CHEP 2013 - October 14-18, 2013 - Amsterdam

Integrating multiple scientific computing needs via a Private Cloud Infrastructure

Stefano Bagnasco
Dario Berzano
Stefano Lusso
Sara Vallero

The present work is partially funded under contract 20108T4XTM of Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale (Italy).

INFN
Istituto Nazionale
di Fisica Nucleare

SIGIL · VNIVERS · AVGVSTAE · TAVRINORVM

The INFN Torino Computing Centre

STORAGE RESOURCES

- 552 SATA/SAS 7.2K disks (1-3 TB each) → 1060 TB (gross) total

COMPUTATIONAL RESOURCES

- 28 physical nodes: 16GB RAM, 8 job-slots each
- 38 Intel/AMD hypervisors (KVM virtualizer): 64GB RAM, 16 job-slots each

SERVICES RESOURCES

- 43 servers, the newest with:
  - 2 Intel 6-core processors / 48GB RAM
  - 2 SAS 10K mirrored disks

LOCAL AREA NETWORK

- 140 1Gbps + 22 10Gbps links

WIDE AREA NETWORK

- 1Gbps (soon 10Gbps)

Stakeholders

PROVISIONED

- WLCG Tier2
  (primarily for ALICE)
- PANDA computing
- Belle-2 computing
- Virtual Analysis Facility for ALICE
  (interactive analysis based on PoD)
- Medical Image Processing
  (local research group)
- Theory (local research group)
- Virtual farm on-demand

Stakeholders

PROVISIONED

- WLCG Tier2
  (primarily for ALICE)
- PANDA computing
- Belle-2 computing
- Virtual Analysis Facility for ALICE
  (interactive analysis based on PoD)
- Medical Image Processing
  (local research group)
- Theory (local research group)
- Virtual farm on-demand

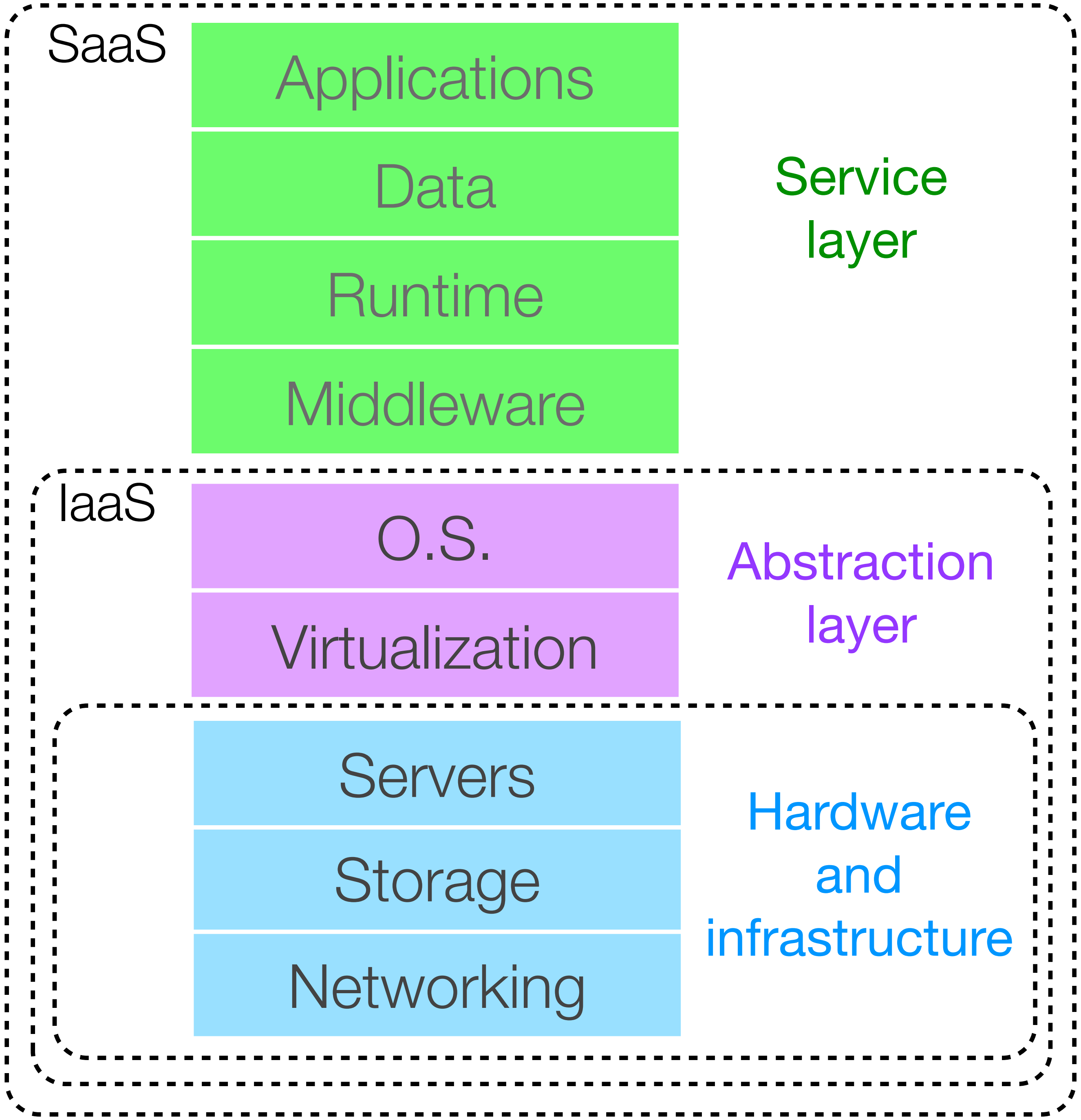
COMING SOON...

- BESIII grid Tier2
  (test infrastructure ready)
- WLCG grid “Tier3” for CMS

Software/Infrastructure as a Service (SaaS/IaaS)

Resources/applications rapidly increasing → manpower is not!

Consolidate available resources to achieve scalability and economies of scale.



SaaS

- Applications
- Data
- Runtime
- Middleware

Service layer

IaaS

- O.S.
- Virtualization

Abstraction layer

- Servers
- Storage
- Networking

Hardware and infrastructure

Software/Infrastructure as a Service (SaaS/IaaS)

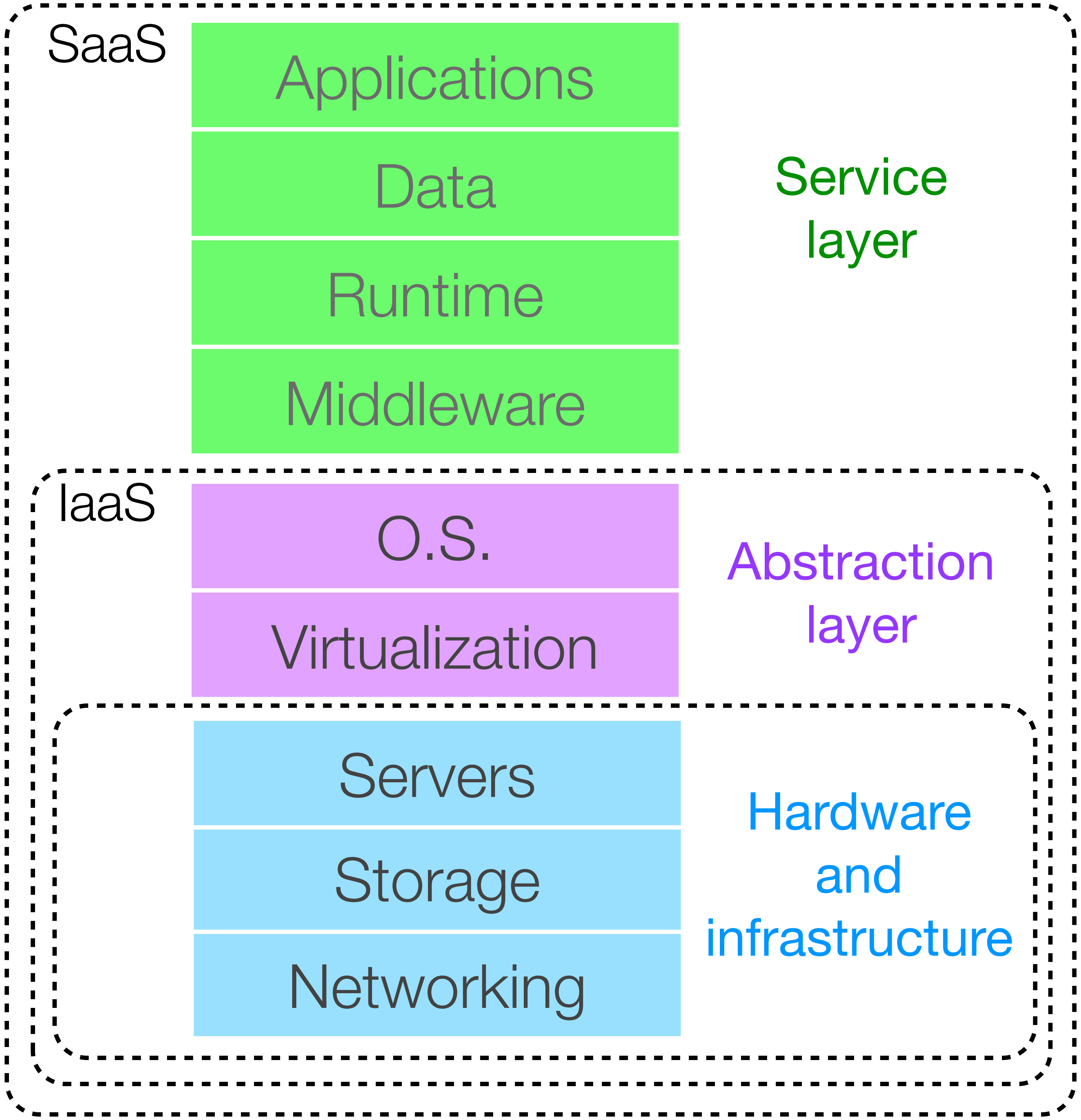
Resources/applications rapidly increasing → manpower is not!

