EGI-Engage

Relocating VM instances between providers, final specification

D4.4

Date	4 March 2016
Activity	WP4
Lead Partner	CSIC
Document Status	FINAL
Document Link	https://documents.egi.eu/document/2661

Abstract

This document provides a specification to relocate Virtual Machine (VM) instances between providers in the EGI Federated Cloud. Firstly, a general overview of the state of the art regarding the migration of VMs is described in the document, including different strategies for migration, from offline migration to live migration. A special emphasis is set on the support for migration that is currently provided by the most used hypervisors in EGI Federated Cloud (KVM and Xen). Different use cases and scenarios are outlined that may benefit from the ability to migrate VMs across sites. Finally, this document proposes a specification for VM migration to be incorporated in EGI Federated Cloud.

The EGI-Engage project is co-funded by the European Union (EU) Horizon 2020 program under Grant number 654142

COPYRIGHT NOTICE

DELIVERY SLIP

	Name	***Partner/Activity***	***Date***
From:	German Moltó, Miguel Caballer	UPV / WP4	4/02/2016
Moderated by:	Małgorzata Krakowiabn	EGI.eu/NA1	
Reviewed by	Peter Solagna Diego Scardaci Johan Montagnat	EGI.eu/SA1 INFN/JRA1 CNRS IDGC/PMB	21/02/2016 25/02/2016 5/02/2016
Approved by:	AMB and PMB		8/03/2016

DOCUMENT LOG

Issue	***Date***	***Comment***	***Author/Partner***
v1.0	02/12/2015	Initial structure and sections 1, 2, 3	Germán Moltó (UPV)
v1.1	12/12/2015	Sections 4, 5 and 6	Miguel Caballer (UPV)
v1.2	22/12/2015	Overall improvements	Germán Moltó (UPV)
v1.3	29/01/2016	1st Internal review	Alvaro Lopez (CSIC) Matthew Viljoen (EGI.eu)
v1.4	25/02/2016	Apply 1st and 2nd review changes	Alvaro Lopez (CSIC) Germán Moltó (UPV)
v1.5	29/02/2016	Apply 3rd review changes	Alvaro Lopez (CSIC) Germán Moltó (UPV)
FINAL	4/02/2016	Final version	Alvaro Lopez (CSIC)

TERMINOLOGY

A complete project glossary is provided at the following page: http://www.egi.eu/about/glossary/

BLCR	Berkeley Lab Checkpoint/Restart
CMF	Cloud Management Framework
CDMI	Cloud Data Management Interface
EGI	European Grid Initiative
GPGPU	General-Purpose Computing on Graphics Processing Units

IaaS	Infrastructure as a Service
KVM	Kernel-based Virtual Machine
NBD	Network Block Device
OCCI	Open Cloud Computing Interface
ONE	Open Nebula
OVF	Open Virtualization Format
GLUE	Grid Laboratory Uniform Environment
GPGPUS	General Purpose Graphical Processing Units
PaaS	Platform as a Service
SaaS	Software as a Service
SAN	Storage Area Network
SCP	Secure Copy Protocol
SSL	Secure Sockets Library
TLS	Transport Layer Security
VM	Virtual Machine
VMI	Virtual Machine Image

Contents

Executive summary

Virtual Machine migration is the process of moving an existing Virtual Machine (VM) from one host into a different one. This technique is used with different purposes, such as the relocation of VMs when maintenance operations must be performed in the original host. Several different techniques exists, from the live migration, that does not stop the running VM (so the applications do not stop their execution) to the offline migration, where a suspended or stopped VM is moved between hosts.

However, all these approaches imply migrating a VM within a single resource provider. Besides, several use cases, benefiting from VM migration across different providers, have been identified during the operation of the EGI cloud infrastructure. Such migration feature is not available, neither in the EGI Federated Cloud nor in the existing Cloud Management Frameworks. Therefore, an analysis on the state of the art techniques for VM relocation has been performed.

This document provides a proposal of a procedure for offline migration, i.e. suspended VMs, in the context of the EGI cloud infrastructure. This design is grounded on the current capabilities, features and architecture of the EGI Federated Cloud, so it is being based on the OCCI standard. Taking into account that this functionality is missing in the existing CMFs, this operation does not exist at the management interface level, so an extension to the OCCI cloud standard, is proposed, as well as a fine-grained error management.

1 Introduction

The EGI Federated Cloud is a collection of private clouds and virtualised resources, built around open standards targeting the requirements of the scientific communities, offering an a scalable and flexible research Cloud e-infrastructure. The Federated Cloud relies on the central services from the EGI Core Infrastructure Platform.[1]The EGI Federated Cloud provides scientists and member of research projects with a flexible environment to run their applications and services. For organizations and institutions that are willing to provide Cloud resources, the EGI Federated Cloud is the proper way to link those institutions into a wide research and public European network of resources. Different research communities have approached the EGI Federated Cloud in areas such as Structural Biology, Software Engineering, Astronomy, Ecology or Linguistics[2].

The EGI Federated Cloud is based on multiple open standards such as OCCI, CDMI, OVF and GLUE, supporting heterogeneous deployments and disparate Cloud Management Frameworks, without imposing a particular restriction on the Cloud technology to be deployed, thus fostering the adoption of standard interfaces and services. A federation of Clouds faces numerous challenges such as authentication and authorization, together with the accountability of the usage of resources. However, many opportunities also arise in order to foster the collaboration among sites and improve the user experience when using such federated infrastructure. One of those opportunities is the migration of VMs across sites within the EGI Federated Cloud. The ability to relocate configured resources, in the shape of Virtual Machines, among different sites paves the way for an enhanced user experience for scenarios that may require this functionality, which will be described in this document.

1.1 Purpose of this document

The purpose of the document is to assess the feasibility and provide an implementable design for introducing migration capabilities of Virtual Machines (VMs) across sites, with a special focus on the EGI Federated Cloud. For that, different migration strategies are identified and properly described. Also, a survey of the current state of the art concerning the migration of VMs across different platforms is produced. Different use cases that may benefit from migration capabilities in EGI Federated Cloud are identified and a set of proposals regarding migration of VMs is presented together with a discussion on the issues, limitations and challenges that lay ahead the usage of these techniques.

[1] The EGI Federated Cloud. https://www.egi.eu/export/sites/egi/infrastructure/cloud/fedcloudflyer2.pdf
[2] The EGI Federated Cloud. https://www.egi.eu/export/sites/egi/infrastructure/cloud/fedcloudflyer2.pdf

1.2 Overview of the document

The rest of the document is structured as follows. Section 2 identifies the use cases that may benefit from the ability to migrate VMs in the EGI Federated Cloud. Section 3 gives an overview of the current trends regarding the migration of Virtual Machines. Section 4 is composed of the VM migration design, as well as the needed additions to the OCCI standard and the error management. Section 6 comprises the work plan regarding the different Cloud Management Frameworks and the needed extensions for the OCCI standard, outlining the next steps towards the final implementation. Finally, section 7 summarizes the document, pointing out the main conclusions.

2 Migration use cases

This section identifies the preliminary use cases that will benefit from migrating a Virtual Machine from one source site to a destination site, both belonging to the EGI Federated Cloud. These use cases specifically target offline migration, as discussed in the previous section. For the purpose of this document, we have identified the following use cases, although we expect that more use cases may arise and benefit from this new functionality:

Move Computation Close to Data

Data-intensive applications benefit from data locality, meaning that the application exploiting that data will run much more efficiently if it is executed close to where the data is. Putting computation and data close together may be done in two different ways: moving the data close to computation, or the computation close to the data.

