cloud**
  - Providers have commercial interests
  - Users have no costs concerning purchase, operation and maintenance of own hardware
  - Critical situation concerning data privacy and security of sensible information
  - Fear for a Lock-in situation!
- **Private Cloud**
  - Providers and users are from the same organization
  - No security or privacy issues
  - Similar operation costs like a non Cloud-based architecture
  - Lock-in situation cannot happen
  - Compatible with the popular public cloud services (in a perfect world!)
- **Hybrid Cloud**
  - Services of private and public clouds are combined to process load peaks or outsource data copies

Everything as a Service (XaaS)

- **1. Layer: Infrastructure as a Service (IaaS)**
  - Users run virtual server instances with optional operations system configurations (restricted by the providers)
  - Administrative user rights
  - Own firewall rules
  - No direct contact to physical hardware for the user
- **2. Layer: Platform as a Service (PaaS)**
  - Scalable running environment and (sometimes) development environment for 1 or 2 programming languages
  - No administrative effort concerning the operation environment
  - More restriction then in IaaS
- **3. Layer: Software as a Service (SaaS)**
  - Applications a run by a provider
  - No need for a local installation at the users site
  - Users do not need to take care about installation, security updates, …
  - Users need to trust the provider concerning the process of their data in the cloud (e.g. E-Mail accounts)



- **4. Layer: Human as a Service (HaaS)**
  - Principle of crowd sourcing
  - Human creativity becomes available as a resource in the cloud
  - Interesting for tasks which are difficult to automate by computers (e.g.: translation, image recognition)

Amazon Web Services (AWS)

http://aws.amazon.com

- Current Situation on the IaaS market
  - Amazon is the market leader with its AWS
  - AWS are a collection of different Cloud services
  - Billing according to consumption
  - Very dynamic development
- Popular services within the AWS are EC2, S3, EBS…

Elastic Compute Cloud (EC2)	Service for virtual servers (instances)
Simple Storage Service (S3)	Service for Web objects
Elastic Block Store (EBS)	Service for persistent data storage volumes
SimpleDB	Distributed database management system
Simple Queue Service (SQS)	Service for Message Queues
Elastic Load Balancing (ELB)	Service for Load Balancer to distribute traffic to different EC2 instances
Mechanical Turk	Market place for HuaaS/Crowdsourcing

Commercial Cloud Service Providers (small selection)

- Besides the AWS, lots of well-established public cloud service offers exist
- Commercial Cloud Systems are often proprietary and not all aspects of their architecture are open
  - Constitution of own private cloud IaaS or PaaS is not always possible
  - Construction of hybrid clouds is even more difficult

Overview of Private Cloud PaaS Frameworks

- Only few private cloud PaaS solutions available
- Number of available solutions is shorter than it appears in the first sight

10gen	http://www.10gen.com
Reasonably Smart	http://reasonablysmart.com
AppScale	http://appscale.cs.ucsb.edu
typhoonAE	http://code.google.com/p/typhoonae/

Overview of Private Cloud IaaS Frameworks

- Lots of Private Cloud IaaS solutions available at first sight
  - All of them are Open Source!
- Already used in science projects
  - CERN builds an Cloud Environment with OpenNebula with the goal to manage up to 45,000 Virtual Machine instances

Cloud.com CloudStack	http://cloud.com
abiCloud	http://www.abicloud.org
OpenNebula	http://www.opennebula.org
Nimbus	http://www.nimbusproject.org
Tashi	http://www.pittsburgh.intel-research.net/projects/tashi/
Enomaly ECP	http://src.enomaly.com
OpenECP	http://www.openecp.org
Eucalyptus	http://open.eucalyptus.com

Private Cloud IaaS im Detail (3)