Consolidate available resources to achieve scalability and economies of scale.

Separate APPLICATION and INFRASTRUCTURE management.

Data-centers become providers of:

- computing resources and storage
- higher-level services



SaaS

Applications

Data

Runtime

Middleware

Service layer

IaaS

O.S.

Virtualization

Abstraction layer

Servers

Storage

Networking

Hardware and infrastructure

Software/Infrastructure as a Service (SaaS/IaaS)

Resources/applications rapidly increasing → manpower is not!

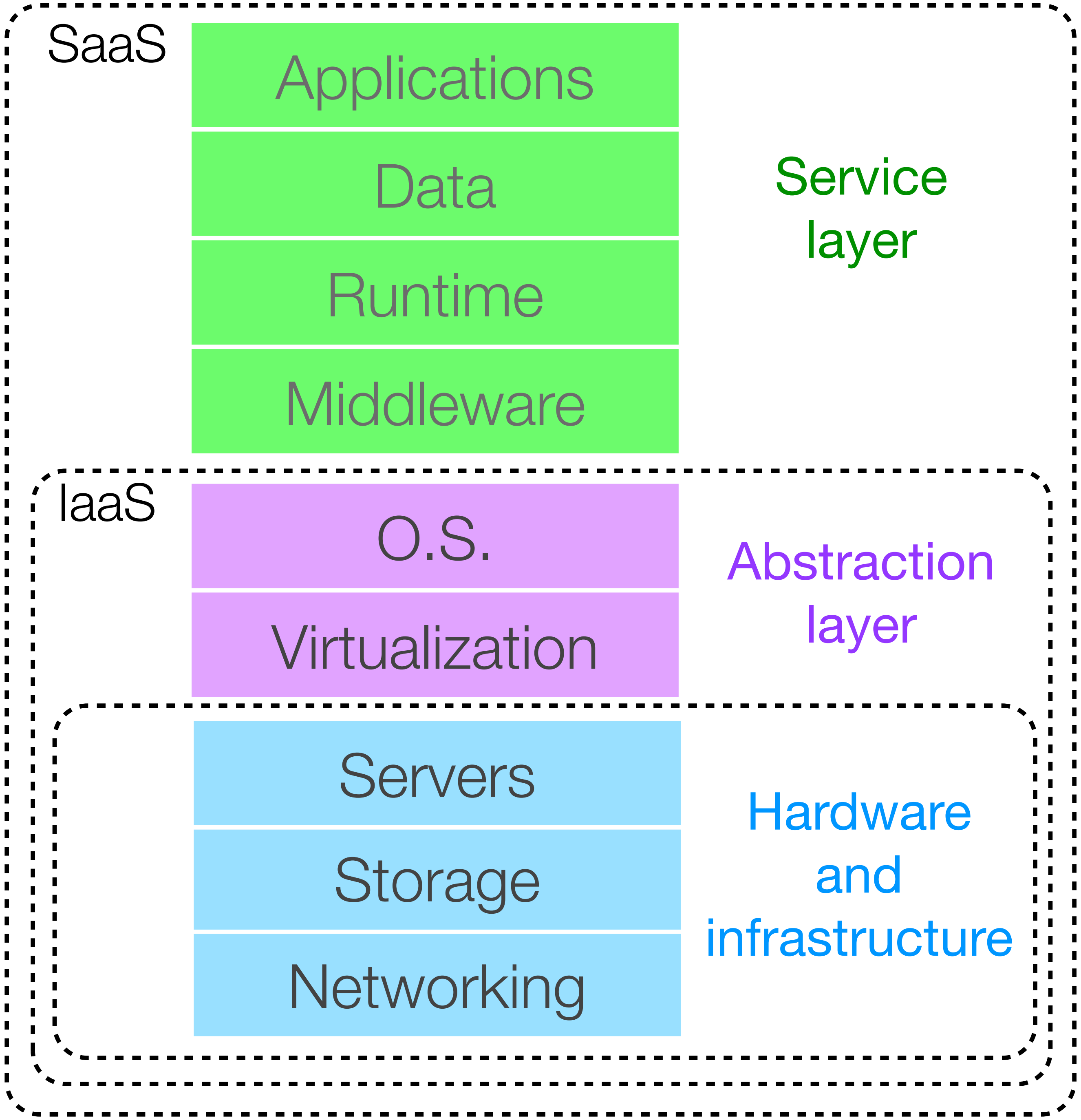
Consolidate available resources
to achieve scalability and
economies of scale.

Separate APPLICATION and
INFRASTRUCTURE management.

Data-centers become providers of:

- computing resources and storage
- higher-level services

→ the underlying paradigm
(cloud computing) is transparent
to the consumer



SaaS

Applications

Data

Runtime

Middleware

Service
layer

IaaS

O.S.

Virtualization

Abstraction
layer

Servers

Storage

Networking

Hardware
and
infrastructure

A Private Cloud

Cloud-computing is a resources management strategy which allows:

- on-demand network access to configurable resources
- pooling of resources to serve multiple consumers using a multi-tenant model
- elastic provisioning or releasing of capabilities to scale with demand

... these goals are naturally achieved with virtualization.

↓

PROOF as a Service on the Cloud
D. Berzano - Tuesday 4:51 PM

A Private Cloud

Cloud-computing is a resources management strategy which allows:

- on-demand network access to configurable resources
- pooling of resources to serve multiple consumers using a multi-tenant model
- elastic provisioning or releasing of capabilities to scale with demand

... these goals are naturally achieved with virtualization.

PROOF as a Service on the Cloud
D. Berzano - Tuesday 4:51 PM

The INFN Torino cloud is **private**:

resources allocation is operated solely by the organization itself.

A Private Cloud

Cloud-computing is a resources management strategy which allows:

- on-demand network access to configurable resources
- pooling of resources to serve multiple consumers using a multi-tenant model
- elastic provisioning or releasing of capabilities to scale with demand

... these goals are naturally achieved with virtualization.

PROOF as a Service on the Cloud
D. Berzano - Tuesday 4:51 PM

The INFN Torino cloud is **private**:

resources allocation is operated solely by the organization itself.

