

Using Semantic Web Ontology for Intercloud Directories and Exchanges

David Bernstein
Huawei Technologies, USA
2330 Central Expressway
Santa Clara, CA 95050
dbernstein@huawei.com

Deepak Vij
Cloud Strategy Partners, LLC
3260 Nipoma Court
San Jose, CA 95135
deepak@cloudstrategypartners.com

*For submission to ICOMP'10 - The 2010 International Conference on Internet Computing, Las Vegas, NV Jul 12-15 2010
as part of WORLDCOMP'10 - The 2010 World Congress in Computer Science, Computer Engineering, and Applied Computing*



ICOMP'10 - The 2010 International Conference on Internet Computing



Pre-acceptance draft, not for distribution

Using Semantic Web Ontology for Intercloud Directories and Exchanges

David Bernstein
Huawei Technologies, USA
2330 Central Expressway
Santa Clara, CA 95050
dbernstein@huawei.com

Deepak Vij
Cloud Strategy Partners, LLC
3260 Nipoma Court
San Jose, CA 95135
deepak@cloudstrategypartners.com

*For submission to ICOMP'10 - The 2010 International Conference on Internet Computing, Las Vegas, NV Jul 12-15 2010
as part of WORLDCOMP'10 - The 2010 World Congress in Computer Science, Computer Engineering, and Applied Computing*

Contact Author: David Bernstein. Keywords: "Cloud Computing", "Intercloud", "Semantic Web", "RDF", "Ontology"

Abstract

The concept of a cloud operated by one service provider or enterprise interoperating with a clouds operated by another is a powerful idea. So far that is limited to use cases where code running on one cloud explicitly references a service on another cloud. There is no implicit and transparent interoperability. This interoperability should be more than cloud to cloud, it should embody 1-to-many and many-to-many models. Working groups have proposed building a layered set of protocols to solve this interoperability challenge called "Intercloud Protocols". Instead of each cloud provider establishing connectivity with another cloud provider in a Point-to-Point manner resulting into n^2 complexity problem, Intercloud Directories and Exchanges will help facilitate as mediators for enabling connectivity and collaboration among disparate cloud providers. This paper proposes a mechanism for such mediation utilizing a resources catalog approach, defined using the Semantic Web Resource Definition Framework (RDF) and a common Ontology of Cloud Computing Resources across heterogeneous cloud providers.

1. Introduction

Cloud Computing has emerged recently as a label for a particular type of datacenter. For the purposes of this paper, we define Cloud Computing as a datacenter/s which:

1. May be hosted by anyone; an enterprise, a service provider, or a government.
2. Implement a pool of computing resources and services which are shared amongst subscribers.
3. Charge for resources and services using an "as used" metered and/or capacity based model.

4. Are usually geographically distributed, in a manner which is transparent to the subscriber (unless they explicitly ask for visibility of that).
5. Are automated in that the provisioning and configuration (and de-configuration and un-provisioning) of resources and services occur on a "self service" basis, usually programmatic request of the subscriber, occur in an automated way with no human operator assistance, and are delivered in one or two orders of seconds.
6. Resources and services are delivered virtually, that is, although they may appear to be physical (servers, disks, network segments, etc) they are actually virtual implementations of those on an underlying physical infrastructure which the subscriber never sees.
7. The physical infrastructure changes rarely. The virtually delivered resources and services are changing constantly.
8. Resources and services may be of a physical metaphor (servers, disks, network segments, etc) or they may be of an abstract metaphor (blob storage functions, message queue functions, email functions, multicast functions, all of which are accessed by running of code or script to a set of API's for these abstract services). These may be intermixed.

Cloud Computing services as defined above are best exemplified by the Amazon Web Services (AWS) [1][2] or Google AppEngine [3][4]. Both of these systems exhibit all eight characteristics as detailed above. Various companies are beginning to offer similar services, such as the Microsoft Azure Service [5], and software companies such as VMware [6] and open source projects such as UCSB Eucalyptus [7][8] are creating software for building a cloud service.