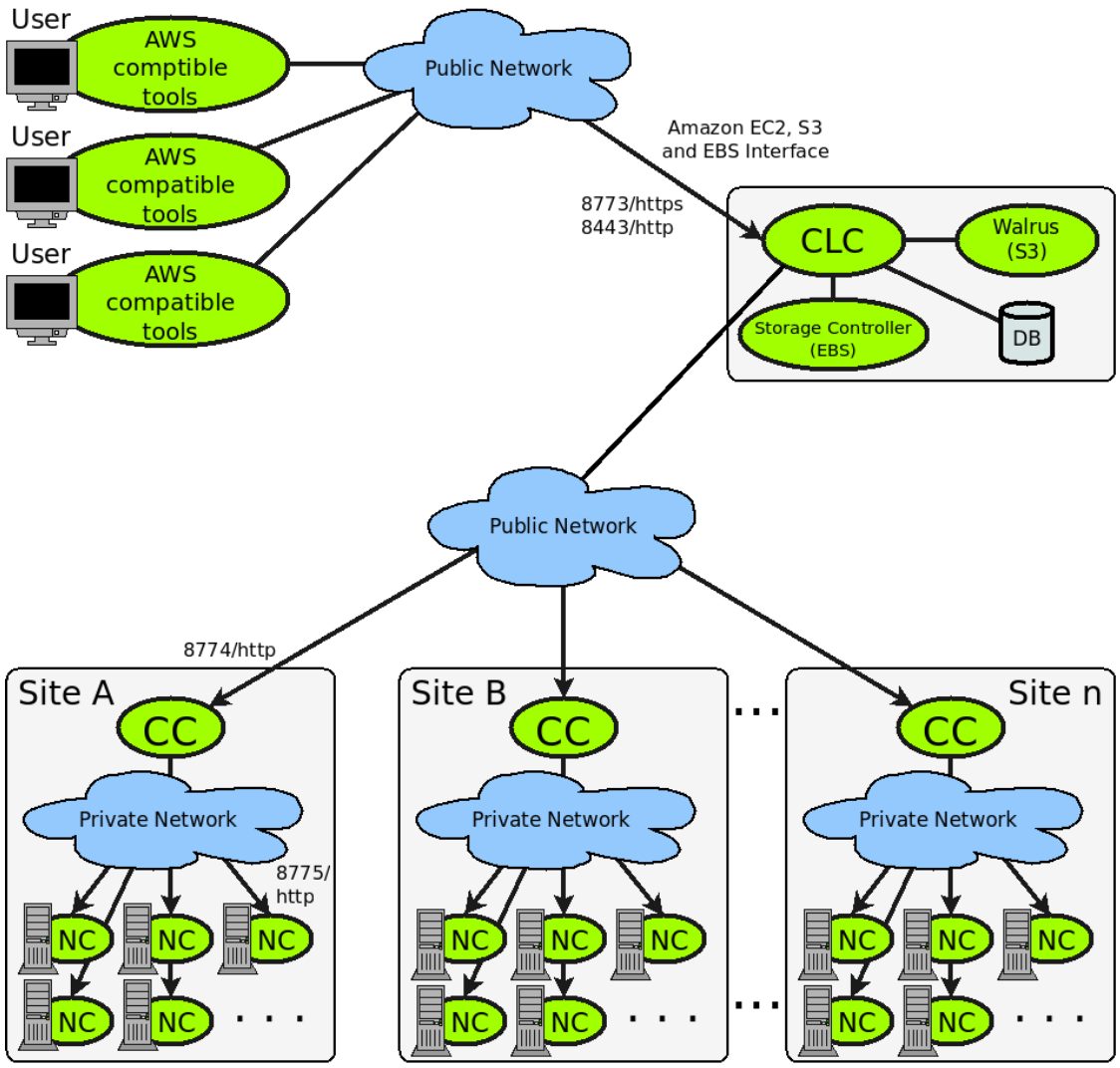
- Nimbus
  - Build on top of the Grid middleware Globus 4
  - EC2 API implemented partly
    - describe images
    - describe, run, reboot und terminate instances
    - add und delete keypair
  - EC2-compatible resources can be used via remote (=> Hybrid Cloud)