Towards an **hybrid** cloud:

INFN and Italian HTC community are moving towards a national cloud infrastructure

→ R&D ongoing to develop the technical tools to expose the infrastructure to standard interfaces

Our philosophy

Our philosophy

KEEP IT SIMPLE

- manageability and flexibility over performance
- don’t use too many tools
- simple images + contextualization (shell-script+puppet)

Our philosophy

KEEP IT SIMPLE

- manageability and flexibility over performance
- don’t use too many tools
- simple images + contextualization (shell-script+puppet)

STAY MAINSTREAM

- use stable and widely-used tools:
  - **OpenNebula** cloud controller
  - **GlusterFS** distrib. filesystem
  - **OpenWRT** for network management

Our philosophy

KEEP IT SIMPLE

- manageability and flexibility over performance
- don’t use too many tools
- simple images + contextualization (shell-script+puppet)

STAY MAINSTREAM

- use stable and widely-used tools:
  - **OpenNebula** cloud controller
  - **GlusterFS** distrib. filesystem
  - **OpenWRT** for network management

BE USER-ORIENTED

- agile development cycle
- provide resources asap
- add functionalities when needed

The ingredients

OpenNebula as
cloud management toolkit:

- open source
- wide user community
- fits our use-cases
- easy to customize (shell and ruby scripts)

The ingredients

OpenNebula as
cloud management toolkit:

- open source
- wide user community
- fits our use-cases
- easy to customize (shell and ruby scripts)

OpenWRT for Virtual Routers:

- light-weight Linux distribution for embedded systems
- tools for network configuration and management

The ingredients

OpenNebula as
cloud management toolkit:

- open source
- wide user community
- fits our use-cases
- easy to customize (shell and ruby scripts)

OpenWRT for Virtual Routers:

- light-weight Linux distribution for embedded systems
- tools for network configuration and management

GlusterFS as filesystem:

- mimics RAID functionalities at FS level (aggregates *bricks* on different machines)
- easy to set-up in basic configuration
- easy management (on-line addition/removal/replacement of bricks)
- flexible to cater different needs with a single tool
- proven robustness
- horizontal scalability:
  - peer-to-peer synchronizations
  - data access directly form host node

Two clusters of VMs

SERVICES-CLASS VMs

- provide critical services
- in/out-bound connectivity
- public and private IP
- live migration
- no particular local disk I/O requirements

Two clusters of VMs

SERVICES-CLASS VMs

- provide critical services
- in/out-bound connectivity
- public and private IP
- live migration
- no particular local disk I/O requirements

- server-class hardware
- shared image repository
- resiliency-optimized FS for shared system disks (RAID1)
- currently 4 hosts

Two clusters of VMs

SERVICES-CLASS VMs

- provide critical services
- in/out-bound connectivity
- public and private IP
- live migration
- no particular local disk I/O requirements

- server-class hardware
- shared image repository
- resiliency-optimized FS for shared system disks (RAID1)
- currently 4 hosts

WORKERS-CLASS VMs

- computational work-force (e.g. grid worker nodes)
- private IP only
- high storage I/O performance

Two clusters of VMs

SERVICES-CLASS VMs

- provide critical services
- in/out-bound connectivity
- public and private IP
- live migration
- no particular local disk I/O requirements

- server-class hardware
- shared image repository
- resiliency-optimized FS for shared system disks (RAID1)
- currently 4 hosts

WORKERS-CLASS VMs

- computational work-force (e.g. grid worker nodes)
- private IP only
- high storage I/O performance

- lower-class hardware
- cached image repository
- performance-optimized file system
- currently 35 hosts

Multipurpose storage: use cases

SHARED FS FOR SERVICES-CLASS INSTANCES

- replicated on two servers for redundancy
- replica is synchronous
- self-healing enabled
- little I/O
- low performance required

VM IMAGE REPOSITORY

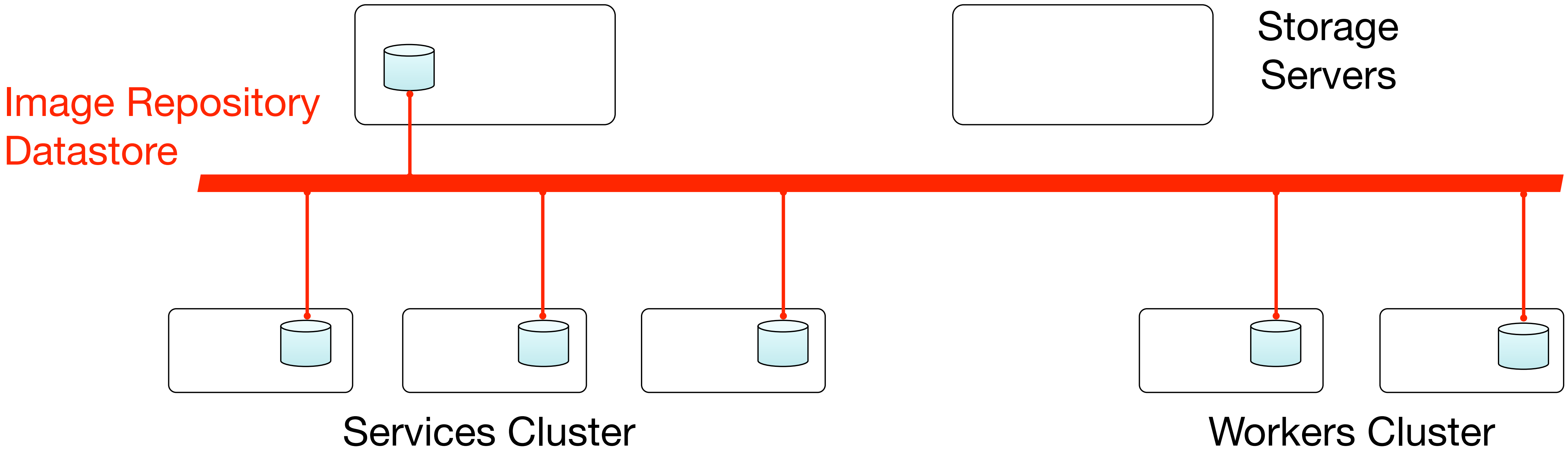
- images in raw or qcow2 format
- locally cached on workers cluster

DATA STORAGE

- pool of aggregated disks (50 TB)
- very high throughput towards many concurrent clients

Multipurpose storage: set-up

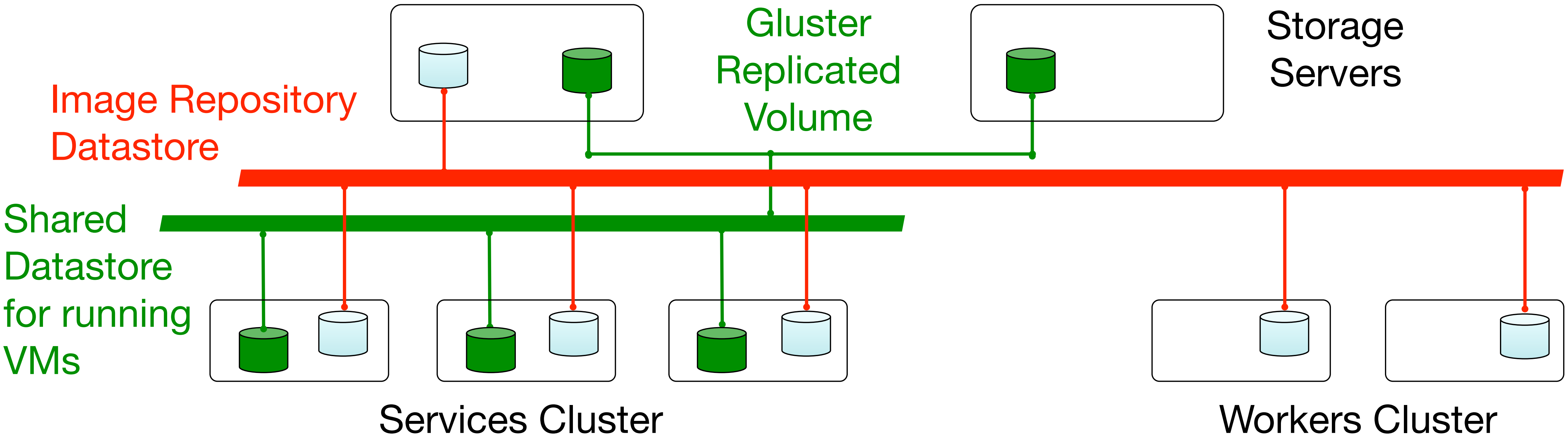
2 Storage Servers with 10 Gbps links provide their Fibre Channel SAN LUNs to build bricks through GlusterFS