- Bringing data close to computation is assumed to be an expensive task that leads to increased network congestion.
- Bringing computation close to data is generally assumed to be a much better approach, especially regarding large datasets (i.e. Big Data).

In this context, the relocation of Virtual Machines between sites may enable users to move their computations (in the shape of VMs) easily to where the data is being stored, without the need of rebuilding their running instances. This reduces the latency to access the data and it is expected to provide enhanced throughput. A VM can be therefore migrated from one site to another, considering the data to be processed.

Migration to Prevent Redeployment of Software

Complex scientific applications might require the deployment of specific versions of Operating Systems, numerical libraries, external applications together with specific configurations of the execution environment to satisfy the requirements of such applications.

A manual installation of such applications is both time-consuming and error-prone. This is why in the EGI Federated Cloud there is available the EGI Applications Database (AppDB)[3], a central service that stores and provides to the public information about:

- Software solutions, in the form of native software products, virtual appliances and/or software appliances.
- The programmers and the scientists who are involved, and

[3] https://appdb.egi.eu

- Publications derived from the registered solutions.

Therefore, Virtual Machine Images with pre-installed applications are available in the AppDB to ease the process of deployment of specific software applications in the EGI Federated Cloud. In addition, it is possible to deploy a plain OS VM and to perform the installation of the applications either manually or in an automated fashion. Indeed, with the advent of automatic configuration (DevOps) tools such as Ansible[4], Puppet[5] or Chef[6] it is becoming more common to create high-level recipes in which the user describes the software and configuration requirements of their applications. This way, the process of application deployment is automated. However, the usage of DevOps tools requires specific software skills that may be out of reach for existing scientists who may be responsible for the deployment of certain applications. Indeed, it is still common to find manual configuration and deployments of customized virtual infrastructures. Of course, in IaaS Clouds, the deployment of a Virtual Machine turns the user into the administrator of the VM, enabling the installation and configuration of software to fit the particular needs of the user. Also, in the specific cases or applications with numerous dependencies that have to be installed from sources, the compilation and installation process is also time-consuming.

In this use case, a user that has a VM deployed and running in one of the sites of the EGI Federated Cloud specifically customized with his/her application and execution environment wants to migrate this VM to another site without taking a snapshot of the VM. This is required when a user needs to quickly migrate the VM from one location due to an upcoming downtime, or because the existing CMF does not provide support for snapshots. The user expects to have her VM automatically transferred to the destination site and to be able to access it, of course, with changes in the network configuration (a different IP address, for example). The very same contents of the VM are expected to be maintained upon migration.

Migration to Prevent the Loss of Execution Progress of a Long Running Application

Consider the case of a long running application that takes several days to be executed on a certain VM provisioned from a site in the EGI Federated Cloud. If a notification that a certain site is going to be decommissioned before the expected finish date of the job, then measures have to be taken in order to prevent the loss of the progress of the execution of the job. If the application supports the ability to perform checkpoints, then a viable approach would be to checkpoint the application before performing a migration of the VM to another site. This would allow having the very same execution environment and the very same application state recovered from the latest checkpoint without requiring the burden of re-deploying the application in another VM provisioned from a

[4] http://www.ansible.com
[5] http://puppetlabs.com/
[6] http://chef.io/

different site. Note that the application has to be checkpointed because the VM will be powered off in order to perform an offline migration.

Checkpointing can be application-dependent or it can be seamlessly provided with the help of different third-party tools such as BLCR (Berkeley Lab Checkpoint/Restart). This library aims at providing a robust, production quality implementation that checkpoints a wide range of applications, without requiring changes to be made to application code. This work focuses on checkpointing parallel applications that communicate through MPI, although sequential applications can be checkpointed as well.

From the point of view of the user, the following actions would happen:

1. The user connects to the VM and checkpoints the application.
2. The user initiates the migration process
3. Virtual machine is migrated to the destination site and it is started
4. The user connects to the VM and resumes the application.

3 Overview of techniques to migrate Virtual Machines

This section starts with a definition of the different approaches for Virtual Machine migration currently described in the scientific literature. Then, it provides a thorough description of these solutions.

3.1 Definitions

The migration of a VM is the ability to move such VM from one physical host (host 1) to another physical host (host 2). This is possible because guest virtual machines are running in a virtualized environment abstracting the execution hardware. According to "KVM live migration"[7], migration is useful for:

Load balancing. Virtual machines can be moved to host physical machines with lower usage when their host physical machine becomes overloaded, or another host physical machine is under-utilized.

- **Hardware independence**. When we need to upgrade, add, or remove hardware devices on the host physical machine, we can safely relocate guest virtual machines to other host physical machines. This means that guest virtual machines do not experience any downtime for hardware improvements.
- **Energy saving**. Virtual machines can be redistributed to other host physical machines and can thus be powered off to save energy and cut costs in low usage periods.
- **Geographic migration**. Virtual machines can be moved to another location for lower latency or in case of incidents in the current sites.

According to different authors [8,9] there are three types of migration:

Cold migration (also known as Offline migration). This procedure involves shutting down the VM on host 1, transferring the VM information to host 2 and, finally, restarting the VM on host 2.

- **Warm migration**. This procedure involves suspending the VM on host 1, transferring the contents of the RAM and CPU registers to host 2 and, later, resume the VM on host 2.

[7] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Administration_Guide/chap-Virtualization_Administration_Guide-KVM_live_migration.html
[8] Live migration in Oracle VM Server SPARC 2.1. https://blogs.oracle.com/jsavit/entry/live_migration_in_oracle_vm
[9] https://nsrc.org/workshops/2014/sanog23-virtualization/raw-attachment/wiki/Agenda/migration-storage.pdf

- **Live migration**. This procedure involves copying across the RAM to host 2 while the VM continues to run on host 1, mark the dirty RAM pages and re-copy and, finally, perform a brief suspension (in the order of a second) for the final copy.

Cold Migration

According to VMware vSphere "Cold Migration"[10] section, cold migration is the migration of a powered-off virtual machine. With this approach, it is needed to move the associated disks from one datastore to another, therefore virtual machine images are not required to be on shared storage. However, the virtual machine to migrate must be powered off prior to beginning the cold migration process. Notice that if a virtual machine is configured to have a 64-bit guest operating system an error will arise if the target host does not support 64-bit operating systems. Indeed, CPU compatibility checks do not apply when migrating a virtual machine with cold migration.

Warm Migration

According to VMware vSphere "Migrating a Suspended Virtual Machine"[11] section, when migrating a suspended virtual machine, it is also needed to move the associated disks from one datastore to another. The virtual machines are not required to be on shared storage. When you migrate a suspended virtual machine, the new host for the virtual machine must meet CPU compatibility requirements (i.e. CPU architecture), because the virtual machine must be able to resume executing instructions on the new host.

Live Migration

Hypervisors such as KVM, Xen and VMware ESXi support live migration among physical hosts without any downtime provided that i) the Virtual Machine Image is located on a shared storage among the source and destination physical machines, and ii) both physical machines reside in the same subnet.

Live Migration involves copying the memory pages from source to destination machines. The time involved in the live migration depends on the memory size of the VM but it is much more dependent on the rate at which dirty pages are created, which is related to the application usage of memory. As Clark et al.[12] noted, if memory-intensive applications running on the VM produce dirty pages faster than the rate of copying, then the copy of pages work will be in vain.

---

[10] https://pubs.vmware.com/vsphere-50/index.jsp?topic=%2Fcom.vmware.vsphere.vcenterhost.doc_50%2FGUID-326DEC3C-3EFC-4DA0-B1E9-0B2D4698CBCC.html

[11] https://pubs.vmware.com/vsphere-50/index.jsp#com.vmware.vsphere.vcenterhost.doc_50/GUID-F7B32670-A3DB-412E-B778-F5EDBEC6138B.html

[12] Clark, Christopher, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. 2005. "Live Migration of Virtual Machines." In NSDI'05 Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, USENIX Association, 273–86. http://dl.acm.org/citation.cfm?id=1251203.1251223 (March 25, 2015).

