Portability and Interoperability between Clouds: Challenges and Case Study (Invited Paper)

Dana Petcu

Institute e-Austria Timişoara and West University of Timişoara, Romania
petcu@info.uvt.ro
http://web.info.uvt.ro/~petcu

Abstract. The greatest challenge beyond trust and security for the long-term adoption of cloud computing is the interoperability between clouds. In the context of world-wide tremendous activities against the vendor lock-in and lack of integration of cloud computing services, keeping track of the new concepts and approaches is also a challenge. We considered useful to provide in this paper a snapshot of these concepts and approaches followed by a proposal of their classification. A new approach in providing cloud portability is also revealed.

Keywords: Cloud computing, portability, interoperability.

1 Introduction

Still being in the early stage of developments, cloud computing suffers from the classical problem of too many different approaches. Practically, every new cloud provider comes with its own solution of interfacing with its resources or services. This variety is a reflection not only of different potential angles of approaching the concepts of cloud computing, but it is also a reflection of the variety of underlying offer on the market in what concerns the models of storage, networking, processes licensing or even integration of own resources in a cloud.

The portability and interoperability between clouds are important not only for the protection of the end user investments but also for the cloud ecosystem and market, business applications and data being currently strangled in silos. The main aim of interoperability is to allow the achievement of the full advantage of the cloud properties like elasticity and pay-as-you-go, not of a vendor infrastructure, platform or service. Nowadays, many companies still do not tie their critical applications to specific cloud providers services due to the underlying proprietary technology. Therefore the latest three years have been marked by the development of a considerable number of approaches to tackle with the issues of portability and interoperability. The selection of the appropriate approach to deal with these issues is in danger to become itself a challenge. Therefore we considered useful to present in this paper an overview of the different concepts and approaches undertaken world wide (in Section 2) and thereafter to go deeper into a particular solution currently under development (Section 3).

2 Challenges in Interoperability and Portability

2.1 Defining the Cloud Interoperability

A first challenge in cloud interoperability consists in its definition. As general term, interoperability is a property referring to the ability of diverse systems and organizations to work together (inter-operate). In computer world, this property has the concrete meaning of exchanging information and use of the information that has been exchanged between two or more systems or components. It is a property of a product or system, whose interfaces are completely understood, allowing it to work with other products or systems.

The most simple way to describe the cloud interoperability is by its most used motos like "avoid vendor lock-in", "develop your application one, deploy anywhere", "enable hybrid clouds", or even "one API to rule them all". Browsing the literature, one can found various definitions referring it as the ability to:

- abstract the programmatic differences from one cloud to another;
- translate between the abstractions supported by different clouds;
- flexible run applications locally or in the cloud, or in a combination;
- move applications from one environment to another or run in multiple clouds;
- move services, processes, workloads, and data between clouds;
- use same management tools, server images, software in multiple clouds;
- communicate between providers, port application and data between them;
- federate multiple clouds to support a single application.

2.2 Use Cases of Multiple Clouds

The above enumerated definitions of cloud interoperability are consequences of the different needs of the customers who are dealing with the use of multiple clouds. Use cases of multiple clouds have been reported in numerous papers in the last half decade. In a try to classify them, we can follow the NIST proposal [2] concerning the deployment scenarios in multiple clouds:

Serially, one cloud at a time, with three scenarios: (a) migration between clouds; (b) interface across multiple clouds; (c) work with a selected cloud;
Simultaneously, several clouds at a time, when operate across multiple clouds.

Beyond the well know use cases exposed in the white paper of Cloud Computing Use Cases [1] we draw the attention to the following use cases (after [2,3,4,5]):

Change cloud vendors: migrate some or all of a set of existing services to a new vendor. Customers may will to change providers to make an optimal choice regarding utilization, expenses or earnings. Other reasons to port application or data from one provider to another can be: provider out of business; better options in market than current ones; technology changes; contract termination; legal issues, etc.
Distributed deployment: the application may be distributed across two or more providers and administrative domains simultaneously:

FEDERATED CLOUDS: the providers agree how to mutually enforce policy and governance, establishing a common trust boundary;

- *Scale-out:* in an event occurs unexpectedly or in a peak, the company can still operate its cloud stably by distributing dynamically its load between the resources of its own cloud and a community cloud;
- *Mutual backup and recovery from a disaster:* if an event damages the provider's cloud or causes power outage, cloud resources in other providers are used to restore the services of that provider.