Multipurpose storage: set-up

2 Storage Servers with 10 Gbps links provide their Fibre Channel SAN LUNs to build bricks through GlusterFS

SERVICES FS for running instances:
shared among Service Cluster hosts to allow for live-migration

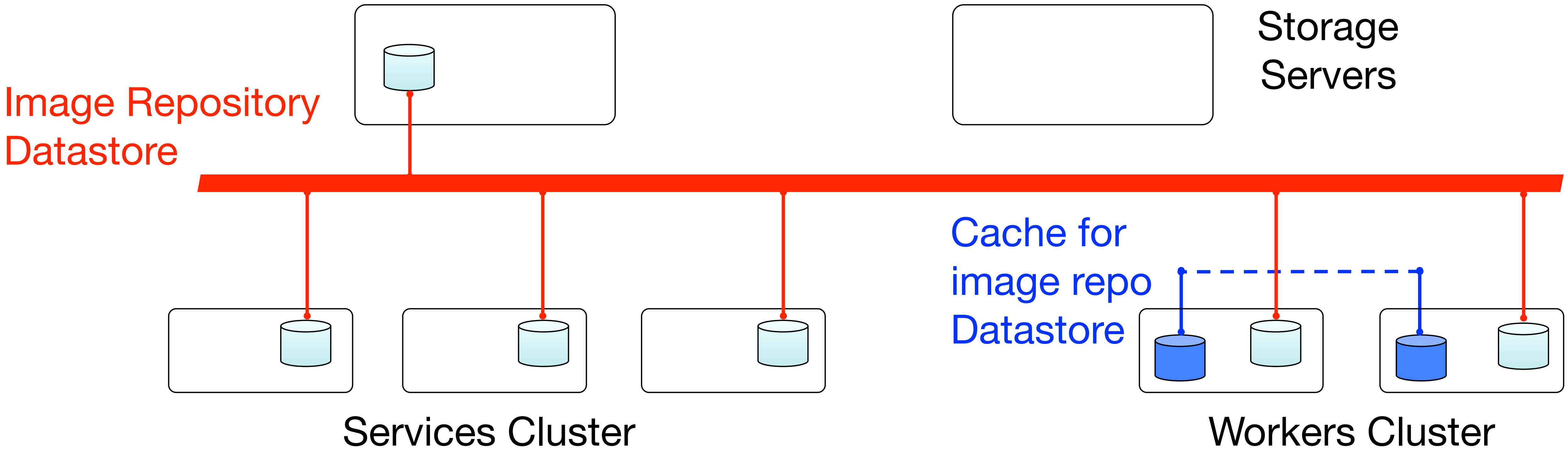

Gluster Replicated Volume
Storage Servers
Image Repository Datastore
Shared Datastore for running VMs
Services Cluster
Workers Cluster

Multipurpose storage: set-up

2 Storage Servers with 10 Gbps links provide their Fibre Channel SAN LUNs
to build bricks through GlusterFS

WORKERS FS for running instances:

- local to hypervisors to increase I/O throughput
- images repo locally cached to reduce start-up time
- custom torrent-like tool (scp Wave + rsync) to synchronize local copies



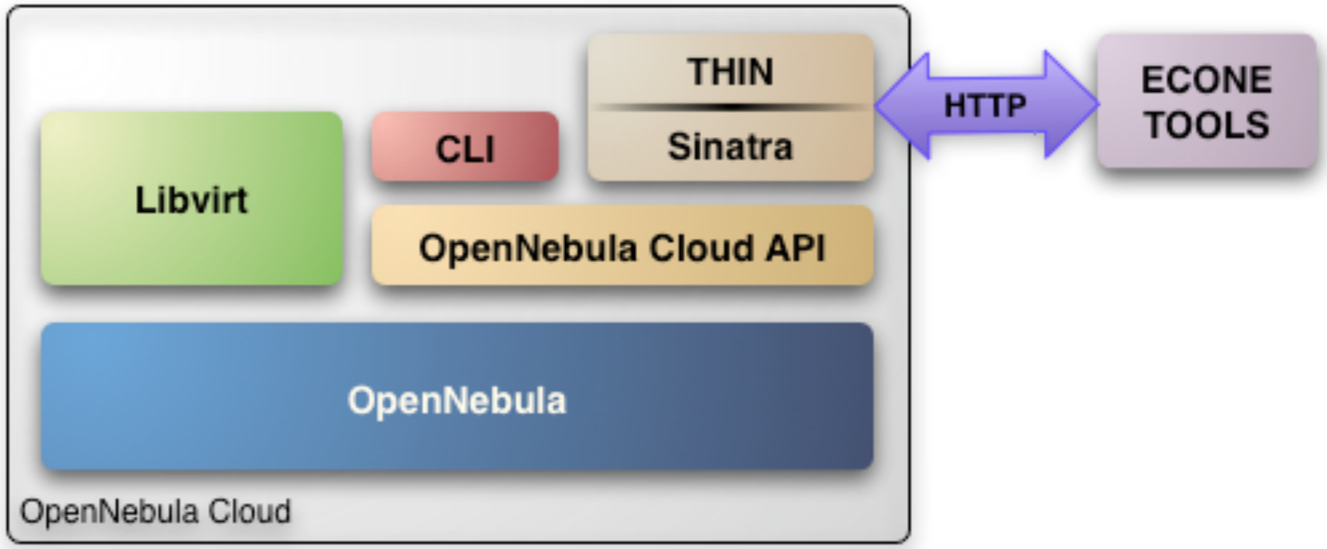
Stay mainstream...

- independently from the complexity of the infrastructure, our Cloud exposes a standard interface (EC2)
- support contextualization via a widely used tool (CloudInit)

Exposing a standard Cloud interface

EC2 QUERY

- OpenNebula web service to manage VMs through *Amazon EC2 API*
- can be used alongside the native OpenNebula client
- implemented upon the OpenNebula Cloud API (OCA)
- Sinatra as light web framework
- includes basic tools to use the query service (econe-tools)



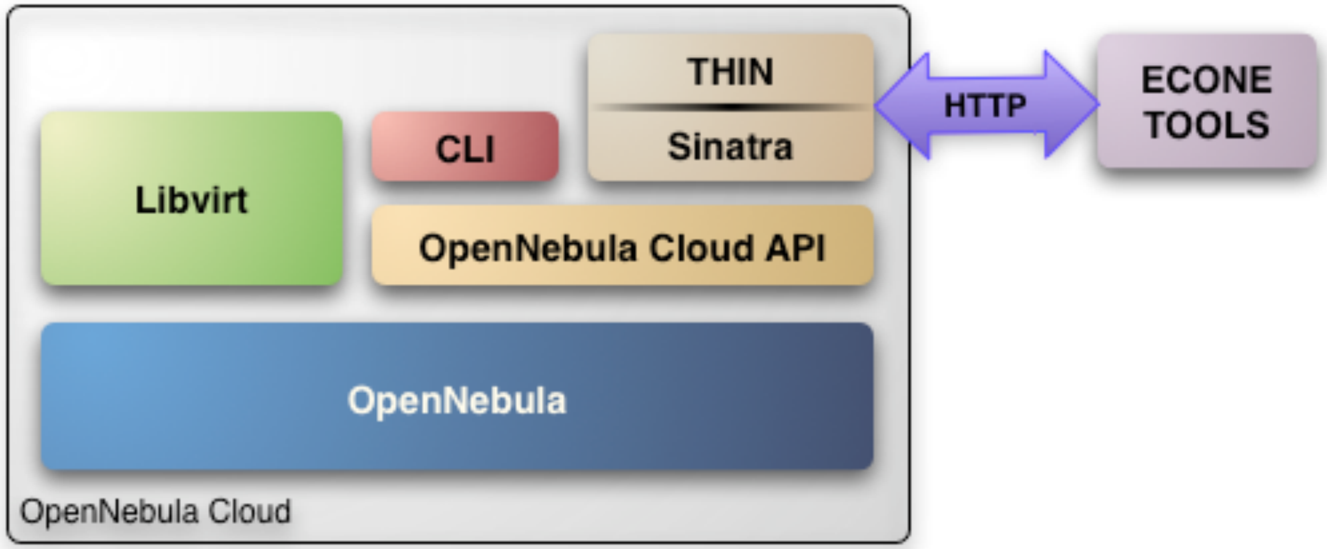
Reference EC2 Query: http://goo.gl/kQh6w5