- Eucalyptus
  - One of the most popular private cloud IaaS solutions
  - May 2008: Version 1.0
  - EUCALYPTUS - Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems
  - Emulates the most popular AWS services
    - API compatible to Amazon EC2
    - Includes „Walrus“, a S3 compatible storage service
    - Includes „Storage Controller“, an EBS compatible storage service

Eucalyptus — Components

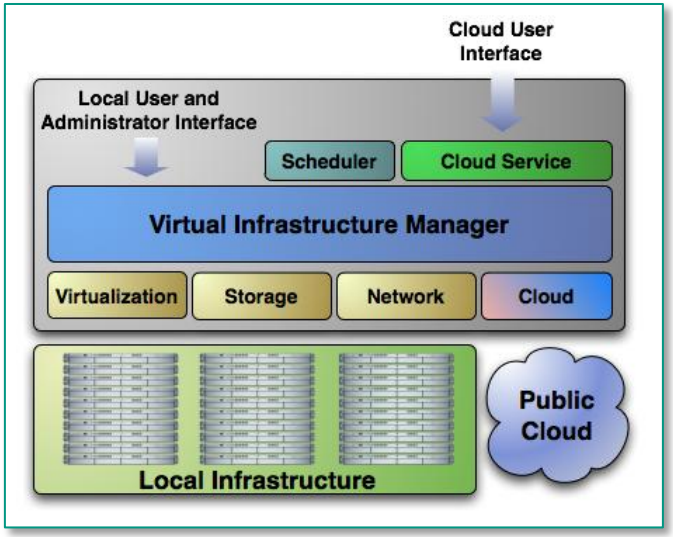
http://open.eucalyptus.com

- Cloud Controller (CLC)
  - Operates like a meta scheduler in the Cloud
  - Collects resource information from the CCs
- Cluster Controller (CC)
  - Schedules the distribution of virtual machines to the NCs
  - Collects free resource information from the NCs
- Node Controller (NC)
  - Runs on every worker node in the cloud
  - Xen hypervisor or KVM running
  - Provides resource information to the CC
- Walrus
- Storage Controller



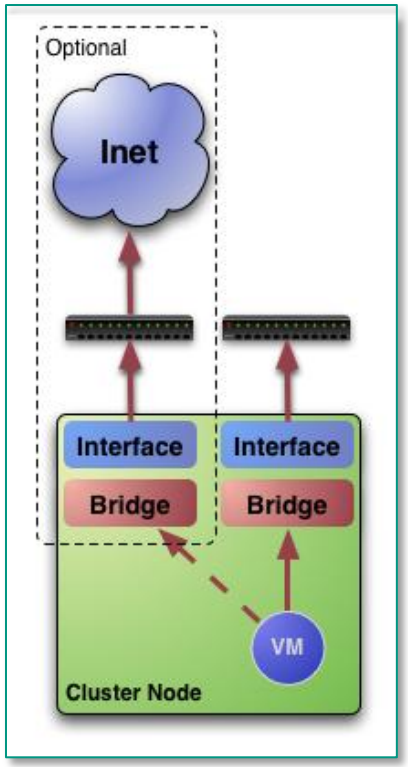
OpenNebula – Introduction

- OpenNebula is an open-source toolkit to easily build any type of cloud: **private**, **public** and **hybrid**.
- OpenNebula supports **KVM, Xen and VMware**
- OpenNebula has been designed to be integrated with any networking and storage solution and so to fit into any existing data center.
- Primary Objective: Efficient Management of VM Instances
  - CERN Cloud Instance: ~ 7.500 VMs on 400 cluster nodes; Future: more than 40.000 VMs
- Only a small part of the EC2 API implemented since OpenNebula 2.0 Beta1
  - describe images
  - describe, run, reboot und terminate instances
- Trivial architecture
  - Easy to implement additional features
  - Easy to debug because of central log data
- Nodes can be grouped, Important for HPCaaS and network latency (e.g. MPI)
- No storage service included