HYBRID CLOUD: applications cross a private-public trust boundary or span multiple clouds (simultaneous use of multiple clouds and both administrative domains and trust boundaries are crossed);

- *Use different cloud services at the same time,* to deploy on-production applications, run tests, or build test environment;
- *Manage selected resources in different clouds,* e.g. e-mail in one cloud, applications in another, and storage in a third one, all of them interacting;
- *Social networks applications* built using multi-tiered web technologies and serving dynamic content to users with unpredictable access and interaction patterns. Each component runs in a separate virtual machine hosted in data centers owned by different providers and must dynamically scale. New plug-ins are created by developers (freedom to choose provider), added to the system and used by others, potentially spiking. An application can have hundreds services in dozens of data centers.

2.3 Types and Targets of Cloud Interoperability

Several criteria for classification of interoperability types can be identified:

AGREEMENT LEVEL

- *syntactic,* when the systems are capable of communicating and exchanging data (using specified data formats, communication protocols).
- *semantic,* as automatically interpretation of the information exchanged, meaningfully and accurately, in order to produce useful results (using a common information exchange reference model);

ADOPTION LEVEL

- *by design* when vendors or individuals use a standard documentation to make products (no specific liability or advantage for any customer for choosing one product over another);
- *post-facto,* result of absolute market dominance of a particular product;

DEPLOYMENT LEVEL

- *horizontal* interoperability of the services at the same deployment level (likely within IaaS, harder to the customized PaaS and SaaS);
- *vertical* interoperability (vertical supply chain) when cloud services can be build on top of other cloud services from other deployment level;

PATTERNS OF INTERACTIONS BETWEEN CLOUDS

- *synchronous:* direct calls, suitable only for very specific real-time applications where the response time is critical;

  – *asynchronous:* applications are loosely coupled so that consuming application do not wait for a response (best option in cloud).

The targeted levels for cloud interoperability are the followings (after [6,7,8]):

– *business* level, achieved between different business strategies that are imposed on the services, regulations on the services, and mode of use.;
– *semantic* level, with focus on functions calls and responses for consumers requests as well as message contents;
– *application/service* level when automated, generalized and extensible solutions are provided to use new resources (other than re-compiling, there are no further changes required of the application). Components could be reconfigured while running, or with limited interruption, to respond to changes in usage patterns or resource availability. Application configuration must be resilient to changes in the configuration within each cloud – for example scaling or migration of computational resources. From simple execution-unaware applications using multiple environments, to applications with multiple distributed components, the complexity is non-uniform and depends upon the application. A pre-requisite is infrastructure independent programming;
– *management* when a management application coordinate and control components in multiple clouds. Standardized functionality for deployment and migration of VMs is required. Interactions between actors responsible for application management and infrastructure management is needed;
– *technology/infrastructure* level, achieved by agreeing on or accepting particular encoding scheme for requests and response, selection of communication protocol or middleware, language, and the platform for working environment;
– *image/data* looks at VM images, applications, or databases, and how they can be deployed on another host without modification (as pre-deployed, ready to run applications packaged as VMs, namely "work loads").
– *network* seek support to uniform access to individual resources and concatenation/federation (standards for allocation and admission control are needed).

The evolution of interoperability issues has take three stages (after [9]):

Migration: refers to portability of VMs allowing to move VMs between clouds, to create VMs locally and import them, and to share VMs with others (OVF is providing standard packaging, but does not address the problem of different virtual disk formats; further migration/ conversion tools are needed);
Federation: targets networking, portable VMs being moved to the cloud without reconfiguring anything including network settings (transparent migration) or move across multiple clouds and multiple hypervisors;
Burst: targets APIs; as portable VMs and ability to seamlessly integrate deployments are in place, migration and federation "on demand" come in place; interoperability efforts are related to storage (CDMI) and compute (OCCI).