Exposing a standard Cloud interface

EC2 QUERY

- OpenNebula web service to manage VMs through *Amazon EC2 API*
- can be used alongside the native OpenNebula client
- implemented upon the OpenNebula Cloud API (OCA)
- Sinatra as light web framework
- includes basic tools to use the query service (econe-tools)

Provides an interface to the private cloud compatible with the Amazon EC2 Query API



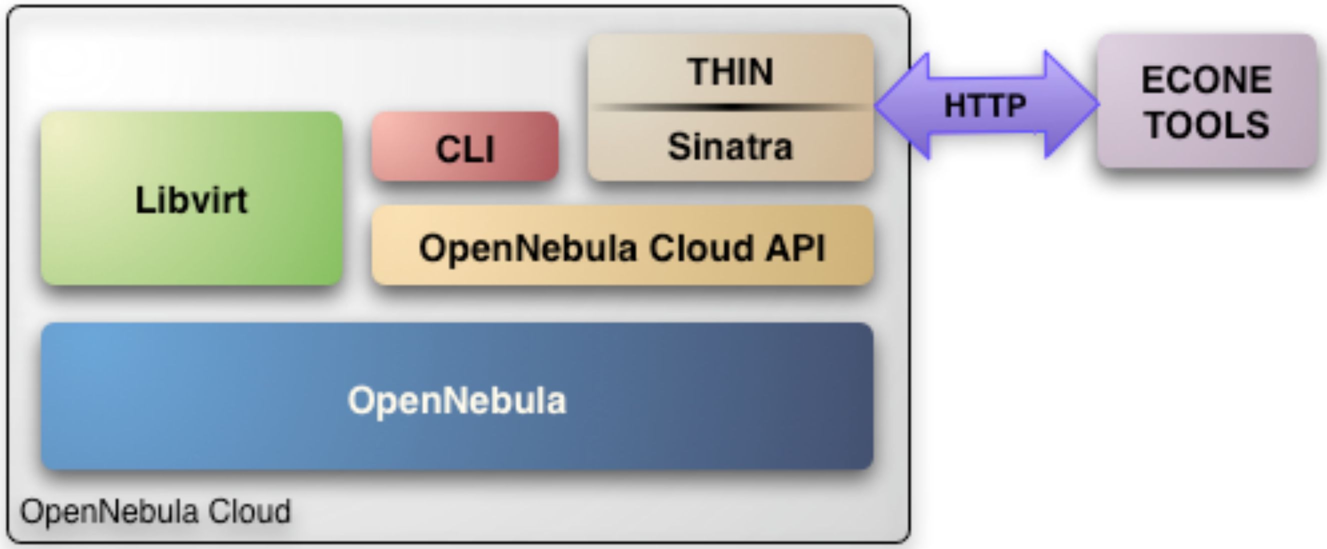
Reference EC2 Query: http://goo.gl/kQh6w5

Exposing a standard Cloud interface

EC2 QUERY

- OpenNebula web service to manage VMs through *Amazon EC2 API*
- can be used alongside the native OpenNebula client
- implemented upon the OpenNebula Cloud API (OCA)
- Sinatra as light web framework
- includes basic tools to use the query service (econe-tools)

Provides an interface to the private cloud compatible with the Amazon EC2 Query API



Reference EC2 Query: http://goo.gl/kQh6w5

EUCA-TOOLS

- native Eucalyptus command-line tools (open-source) http://goo.gl/qiSmXb
- interact with AWS-compatible web services
- more functionalities than econe-tools (e.g. pass user-data as file)

CloudInit

Docs:
http://goo.gl/jNebHc
http://goo.gl/UVbDZb

Source code:
http://goo.gl/hU1G8v

WHAT?

- Ubuntu package to handle early initialization of a cloud instance
- default on Ubuntu images on EC2
- available for multiple Linux distros
- Python source code
- boto library to fetch configuration data
- datasources:
  - meta-data (from the cloud stack): hostname, ssh keys, ephemeral mount-points etc...
  - user-data: ...anything else!

CloudInit

Docs:
http://goo.gl/jNebHc
http://goo.gl/UVbDZb

Source code:
http://goo.gl/hU1G8v

WHAT?

- Ubuntu package to handle early initialization of a cloud instance
- default on Ubuntu images on EC2
- available for multiple Linux distros
- Python source code
- boto library to fetch configuration data
- datasources:
  - meta-data (from the cloud stack): hostname, ssh keys, ephemeral mount-points etc...
  - user-data: ...anything else!

WHY?

- increasingly popular
- modular and customizable: custom modules and *part-handlers*
- several configuration formats: cloud-config (yaml), shell scripts, include urls, gzip or MIME archives...

CloudInit

Docs:
http://goo.gl/jNebHc
http://goo.gl/UVbDZb

Source code:
http://goo.gl/hU1G8v

WHAT?

- Ubuntu package to handle early initialization of a cloud instance
- default on Ubuntu images on EC2
- available for multiple Linux distros
- Python source code
- boto library to fetch configuration data
- datasources:
  - meta-data (from the cloud stack): hostname, ssh keys, ephemeral mount-points etc...
  - user-data: ...anything else!

WHY?

- increasingly popular
- modular and customizable: custom modules and *part-handlers*
- several configuration formats: cloud-config (yaml), shell scripts, include urls, gzip or MIME archives...

CloudInit and OpenNebula

- since version 0.7.3 OpenNebula context disk supported as data-source
- not many examples yet
- small change applied to enable usage of user-data *file* with euca-tools

Changing the contextualization strategy

To achieve portability of custom instances to any EC2-compatible cloud, provided:
CloudInit installed on base images + single user-data file (MIME arch.)

Changing the contextualization strategy

To achieve portability of custom instances to any EC2-compatible cloud, provided:
CloudInit installed on base images + single user-data file (MIME arch.)

- complex contextualization (worker-nodes and servers)
  → base image is a minimal OS installation

Changing the contextualization strategy

To achieve portability of custom instances to any EC2-compatible cloud, provided:
CloudInit installed on base images + single user-data file (MIME arch.)

- complex contextualization (worker-nodes and servers)
  → base image is a minimal OS installation

- custom modules to handle needed context blocks (partition filesystem, install cvmfs, install and configure grid software etc...)
  → in form of part-handlers, to be modified only when new features are needed

Changing the contextualization strategy

To achieve portability of custom instances to any EC2-compatible cloud, provided:
CloudInit installed on base images + single user-data file (MIME arch.)

- complex contextualization (worker-nodes and servers)
  → base image is a minimal OS installation
- custom modules to handle needed context blocks (partition filesystem, install cvmfs, install and configure grid software etc...)
  → in form of part-handlers, to be modified only when new features are needed
- configuration in simple cloud-config-like format (yaml)
  → eventually the only thing the user should modify!

Changing the contextualization strategy

To achieve portability of custom instances to any EC2-compatible cloud, provided:
CloudInit installed on base images + single user-data file (MIME arch.)

- complex contextualization (worker-nodes and servers)
  → base image is a minimal OS installation
- custom modules to handle needed context blocks (partition filesystem, install cvmfs, install and configure grid software etc...)
  → in form of part-handlers, to be modified only when new features are needed
- configuration in simple cloud-config-like format (yaml)
  → eventually the only thing the user should modify!
- script to embed all the information (part-handlers, configuration, keys...) in a single MIME archive

Changing the contextualization strategy

To achieve portability of custom instances to any EC2-compatible cloud, provided:
CloudInit installed on base images + single user-data file (MIME arch.)

- complex contextualization (worker-nodes and servers)
  → base image is a minimal OS installation
- custom modules to handle needed context blocks (partition filesystem, install cvmfs, install and configure grid software etc...)
  → in form of part-handlers, to be modified only when new features are needed
- configuration in simple cloud-config-like format (yaml)
  → eventually the only thing the user should modify!
- script to embed all the information (part-handlers, configuration, keys...) in a single MIME archive
- instance VM with euca-tools passing the MIME archive

Changing the contextualization strategy

To achieve portability of custom instances to any EC2-compatible cloud, provided:
CloudInit installed on base images + single user-data file (MIME arch.)