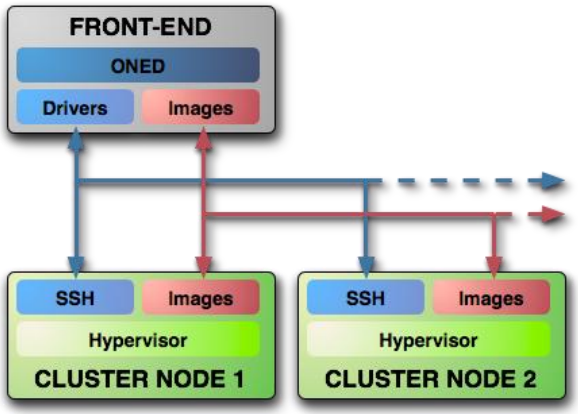
OpenNebula – Structure Notes

- Installation:
  - Documentation available for Ubuntu, CentOS, Debian, OpenSUSE, MacOS, …
    see: http://opennebula.org/documentation:documentation
- Structure:
  - Separation in Front-End and Cluster Nodes
  - Communication based on SSH (password-less login via SSH keys) and Ruby scripts
  - Front-End uses the **libvirt library** to control the Hypervisor on the Cluster Nodes via SSH
  - To provide one or more physical networks for the VMs, the cluster nodes have to be set up with Ethernet Bridges
- Two operation methods for VM Deployment:
  - via SSH
    - Images are copied via SSH to the Cluster Node partitions
  - on a Shared File System
    - Live Migration is possible
    - FS should be performant enough to manage high I/O -> SAN mount

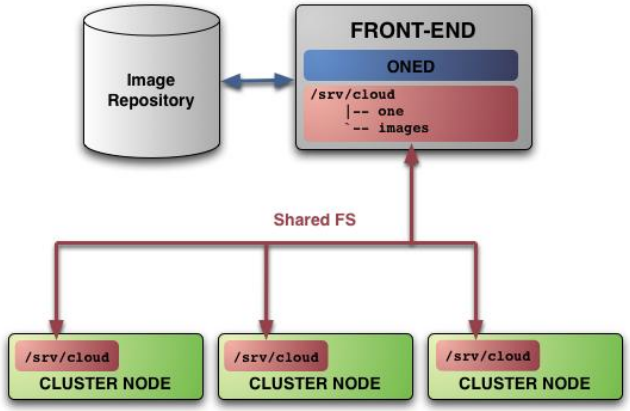


OpenNebula – Private Cloud Tutorial Instance

- 7x Dell Blades - Dual Intel Xeon Quad Core 2,66 GHz / 16 GB Ram:
  1 Front-End + 6 Cluster Nodes (48 Cores)
- Connection: 1 Gigabit Ethernet
- **Image Deployment via SSH**
- Based on Ubuntu 10.04 LTS Server
- Virtualization Technology: **KVM** Hypervisor
- Version: OpenNebula 2.0 Beta1
- Installation can be found under `/srv/cloud/one` on the front-end



Deployment via SSH



Shared File System

Exploring the Private Cloud

- **Hands on…** explore the Cloud with some basic OpenNebula commands:

Cluster Node Management: **onehost <list top show create delete enable disable …>**
Check out how many cluster nodes are available with **onehost list**.
Explore the details of one cluster node with **onehost show *host_id***

Virtual Network Management: **onevnet <list show create delete …>**
Check out which virtual networks are available with **onevnet list**.
Explore the details of one virtual network with **onevnet show *vnet_id***

Virtual Machine Management: **onevm <create delete migrate supend resume …>**
Check out how many virtual machines are running with **onevm list** or **onevm top**.
Explore the details of one virtual machine with **onevm show *vm_id***

Image Management: **oneimage <list show ...>**
Check out how many images are available with **oneimage list**
Explore the details of one image with **oneimage show image_*id***

Cloud User Management: **oneuser <create delete list>**
Only available for the cloud admin to create and delete cloud users.