The federation of clouds can be organized in different ways:

HORIZONTAL FEDERATION. Two or more cloud providers join together to create a federated cloud [10]: participants who have excess capacity can share their resources, for an agreed-upon price, with participants needing additional resources (avoiding over provisioning of resources to deal with spikes in capacity demand). Challenges are related to: finding the "best" cloud for a workload by balancing among parameters like QoS and cost; a logical topology is maintained regardless the physical location of the components.

INTERCLOUD. Federation of clouds with common addressing, naming, identity, trust, presence, messaging, multicast, time domain and application messaging [4]. The responsibility for communicating is on the providers' side. The applications are integrating services from multiple clouds and are scaling across multiple clouds. The overall goal is to create a computing environment that supports dynamic expansion or contraction of capabilities for handling variations in demands. Dynamic workload migration is possible.

CROSS-CLOUD. The federation establishment between a cloud needing external resources and a cloud offering resources (not necessarily in agreement), passes through three main phases [11]: discovery, looking for available clouds; match-making, selecting the ones fitting the requirements; authentication, establishing a trust context with the selected clouds.

SKY COMPUTING. Offers a combined use of multiple clouds. Resources, applications and platforms across clouds are used. New services other than those of each individual cloud are provided. Transparency of multiple clouds is provided offering a single-cloud like image. Sky providers are consumers of cloud providers' services offering (virtual datacenter-less) dynamicity [12].

The main supporting actors [2,4] assisting in federation implementation are:

Broker, an entity that manages the use, performance and delivery of cloud services, and negotiates relationships between providers and consumers;

Auditor, a third-party that conducts an independent audit of the operations and determines the security of the cloud implementation;

Coordinator, for exporting cloud services and their management driven by market-based trading and negotiation protocols for optimal QoS delivery;

Exchange, acting as a market maker enabling capability sharing across multiple cloud domains through its match making services.

Orchestrator, for the network of clouds.

2.4 Cloud Portability

While the interoperability is the successful communication between or among systems, portability is the ability to use components or systems lying on multiple hardware or software environments.

The types and solutions for portability are classified (after [8,13]) as follows:

functional portability: ability to define application functionality QoS details in a platform-agnostic manner. OVF provides a basis for portability but does not address complex configuration or interactions with any supporting systems. Domain specific languages are expected to bridge the gap between executable artifacts and high-level semantic models.

data portability: ability for a customer to retrieve application data from one provider and import this into an equivalent application hosted by another provider. Achieving data portability depends on standardization of data import and export functionality between providers. It is necessary to provide a platform-independent data representation, and generate the specific target representations and even the code for the application's data access layer;

services enhancement: use metadata added through annotations. Control APIs allow infrastructure to be added, reconfigured, or removed in real time, either by humans or programmatically based on traffic, outages or other factors.

The requirements of portability at different deployment levels are as follows (after CSA - Cloud Security Alliance documents):

SaaS: the cloud customer is substituting software applications with new ones. The focus is on preserving or enhancing the functionality provided by the application. Portability is evaluated based on open source code base, proprietary or open standard data formats, integration technologies and application server/operating system.

PaaS: some degree of application modification will be necessary to achieve portability. The focus is on minimizing the amount of application rewriting while preserving or enhancing controls, and a successful data migration. Portability is evaluated based on proprietary or open source programming languages for application development, proprietary or open data formats, tight integration or loose coupled services, abstraction layers for queuing and messaging services.

IaaS: the applications and the data migrate and run at a new cloud provider. Portability is evaluate based on ability to port VMs and the underlying configurations across infrastructure providers.

2.5 Interoperability and Portability Requirements

The main requirements for cloud interoperability are (after [4,5,10]):

programming: move from one provider to another without dramatic reimplementation; common set of interfaces; standard API enabling an entity to build something once, then use it to monitor and control a variety of platforms; work with both cloud-based and enterprise-based applications using a single tool set that can function across existing programs and multiple cloud providers; new programming models; ontology of cloud; high level modelling.