- complex contextualization (worker-nodes and servers)
  → base image is a minimal OS installation
- custom modules to handle needed context blocks (partition filesystem, install cvmfs, install and configure grid software etc...)
  → in form of part-handlers, to be modified only when new features are needed
- configuration in simple cloud-config-like format (yaml)
  → eventually the only thing the user should modify!
- script to embed all the information (part-handlers, configuration, keys...) in a single MIME archive
- instance VM with euca-tools passing the MIME archive
- user-data are securely shipped to the instance via the OpenNebula context disk

Example: work in progress...

At the moment part of the configuration data are taken from an http server (not directly embedded in the MIME)

Example: work in progress...

At the moment part of the configuration data are taken from an http server (not directly embedded in the MIME)

Configure your instance:

```
#multiple-config
# vim: syntax=yaml

# Configure local file-system
localfs:
   parts:
      # below percent of VG
      cvmfs: 17
      cvmfs-mount: /var/lib/cvmfs
      home: 58
      tmp: 8
      # below percent of FREE
      swap: 100

# Configure CVMFS
cvmfs:
  install: true
  version: 2.1.14
  local:
    repositories: alice.cern.ch...
    http-proxy: http://t2-squid-01...
    cache-base: /var/lib/cvmfs
    quota-limit: 18000
```

Example: work in progress...

At the moment part of the configuration data are taken from an http server (not directly embedded in the MIME)

Configure your instance:

```
#multiple-config
# vim: syntax=yaml

# Configure local file-system
localfs:
   parts:
      # below percent of VG
      cvmfs: 17
      cvmfs-mount: /var/lib/cvmfs
      home: 58
      tmp: 8
      # below percent of FREE
      swap: 100

# Configure CVMFS
cvmfs:
  install: true
  version: 2.1.14
  local:
    repositories: alice.cern.ch...
    http-proxy: http://t2-squid-01...
    cache-base: /var/lib/cvmfs
    quota-limit: 18000
```

Run a script to produce the MIME archive:

```
From nobody Tue Oct  8 16:11:36 2013
Content-Type: multipart/mixed;
boundary="===============181078353...
MIME-Version: 1.0

--===============1810783535439179425...
Content-Type: text/x-include-url; charset="us-
ascii"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
filename="include_modules.txt"

#include
http:srm-dom0.to.infn.it/test/cc_localfs.py
http:srm-dom0.to.infn.it/test/cc_cvmfs.py

...
--===============1810783535439179425...
Content-Type: text/localfs-config; charset="us-
ascii"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
filename="localfs-config.ccfg"

localfs:
  parts:
      cvmfs: 17
```

Example: work in progress...

At the moment part of the configuration data are taken from an http server (not directly embedded in the MIME)

Configure your instance:

```
#multiple-config
# vim: syntax=yaml

# Configure local file-system
localfs:
   parts:
      # below percent of VG
      cvmfs: 17
      cvmfs-mount: /var/lib/cvmfs
      home: 58
      tmp: 8
      # below percent of FREE
      swap: 100

# Configure CVMFS
cvmfs:
  install: true
  version: 2.1.14
  local:
    repositories: alice.cern.ch...
    http-proxy: http://t2-squid-01...
    cache-base: /var/lib/cvmfs
    quota-limit: 18000
```

Run a script to produce the MIME archive:

```
From nobody Tue Oct  8 16:11:36 2013
Content-Type: multipart/mixed;
boundary="===============181078353...
MIME-Version: 1.0

--===============1810783535439179425...
Content-Type: text/x-include-url; charset="us-
ascii"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
filename="include_modules.txt"

#include
http:srm-dom0.to.infn.it/test/cc_localfs.py
http:srm-dom0.to.infn.it/test/cc_cvmfs.py

--===============1810783535439179425...
Content-Type: text/localfs-config; charset="us-
ascii"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
filename="localfs-config.ccfg"

localfs:
  parts:
      cvmfs: 17
```

Instance your VM *à la EC2:*

```
euca-run-instances ami-000070 \
                         -t m1.large \
                       -f myMIME.txt \
```

Example: work in progress...

At the moment part of the configuration data are taken from an http server (not directly embedded in the MIME)

Configure your instance: → Run a script to produce the MIME archive: → Instance your VM *à la EC2:*

```
#multiple-config
# vim: syntax=yaml

# Configure local file-system
localfs:
   parts:
      # below percent of VG
      cvmfs: 17
      cvmfs-mount: /var/lib/cvmfs
      home: 58
      tmp: 8
      # below percent of FREE
      swap: 100

# Configure CVMFS
cvmfs:
  install: true
  version: 2.1.14
  local:
    repositories: alice.cern.ch...
    http-proxy: http://t2-squid-01...
    cache-base: /var/lib/cvmfs
    quota-limit: 18000
```

```
From nobody Tue Oct  8 16:11:36 2013
Content-Type: multipart/mixed;
boundary="===============181078353...
MIME-Version: 1.0

--===============1810783535439179425...
Content-Type: text/x-include-url; charset="us-
ascii"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
filename="include_modules.txt"

#include
http:srm-dom0.to.infn.it/test/cc_localfs.py
http:srm-dom0.to.infn.it/test/cc_cvmfs.py
...
--===============1810783535439179425...
Content-Type: text/localfs-config; charset="us-
ascii"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
filename="localfs-config.ccfg"

localfs:
  parts:
      cvmfs: 17
```

```
euca-run-instances ami-000070 \
                      -t m1.large \
                    -f myMIME.txt \
```

↓

Check the logs:

- each part-handler produces a detailed log-file
- a final script checks the logs for errors
- summary log with a list of failed handlers

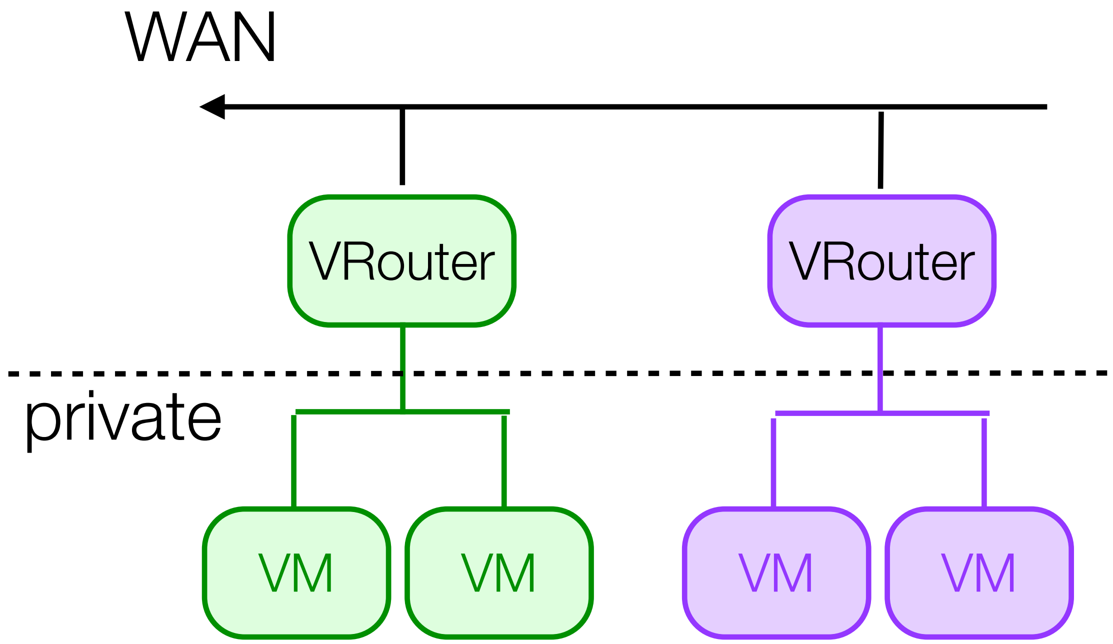
→ **next**: shut-down the instance in case of failures

Be user-oriented...

- private networks
- private farms

Networking: VRouter and Elastic IPs

Provide users with fully featured class-C private networks.

