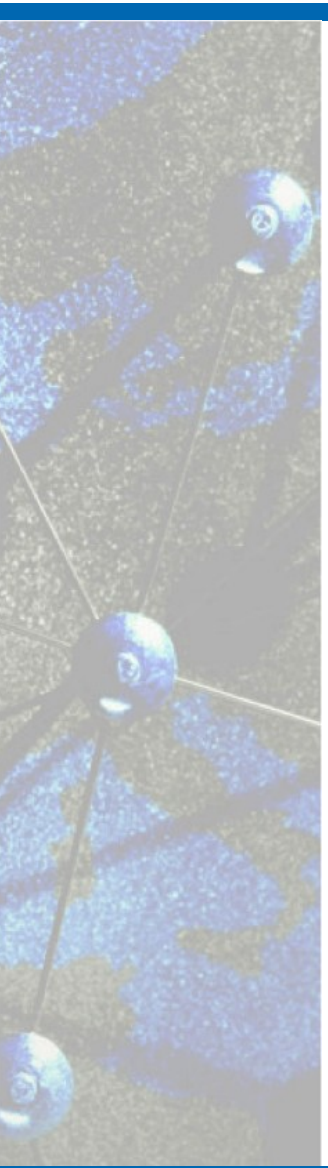


Best Practices for Cloud Application Architecture

Björn Hagemeier, Boris Parák, Miroslav Ruda

e-infrastructure

Agenda

- Thinking in terms of IaaS
- Application architecture
- Image preparation and customization
- Contextualization
- Storage resources