application: able to span multiple cloud services; data exchange; data portability; private cloud applications obtain resources from public cloud when excess capacity is needed; location-free applications; workflow management;

monitoring: SLA and performance monitoring; SLAs in support of governance requirements; deliver on demand, reliable, cost-effective, and QoS aware services based on virtualization technologies while ensuring high QoS standards and minimizing service costs; QoS and SLA items to be guaranteed end-to-end; monitoring and management of load balanced applications in an elastic environment; scalable monitoring of system components; service monitoring and audit; publish sets of benchmarks to evaluate performance factors;

deployment: provision resources from multiple cloud services with a single management tool; agreements between providers; service discovery; common platforms to ensure users can navigate between services/applications; enabling a service hosted on one platform to automatically call a service hosted by another; automatically provision services and manage VM instances; virtual organization management; resource discovery and reservation procedure; service setup procedure; interworking between clouds and the network – routing optimization based on monitoring; automatism and scalability – home cloud using discovery mechanisms able to pick out the right foreign clouds; support for multiple hypervisor technologies; application service behavior prediction;

authentication, authorization, security: single sign-on for users accessing multiple cloud; integration of different security technologies (home cloud able to join federation without changing security policies); digital identities; identity federation and management across vendors; standards for security specification, platform components and configuration; authentication procedure when use different domains; cloud trust mechanisms; auditing, security mechanisms for authentication and authorization;

market: economic models driven optimization techniques; market driven resource leasing federation – application service providers host their services based on negotiated SLAs driven by competitive market prices; flexible mapping of services to resources to maximize efficiency, cost-effectiveness, and utilization; accounting; license flexibility.

2.6 Approaches to Cloud Interoperability and Portability

The approaches to interoperability and portability can be classified in building and using: (1) open APIs; (2) open protocols; (3) standards; (4) layers of abstractions; (5) semantic repositories; (6) domain specific languages.

Open APIs are for example jClouds (Java), libcloud (Python), Cloud::Infrastructure (Perl), Simple Cloud (PHP), Dasein Cloud (Java), proprietary APIs being for example Micosoft Azure (.NET) or Fujitsu API. A short list includes (after [14,15]):

JCloud provides a framework for interacting with blob storage, queue storage, and compute resources for a variety of clouds (requires service and location information to access a data item).

Dasein Cloud includes blob storage, compute, and network abstractions (does not include table or queue-based storage, and aiming to interface with all aspects of cloud infrastructures it is quite complex);

CloudLoop provides a filesystem-like interface to blob storage (does not support other abstractions such as tables and queues);

SimpleCloud from IBM, Microsoft, Zend, Nirvanix, Rackspace and GoGrid supports storage management (storage, queues, and databases) and is a PHP API for interacting with blob, table, and queue storage services of several providers (PHP is typically limited to web applications);

OpenNebula provides cloud users and administrators with choice across popular cloud interfaces, hypervisors and clouds, for hybrid cloud computing deployments, and with a flexible software that can be installed (expose most common cloud interfaces, such as vCloud and EC2; use open community specifications, such us OCCI, and open interfaces, such as libcloud and deltacloud; support the combination of local private infrastructure with EC2 and others through the deltacloud adaptor);

ServerTemplates from RightScale, enables portability, but also lets users take advantage of the unique capabilities of different clouds: it configures servers for a specific cloud, application architecture and operating system;

Appistry, AppZero, and 3Tera created suites providing a layer of abstraction between the programmer and the cloud platforms. Developers create applications for this intermediate layer, which then supports and manages multiple hypervisors or external cloud platforms;

OpenStack has a similar proposal and is a combination of Rackspace cloud architecture and NASA's Nebula.

The above APIs for multiple clouds can be classified as follows (after [15]):

API with multiple independent implementations (Eucalyptus compatibility with EC2, AppEngine with AppScale). The shared API is driven by the initial provider and periods of inconsistency between implementations can appear; moreover, two implementations of the same API are not equivalent in terms of scalability, features, maturity, and customer support.

API runnable on multiple clouds not necessarily through multiple independent implementations (e.g. MapReduce and Hadoop). They focus on particular application model, not necessarily fit for any developers requirements. Significant developer time investment for configuring, deploying, maintaining these services is needed. Implementations are tailored to the specific vendor offering the service (the configuration may differ between vendors).

Separate the application into "app-logic layer" and "cloud layer" (with code written for each cloud provider). It is the most general option, but requires a time and complexity investment by a developer to initially create the layers and further maintain them over time as the APIs change.

Set of standards this is the best solution in terms of cloud interoperability, but it is far from being realized in the commercial space.