State of the Art on Migration of VMs

Cloud computing is a network-based computing paradigm where resources, software, data and information are provided through the Internet as a shared pool of services (following the IaaS - Infrastructure as a Service, PaaS - Platform as a Service or SaaS - Software as a Service model), which may be packaged in virtual machines. This technology is related to high efficiency computing and high power computing by means of storage, memory, processing and bandwidth. Thus, enterprises get their applications up and running faster, with improved manageability and less maintenance. It gives clients a tool to more rapidly adjust resources to meet fluctuating and unpredictable business demand (scale up when computing needs increase and then scale down when demands decrease).

Cloud computing providers are entities which offer production services and components via a private (private cloud) or a public network (cloud). The architectures allow the deployment of resources (e.g., networks, servers, etc.) for data storing or for applications support. By aggregating large amounts of resources, providers are becoming a highly demanded service or utility due to the advantages of high computing power, cheap cost of offered services, high performance, scalability, accessibility as well as availability.

Several key challenges take place in the specific case of a federated Cloud infrastructure, such as the EGI Federated Cloud. One of these challenges is the relocation of virtual machines (VMs) instances between providers.

The fully automatic relocation of service instances between cloud providers requires software libraries, standards and frameworks that abstract cloud computing services to common interfaces where data optimisation, runtime architecture adaptation, and goal-oriented service instance relocation must play an important role in proposed solutions. This process, intuitively, takes place in several steps: stopping/suspending the virtual machine in the source provider, creating a copy of it in the destination provider; moving the data from the block storage to the new virtual machine; and finally performing the reconfiguration. Furthermore, there are several actors involved in this process, such as cloud brokers to discover cloud provider services; migrators to support adaptation, communication channels, among others (see "An Open Framework for Relocation of Cloud Services"[13] for details).

In "An Open Framework for Relocation of Cloud Services."[14], authors present an interesting point of view when considering VMs migration. They describe a framework, called FluidCloud, for addressing relocation of IaaS-based service instances, relocation of PaaS-based service instances

[13]Andy Edmonds, Thjis Metsch, Erik Elmroth, Jamie Marshall, and Petov Ganschosov. FluidCloud: An Open Framework for Relocation of Cloud Services. In The 5th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud'13), June 25-26, 2013, San Jose, CA, 2013.

[14]Andy Edmonds, Thjis Metsch, Erik Elmroth, Jamie Marshall, and Petov Ganschosov. FluidCloud: An Open Framework for Relocation of Cloud Services. In The 5th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud'13), June 25-26, 2013, San Jose, CA, 2013.

and service instances adaptation: IaaS to PaaS. To achieve these goals, both the source provider and the destination provider must support APIs that allow these communications. A proof-of-concept, implemented in Python, considering the relocation of IaaS components, and runtime cost, between OpenStack[15] and SmartOS[16] was developed to prove VM migration feasibility and the designed architecture capability.

Focusing on cold migration (or offline migration), this can be achieved effectively between hosts that have the same hypervisor. However, migrating a powered-off VM between hosts with different hypervisors typically requires a change in the underlying virtual machine image format. There exist tools that aim at converting virtual machines across hypervisors. This is the case of virt-v2v[17] that converts virtual machines from Xen and VMware hypervisors to run on KVM. According to the documentation in "Red Hat Enterprise Linux 6. V2V Guide"[18] virt-v2v can currently convert virtual machines running Red Hat Enterprise Linux and Windows on Xen, KVM and VMware ESX / ESX(i) hypervisors. virt-v2v enables para-virtualized (virtio) drivers in the converted virtual machine if possible. It supports the following guests: Red Hat Enterprise Linux (versions 3.9, 4, 5 and 6) and Windows (XP, Vista, 7, Server 2003, Server 2008). The following hypervisors are supported: Xen and KVM (all versions released by Red Hat) and VMware ESX / ESX(i) versions 3.5, 4.0, 4.1, 5.0, 5.1. There is further information on how to convert a VM created for VMware to be run with KVM[19].

Concerning warm migration, problems may arise when trying to migrate a suspended VM between different sites. Indeed, the usage of different source and destination architectures and the usage of disparate hypervisors that can be found in a federated Cloud infrastructure discourage the usage of this approach. There are known issues concerning warm migration in different hypervisors such as the case of Oracle VM Server for SPARC (formerly known as Logical Domains) , where a failure can occur when there is a problem migrating the runtime state of the logical domain channels of the guest. This problem has occurred when the migrating domain has an unplumbed virtual network interface or has a sparse memory configuration[20]. Also, suspending a VM typically uses CPU-specific power management states. Therefore, if a VM is migrated to a physical host that lacks these power management states, the VM will not restart properly, as indicated in "Troubleshooting the top five virtual machine migration errors"[21]. Indeed, before

[15] Openstack: https://www.openstack.org
[16] Smartos: https://smartos.org
[17] http://libguestfs.org/virt-v2v/
[18] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/V2V_Guide/index.html#sect-V2V_Guide-We_Need_Feedback
[19] http://www.linux-kvm.org/page/How_To_Migrate_From_Vmware_To_KVM
[20] Warm Migration Can Fail With an Unknown migration failure Message. http://docs.oracle.com/cd/E19604-01/821-0404/auto50/index.html
[21] http://searchservervirtualization.techtarget.com/tip/Troubleshooting-the-top-five-virtual-machine-migration-errors

migrating a VM, the required extensions have to enable in the BIOS, as is the case of the virtualization features such as Intel VT or AMD-V.

Indeed, there exist migration procedures in order to transition Virtual Machines from one hypervisor to another that advice on using a cold migration approach. This is the case of the Xen to KVM Migration Guide[22] which describes a procedure for server administrators to migrate their existing Xen based environments to KVM. The document states that, as of now, there are no mature tools to automatically convert Xen VMs to KVM. There is, however, a technical solution that helps convert Xen virtual machines to KVM which is, basically: i) Make a backup copy of the original Xen VM Guest; ii) OPTIONAL: Apply changes specific to paravirtualized guests; iii) Obtain information about the original Xen VM Guest and update it to KVM equivalents and iv) shut down the guest on the Xen host, and run the new one under the KVM hypervisor. They specifically indicate that the Xen to KVM migration cannot be done live while the source VM Guest is running. Therefore, before running the new KVM-ready VM Guest, users are advised to shut down the original Xen VM Guest.

Live migration of VMs, is a special case of migrating VMs where minimal or no service disruption is introduced. In this way, Clark et al.[23] present the design, implementation and evaluation of a high-performance OS migration built on top of the Xen[24] virtual machine monitor (VMM). They consider the migration process as a transactional interaction between the two hosts involved by following these six steps:

Pre-Migration, where an active VM on physical host A is selected. Then, a target host may be preselected where the resources required to receive migration will be guaranteed;

1. Reservation, where a request is issued to migrate an OS from host A to host B. The necessary resources must be available on B and a VM container of that size must be reserved
2. Iterative Pre-Copy, where all pages are transferred from A to B, during the first iteration.
3. Stop-and-Copy, where the running OS instance at A is suspended and its network traffic is redirected to B. CPU state and any remaining inconsistent memory pages are then transferred. At the end of this stage there is a consistent suspended copy of the VM at both A and B. The copy at A is still considered to be primary and is resumed in case of failure.

