Open Standards, An Open Cloud

Andy Edmonds (Intel), Thijs Metsch (Platform Computing), Eugene Luster (R2AD)

DMTF APTS mtg. - May 2011

Introduction & Executive Summary

This paper documents observations and action items of the Open Grid Forum's (OGF)

Open Cloud Computing Interface (OCCI) [1] working group’s perspective resulting from the collaborative efforts of attending Standards Development Organization (SDO) sessions at the Distributed Management Task Force (DMTF) [2] Alliance Partner Technical Symposium (APTS). DMTF hosted this face-to-face event to investigate the possibility of integrating open cloud computing interoperability and portability standards efforts of Open Virtualisation Format (OVF), Cloud Data Management Interface (CDMI) [3] and OCCI.

In this document we describe how through leveraging mature open standards (related to Clouds, Grids and Storage), interoperable clouds can be created now. We demonstrate the use of the major contemporary innovative cloud interface specifications to realise an open standards-based, interoperable, cloud offering.

The motivation of this document is to demonstrate that currently available features in cloud standards from various SDOs are enough to create a cloud offering as described in the following sections. This document can be seen as one of the first steps of a deep-dive into cloud standards integration, especially between OCCI, OVF [4] and CDMI. Last but not least what is detailed in this document can be used to influence collaborations between OCCI-wg and CMWG as well as efforts like cloud-standards.org, SIENA [5], NIST [6] and others.

In order to describe how this integration of standards can happen, we will use a simple scenario. That scenario is one where a startup service provider wants to deploy, scale, migrate and redeploy their new Hadoop’s [7] based MapReduce service.

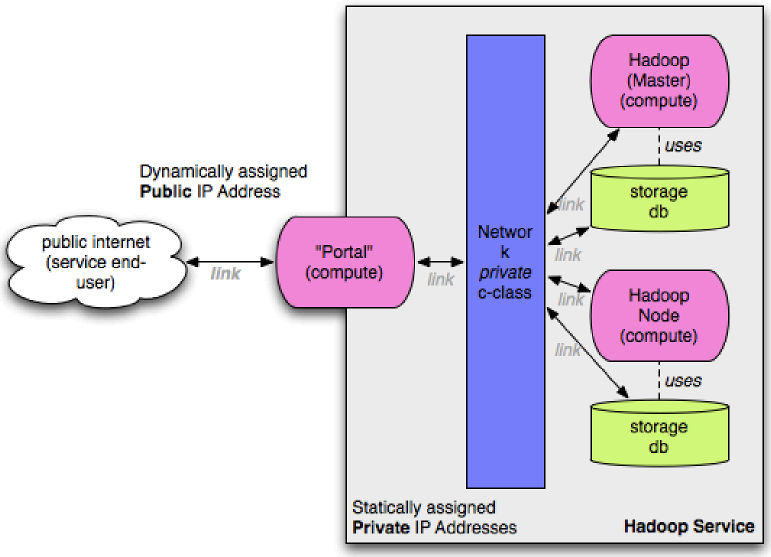
In order to implement/execute the scenario, the following enabling standards are used:

1. **DMTF’s OVF** - providing a means to package virtual infrastructure deployments, which can be exported or imported into an infrastructure service offering.
2. **SNIA’s CDMI** - providing an API that allows for the runtime management of storage infrastructure service offerings.
3. **OGF’s OCCI** - providing an API that allows for the runtime management of infrastructure as a service offerings.

It should be noted that aspects such as authorisation and authentication are not dealt with in this document as these issues are orthogonal ones dealt with by other technologies (e.g. OAuth, OpenID, etc).

The Scenario

The scenario is one where a startup company wishes to offer a MapReduce service to their clients. As the service offering is new it will be offered to a limited set of beta users. This limitation allows the startup company's architects to create an initial deployment architecture that suits the initial resource requirements to deliver the service. The service's deployment architecture is as follows:



Once the service has been deployed, the number of users makes it necessary that the service scales. This means additional resources must be added on-the-fly to serve the demand. Next to the scaling, migrations are considered part of the overall scenario. In case that the infrastructure provider suffers a major outage the startup is forced to migrate the service. This forms the final piece of the scenario in which the service provider moves the complete deployment over to a new more suitable infrastructure provider.

The scenario consists of two distinct phases:

1. Service deployment and scaling
2. Service migration and redeployment

To each of these phases there are a number of steps which will be detailed in relation to the standards used to execute each step.