Open protocols are for example OCCI (HTTP) or Deltacloud (RedHat), while proprietary protocols are for example Amazon EC2 or VMware vCloud.

vCloud (with TCloud extension) offers the concepts of instance template, storage and network, image element, and vApp (containing one or more VMs).

DeltaCloud by Red Hat, abstract the differences between diverse clouds (supports only computational resources).

OCCI is a specification for remote management of cloud infrastructure, allowing the development of tools for common tasks including deployment, autonomic scaling and monitoring. Its API is based on three concepts: compute, storage and network. It relies on the HTTP protocol.

Many groups are working on cloud computing standards (interoperability ab-initio). The most active groups are (after [16]) CloudAudit, CSA, DMTF, ETSI TC CLOUD, OGF, OMG, OCC, ASIS, SNIA, CCIF, GICTF, ODCA. The OCCI Working Group of OGF, for example, develops mainly a practical specification related to IaaS. DMTFs Open Cloud Standards Incubator (OCSI) focuses on standardizing interactions between cloud environments by developing cloud resource management protocols. Cloud Manifesto is an initiative supported by several companies arguing that cloud should capitalize on standards.

The two widely adopted standards are:

OVF is a DMTF standard that describes virtual appliances for deployment across heterogeneous virtualization platforms (i.e. different hypervisor), allowing the users to deploy their virtual appliances at every cloud provider.

CDMI is a SNIA standard for data management specifying a functional manner on how applications create, retrieve, update, delete data from the cloud.

Technical requirements versus standards were recently discussed in [2]:

Creating, accessing, updating, deleting data in clouds (cross-cloud): standard interfaces to metadata and data objects are needed, current solution is CDMI;

Moving VMs & virtual appliances between clouds (migration, hybrid clouds, recovery, bursting): need a common VM description format – solution: OVF;

Selecting vendor for externally hosted cloud (cost-effective reliable deployment): resource and performance requirements description languages, no solution;

Portable tools for monitoring and managing clouds (simplifies operations as opposed to individual tools per cloud): standard management interfaces to IaaS resources, current solution OCCI;

Moving data between clouds (migration, cross-cloud): standard metadata/data formats for movement; vendor mappings between cloud data and standard formats; standardized query languages – no solution;

Single sign-on access to multiple clouds (simplified access, cross-cloud): federated identity and authorization, solutions from OpenID, OAuth, OASIS, CSA;

Orchestrated processes across clouds and enterprise systems (enhanced applications): standards for APIs/data, solutions from SOA, Intercloud/IEEE;

Discovering cloud resources (selection of clouds): description languages for available resources, catalog interfaces, current solution from DMTF, TM Forum;

Evaluating SLAs and penalties (selection of appropriate cloud resources): SLA description language, no solution;

Auditing clouds (ensure regulatory compliance; verify information assurance): auditing standards, verification check lists, solution from CSA CloudAudit.

There are several barriers in standardization (after [3]): (1) Each vendor likes to put barriers to exit for their customers, unless there is some pressure from their big customers. (2) Each cloud provider offers differentiated services, and want to have special services to attract more customers (a common standard may regulate them). (3) Cloud providers will possibly not agree with an easy and standardized manner to export/import cloud configurations. (4) Standards are nascent and will take years to fully develop. (5) There are numerous standards being developed simultaneously and consensus may be difficult, if not impossible, to attain. (6) As there are a number of different cloud computing models (e.g. SaaS, PaaS, IaaS), this indicates that different standards are needed for each model, rather than one overarching set of standards.

Different layers of abstractions are used for example in the case of:

RESERVOIR: service providers are the mediators between infrastructure providers and end-users (single clients or businesses). Service manifests, formally defining a contract and SLA, play a key role in the whole architecture [10];

SLA@SOI: Framework where cloud services can be traded as economic goods and SLA agreements can be established between customers and service/ business providers, service providers and all the way down to infrastructure demands monitoring of the services life-cycle;

CSAL: Abstraction layer, that provides blob, table, and queue abstractions across multiple providers and presents applications with an integrated namespace thereby relieving applications of having to manage storage entity location information and access credentials [15];

RASIC: An open, generic reference architecture for semantically inter-operable clouds introducing an approach for the design, deployment and execution of resource intensive SOA services on top of semantically interlinked clouds [16].