[22] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Administration_Guide/chap-Virtualization_Administration_Guide-KVM_live_migration.html

[23] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. pages 273–286, may 2005.

[24] Xenproject: http://xenproject.org, 2015

4. Commitment, where host B indicates to A that it has successfully received a consistent OS image. Host A may now discard the original VM, and host B becomes the primary host;
5. Activation, where the migrated VM on B is activated.

Thus, they integrated live OS migration into the Xen virtual machine monitor, enabling rapid movement of interactive workloads within clusters and data centers, with minimal impact on running services and reducing total downtime to below discernible threshold.

Following the live migration approach, taking into account suspension of VMs, Kang et al.[25] designed and developed a practical prototype of a best-effort middleware for precopy-based VM migration across datacenters. They take advantage of the mobility of VMs to implement a disaster-recovery system[26] which can be deployed on a large-scale testbed such as Emulab[27] or CloudLab[28]. In the same sense, Hirofuchi et al. [[29], [30], [31]] developed an advanced VM consolidation system, enabling relocation of VMs onto new server nodes without stopping guest operating systems, based on postcopy live migration. Experiments were performed, with instances of KVM[32] and Amazon EC2[33], where feasibility was evaluated. Results show that the system achieves a higher degree of performance assurance than using precopy migration.

In general terms, there are no works that explicitly treat the topic of relocating VM instances across providers. Most of the works focus on the intent of reducing time of migration, improving data transfer (disk blocks, memory pages, etc.), improving performance assurance and energy efficiency, improving both precopy and postcopy VM migration. Therefore, they indirectly stop, suspend or checkpoint the VM that will migrate.

---

[25] Tae Seung Kang, Mauricio Tsugawa, Andrea Matsunaga, Takahiro Hirofuchi, and Jose A.B. Fortes. Design and Implementation of Middleware for Cloud Disaster Recovery via Virtual Machine Migration Management. In 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, pages 166–175. IEEE, dec 2014

[26] Tae Seung Kang, Mauricio Tsugawa, Jose Fortes, and Takahiro Hirofuchi. Reducing the Migration Times of Multiple VMs on WANs Using a Feedback Controller. In 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, pages 1480–1489. IEEE, may 2013

[27] https://www.emulab.net

[28] Cloudlab: https://www.cloudlab.us

[29] Takahiro Hirofuchi, Hidemoto Nakada, Satoshi Itoh, and Satoshi Sekiguchi. Reactive consolidation of virtual machines enabled by postcopy live migration. In Proceedings of the 5th international workshop on Virtualization technologies in distributed computing - VTDC '11, page 11, New York, New York, USA, jun 2011. ACM Press

[30] Takahiro Hirofuchi, Hidemoto Nakada, Satoshi Itoh, and Satoshi Sekiguchi. Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension. In 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pages 73–83. IEEE, 2010

[31] Takahiro Hirofuchi, Hidemoto Nakada, Satoshi Itoh, and Satoshi Sekiguchi. Reactive Cloud: Consolidating Virtual Machines with Postcopy Live Migration. IMT, 7(2):614–626, 2012

[32] KVM: http://www.linux-kvm.org, 2015

[33] Amazon EC2: https://aws.amazon.com/ec2, 2015

Concerning live migration, the VM disk image has to be accessible from the new host after the migration. Therefore, if no shared storage is available, then the disk image has to be copied across hosts which is a slow process limiting the approach to cold migration. An alternative to a shared storage is to use a distributed storage. For that, different projects exist such as Sheepdog[34], which is a distributed object storage system for volume and container services, and manages the disks and nodes intelligently. Sheepdog features ease of use, simplicity of code and can scale out to thousands of nodes. There exists also Ceph[35] where, according to the documentation[36], the most frequent Ceph Block Device use case involves providing block device images to virtual machines. For example, a user may create a "golden" image with an OS and any relevant software in an ideal configuration, then create a snapshot of the image, and finally clone the snapshot (usually many times). The ability to make copy-on-write clones of a snapshot means that Ceph can provision block device images to virtual machines quickly, because the client doesn't have to download an entire image each time it spins up a new virtual machine. Other distributed storage systems are GlusterFS[37] or even the DRBD[38] replicated storage, as used in Ganeti[39].

Concerning warm migration, problems may arise when trying to migrate a suspended VM between different sites. Indeed, the usage of different source and destination architectures and the usage of disparate hypervisors which can be adopted in a federated Cloud infrastructure discourage the usage of this approach. There are known issues concerning warm migration in different hypervisors such as the case of Oracle VM Server for SPARC (formerly known as Logical Domains), where a failure can occur when there is a problem migrating the runtime state of the logical domain channels of the guest. This problem has occurred when the migrating domain has an unplumbed virtual network interface or has a sparse memory configuration[40]. In addition, suspending a VM typically uses CPU-specific power management states. Therefore, if a VM is migrated to a physical host that lacks these power management states, the VM will not restart properly, as indicated in "Troubleshooting the top five virtual machine migration errors"[41]. Indeed, before migrating a VM, the required extensions have to enable in the BIOS, as is the case of the virtualization features such as Intel VT or AMD-V.

[34] https://sheepdog.github.io/sheepdog/
[35] http://docs.ceph.com/docs/master/
[36] QEMU and Block Devices. http://docs.ceph.com/docs/hammer/rbd/qemu-rbd/
[37] http://www.gluster.org/
[38] http://drbd.linbit.com/
[39] http://docs.ganeti.org/ganeti/2.6/html/install.html#installing-drbd
[40] Warm Migration Can Fail With an Unknown migration failure Message. http://docs.oracle.com/cd/E19604-01/821-0404/auto50/index.html
[41] http://searchservervirtualization.techtarget.com/tip/Troubleshooting-the-top-five-virtual-machine-migration-errors

There exist migration procedures in order to transfer Virtual Machines from one hypervisor to another adopting a cold migration approach. This is the case of the Xen to KVM Migration Guide[42] which describes a procedure for server administrators to migrate their existing Xen based environments to KVM. The document states that, as of now, there are no mature tools to automatically convert Xen VMs to KVM. There is, however, a technical solution that helps convert Xen virtual machines to KVM which is, basically: i) make a backup copy of the original Xen VM Guest; ii) optional: apply changes specific to paravirtualized guests; iii) obtain information about the original Xen VM Guest and update it to KVM equivalents and iv) shut down the guest on the Xen host, and run the new one under the KVM hypervisor. They specifically indicate that the Xen to KVM migration cannot be done live while the source VM Guest is running. Therefore, before running the new KVM-ready VM Guest, users are advised to shut down the original Xen VM Guest.

Focusing on cold migration (or offline migration), this can be achieved effectively between hosts that have the same hypervisor. However, migrating a powered-off VM between hosts with different hypervisors typically requires a change in the underlying virtual machine image format. There exist tools that aim at converting virtual machines across hypervisors. This is the case of virt-v2v[43] that converts virtual machines from Xen and VMware hypervisors to run on KVM. According to the documentation in "Red Hat Enterprise Linux 6. V2V Guide"[44] virt-v2v can currently convert virtual machines running Red Hat Enterprise Linux and Windows on Xen, KVM and VMware ESX / ESX(i) hypervisors. virt-v2v enables para-virtualized (virtio) drivers in the converted virtual machine if possible. It supports the following guests: Red Hat Enterprise Linux (versions 3.9, 4, 5 and 6) and Windows (XP, Vista, 7, Server 2003, Server 2008). The following hypervisors are supported: Xen and KVM (all versions released by Red Hat) and VMware ESX / ESX(i) versions 3.5, 4.0, 4.1, 5.0, 5.1. There is further information on how to convert a VM created for VMware to be run with KVM[45].