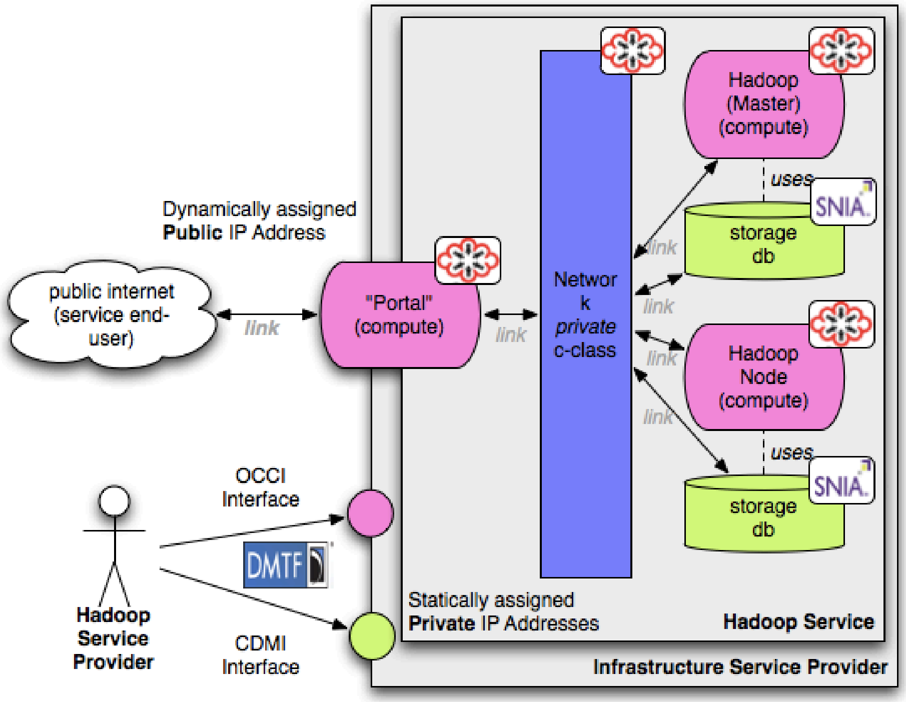
The overall goal of the startup is to utilise the infrastructure provider's CDMI and OCCI interfaces and to supply the provider with OVF service descriptions, which it can understand. It is these provider-offered capabilities that enable interoperability for the startup.

Service Deployment and Scaling

To create the initial deployment as shown in the figure above and demonstrate scaling, OCCI and CDMI will be used. This phase consists of the following:

1. **Create the initial deployment** based on the service's architecture. This can be accomplished using OCCI and CDMI or by OVF import.
2. Once the service matures and users increase, **Scale the service deployment**. This can be done incrementally using OCCI

The figure below shows the deployment of the startup's service in the context of these management APIs.



Create the Initial Deployment

At this point the aim is to create the initial service deployment, based on the architecture, so that it can be offered out to beta users. Overall, the process of setting up the service offering consists of a set of steps which reflects the setup as describe in the previous diagram:

1. Create Internal/Private Infrastructure
    a. Upload the virtual machine images to be executed (there are three: Portal, Hadoop Master and Slave)
        i. Upload the virtual machine images to the cloud provider. This results in a CDMI management endpoint
        ii. Register the virtual machine images as OCCI OS Templates
    b. Create the private network
        i. Use OCCI's IPNetwork Mixin (192.168.0.1/24)
    c. Create Storage Volume A (5GB)
        i. Use the CDMI interface for this
    d. Create Storage Volume B (100GB)
        i. Use the CDMI interface for this
    e. Create a hadoop master node
        i. Use hadoop master template along with the OCCI compute kind request
        ii. Connect to Storage A using OCCI's StorageLink (use CDMI identifier to relate CDMI resource to OCCI resource)

    f. Create a hadoop slave node
        i. Use hadoop slave template along with the OCCI compute kind request
        ii. Connect to Storage B using OCCI's StorageLink (use CDMI identifier to relate CDMI resource to OCCI resource)
    g. Connect Master node to private network
        i. Use OCCI's IPNetworkInterface
    h. Connect Slave node to private network
        i. Use OCCI's IPNetworkInterface
2. Create External/Public Infrastructure
    a. Create the portal node
        i. Use the portal OS template along with the OCCI compute kind request
    b. Connect the portal node to private network
    c. Connect the portal node to the public network

When the service is deployed, numerous RESTful resources will be present (such as http://example.com/compute/master) and will be manageable through their respective API. Also the overall service could be represented by a RESTful resource http://example.com/mapreduce/service1 which could be an OCCI representation (with actions, links etc) of the overall service.