The semantic can be applied to the interface level, the component level, and the data level by utilizing a generic semantic cloud resource data model. Semantics can be also used to annotate the services and the applications deployed by the users on top of the cloud (the challenges to be addressed involve the specifications of the appropriate properties such that the service can be deployed and efficiently executed in the infrastructure). In this context, Unified Cloud Computing [17] is an attempt to create an open and standardized cloud interface for the unification of various cloud APIs; for a unified interface RDF is used to describe a semantic cloud data model (taxonomy and ontology). A functional implementation of UCI is available for on Amazon EC2 or Enomaly ECP.

Domain specific languages provide a cloud neutral application creation strategy. They create a cloud specific application for the platform or interpret it in a cloud specific VM. The complexity of managing heterogenous applications is hidden from the user. Few solutions are available on the market.

3 Case Study: Interoperability in mOSAIC

mOSAIC is a project partially funded by the European Commission in the period 2010-2013 aiming to provide an open source API and platform for supporting applications using multiple clouds. We considered useful to presents its position versus the concepts discussed in the previous section as providing a real snapshot of current activities in the field of cloud portability and interoperability.

mOSAIC approach for interoperability consists in building an open API, offering a layer of abstraction, and applying semantic to interface and component level (see section 2.6). The main targets are hybrid clouds' scenarios and vertical interoperability by design (sections 2.2 and 2.3). The platform best suited applications are the long-time running ones (like in the social network or management of resources scenarios), but entry points in the platform are provided to deploy legacy or scientific applications (using different services at the same time) [18].

In what concerns the API design, mOSAIC team decided to separate the application into application-logic layer and cloud layer (see section 2.6). The application developer do not care about the infrastructure. A descriptor of the application is used to select the proper clouds (it presents the components and their interactions). The code for connecting to a specific cloud is generated at run time using what is named in mOSAIC, interoperability API [19], drivers for a certain category of cloud technologies and connectors to particular cloud services. The applications are decomposed in components runnable of top of different clouds [20]. Cloudlets are the means offered to developer to create components. A cloudlet runs in a container managed by the platform and can have multiple instances. A container hosts a single cloudlet, even if it may host multiple instances of the same cloudlet. Containers can run on multiple clouds. A component is a container and its hosted cloudlet (instances).

Exchanges between clouds are done using cloud based message queues technologies to ensure the syntactic interoperability (see section 2.3), while the high level APIs are ensuring the semantic interoperability. A asynchronous pattern of interaction is expected and an event-driven architecture was adopted.

Based on the current state of the art, mOSAIC is a representative for the "burst" phase in cloud interoperability (see section 2.3). It is build on top of the existing solutions for portable VMs deployment, providing "on demand" federation of cloud services. While it relies upon the current solutions for interoperability at technology–infrastructure, image–data and network level, mOSAIC targets application–service, management and semantic levels of interoperability, supporting applications decoupled from the execution, deploying and controlling the running components [21], and allowing semantic processing based on a particular cloud ontology. The level of deployment targeted by mOSAIC is PaaS (see section 2.4). Being able to combine services from different clouds (without the need establishing a trust context in the selected clouds), to offer transparency of multiple clouds, and to provide new services on top of the existing ones, mOSAIC can be considered a sky computing provider (see section 2.3). As supporting actors the platform uses brokers and coordinators.

4 Final Remarks and Conclusions

Far from being solved, the cloud interoperability and portability problems are treated by various approaches. A deep analysis of these approaches show a limited trace of consensus and worries can be raised about their completeness in the near future. The current paper intended to "draw the line" and to point towards the most successful or promising approaches with the hope to pave the way for further research and development activities in using multiple clouds.

Acknowledgment. This research is partially supported by the grant FP7-ICT-2009-5-256910 (mOSAIC).