Notice that the requirements for live migration are commonly (and easily) met in an on-premises Cloud deployment, but are obviously not met on a federated Cloud infrastructure such as EGI Federated Cloud. Therefore, live and warm migration will not be considered as a feasible strategy for EGI Federated Cloud in the scope of this document.

3.2 Migration Support of Popular Hypervisors

This subsection collects the current support to migration provided by the most popular hypervisors adopted in the EGI cloud federation. This is of special interest for the purpose of this

[42] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Administration_Guide/chap-Virtualization_Administration_Guide-KVM_live_migration.html
[43] http://libguestfs.org/virt-v2v/
[44] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/V2V_Guide/index.html#sect-V2V_Guide-We_Need_Feedback
[45] http://www.linux-kvm.org/page/How_To_Migrate_From_Vmware_To_KVM

document since the ability to integrate a migration scheme in EGI Federated Cloud is very much dependent on the capabilities supported by the underlying hypervisors.

3.2.1 KVM

According to the KVM documentation[46], KVM supports both offline and live migration. Upon successful completion, the migrated VM continues to run on the destination host. Of course, you can migrate a guest between an AMD host to an Intel host and back. Naturally, a 64-bit guest can only be migrated to a 64-bit host, but a 32-bit guest can be migrated at will. The following requirements have to be met:

- The VM image is accessible on both source and destination hosts (located on a shared storage, e.g. using NFS).
- It is recommended that an images directory would be found on the same path on both hosts
- The source and destination hosts must be on the same subnet.
- The guest on the destination must be started the same way it was started on the source.

3.2.2 Xen

Like KVM, Xen supports offline (cold) and live (hot) migration ([47],[48],[49],[50]) with almost no service interruption. In this case, offline migration suspends the guest on the original host, transfers it to the destination host and then resumes it once the guest is fully transferred. The persistent storage for the VMs must be shared at the same location or without sharing by using Network Block Devices (NBD). The latter allows one to share the disk of a host to another. The disadvantage of this is that whilst the VM can be migrated, we cannot shutdown the host that contains the storage. Some preliminary considerations regarding the VM Host Server should be taken into account:

- All VM Host Server systems should use a similar CPU. The frequency is not so important, but they should be using the same CPU family.
- All resources that are used by a specific guest system must be available on all involved VM Host Server systems. This means that the network bridges must be in the same subnet (cold migration between different subnets will work but will most likely need to have its networking reconfigured), and all used block devices must exist on both VM Host Server systems.

[46] KVM Migration. http://www.linux-kvm.org/page/Migration
[47] http://wiki.xenproject.org/wiki/Migration
[48] https://www.suse.com/documentation/sles-12/book_virt/data/sec_xen_manage_migrate.html
[49] http://wiki.prgmr.com/mediawiki/index.php/Chapter_9:_Xen_Migration
[50] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Virtualization/chap-Virtualization-Xen_live_migration.html

- Using special features like PCI Pass-Through may be problematic. These should not be implemented when deploying for an environment that should migrate VM Guest systems between different VM Host Server systems.
- For fast migrations, a fast network is mandatory. If possible, use GB Ethernet and fast switches. Deploying VLAN might also help to avoid collisions.

To enable migration with Xen, passwordless SSH connections should be enabled between the hosts so as to start the migrating process.

3.2.3 VMware

VMware, through the vSphere and vMotion platforms, supports several migration types[51] such as migration of a suspended VM where, optionally, we can relocate configuration and disk files to new storage location; and, migration of powered-on virtual machine to a new host with the ability to move virtual disks, virtual machine's storage or configuration file to a new datastore or VM without interruption. In this environment, both migration types sometimes are called hot migration, because they allow migration of a virtual machine without powering it off.

For live migration, which can be performed only between hosts in the same datacenter that are managed by a vCenter Server system (for moving it to a different datacenter we must power off the virtual machine), the system performs the following tasks:

- Uses QueryVMotionCompatibility_Task function to check two hosts are compatible.
- Uses CheckMigrate_Task function to check whether migration is feasible. For example, if two hosts are not compatible, virtual machines cannot be migrated from one to the other.
- Uses CheckRelocation_Task function to check whether relocation is possible.

Migration with storage, across vMotion, allows us to move a running virtual machine from one VMFS volume to another. All datastore types are supported, including local storage, VMFS, and NAS (network attached storage). We can place the virtual machine and all its disks in a single location, or select separate locations for the virtual machine configuration file and each virtual disk. The virtual machine remains on the same host during the process.

3.3 Discussion on the Migration Approaches

Once reviewed the state of the art regarding migration of VMs and the current support provided by hypervisors, it is important to define the feasible migration strategy that could be adopted in a federated cloud infrastructure such as the EGI Federated Cloud.

Warm and live migrations are only feasible to perform when using the same hypervisor technologies. Therefore, implementing such an approach in an heterogeneous Cloud infrastructure as the EGI Federated Cloud is not an option. Indeed, relocating suspended (not

[51] http://pubs.vmware.com/vsphere-60/index.jsp#com.vmware.wssdk.pg.doc/PG_VM_Manage.13.2.html

powered off) VMs between sites can lead to inaccessible VMs after the expensive procedure of transferring the disks and memory state among geographically distant sites. This would be a waste of bandwidth and time for users without the certainty that their VMs would have been successfully migrated from one Cloud deployment to another Cloud deployment,

Moreover, inter-site live and warm migration is not supported by any Cloud Management Framework. In the hypothetical case where all the EGI Federated Cloud providers were using the same hypervisor technology, supporting warm or live migration would require a significant effort in order to modify and adapt the existing CMFs to support it.

Concerning the diversity of hypervisors and the panoply of heterogeneous hardware that is available in such geographically disperse federated infrastructure, offline migration is the safest approach to Virtual Machine migration that can be adopted.

4 VM Migration Design

Once reviewed the current state of the art on migration of Virtual Machines, the capabilities supported by the hypervisors, and the use cases that are foreseen to benefit from this approach, this section proposes a schematic approach for migration of Virtual Machines across sites in EGI Federated Cloud.

As described previously, this section will focus on the migration of stateless (i.e. powered off) VMs. This means that the VMs will be powered off and only the VM disk will be moved to the destination site where the migrated VM must be powered on again. As stated in the previous sections, this is considered offline migration.