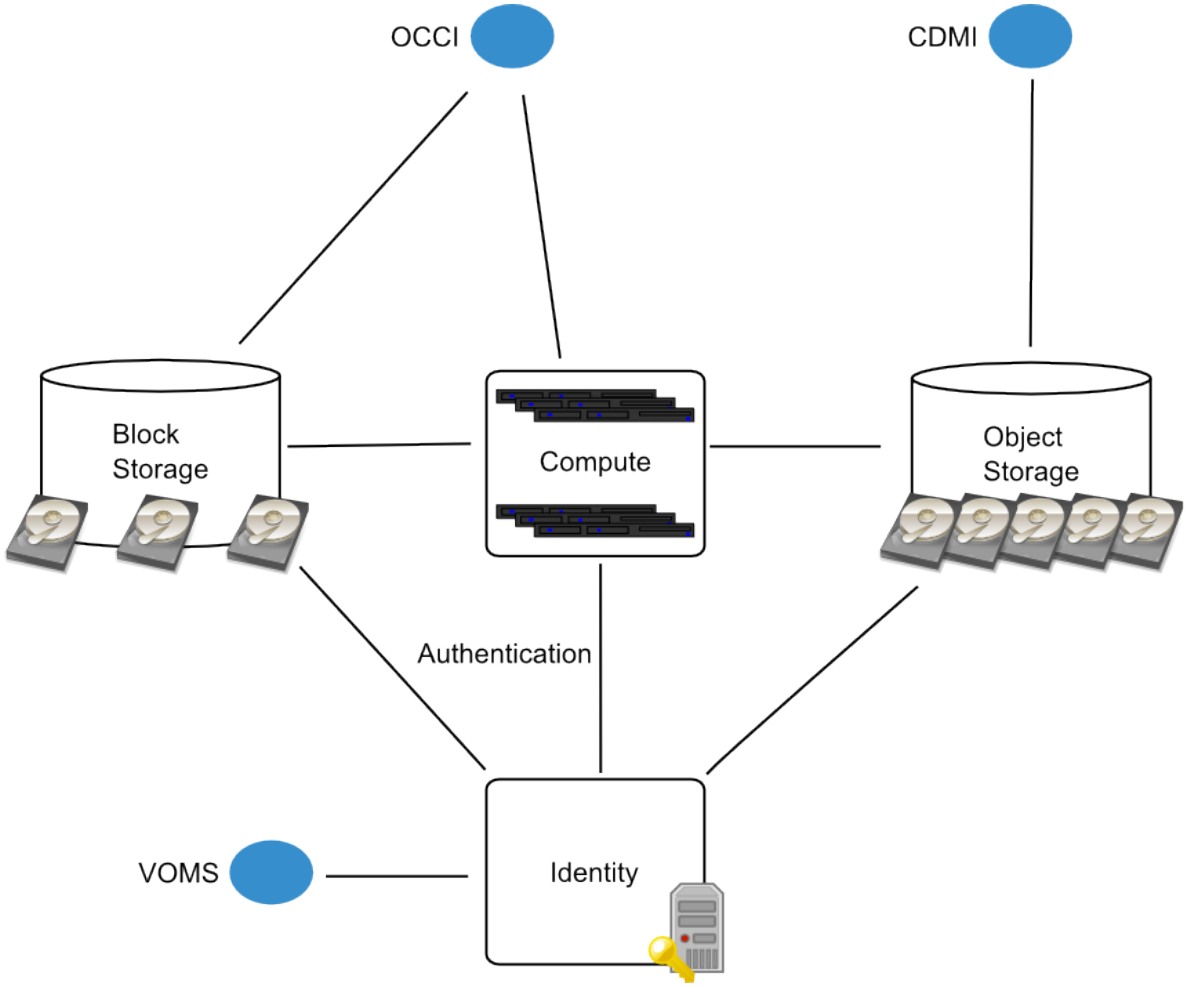
Part I

Infrastrucuture as a Service – IaaS

IaaS

Components

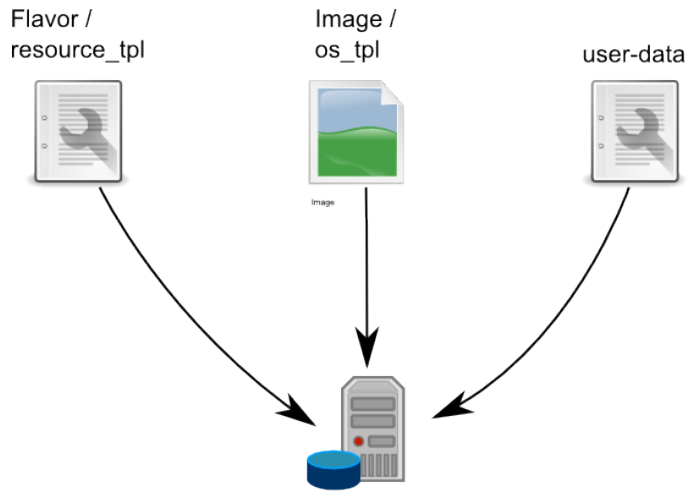
- Images
- Instances
- Block storage
- Object storage
- Network



There are various answers to this question

- Typically
  - A blob of data representing the contents of a disk or file system
  - Usually contains the root file system contents of an operating system
- Image is usually used as a term for a *template* to create a virtual machine or *instance* of it.
  - In OCCI terms, an image is an *Operating System Template* – *OS Template*
- Application data can be conveyed within images

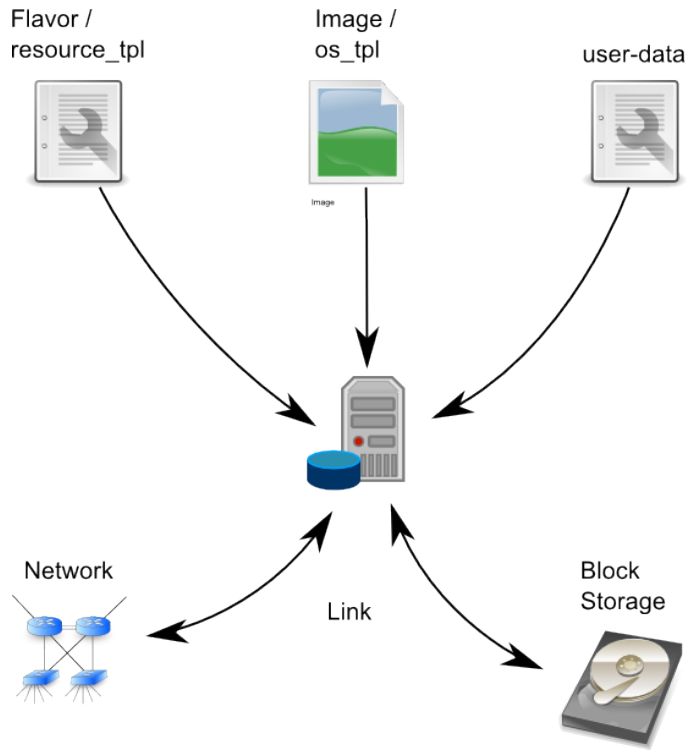
- A virtual machine (VM)
  - ... created from an *image*
  - ... given an amount of *resources*
  - ... and additional parameters (aka. *user-data*)
- User data is used to pass parameters to the instance
  - This is used in the *contextualization* process
  - More details on this later
- Links are used to associate
  - Block storage
  - Public networks

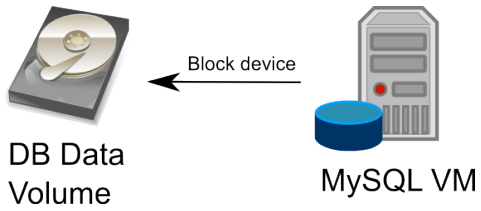


IaaS

What is an instance?