Scale the Service Deployment

Over time the deployed service might need to horizontally[1] scale (positively and negatively). In the context of the scenario, the reason for the scaling is either that the current set of beta-users find huge value in the new service and so are placing more workloads onto the service or that as the service matures the provider increases the number of beta-users. This means that OCCI-managed virtual machines with Hadoop's Task Trackers and Data Nodes must be added on-demand. Hadoop's Name Node and Job tracker reside on the master node and might not need scaling at this stage.

There is only one step in this phase, namely, the operation of adding a new compute instance to the service and so horizontally scaling the service. This can be easily achieved by the following calls to the OCCI compatible interface:

1. Create new Hadoop slave node
    a. Use Hadoop slave template
    b. Connect to Storage B using OCCI's StorageLink (use CDMI identifier to relate CDMI resource to OCCI resource)
2. Connect Slave node to private network
    a. Use OCCI's IPNetworkLink

It is assumed that the Hadoop slave node has been configured *a priori* in such a manner that it

[1] Horizontal scaling here means to add nodes to the service and distribute the workload across those nodes.

registers with the Hadoop master node.

Service Migration and Redeployment

In the scenario having had poor service delivery from their current provider, the startup decides to migrate their current deployment to a new provider. In this case all three standards need to work together to ensure migration and redeployment from cloud A to B can be achieved. The overall process takes 4 steps:

1. **Service lookup and discovery -** Once the startup decides to move providers, the process of understanding if the new replacement provider can support the same features as the previous one can be accomplished by using the service discovery mechanisms of OCCI (query interface) and CDMI (meta-data interface)[2]. Only by being assured that the replacement provider can support the previous features, can the startup execute on service export, migration and data import.
2. **Service export** - A representation of the Service would be retrieved in OVF format. This is done by requesting a representation of the service in OVF. This is accomplished using HTTP content-negotiation. This will result in an OVF file with external links to CDMI managed data (application data, virtual machine images etc).
3. **Migration** - The CDMI interface should take care of the actual migration of the data from cloud A and B. This can be achieved through the serialize and deserialize features of CDMI.
4. **Service import** - With the service exported as an OVF representation and the related data now migrated to the new service provider, the original service needs to be instantiated. The instantiation is performed using the exported OVF representation. That OVF file is supplied to the service provider who, on the behalf of the client, instantiates the resources as described in the OVF and attaches the related data (managed by CDMI).

If the migration is a 'cold' migration (Service will be passivated and down during migration), OCCI's features to stop and restart the Virtual Machines, Network resources, etc. can be used. To support a live migration where there is little or no down-time of the service, the relevant capabilities (e.g. live migration across sub-nets) would need to be supported by the provider rather than the specifications.

Integrating Cloud Standards

The scenario described earlier shows that there are some interactions needed between several (currently available) cloud standards. The scenario demonstrates the need to both scale and migrate. In both cases three standards play an important role. OCCI is suited as the runtime

[2] Currently the CDMI interface does not live at a standardised URL. See section entitled *"Expose the Capabilities' Query Interface Through Well-Known URIs"* on how this can be resolved.

management API which triggers the operations. OVF is suited for portability reasons and CDMI is very appropriate for data movement and format migration. Although CDMI could also be used for some management tasks (e.g. Importing and Uploading Images through CDMI) - further extension to the current specification would be required.

To be able to ensure that the scenario can be realized the standards need to be integrated. The next sections focus on how the standards could interact, and also the changes that might be needed or nice to have for some of the standards.

Overall Topics Needed for Integration

One of the biggest challenges currently found is the migration of data. This not only means the transport of raw data from cloud A to B but also the possible conversion of data formats. For example VMware's file format, which is supported by cloud service provider A, might need to be converted to a VirtualBox format for cloud service provider B.

The data conversion and migration issue is currently not well documented and probably deserves further investigation. The migration process between two clouds might be triggered by OCCI but actually also involve CDMI and OVF. CDMI could to take care of the data conversions[3], whereas descriptions in the OVF representation might change during such an operation.