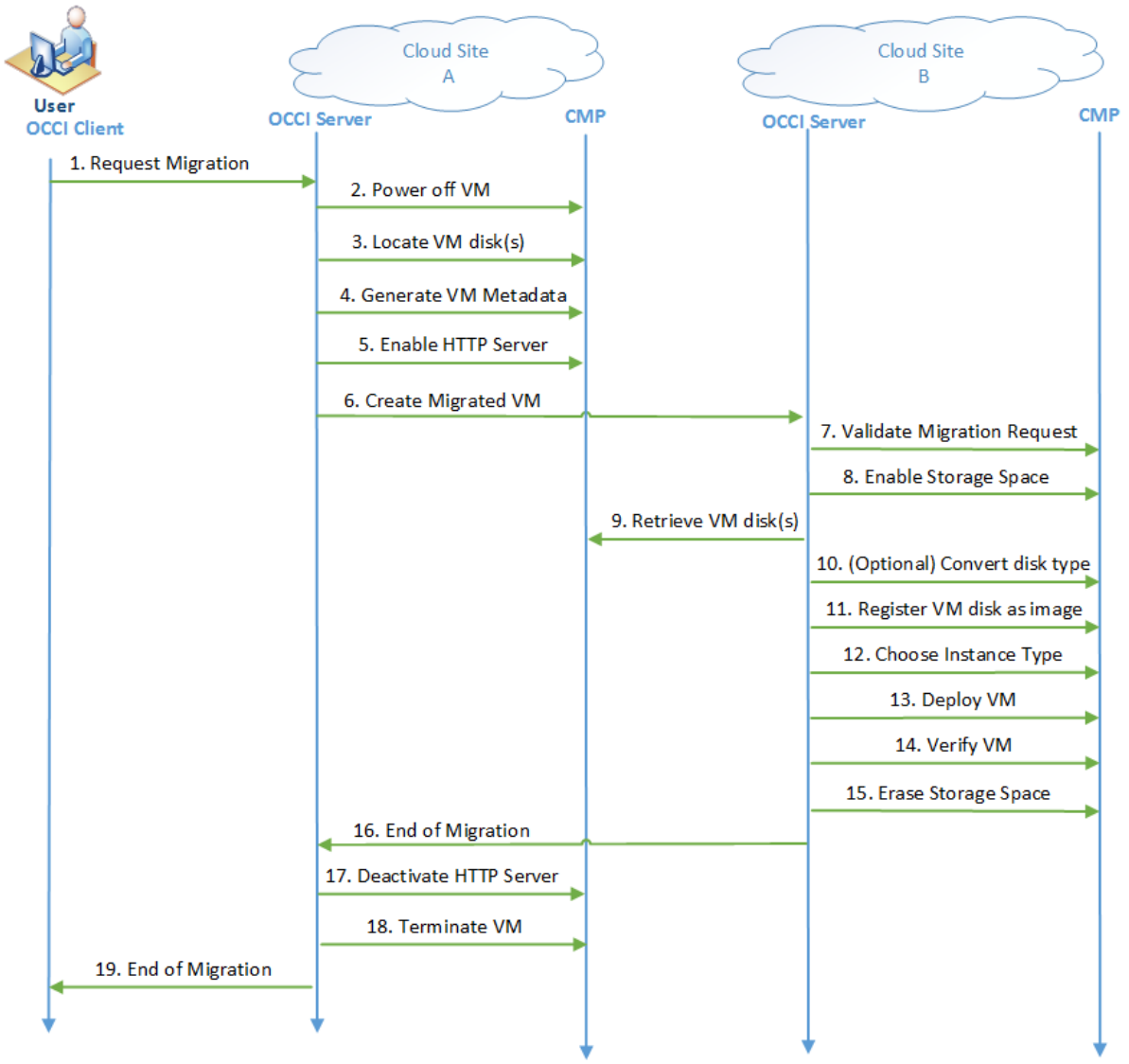
For better interoperability we advocate for introducing the migration support in the OCCI (Open Cloud Computing Interface) specification[52]. OCCI is a RESTful protocol and API for all kinds of management tasks. In the last years it has evolved into a flexible API with a focus on interoperability. Indeed, OCCI is a boundary protocol and API that acts as a service front-end to each site's CMF (e.g. OpenNebula, OpenStack, etc.).

In the EGI Federated Cloud, OCCI is the standard interface to interact with CMFs. Therefore, introducing support for migration in OCCI would be a natural step that would preserve interoperability among sites.

The migration process of a VM from site A to site B will include the steps shown in the following figure. It is assumed that both sites belong to EGI Federated Cloud and, therefore, each one has an OCCI endpoint that has been extended to support the migration lifecycle described in this document. More on the additions required to the OCCI standard to support migration will be discussed in the next sections.

[52] Open Cloud Computing Interface (OCCI). http://occi-wg.org

4.1 Migration process



We provide here an overview of the VM migration cycle. In the following sections we identify the main challenges that lay ahead this procedure.

1. The user initiates the migration procedure by means of her OCCI client. For that a request to the OCCI endpoint of Cloud site A is performed specifying:
   a) The OCCI identifier of the VM that wants to be migrated,
   b) The endpoint of Cloud site B (the destination of the VM).

2. The OCCI server of site A powers off the VM since an offline migration will be performed.
3. The OCCI server of site A locates the disk(s) of the VM. The location of these files depends both on the specific CMF and the storage system employed (shared filesystem, SAN, etc.).
4. The OCCI server of site A generates a set of metadata for the VM including (but not restricted to):
   a) The hypervisor for which the disk images where created.
   b) The size of the VM disk(s)
   c) The format of the VM disk(s) (e.g. qcow2, raw, vmdk, etc.)
   d) The resources attached to the VM (i.e. the VM flavour))
5. The OCCI server of site A generates a temporary or single-use secured HTTP URL to expose the VM disk(s) (i.e. the VM image itself and any data partitions available) so that the destination site can retrieve the disk(s). Basic security measures such as disabling directory listing, use of .htaccess and adoption of SSL/TLS can be employed to further restrict the exposure of this service.
   a) A temporary URL such as https://user:pass@CloudSiteA.org/scratch/long-uuid.img is generated.
6. Once the OCCI server of site A has generated such an URL, the OCCI server of site A contacts the OCCI server of site B to initiate the migration of the VM supplying the following information:
   a) URL where the VM disk(s) are available to be retrieved from site A.
   b) Metadata for the VM (as indicated in step 4).
   c) Instance type, in terms of vCPUs, RAM, and special requirements (e.g. access to GPGPUS).
7. The OCCI server of site B receives the migration request, downloads the metadata and checks that:
   a) Cloud site B supports the special requirements indicated (if any).
   b) It will be possible to change the VM disk type into a compatible format accepted by the CMF and hypervisor configuration at site B.
8. The OCCI server of site B enables a scratch space to temporarily host the VM disk(s). Notice that the amount of data required can be obtained via the metadata information.
9. The OCCI server of site B downloads the image by means of an HTTP/HTTPS client (e.g. using wget / curl).

10. (Optional) If it is necessary, the VM disk is transformed into an appropriate format. For that, tools such a qemu-img can be used. The format of the disk images strongly depends on the underlying hypervisor. Therefore, migration between sites with different hypervisors may require (not always) transforming the disk type.
11. The OCCI server of site B registers in the CMF's catalog of Virtual Machine Images (VMIs). Each CMF has a different internal catalog system for VMIs. For example, OpenStack uses Glance while OpenNebula has its own repository.
12. The OCCI server of site B chooses an appropriate instance type considering the description of the instance type provided in the request. An upper bound, in terms of capacity of resources, should be chosen in order to maintain the level of service for that VM at site B.
13. The OCCI server of site B deploys the VM out of the registered VMI with the appropriate instance type.
14. The OCCI server of site B verifies that the VM is properly executing. For that, a set of general checks is assessed in order to determine whether the VM has successfully booted. This checks would include (but are not limited to): verifying that the VM responds to ping requests or if an SSH server is listening on port 22 or if there is a certain service on a certain port. For that, a subset of the most common-used ports can be employed. In case the VM is on a VLAN inaccessible to the OCCI endpoint, the latter would require confirmation from the user to verify the integrity of the VM.
15. The OCCI server of site B erases the temporary storage allocated for the VM disk(s)
16. The OCCI server of site B notifies the OCCI server of site A that the migration procedure has finished (whether successfully or if an error in any of the previous stages has occurred).
17. The OCCI server of site A deactivates the temporary HTTP server and eliminates the associated temporary files for the VM disk(s).
18. The OCCI server of site A terminates the VM.
19. The OCCI server of site A confirms the OCCI client the end of the migration procedure.

Note that the result of the migration procedure is either:

- The VM is left running on site B.
- The VM is left powered off on site A.

The migration procedure is intended to run as a transaction. Therefore, if any error occurs during any of the aforementioned steps, a rollback process is initiated in order to clean the allocated resources (VMs, files, etc.). Notice that the migration can be asynchronously handled by the OCCI Server of site B by providing the OCCI client with an identifier to track the status of the migration.