Virtual Networks I

- A Virtual Network in OpenNebula
  - Defines a MAC/IP address space to be used by VMs
  - Each Virtual Network is associated with a physical network through a bridge
- Virtual Network definition
  - **Name** of the Network
  - **Type**
    - **Fixed**, a set of IP/MAC leases
    - **Ranged**, defines a network range
  - **Bridge**, name of the physical bridge in the physical host where the VM should connect its network interface

```
# Ranged VNET template file
NAME                = "Red LAN"
TYPE                = RANGED
BRIDGE              = eth0
NETWORK_SIZE        = C
NETWORK_ADDRESS     = 192.168.169.0
```

```
# Fixed VNET template file
NAME                = "Blue LAN"
TYPE                = FIXED
BRIDGE              = br0
LEASES              = [IP=192.168.170.11]
LEASES              = [IP=192.168.170.12]
LEASES              = [IP=192.168.170.13]
```

- **Hands on…** create your own fixed Virtual Network with two IPs.

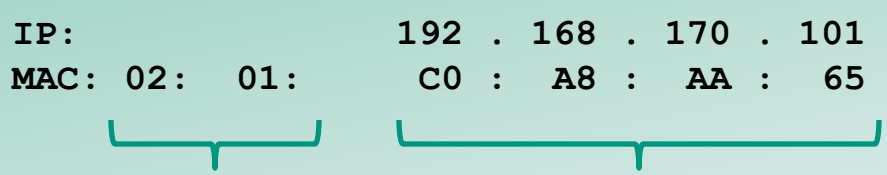
Virtual Networks II

- How to use a Virtual Network with your VMs
  - Define NICs attached to a given virtual network. The VM will get a NIC with a free MAC address in the network and attached to the corresponding bridge

```
#A VM with two interfaces each one in a different vlan
NIC      = [NETWORK="Blue LAN"]
NIC      = [NETWORK="Red LAN"]

#Ask for a specific IP/MAC
NIC      = [NETWORK="Blue LAN", IP = 192.168.0.11]
```

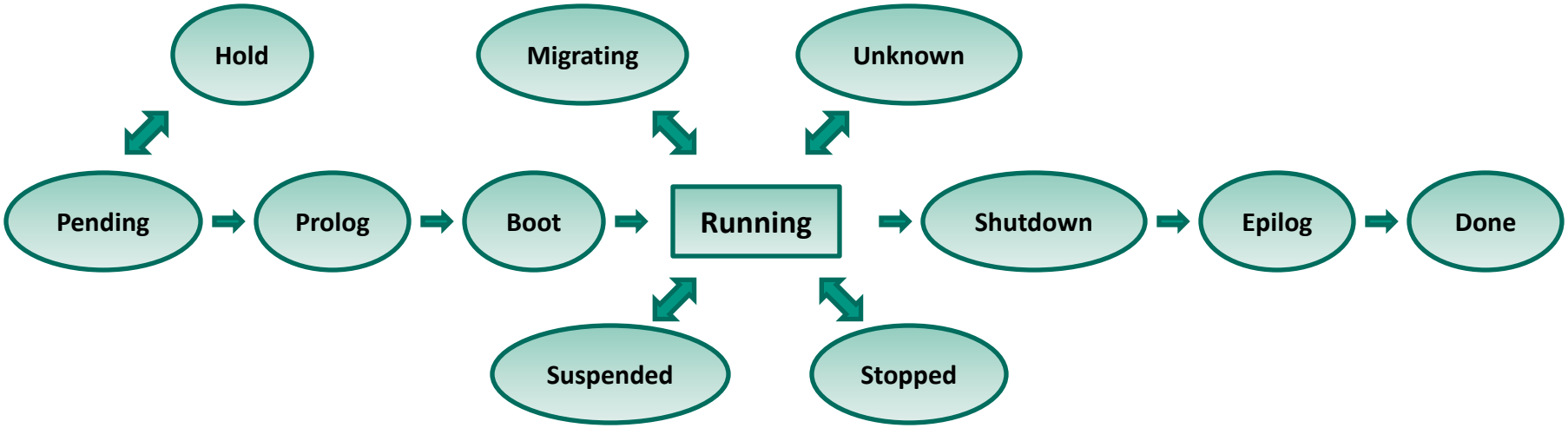
  - Prepare the VM to use the IP. Sample scripts to set the IP based on the MAC are provided for several Linux distributions.