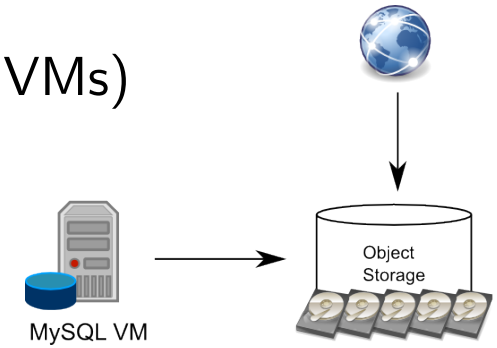
- A virtual machine (VM)
  - ... created from an *image*
  - ... given an amount of *resources*
  - ... and additional parameters (aka. *user-data*)
- User data is used to pass parameters to the instance
  - This is used in the *contextualization* process
  - More details on this later
- Links are used to associate
  - Block storage
  - Public networks





Storage usually comes in two flavors

- Block Storage
  - Can be attached to VM instances as block devices
  - Transparent to applications running within VM
  - Used to store data persistently (until the block device is deleted)
- Object Storage
  - Accessible from anywhere (within and outside VMs)
  - Virtually infinitely scalable
  - Standard to access: CDMI
- More details on storage later



Network

Network as a resource, or Software Defined Networks (SDN) do not play a role in the EGI Federated Cloud.

However

- At some sites users need to explicitly add public IP addresses to instances to make them available to the world
- This can be achieved through the OCCI interface by means of links
- Only public IPs make the VM accessible from the outside world

Part II

Images

Starting points

- Existing images
- Installation media
- Special tools for image creation

Our recommendation

- Existing images
  - Basic images of popular LINUX distributions available
    - Ubuntu
    - Debian
    - ScientificLinux

Image Creation

Starting from existing images

- Download image from repository
- Start image
  - Start it as a VM
    - Caveat: need local virtualization platform, e. g. KVM
  - Alternative: start as a cloud instance and create snapshot
    - Caveat: creating snapshots not standardized among platforms
- Make your updates and modifications
- Clean up as you would do for a new image
- Re-package the image

An example

```
1  $ truncate -s 1G debian-7.4.0.img
2  $ kvm -cdrom debian-7.4.0-amd64-netinst.iso
       debian-7.4.0.img
```

During installation

- Select “Install”
- Give a strong root password despite it being locked later on
- Use manual partitioning
  - Avoid creating a swap partition and put everything in one partition
  - Additional block devices can and should be added at runtime
  - *Do not* use LVM based setups

During installation

- Software installation
  - Install as few packages as possible
  - Default set of system utilities
  - SSH server
- Boot loader on MBR

- Users and passwords
  - To disallow password base logins, put * or ! in the password field in `/etc/shadow`
- Log files
- Excessive data
  - Find it using du or ncdu
- ...?

We posted the general procedure in the EGI Blog: `http://goo.gl/ju7vgP`

Image Creation

Log Files

- Avoids leaking of potentially personal/sensitive data, though not likely right after installation
- Reduces image size if lots of logging data was created during a longer installation process
- We've used the option to truncate log files to size 0
  - Keeps files and their permissions

```
1  # find /var/log -type f -exec truncate -s 0 {} \;
```

Excessive Data

Look for

- Old Linux kernels and their modules in `/boot` and `/lib/modules`
- Undesired services
  - some distributions install a number of services by default
  - web, mail, etc.
  - remove them also for security reasons
- Unused packages
  - after removal of undesired services, some dependencies may not be needed anymore
- Clear the package cache
  - apt-get clean
  - yum clean all
  - zypper clean –all

You can go on a hunt for the biggest directory structures of your image. The following tools will help you