4.2 Discussion on the Migration Process

Notice that the migration process that has been described faces some challenges within a federated Cloud infrastructure. This section identifies such challenges.

4.2.1 Instance Types

Concerning step 12 in the VM Migration workflow described in section 4, the OCCI server of site B has to choose an appropriate instance type that has at least the same amount of resources as those of the original instance type at site A. The instance types currently supported at each site in the EGI Federated Cloud are published in the BDII service. This information is publicly available via LDAP mechanisms. However, in OCCI 1.1 there is no common agreement on the semantics of each instance type and the specific types that each site is entitled to support.

For example, the following figure shows the computing capabilities of an instance type named “large”. This is an excerpt of output obtained by the execution of the following command:

```
ldapsearch -x -H ldap://ngiesbdii.i3m.upv.es:2170 -b 'o=glue'
```

```
# resource_tpl#large_fc-one.i3m.upv.es, fc-one.i3m.upv.es_cloud.compute, cloud, UPV-GRyCAP, glue
dn: GLUE2ResourceID=resource_tpl#large_fc-one.i3m.upv.es,
        GLUE2ServiceID=fc-one.i3m.upv.es_cloud.compute,
        GLUE2GroupID=cloud,GLUE2DomainID=UPV-GRyCAP,o=glue
GLUE2ExecutionEnvironmentPlatform: amd64
GLUE2ExecutionEnvironmentCPUVendor: virtual vendor
GLUE2ExecutionEnvironmentLogicalCPUs: 4
GLUE2ResourceManagerForeignKey: fc-one.i3m.upv.es_cloud.compute_manager
objectClass: GLUE2Entity
objectClass: GLUE2Resource
objectClass: GLUE2ExecutionEnvironment
GLUE2ExecutionEnvironmentCPUMultiplicity: multicpu-multicore
GLUE2ResourceID: resource_tpl#large_fc-one.i3m.upv.es
GLUE2ExecutionEnvironmentOSFamily: linux
GLUE2ExecutionEnvironmentVirtualMachine: TRUE
GLUE2ExecutionEnvironmentMainMemorySize: 8196
GLUE2ExecutionEnvironmentConnectivityOut: TRUE
GLUE2ExecutionEnvironmentConnectivityIn: TRUE
GLUE2ExecutionEnvironmentCPUModel: virtual model
GLUE2EntityName: resource_tpl#large
GLUE2ExecutionEnvironmentPhysicalCPUs: 4
GLUE2ExecutionEnvironmentComputingManagerForeignKey:
        fc-one.i3m.upv.es_cloud.compute_manager
```

Notice that information about the instance type is provided. In particular, the main memory size is identified (8 GBytes), the number of virtual CPUs (4), the number of physical CPUs (4) and the ability to receive inbound connections, together with another attributes. However, since there is not a common set of instance types across different Cloud sites, there has to be a process of

matchmaking in order to choose an appropriate instance type that provides at least the same amount of resources as the instance type in the source Cloud site. A list of supported instance types per site can be obtained by querying the BDII service at each site.

An alternative solution would be that all the sites in the EGI Federated Cloud agreed to a common set of instance types. Note that, although the Cloud sites are not enforced to support all of the agreed instance types, at least a common definition of the capabilities supported for each instance type would be beneficial for the process of VM migration.

Consider the case of migrating a VM running in an OpenNebula site (which does not natively support the concept of flavours) to an OpenStack site in which there is support for certain flavours. Flavours are virtual hardware templates defining sizes for RAM, disk, number of cores, and so on. The default OpenStack installation provides five flavours (m1.tiny, m1.small, m1.medium, m1.large and m1.xlarge). Therefore, the resources allocated to the VM must be at least those originally allocated (e.g. use the nearest upper flavour). In EGI Federated Cloud certain flavours are defined through OCCI, although there is no common agreement on the features that each flavour should expose. Using a de facto standard such as the Amazon EC2 instance types[53] or the default OpenStack flavours[54] may be used as starting points to define those instance types.

In this sense, OCCI version 1.2 will include the OCCI Resource Template Profile[55] as well as the JSON rendering of the OCCI objects and attributes. On the one hand, the former consists on a set of well-defined instances of the OCCI compute resource types. The OCCI family of specifications defines a consistent way of defining compute resource requirements through the Compute resource type. In particular, in federated IaaS Clouds, as is the case of the EGI Federated Cloud, it is convenient to provide the user with a consistent set of resource templates (flavours) to be used across the sites within the federation. The adoption of such specification in the context of the EGI Federated Cloud will be very beneficial to the process of VM migration. This way, consistent resource templates would be used for both the source VM and the destination VM. On the other hand, the JSON rendering makes possible to advertise Mixin (i.e. flavours) model attributes (those pre-set defaults such as CPU, RAM memory, disk) so it will be no longer needed to do a manual matchmaking of all the published flavours, being the information directly available at the OCCI endpoint.

4.2.2 Capacity Leasing

In addition, the concept of reservation is not currently very much supported by the CMFs. In the case of OpenStack, there are works that aim at including capacity reservation, although they are not production ready. This is the case of Blazar[56], OpenStack related Reservation-as-a-Service

[53] EC2 Instance Types. https://aws.amazon.com/ec2/instance-types/?nc1=h_ls
[54] http://docs.openstack.org/openstack-ops/content/flavours.html
[55] Draft of OCCI 1.1 profile for VM templates/flavours. https://www.ogf.org/pipermail/occi-wg/2015-March/003599.html
[56] https://wiki.openstack.org/wiki/Blazar

project. A Blazar user can request the resources of cloud environment to be provided ("leased") to his project for specific amount on time, immediately or in future. Both virtual (Instances, Volumes, Networks) and hardware (full hosts with specific characteristics of RAM, CPU and etc.) resources can be allocated via "lease". This would allow for virtual instance reservation, which mostly looks like usual instance booting for user - he/she only passes special hints to Nova containing information about future lease - lease start and end dates, its name, etc. Special Nova API extensions parse this parameter and use them to call Blazar, passing to it the ID of just created instance. By default, it starts lease at the moment of request and gives it one month of lifetime.

Concerning OpenNebula, capacity leasing was initially supported by means of Haizea[57], opensource virtual machine-based lease management architecture. Haizea is a piece of software that, in combination with the OpenNebula virtual infrastructure manager, can be used to manage a Xen, KVM, or VMWare cluster, allowing you to deploy different types of leases that are instantiated as virtual machines (VMs). However, the latest release of Haiza was in 2009 and for OpenNebula 1.4. It does not seem that such capacity leasing exists in modern versions of OpenNebula.

Without capacity leasing mechanisms available at the CMF, there exists a chance that once the disk has been transferred to the destination site the VM cannot be started because there are no available resources (in terms of memory or CPUs). This could be mitigated by deploying at the destination site a transient VM with the very same resource requirements, to be executed during the transfer. Once the migrated VM has to be powered up at the destination site, you just terminate the transient VM to be able to accommodate the migrated VM. Note, however, that by no means this represents a guarantee that the required resources will be available when the VM is deployed.

4.2.3 VM disk transfer

Concerning the VM disk transfer, different approaches may be used in order to perform the disk transfer between sites. The proposed solution uses the HTTP protocol as it is a simple and fast way to setup a temporary server but other protocols can be used. GridFTP[58] would be the most high-performance approach to such data transfer. In particular, GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. The GridFTP protocol is based on FTP, the highly-popular Internet file transfer protocol. It comprises a set of protocol features and extensions defined already in IETF RFCs and it includes a few additional features to meet requirements from current data grid projects.