IP-MAC address correspondence



Virtual Machines I

- Preparing a VM to be used with OpenNebula
  - You can use any VM prepared for the target hypervisor
  - Prepare master images: Install once and deploy many;
  - Do not put private information (e.g. ssh keys) in the master images, instead use `CONTEXT` (see later)
  - Pass arbitrary data to a master image using `CONTEXT`
- Virtual Machine Life-cycle:



Virtual Machines II

- Virtual Machines are defined in a VM template file
- Each VM has an unique ID in OpenNebula, the **VM_ID**
- All log files are stored in `/srv/cloud/one/var/<VM_ID>` on the head node
- The images will be copied via a SSH connect to the cluster nodes

- A Virtual Machine template in OpenNebula consists of
  - a **capacity** section in terms of name, memory and cpu
  - a set of **NICs** attached to one or more virtual networks
  - a set of **disk images**, to be "transferred" to/from the execution host
  - …

Virtual Machine Definition File (VM template) I

```
#---------------------------------------------------------
# Capacity Section
#---------------------------------------------------------
NAME      = "vm-example"
CPU       = "percentage of CPU divided by 100 required for the Virtual Machine"
MEMORY    = "amount of requestet MEM"
VCPU      = "number of virtual cpus"

#---------------------------------------------------------
# OS and boot options
#---------------------------------------------------------
OS        = [
          kernel              = "path_to_os_kernel",        # para-virtualization
          initrd              = "path_to_initrd_image",     # para-virtualization
          kernel_cmd          = "kernel_command_line",
          root                = "device to be mounted as root",
          bootloader          = "path to the boot loader exec",
          boot                = "device to boot from" ]

#---------------------------------------------------------
# Features of the hypervisor
#---------------------------------------------------------
FEATURES  = [
          pae       = "yes|no",         # optional, KVM
          acpi      = "yes|no" ]        # optional, KVM
```

Virtual Machine Definition File (VM template) II

```
#---------------------------------------------------------
# VM Disks
#---------------------------------------------------------
DISK      = [
          type      = "image|floppy|disk|cdrom|swap|fs|block",
          source    = "path_to_disk_image_file|physical_dev",
          format    = "type for fs disks",
          size      = "size_in_GB",
          target    = "device_to_map_disk",
          bus       = "ide|scsi|virtio|xen",
          readonly  = "yes|no",
          clone     = "yes|no",
          save      = "yes|no" ]

#---------------------------------------------------------
# Network Interface
#---------------------------------------------------------
NIC       = [
          network   = "name_of_the_virtual_network",
          target    = "device_name_to_map_if",
          ip        = "ip_address",
          bridge    = "name_of_bridge_to_bind_if",
          mac       = "HW_address",
          script    = "path_to_script_to_bring_up_if",
          model     = "NIC model" ]
```

Virtual Machine Definition File (VM template) III

```
#-------------------------------------------------------------
# I/O Interfaces
#-------------------------------------------------------------
INPUT     = [
          type      = "mouse|tablet",
          bus       = "usb|ps2|xen" ]
GRAPHICS  = [
          type      = "vnc|sdl",
          listen    = "IP_to_listen_on",
          port      = "port_for_VNC_server",
          passwd    = "password_for_VNC_server" ]

#-------------------------------------------------------------
# RAW Hypervisor attributes
#-------------------------------------------------------------
RAW       = [
          type      = "xen|kvm",
          data      = "raw_domain_configuration" ]
#-------------------------------------------------------------
# CONTEXT Section used for Customization of VMs
#-------------------------------------------------------------
CONTEXT   = [ ... ]           # see later
```