Integrating the Storage Management API

OCCI presents a simple representation of storage management capabilities. This allows for the most basic storage-related operations to be carried out. One of the goals of OCCI is integration of existing standards and not to reinvent the wheel. In the case where a provider wishes to expose a richer management interface to storage then it is recommended by OCCI to use SNIA's CDMI. This recommendation is accounted for in the specification, which details how CDMI managed storage can be represented in the OCCI infrastructure model (See section 3.4.3 of the OCCI Infrastructure Model Specification [8])

Integrating OCCI and CDMI

Since both specifications already leverage on each other integration of both can already be achieved at a high-level. The following sections provide the needed information for the integration and lead to ideas for a more tight integration of the two standards.

Using OCCI's StorageLink to Access CDMI's Container

As defined in the OCCI's Infrastructure extension OCCI can be used in conjunction with the SNIA cloud storage standard, CDMI, to provide enhanced management of cloud computing storage and data. In order to integrate the two, OCCI's StorageLink should be used. This will link OCCI managed Resources to CDMI resources.

[3] This is more an aspect of service implementation.

Migration of a Service Using OCCI and CDMI

If [4]a service provider implements both the OCCI and CDMI interface the users will begin a process to initiate and execute the migration. This process is a workflow that will follow the steps as set out in the above section "Service Migration and Redeployment". Another cloud provider (which also exposes the OCCI and CDMI interface) would be queried for the necessary capabilities in order to satisfy that the required service capabilities are present. Upon success and satisfaction of capabilities, the data needs to be migrated between the clouds and then the necessary resources provisioned.

CDMI can be used to address the issue of migration of the data. A new data object can be created at the destination cloud provider. Upon creation of the new data object the source should be a the original data object. See section 15 of the CDMI specification [9].

Considerations for CDMI

The following topics are suggested to further integrate OCCI and OVF into CDMI.

Importing and Uploading Virtual Machine Images Through CDMI (Using OVF)

When importing an OVF file through a CDMI interface it should be possible to assign network and compute resources which are defined in the OVF, however this is currently not possible. CDMI focuses on the Storage side of clouds but if an OVA/OVF is to be imported it could not only take care of the storage assignments but also interact with an OCCI interface to satisfy the complete OVF document's resource needs.

Linking Back From a CDMI Container to an OCCI Interface

The current process of using OCCI and CDMI is described in section 13 of the CDMI specification. Currently, resource instances of the OCCI Infrastructure model can link to the CDMI model. It is OCCI's StorageLink that is used to bind a compute resources to its storage[5]. Therefore this current link has a direction pointing from the OCCI model instances towards CDMI containers.

Adding to this integration, it would be useful to allow links to be directed from CDMI towards resource instances of the OCCI models (a complementary reverse linkage). Semantically this means to link storage resources in CDMI back to their associated OCCI Compute and/or storage resource. In general CDMI users could than see which services are associated to their data objects. In this scenario it would allow discovery of which disks are used for the virtual machines and also which data is used for the MapReduce service itself.

Expose the Capabilities' Query Interface Through Well-Known URIs

[4] If the Service provides does not provide a OCCI and CDMI interface migration cannot happen without direct user interaction. Discovery of whether a provider supports OCCI and CDMI can be done using the /.well-known/ interface.

[5] The semantic meaning here is to bind a disk image to a Virtual Machine instance.

As described in RFC 5785 (http://tools.ietf.org/html/rfc5785) we would recommend exposing the CDMI capabilities interface, which is currently exposed through the path 'cdmi_capablities', (also) under the path '.well-known/com/snia/cdmi'. OCCI shares this method and could expose it's query interface through the path '/.well-known/org/ogf/occi' (even alongside the current path '/-/'). That way a client has a unified way of accessing and querying the capabilities of a service offering. Eventually these namespaces should be registered with IANA.

Integrating Standards to Ensure Portability

Next to the aspect of storage and data in the cloud, portability of Services must be ensured. DMTF's OVF specification presents a way of describing complete Services in a portable way. The OCCI interface could be used to import and export Service definitions in an OVF format. The following sections will elaborate on these ideas.

Integrating OCCI and OVF

OCCI and OVF could simply coexist in parallel with each other. The only addition currently needed would be support of a MIME type which tells the OCCI service provider that the Client wants to retrieve or supply information in an OVF format. The specification of this new MIME type is not included in the OCCI specification. It only supports *text/occi*, *text/plain* and *text/uri-list* at the moment.