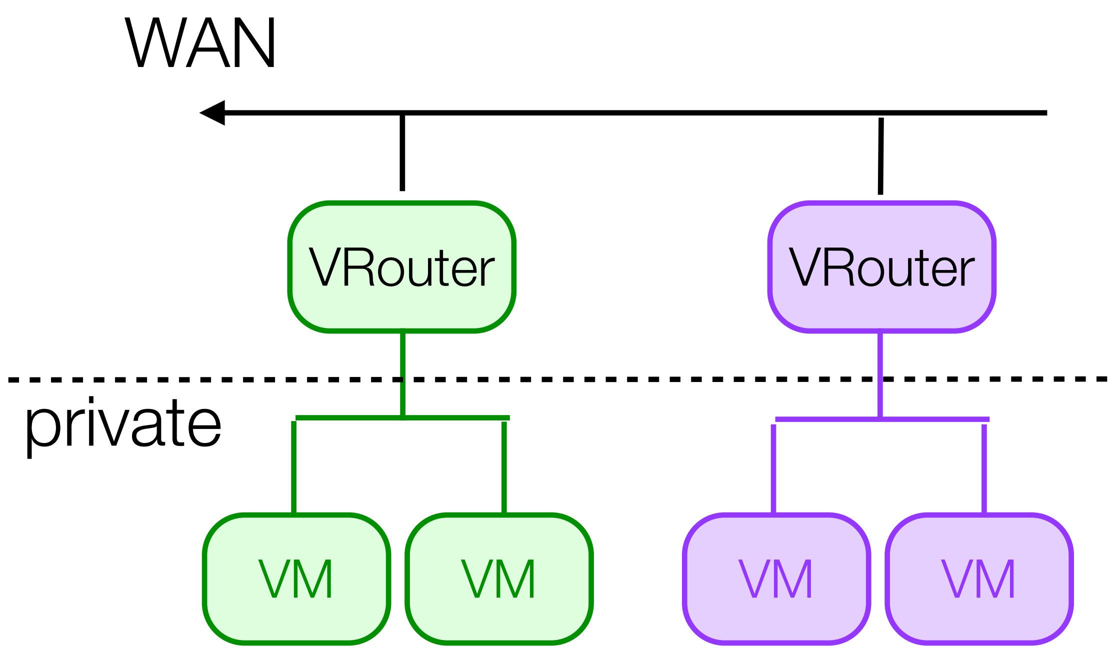


Networking: VRouter and Elastic IPs

Provide users with fully featured class-C private networks.

Network isolation (level 2):

- each user is assigned a Virtual Network
- each network is isolated with ebtables rules on the hypervisor bridge (OpenNebula V-net driver)



Networking: VRouter and Elastic IPs

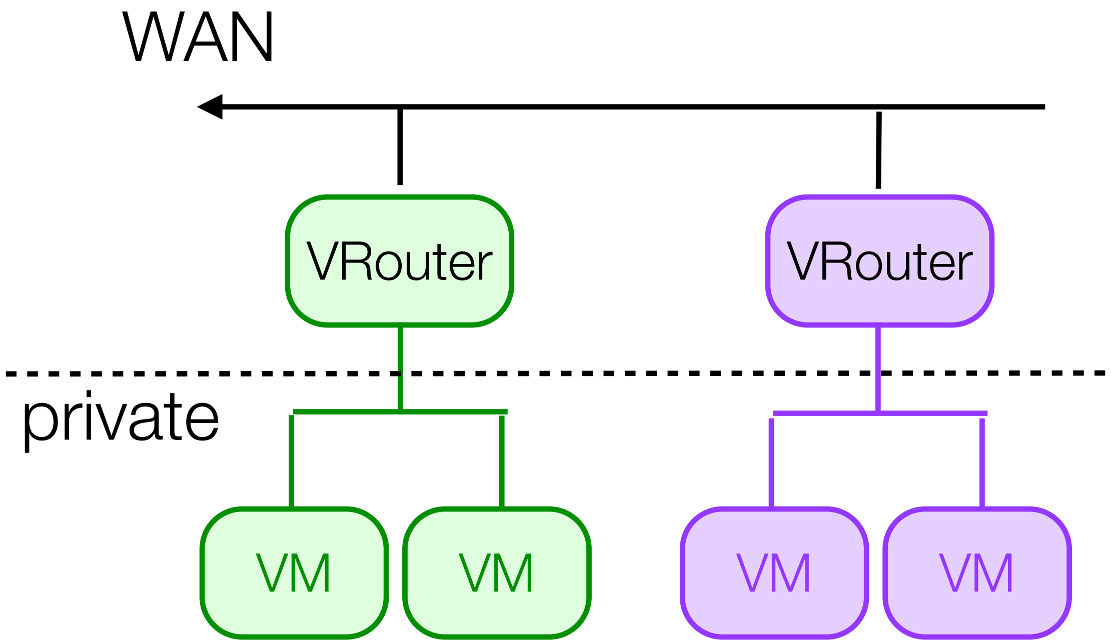
Provide users with fully featured class-C private networks.

Network isolation (level 2):

- each user is assigned a Virtual Network
- each network is isolated with ebtables rules on the hypervisor bridge (OpenNebula V-net driver)

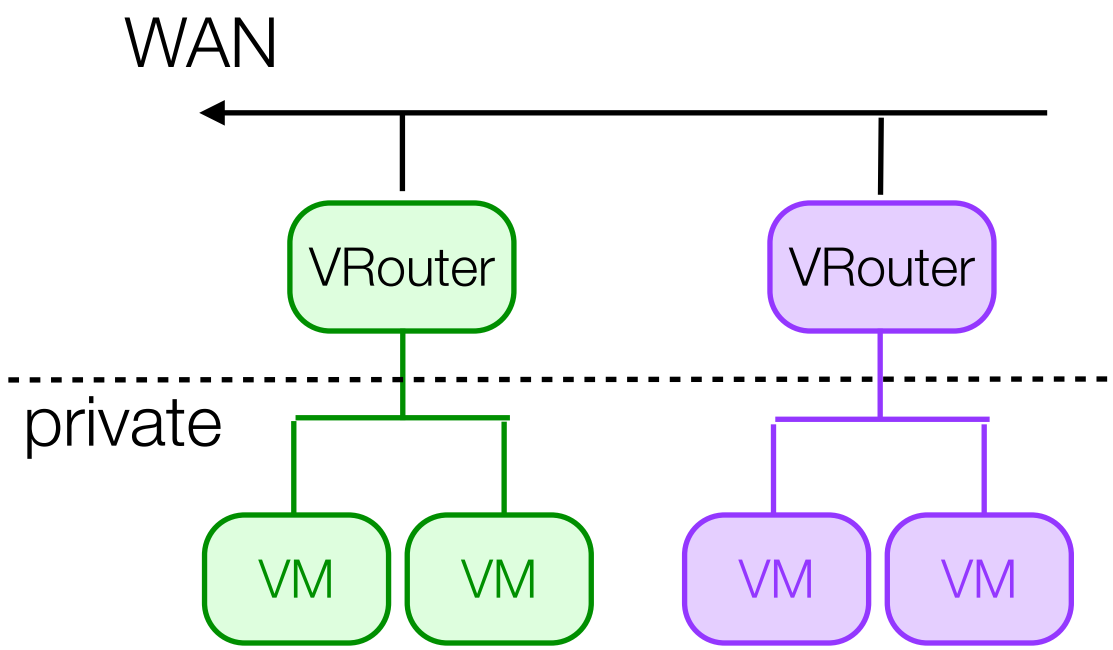
Virtual Routers (level 3):

- private and public IP
- light-weight OpenWRT VM (1CPU, 150 MB)
- DHCP, DNS, NAT functionalities
- Firewalling / port-forwarding
- configuration possible via HTTPS or SSH



Networking: VRouter and Elastic IPs

Provide users with fully featured class-C private networks.



Network isolation (level 2):

- each user is assigned a Virtual Network
- each network is isolated with ebtables rules on the hypervisor bridge (OpenNebula V-net driver)

Virtual Routers (level 3):

- private and public IP
- light-weight OpenWRT VM (1CPU, 150 MB)
- DHCP, DNS, NAT functionalities
- Firewalling / port-forwarding
- configuration possible via HTTPS or SSH

Connectivity remains under sysadmin control, user has no access to the VRouter!

Networking: VRouter and Elastic IPs

Provide users with fully featured class-C private networks.