References

1. Cloud Computing Use Case Discussion Group. Cloud computing use cases-white paper. Version 2.0 (2009)
2. NIST CCSRWG: Cloud Computing Standards Roadmap (2011)
3. Machado, G.S., Hausheer, D., Stiller, B.: Considerations on the interoperability of and between cloud computing standards. In: Procs. OGF27: G2C-Net (2009)
4. Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., Morrow, M.: Blueprint for the Intercloud - protocols and formats for cloud computing interoperability. In: Procs. ICIW 2009, pp. 328–336. IEEE Computer Press, Los Alamitos (2009)
5. GICTF: White paper–use cases and functional requirements for inter-cloud computing (2010)
6. Khattak, A.M., Pervez, Z., Sarkar, A.M.J., Lee, Y.-K.: Service level semantic interoperability. In: Procs. IEEE SAINT 2010, pp. 387–390. IEEE CSP, Los Alamitos (2010)
7. Merzky, A., Stamou, K., Jha, S.: Application level interoperability between clouds and grids. In: Procs. GPC Workshops 2009, pp. 143–150. IEEE CSP, Los Alamitos (2009)
8. Oberle, K., Fisher, M.: ETSI CLOUD–initial standardization requirements for cloud services. In: Altmann, J., Rana, O.F. (eds.) GECON 2010. LNCS, vol. 6296, pp. 105–115. Springer, Heidelberg (2010)
9. Williams, J.L.: An implementors perspective on interoperable cloud APIs. Cloud interoperability roadmaps sessions. In: OMG Technical Meeting (2009)
10. Rochwerger, B., Breitgand, D., Epstein, A., Hadas, D., Loy, I., Nagin, K., Tordsson, J., Ragusa, C., Villari, M., Clayman, S., Levy, E., Maraschini, A., Massonet, P., Munoz, H., Tofetti, G.: Reservoir–when one cloud is not enough. Computer 44(3), 44–51 (2011)
11. Celesti, A., Tusa, F., Villari, M., Puliafito, A.: Three-phase cross-cloud federation model: the cloud SSO authentication. In: Procs. AFIN 2010, pp. 94–101. IEEE CSP, Los Alamitos (2010)
12. Keahey, K., Tsugawa, M., Matsunaga, A., Fortes, J.: Sky computing. Internet Computing 13(5), 43–51 (2009)
13. Sheth, A., Ranabahu, A.: Semantic modeling for cloud computing, part 2. IEEE Internet Computing 14(4), 81–84 (2010)
14. Lee, C.A.: An open cloud computing interface status update (and roadmap dartboard) Cloud interoperability roadmaps sessions. In: OMG Technical Meeting (2009)
15. Hill, Z., Humphrey, M.: CSAL: A cloud storage abstraction layer to enable portable cloud applications. In: Procs. CloudCom 2010, pp. 504–511. IEEE CSP, Los Alamitos (2010)
16. Loutas, N., Peristeras, V., Bouras, T., Kamateri, E., Zeginis, D., Tarabanis, K.: Towards a reference architecture for semantically interoperable clouds. In: Procs. CloudCom 2010, pp. 143–150. IEEE CSP, Los Alamitos (2010)
17. Unified Cloud Interface, `http://code.google.com/p/unifiedcloud/`

18. Di Martino, B., Petcu, D., Cossu, R., Goncalves, P., Máhr, T., Loichate, M.: Building a mosaic of clouds. In: Guarracino, M.R., Vivien, F., Träff, J.L., Cannatoro, M., Danelutto, M., Hast, A., Perla, F., Knüpfer, A., Di Martino, B., Alexander, M. (eds.) Euro-Par-Workshop 2010. LNCS, vol. 6586, pp. 571–578. Springer, Heidelberg (2011)
19. Petcu, D., Craciun, C., Neagul, M., Lazcanotegui, I., Rak, M.: Building an interoperability API for sky computing. In: InterCloud 2011, pp. 405–412. IEEE CSP, Los Alamitos (2011)
20. Petcu, D., Craciun, C., Rak, M.: Towards a cross platform cloud API. Components for cloud federation. In: Procs. CLOSER 2011, pp. 166–169. SciTePress (2011)
21. Petcu, D., Crăciun, C., Neagul, M., Panica, S., Di Martino, B., Venticinque, S., Rak, M., Aversa, R.: Architecturing a sky computing platform. In: Cezon, M. (ed.) ServiceWave 2011 Workshops. LNCS, vol. 6569, pp. 1–13. Springer, Heidelberg (2011)