- du
  - `du -a / | sort -n -r| head -10`
  - `du -hsx *| sort -rh | head -10`
- ncdu
  - an interactive version of the above commands

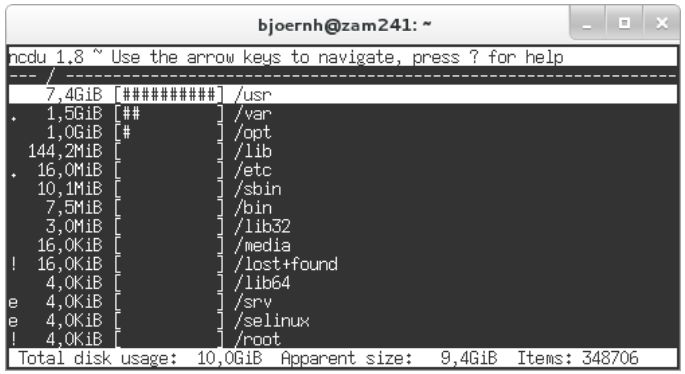


Image Creation

Tools

These tools can be used to deal with images after their creation. They help to create small clean images and access the file systems within for a final touch up.

qemu-img

- mainly used for image conversion, e.g. from raw to (compressed) QCOW2

zerofree

- zero out unused blocks in a file system
- improves compression ratio

kpartx

- Make partitions within image files available as block devices in the current system

Image Creation

Packaging

- Use zerofree to set unused blocks in the file system to '0'

```
# kpartx -av debian-7.4.0.img
add map loop0p1 (254:4): 0 4190208 linear
    /dev/loop0 2048
# zerofree /dev/mapper/loop0p1
# kpartx -d debian-7.4.0.img
```

- Create compressed qcow2

```
$ qemu-img convert -c -f raw -O qcow2 -p
    debian-7.4.0.img debian-7.4.0.qcow2
```

- Including zerofree, we've seen compressed image sizes up to 50% smaller than without it

More details in the EGI Blog: `http://goo.gl/UA6t1Y`

Try and make your images secure. Some simple measures:

- Disable or uninstall unneeded services and other software
- Do not allow unauthenticated access
- Use Strong passwords or disallow passwords

Image Maintainance

Things to avoid

When creating or updating images, the following can occur

- Unneded software may remain installed
  - Particularly true for older Kernel versions and their modules
  - Modules for a single Kernel are roughly 100MB
  - Desktop environment
  - Office suites
  - Games
- Swap files
  - Usually gigabytes in size
  - Do not make much sense for cloud. Use a larger instance instead.

Image Creation

Updates

- Steps to update contents of an image and create a new one
- Is this really possible with *only OCCI* commands?
  - We'd certainly want to support this, i. e. take an instance, create a snapshot of it and use thas as a new image.
- It should be just as easy as creating an image from an existing one.

Part III

Application Architecture

Application Architecture

"Do what you would do anyway to build a highly scalable web application."

– George Reese

- Plan for the worst, instances can and will fail
- Automation, e.g. through contextualization and configuration management will help
  - More on this later as part of contextualization
- Particularly important when running user-facing services