Complete reference and examples for all sections:
http://opennebula.org/doku.php?id=documentation:rel1.4:template

Submitting & Management of VMs

- **Hands on…** define a minimal VM template and create your first VM:

```
#---------------------------------------------------------------
# VM template for the ubuntu image: "ubuntu-lucid"
#---------------------------------------------------------------
NAME        = "my_VM"                                  # define a name for your V
MEMORY      = 512
VCPU        = 2

DISK        = [ image              = "ubuntu-lucid" ]
NIC         = [ NETWORK            = "my_VNET" ]      # enter here your created vnet
```

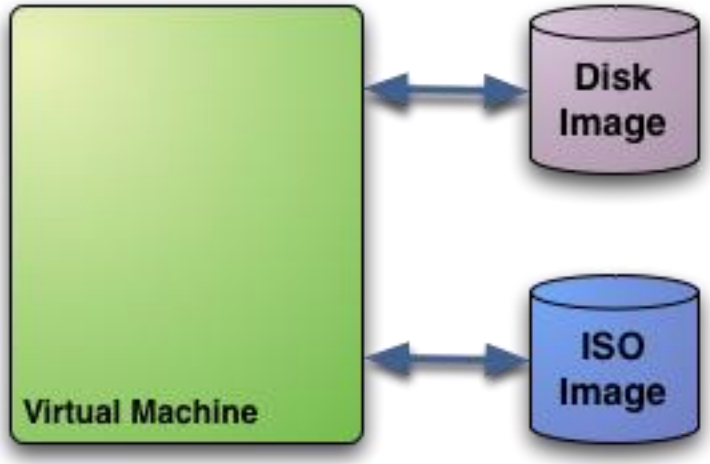
- Submit your VM: **onevm create *vm_template_file***
- Monitor the status for your VM: **onevm top**
- Get detailed information, (e.g. IP): **onevm show *VM_ID***
- Try to login (User: "ubuntu", PW: "ubuntu"): **ssh ubuntu@*VM_IP***
- Take a look to the script file "**/etc/init.d/vmcontext**" which is part of the boot procedure and try to understand how the network will be configured
- Try to perform some VM operation: **onevm <migrate|suspend|resume|delete|…>**
- Optional: Modify the template:
  add another DISK, e.g.: `type="fs", format="ext2", size="100", target="hdb"` and try to mount it in the VM

Customization of VMs

ONE provides a method to modify created VMs. The master image **ubuntu-lucid** is already preconfigured to support the CONTEXT Block:

- The ISO Image will be mounted under **/mnt/context**
- The **init.sh** script will be executed with root privileges
- Afterwards the ISO Image will be un-mounted

```
# VM template
...
CONTEXT = [
  files   = "/path/init.sh /path/id_rsa.pub",
  target  = "hdc",
  host    = "myHostname",
  dns1    = "192.168.42.42",
  … ]
```



context.sh
id_rsa.pub
init.sh

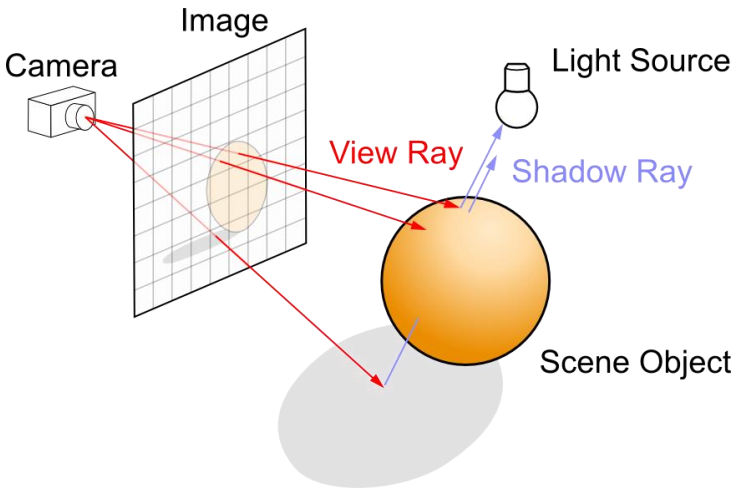
```
#context.sh (created by ONE)
HOST       = "myHostname"
DNS1       = "192.168.42.42"
…
```

```
#init.sh
. /mnt/context/context.sh
hostname $HOST

echo "nameserver $DNS1" > \
    /etc/resolv.conf

cat /mnt/context/id_rsa.pub >> \
    /root/.ssh/authorized_keys
```