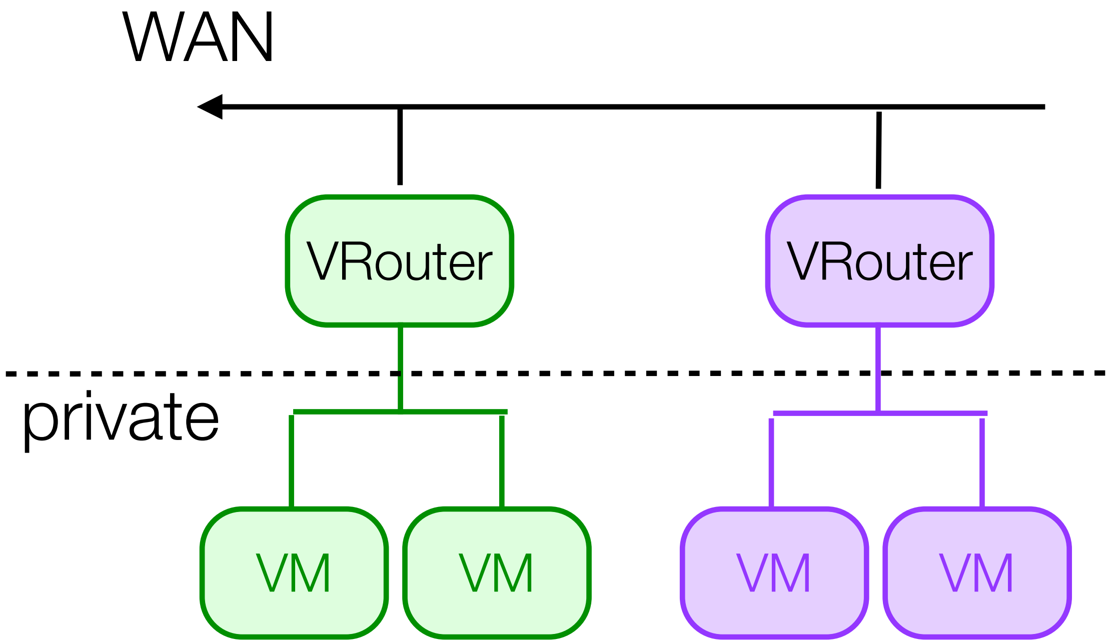
Network isolation (level 2):

- each user is assigned a Virtual Network
- each network is isolated with ebtables rules on the hypervisor bridge (OpenNebula V-net driver)

Virtual Routers (level 3):

- private and public IP
- light-weight OpenWRT VM (1CPU, 150 MB)
- DHCP, DNS, NAT functionalities
- Firewalling / port-forwarding
- configuration possible via HTTPS or SSH

Connectivity remains under sysadmin control, user has no access to the VRouter!



ELASTIC IPs

The VRouter allows to use EC2-compatible APIs (euca-tools) to

→ bind DINAMICALLY a public IP to one of the private VM instances

A model for Virtual Farm provisioning

Wiki: http://goo.gl/Lj5l01

Automated creation of a sandboxed environment within our Private Cloud:

- creation of new OpenNebula user
- creation of isolated Virtual Network
- configuration and instantiation of dedicated VRouter
- 1 elastic public IP assigned
- restrictive quota on the amount of resources
- use only a subset of images with *public* permissions
- configuration simplified through the definition of Amazon-like flavors
- flavors mapped onto OpenNebula templates

Flavor	**CPUs**	**RAM**	**Ephemeral disk**
m1.small	1	512 MB	-
m1.medium	2	2 GB	20 GB
m1.large	4	8 GB	80 GB

A model for Virtual Farm provisioning

Wiki: http://goo.gl/Lj5l01

Automated creation of a sandboxed environment within our Private Cloud:

- creation of new OpenNebula user
- creation of isolated Virtual Network
- configuration and instantiation of dedicated VRouter
- 1 elastic public IP assigned
- restrictive quota on the amount of resources
- use only a subset of images with *public* permissions
- configuration simplified through the definition of Amazon-like flavors
- flavors mapped onto OpenNebula templates

Flavor	**CPUs**	**RAM**	**Ephemeral disk**
m1.small	1	512 MB	-
m1.medium	2	2 GB	20 GB
m1.large	4	8 GB	80 GB

State of the art:

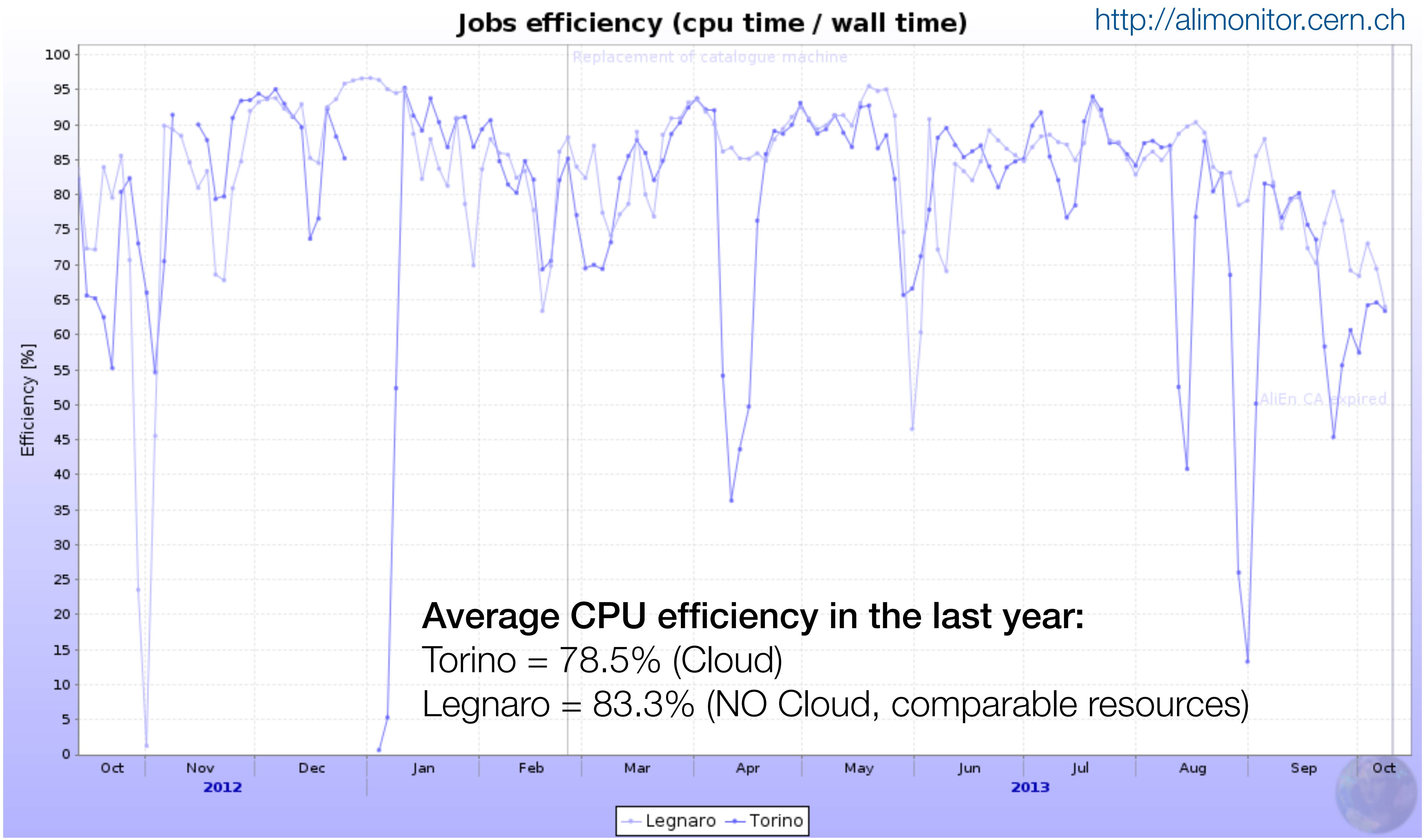
- ready to be tried out by users
- access and control from within the INFN Torino network
- access tools (euca-tools) available on public login hosts

Conclusions and Outlook

- Cloud infrastructure in production for nearly 2 years
- until now IaaS paradigm allowed us to reduce the management effort
- infrastructure now ready for dynamic use cases and self-service provisioning
- coordination with other INFN Cloud development activities
- any new hardware will be included in the Cloud infrastructure and transparently given back to users as virtual resources
- Work in progress: automated reallocation of resources

Back-up

ALICE Tier2 CPU efficiency



http://alimonitor.cern.ch

Average CPU efficiency in the last year:

Torino = 78.5% (Cloud)

Legnaro = 83.3% (NO Cloud, comparable resources)