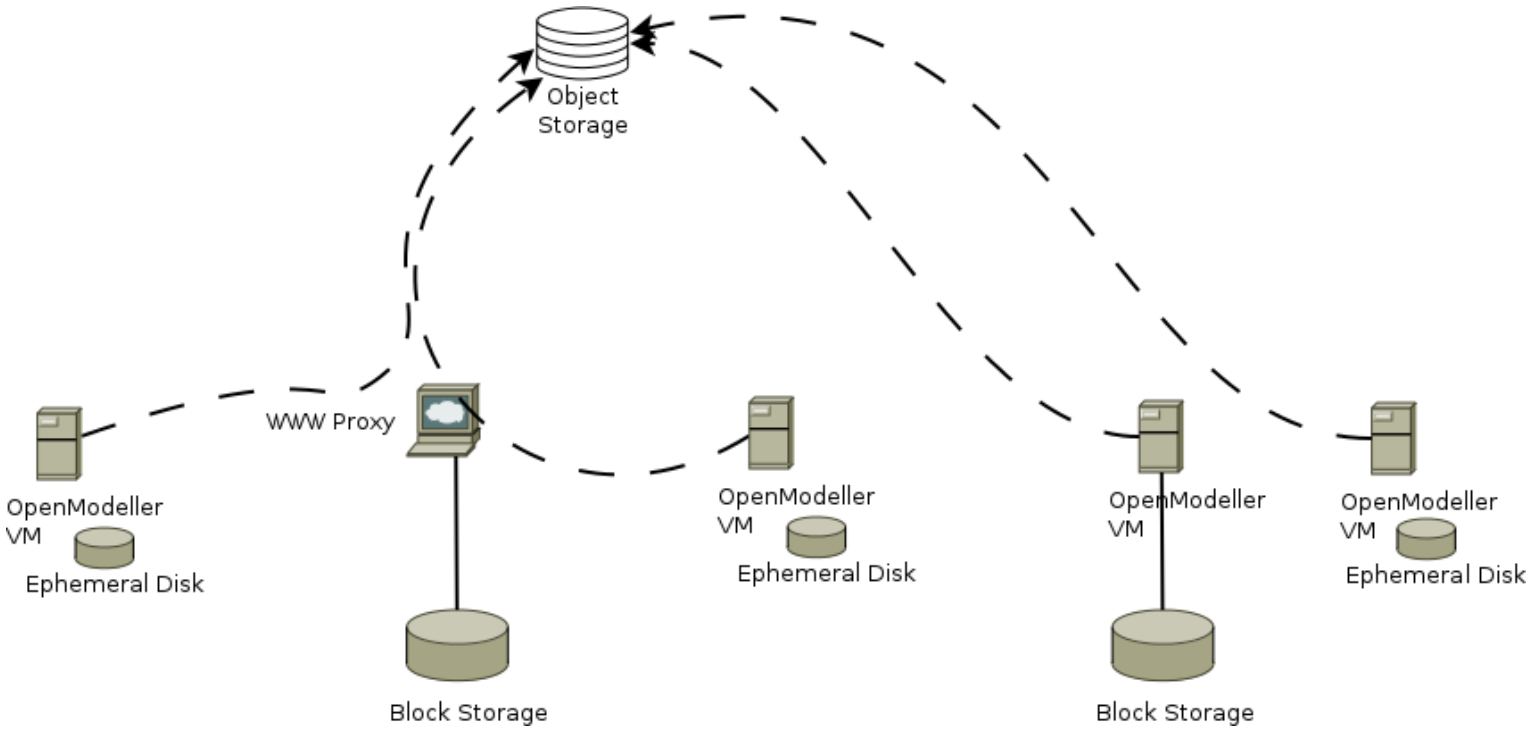
Compute and Data

The cloud is not only about VM instances for processing. It is also about data, storage, network, etc. Try and keep these concerns separated.
Therefore:

- Avoid large data payloads in your VM images. Particularly, if the data is rather static and reused in more than one VM.
- Make use of available storage resources such as block or object storage
  - This will keep you away from the "update trap"
  - Software updates will not require re-distribution of all data contained in images
  - Data updates can be managed centrally and also do not require re-distribution of large images
  - Make use of the same data from multiple instances/images

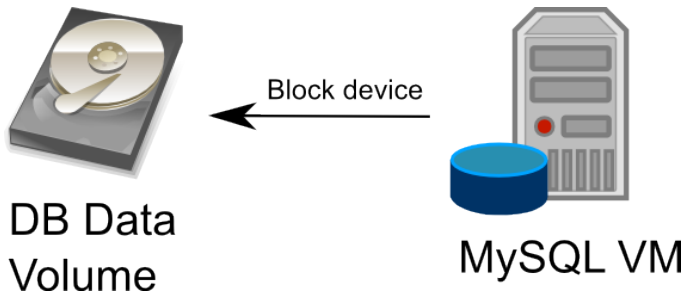
Compute and Data

Caching Example



An Example

- Create a generic VM containing the application and basic configuration, e. g. a MySQL server
- Create a block storage template containing MySQL data directories
- Modify MySQL startup scripts to wait for availability of block device



Also mentioned in “Cloud Application Architectures” by George Reese

```
# tree /mysql/
/mysql
|-- etc
|   `-- mysql
|       |-- conf.d
|       |   `-- mysqld_safe_syslog.cnf
|       |-- debian.cnf
|       |-- debian-start
|       |-- my.cnf
`-- mysql
```

The mysql service startup script has been modified to wait for the /mysql mountpoint to come online. This requires the external block device, e. g. /dev/vdc, to be mounted on that specific mount point.

Part IV

Contextualization

Contextualization

What is it?