- **Hands on…** define a VM template for the Ubuntu Image and try to use the CONTEXT Block (see Handout).

Performing some Rendering Jobs

- high quality pictures can be rendered using ray tracing
- the rendering can be done in parallel
  - regions of a picture
  - singe frames of an animation
- POV-Ray is open source

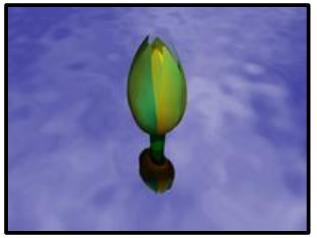


- **Hands on…** define a new CONTEXT section for the Ubuntu Image to perform a rendering job. Divide the complete rendering procedure of the pictures in 2 parts:
  - First VM: 0..50
  - Second VM: 51..99

  See handout!!

ray.pov

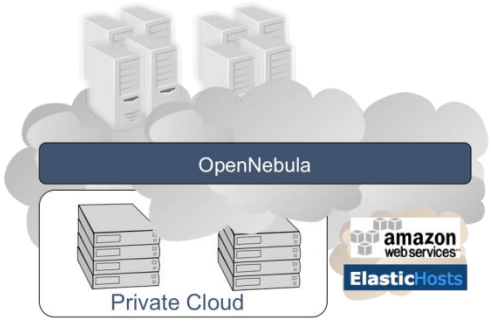
votex.pov

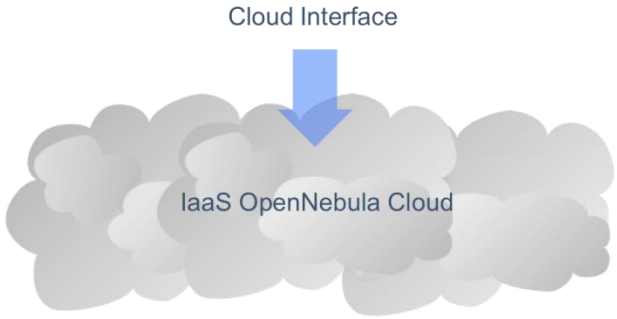


flower.pov

Further Feature

- Hybrid Cloud:
  - Provides the possibility to control AWS / ElasticHosts resources with the same basic ONE commands
  - Creates a simple abstraction layer over the EC2-API-Tools
  - However there is no simple way to deploy own images to AWS / ElasticHosts





- Public Cloud:
  - Extension of a Private Cloud to expose RESTful Cloud interfaces
  - Can be added to you Private or Hybrid Cloud if you want to provide partners or external users with acces to your infrastructure

- EC2 Compatible Management:
  - Since ONE 2.0Beta1 there is the possibility to control ONE resources via EC2 compatible GUI tools, like
    - HybridFox / ElasticFox (Firefox Plug-Ins)
    - KOALA (PaaS Browser Service- http://koalacloud.appspot.com/)

Thank You!

- Links:
  - OpenNebula Website: http://opennebula.org/
  - 2-days Tutorial with detailed information concerning Installation of OpenNebula 1.4 / Hybrid Cloud / Public Cloud by ***Ruben S.Montero, University Complutense of Madrid***: http://dl.dropbox.com/u/4497643/buildingcloudsone1-4.pdf