[57] http://haizea.cs.uchicago.edu
[58] GridFTP. http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/

4.3 OCCI extensions

The OCCI specification, acting as a homogeneous entry point to different CMFs, is the ideal place in which introducing support for the migration operation. The document "Open Cloud Computing Interface – Infrastructure"[59] contains the definition of the OCCI Infrastructure extension for the IaaS domain. It defines additional resource types, their attributes and the actions that can be taken on each resource type.

In particular, this document defines the actions that can be applied to the compute type instances, which represent a Virtual Machine, as showed in 1.

Table 1. Actions for compute type instances

Action Term	*Target State*	*Attributes*
start	Active	-
stop	Inactive	method={graceful, acpioff, poweroff}
restart	active (via stop and start chain)	method={graceful,warm,cold}
suspend	suspended	method={hibernate,suspend}

No migration capabilities of VMs are currently supported by OCCI. Therefore, we need to add two operations to the standard in order to support the workflow identified in section 4. The first one, to initiate the migration process, corresponding to step 1 in the workflow. The second one, to create the VM on the destination site once its disk has been migrated, corresponding to step 6 in the workflow.

Initiate Migration

OCCI is a REST API that defines resource types on which HTTP operations (GET, PUT, POST, DELETE) can be invoked. Therefore, to initiate the migration procedure for a certain VM, one would use the OCCI client to invoke this operation on the corresponding resource of a particular VM identifier specifying the action "migrate".

```
POST /compute/12345?action=migrate
```

Moreover, the parameter "endpoint" must be specified in the body of the request to include the endpoint of the destination site:

```
X-OCCI-Attribute: endpoint = <endpoint_url>
```

Create Migrated VM

To create a VM whose disk (or disks) has been migrated to the destination site, the OCCI server at the source site would invoke this operation of the OCCI server at site B via a POST method to /compute, which is employed to create a new VM, but specifying in the body of the request:

[59] Open Cloud Computing Interface - Infrastructure. https://www.ogf.org/documents/GFD.184.pdf

X-OCCI-Attribute: occi.compute.vmdisk.location = <url>

X-OCCI-Attribute: occi.compute.vmdisk.metadata = <metadata>

Notice that currently, OCCI already allows including the information concerning the instance type of the VM. For example, the following excerpt shows the content of the body for a REST request to create a VM:

```
Category:compute;scheme="http://schemas.ogf.org/occi/infrastructure#";
        class="kind"
Category:compute;scheme="http://opennebula.org/occi/infrastructure#";
        class="mixin"
Category:small;scheme="http://fedcloud.egi.eu/occi/infrastructure/
        resource_tpl#";class="mixin"
Category:uuid_test_0;scheme="http://occi.fc-one.i3m.upv.es/occi/
        infrastructure/os_tpl#";class="mixin";
X-OCCI-Attribute: occi.core.id="1"
X-OCCI-Attribute: occi.core.title="one-1"
X-OCCI-Attribute: occi.compute.architecture="x64"
X-OCCI-Attribute: occi.compute.cores=1
X-OCCI-Attribute: occi.compute.memory=2
```

Notice that the following attributes:

- occi.compute.architecture
- occi.compute.cores
- occi.compute.memory

already define the capabilities of the instance type.

4.4 Error Handling

The migration process should be transactional. Either it worked and the result is the VM running on the remote Cloud site or the VM is kept running in the initial site. This subsection identifies the possible foreseen errors and a proposal for handling them.

- *Unauthorized:* This error occurs when the user that requests the migration of a VM (or the site in charge to manage the migration) is not authorized to perform such operation involving a remote site (e.g. the user is not authorized on the destination site). The migration procedure is aborted and the VM is not powered off. Authorization rights have to be checked at the beginning of the migration process.
- *Unable to Meet Requirements:* This error arises when a request involves a VM whose requirements cannot be fulfilled by the destination site. For example, a 64-bit guest cannot be migrated to a Cloud site where only 32-bit hosts exist. The migration procedure is aborted and the VM is not powered off. This has to be checked at the beginning of the migration process to ensure that the destination site will support the features requested by the VM.

- *Insufficient Resources at Destination Site:* This error arises whenever a migration of a VM with certain resources allocated (basically in terms of amount of memory and number of vCPUs) is requested and the destination site does not support an instance type with that amount of resources.
- *Not Enough Resources:* This error arises whenever a VM cannot be migrated because the destination site does not have enough resources to fulfil the request.
- *Data Transfer Error:* This is triggered when an error in the data transfer occurs, possibly because of a transient network outage. After retrying the operation several times, including resuming the data transfer from the point where it was left when the error occurred to avoid wasting time, the operation would fail.
- *Unable to Resume Virtual Machine:* This error arises whenever the VM cannot be resumed at the destination site, possibly because the process of image conversion to a valid format for the destination hypervisor failed.
- *Verification of VM failed:* This error arises when the migrated VM at the destination site has been started but the verification process fails.

In all these cases the migration procedure is aborted and a rollback process is initiated in order to clean the allocated resources (VMs, files, etc.) maintaining the original VM at site A powered off.

5 Relocation of Virtual Machines: Roadmap

The implementation of this new functionality requires the definition of an OCCI extension, targeting the current OCCI 1.2 extension. After the draft document is ready and it has passed a public comment phase, the functionality will be implemented in the existing client tools and in the current OCCI implementations available: rOCCI[60], ooi[61] and Synnefo[62].

The overall workplan, with tentative dates, for the fulfilment of this task is the following:

1. OCCI extension draft design – March 2016 – June 2016:
   a) Design a draft version of the formal document for the OCCI VM migration extensions.
   b) Public comment of the OCCI extensions.
   c) **Output:** OCCI extensions draft.
2. OCCI extensions first implementation – July 2016 – October 2016:
   a) Implement the first version of the VM relocation mechanism, as described in Section 4.
   b) Initial client tools implementation
   c) **Output:** OCCI implementations, supporting the VM relocation extension.
   d) **Output:** OCCI CLI tools supporting migraiton.
3. Testing phase – September 2016:
   a) Deploy and test the OCCI extensions.
   b) Output: Refined OCCI and CLI implementations.
4. VM relocation assessment – September 2016 – October 2016:
   a) Evaluate VM migration functionality.
   b) Evaluate VM migration process by user communities and EGI Federated Cloud.
   c) **Output:** Modifications to the extension document and refination of VM migration process.
5. OCCI extensions second implementation – November 2016 – January 2017:
   a) Incorporate any missing CMFs to support the extension.

[60] https://github.com/EGI-FCTF/rOCCI
[61] https://launchpad.net/ooi
[62] https://www.synnefo.org/

b) Incorporate changes raised during the assessment phase.

c) Output: Final OCCI implementations supporting migration.

d) Output: Final OCCI CLI tools supporting migration.

6. OCCI extension final document – January 2017 – March 2017:

   a) Incorporate all changes and modifications identified during the implementation phase.

   b) **Output**: OCCI extension final document.

7. VM relocation final implementation – March 2017:

   a) Output: final implementation of VM relocation.

6 Conclusions

This document has provided an overview of the current state of the art concerning the migration of Virtual Machines (VMs). Considering the state of migrating suspended VM between sites and the variety of hypervisors in the EGI Federated Cloud, the safest approach to be considered is offline migration in which the VM has to be powered off before transferring the VM to a new site. A procedure that involves powering off the VM, transferring the disk into the destination site and resuming the VM at destination has been outlined. The challenges that lie ahead this procedure had been identified (e.g. capacity leasing and VM disk transfer). In addition, bindings with OCCI have been established to plan the integration of this procedure into the EGI Federated Cloud.