Mapping OVF to OCCI's Infrastructure Model

The following tables give an overview of how the OCCI attributes can be mapped to the OVF attributes and vice-versa.

OCCI's Compute Resource

Description	OVF	OCCI	Details
Architecture of the CPU (86, x64)	<vssd:VirtualSystemType>[...String...]</vssd:VirtualSystemType>	occi.compute.architecture	Described in a VirtualHardwareSection
# of cores	<rasd:ResourceType>3</rasd:ResourceType> <rasd:VirtualQuantity>1</rasd:VirtualQuantity>	occi.compute.cores	Described in a VirtualHardwareSection
Hostname	<Property ovf:key="hostname" ovf:type="string"> </Property>	occi.compute.hostname	Part of ProductSection
Speed of the CPU	<rasd:ResourceType>3</rasd:ResourceType> <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits> <rasd:Reservation>500</rasd:Reservation>	occi.compute.speed	Described in a VirtualHardwareSection

Amount of memory	<rasd:ResourceType>4</rasd:ResourceType> <rasd:VirtualQuantity>512</rasd:VirtualQuantity>	occi.compute.memory	Described in a VirtualHardwareSection
Status of the resource	N/A	occi.compute.state	

OCCI's Network Resource

Description	OVF	OCCI	Details
A network label	<Property ovf:key="label" ovf:type="string">	occi.network.label	Defined via Properties in the ProductSection
A vlan name	<Property ovf:key="vlan" ovf:type="string">	occi.network.vlan	Defined via Properties in the ProductSection
Status of the resource	N/A	occi.network.state	

OCCI's Storage Resource

Description	OVF	OCCI	Details
Size of the storage device.	<Disk ovf:diskId="vmdisk2" ovf:capacity="536870912"	occi.storage.size	Described in DiskSection
Status of the resource	N/A	occi.storage.state	

Other OCCI resources might also need a mapping, such as some Links and Mixins which are defined by the OCCI infrastructure model extension. Those mappings are straight forward and in-line with the previously described tables.

The service provider currently decides what happens with unmapped attributes. It could happen that retrieving an OCCI managed service in an OVF format results in a minimalistic OVF file which only holds the attributes also present in the OCCI infrastructure representation. The service provider however could choose to hold the OVF representation in parallel to the service deployment in the OCCI infrastructure model in which case the complete OVF representation could be retrieved. We currently would encourage the use of the latter.

Ensure horizontal/vertical scaling

Horizontal and Vertical scaling are currently not covered in detail by the OVF specification, although ranges can be defined in the service description. The use of ranges allow a client to specify a valid set of ranges for an attribute however (Like the number of virtual machines), these may not be enough for scaling, given that scaling should happen based on some logic. The horizontal and vertical scaling of the service can only be achieved by using OCCI in conjunction with OVF.

Mapping of OCCI's resource templates to OVF

The OCCI infrastructure extension describes ways for the user to define resource and OS templates. Templates allow clients of an OCCI implementation to quickly and conveniently apply predefined configurations to OCCI Infrastructure defined types. They are implemented using OCCI's Mixin instances. The OS and Resource template build upon each other:

- OS Templates allow clients specify what operating system must be installed on a requested Compute resource.
- The Resource Template Mixin builds upon the concept of OS Templates. A Resource Template is a provider-defined Mixin instance that refers to a preset Resource configuration (Similar to Amazon's EC2 Instance Types [Small, Large, Extra Large])

It would be desirable to have these templates being described by an OVF representation as well.

Considerations for OCCI

The following two topics are not directly related to OCCI as a boundary protocol. Rather they refer to how service providers implementing OCCI should handle some operations.

Configuring OCCI entities from options in an OVF description

The OVF format allows the configuration of some operations such as describing what should happen upon power-on and power-off operations. Future versions of OVF might even go into details by describing *ActivationEngines* and related aspects[6]. OVF therefore allows some level of detail which describe the semantics of how a Service should be handled. Service providers which allow the provisioning of a complete service using OVF over OCCI need to take care of these configurations/details and configure the Resource Management Framework accordingly.

Ensure that conformance levels described in OVF are met in a setup with OCCI

Closely related to the previous topic are conformance levels which are be described by OVF. Service providers supporting service provisioning using OVF over OCCI need to take care that these levels are met. The conformance levels must be met when instanciating the service. If the levels are not met the client could see this as a voilation of an Agreement or SLA. So therefore if a Service provider cannot implement the conformance level requested in the OVF description the request for provsioning should be denied.