A way to

- inject user credentials on boot
- perform simple configuration tasks
- customize generic images on boot
- pass arbitrary data into virtual machine instances

NOT a way to

- transfer large amounts of data
- perform complex or interactive tasks
- replace orchestration or provisioning tools

Contextualization

How it works?

1. Data is passed to the instance at boot
2. The instance is expected to use it
3. Implementation is usually platform-specific
4. There are attempts to "standardize" the process

Delivery mechanisms: custom-made scripts, direct injection, metadata services, contextualization disks

Contextualization

Using *cloud-init*

- *De facto* standard for multi-platform contextualization
- Supports multiple delivery mechanisms → "datasources"
- Offers implementations of commonly performed tasks → "modules"
- Configuration in the form of a YAML file → "user_data"

For details, see
`http://cloudinit.readthedocs.org/en/latest/index.html`

- Advanced task automation and parallelization
- Typically using a declarative approach
- Slow start, considerable advantages later on
- Preferred method for the cloud environment

Tools: puppet, chef, ansible, salt, CFEngine, ...

Contextualization

Usage example – Simple

```
#cloud-config

groups:
  - cloud-users

users:
  - name: cloudman
    gecos: Cloud H. Man
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: cloud-users, admin
    lock-passwd: true
    ssh-authorized-keys:
      - <ssh pub key 1>
      - <ssh pub key 2>
```

Contextualization

Usage example – Advanced

```
1  #cloud-config
2
3  mounts:
4  - [vdc,/data,ext4]
5
6  package_upgrade: true
7
8  packages:
9  - qemu-guest-agent
10
11 phone_home:
12   url: http://my.master.org/instances/$INSTANCE_ID
13   post: [ pub_key_rsa, instance_id ]
14
15 final_message: "The system is ready!"
```

Contextualization

Usage example – Provisioning

```
1  #cloud-config
2
3  puppet:
4   conf:
5     agent:
6       server: "my.master.org"
7       certname: "%i.%f"
8
9     ca_cert: |
10      -----BEGIN CERTIFICATE-----
11      ... omitted ...
12      -----END CERTIFICATE-----
```

Part V

Storage

Storage

What is storage?

- In this context, "virtualized" storage
- Place to store your data in the cloud
- Two distinctive types of storage

  block storage
  : local block devices connected to virtual machines, file system based

  object storage
  : remote, network-accessible and file based

- Different applications and underlying technologies

- Indistinguishable from ordinary physical drives
- Connected directly to a virtual machine; emulating *IDE*, *SCSI* or *SATA*
- Easier to integrate and use, no need to change the application
- Option to (dis)connect devices at runtime
- Ideal for short-term storage (input data & results) or caching
- **NOT** intended for long-term storage or large amounts of data

- Remote storage accessible via a management protocol (*CDMI*, *S3*, Swift, . . .)
- File-oriented approach, each file available as an independent object
- Ideal for mid-term to long-term storage, large amounts of data
- Option to share data between users or groups (e.g. read-only public access)
- Direct use requires support in applications

Storage

Examples I.

Problem My application works with a considerable amount of disposable data. Read/write random access to files is required.

Solution Use on-the-fly created block storage. It's designed to accommodate random read/write access to files. Your data is disposable by nature, there is no need to move it somewhere at the end of virtual machine's life-cycle.

Problem My application works with structured data in a database. Data is persistent by nature.

Solution Use persistent block storage. Configure the database to store its data files in a separately mounted block device.

Problem My application requires a large read-only data set. Data is the same across multiple instances and organized into a smaller number of large files. Random access is not required. Small parts of the data set are needed for specific actions.

Solution Use object storage to eliminate data duplication. Keep images small. On-demand access to data for every instance.

Problem I need to store the results of my computation and possibly share them with my colleagues. Data should be stored for a longer period of time and still accessible.

Solution Use object storage. After your computation is finished, store the resulting files and terminate all running instances.

Problem I have a complex application requiring a random read/write access to a large data set. Changes are disposable by nature. Results are stored separately. A single "run" takes a long time.

Solution Use a combination of block and object storage. Start instances with larger amounts of local block storage, pull data from object storage and cache it locally. Store results in object storage after you are done.

Summary

Pointers

- You will find more information in the EGI Wiki starting at `https://wiki.egi.eu/wiki/VT-CloudCaps:BestPractices`
- George Reese – Cloud Application Architectures: Building Applications and Infrastructure in the Cloud