Considerations for OVF

The following topics should be considered to enable a further integration of OVF and OCCI.

Describe Network devices like Virtual Switches in OVF

Network features can already be described in OVF, although it would be desirable to be able to describe complete network setups like the one used in the Scenario section in the service

[6] These topics are for example described in this http://www.ibm.com/developerworks/websphere/techjournal/0708_he/0708_he.html article.

description. It is noted that OVF 2.* will be more supportive for these descriptions.

Define a mime-type for OVF

The OCCI client relies on correct mime-types while requesting operations or receiving information from a service. The mime-type information is used by the service provider to verify and understand how the information has been sent to it. The client can always request in which mime-type they want to receive the requested information. For an importing and exporting feature for OVF files clients would be posting OVF files to an OCCI service or request the current state in an OVF format. For best interoperability we would recommended that a mime-type for OVF is registered.

Related Work and Outlook

While investigating the scenario used in this document it was found that some topics would be interesting for further investigation:

1. Service Level Agreements (SLAs)
    a. An SLA extension for OCCI is currently under development and could be used.
    b. SLAs could be described in OVF and enforced using this OCCI extension.
    c. Monitoring, enforcement and setup of SLAs for storage/data can be integrated through CDMI using the OCCI extension. This monitoring extension, when coupled with triggers and actions, will allow for further standardised auto-scaling.
2. Features which OCCI should incorporate from other Standards
    a. CDMI's Queues and Notification mechanisms are found highly usable and it should be investigated if those could be integrated into OCCI.
    b. URI escaping as described in RFC 3986 might be needed for OCCI's current rendering.
    c. Adding a JSON rendering extension to OCCI's current text and HTTP header rendering might allow even tighter integration of OCCI and CDMI.

Conclusions

The scenario described in the first sections was used to step through different processes during the service lifetime, in particular migration and scaling of the service. It is intended to reflect a real-world service offering and demonstrate that by using today's standards it is possible to realise such a setup. Important to note is that several standards need to be integrated to achieve this goal.

Further sections in the document discussed how the three standards used in this document can be integrated and where open issues reside. The open issues requiring further investigation or improvement were shown and described. The overall integration is sufficient to achieve the complete deployment, migration and scaling of the service described in the scenario section using today available versions of the Specification. Althought the modification mighe be required to ensure that every cloud provider uses the same semantics while implementing the Standards.

It is noted here that bringing in a fourth currently available specification might be desirable. CSA's CloudAudit has become accepted by the industry and adds important features to the scenario in this document. Auditing clouds and conformance levels was left out of scope in this document but would be useful to investigate further.

Although currently focusing more on the resource (Infrastructure) level of the service deployment a more PaaS based scenario (like a Node.js or GAE service) using cloud standards could be described in future revisions of this document.

Acknowledgement

The authors would like to thank Mark Carlson (Oracle) and David Slik (NetApp) for inputs on the integration of CDMI and OCCI.

We would like to thank Winston Bumpus (VMware) and Jeff Wheeler (Huawei) for their help on OVF-related topics.

The following reviewers work was greatly appreciated while creating, editing and reviewing this document:

- Lawrence Lamers (VMware)
- Sebastian Schoenberg (Intel)
- John Kennedy (Intel)
- Finian Rogers (Intel)

References

1. Open Cloud Computing Interface working group community website, http://www.occi-wg.org
2. Distributed Management Task Force website, http://www.dmtf.org
3. Cloud Data Management Interface website, http://www.snia.org/cloud
4. Open Virtualization Format, http://www.dmtf.org/standards/published_documents/DSP0243_1.1.0.pdf
5. Standards and Interoperability for eInfrastructure Implementation Initiative website, http://www.sienainitiative.eu
6. NIST Cloud Computing Program, http://www.nist.gov/itl/cloud/
7. Apache Hadoop website, http://hadoop.apache.org/

8. Open Cloud Computing Interface - Infrastructure, Open Grid Forum, GFD-P-R.184, 2011, http://ogf.org/documents/GFD.184.pdf
9. Cloud Data Management Interface, Storage Network Initiative Alliance, 2010, http://www.snia.org/tech_activities/standards/curr_standards/cdmi/CDMI_SNIA_Architecture_v1.0.pdf