In case 8, where the resources and services are of a physical metaphor, the cloud is said to be exposing “Infrastructure as a Service”, or IaaS. In the last case described above (number 8), where the resources and services are of an abstract metaphor, the cloud is said to be exposing “Platform as a Service”, or PaaS. A PaaS cloud looks like a remote, virtual, distributed implementation of a managed code container, or “Application Server”, similar to J2EE [9] or .NET [10]. The terms are well accepted now [11].

Use Cases and Scenarios for Cloud IaaS and PaaS interoperability [12][13] have been detailed in the literature along with the challenges around actually implementing standards-based Intercloud federation and hybrid clouds. Work detailing high level architectures for Intercloud interoperability were proposed next [14][15]. More recently, specific implementation approaches for Intercloud protocols [16][17] have been proposed, including specifically Extensible Messaging and Presence Protocol (XMPP) [18][19] for transport, and using Semantic Web [20] techniques such as Resource Description Framework (RDF) [21] as a way to specify resources.

Following that work outlining approaches for Intercloud protocols, a detailed analysis on the feasibility of XMPP was explored after that [22]. The work went through considerable detail to implement various XMPP-based control plane operations for Intercloud:

- Fitting XMPP into an Intercloud Topology
- Securing the XMPP conversation using TLS
- Authentication over XMPP using SAML
- Service Invocation over XMPP using IO Data XEP, XMPP Web Services for Java (xws4j)
- RDF and SPARQL within XMPP
- XMPP Java API to a Cloud Service

The conclusion was that for each of these techniques it found XMPP to be flexible and usable. This paper moves to the next topic, by continuing to investigate the blueprint set out as referenced [16][17]. We now investigate how Cloud Computing resources can be described, cataloged, and mediated using Semantic Web Ontologies, implemented using RDF techniques.

2. Intercloud Topology

Cloud instances must be able to dialog with each other. One cloud must be able to find one or more other clouds, which for a particular interoperability scenario is ready, willing, and able to accept an interoperability transaction with and furthermore, exchanging whatever

subscription or usage related information which might have been needed as a pre-cursor to the transaction. Thus, an Intercloud Protocol for presence and messaging needs to exist which can support the 1-to-1, 1-to-many, and many-to-many Cloud to Cloud use cases.

The vision and topology for the Intercloud we will refer to [12][13] is as follows. At the highest level, the analogy is with the Internet itself: in a world of TCP/IP and the WWW, data is ubiquitous and interoperable in a network of networks known as the “Internet”; in a world of Cloud Computing, content, storage and computing is ubiquitous and interoperable in a network of Clouds known as the “Intercloud”; this is illustrated in Figure 1.

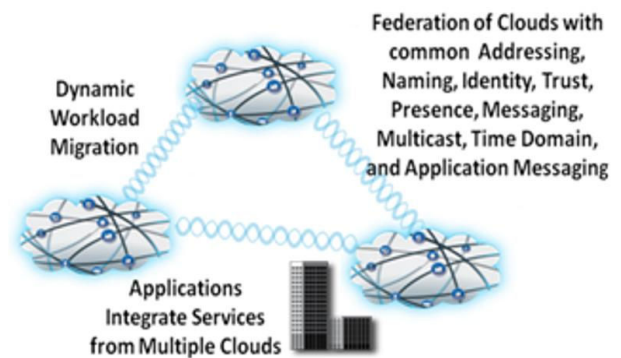


Figure 1. The Intercloud Vision

The reference topology for realizing this vision is modeled after the public Internet infrastructure. Again, using the generally accepted terminology [11][12][13][14][15][18][19], there are Public Clouds, which are analogous to ISP’s and Service Providers offering routed IP in the Internet world. There are Private Clouds which is simply a Cloud which an organization builds to serve itself.

There are Intercloud Exchanges (analogous to Internet Exchanges and Peering Points) where clouds can interoperate, and there is an Intercloud Root, containing services such as Naming Authority, Trust Authority, Directory Services, and other “root” capabilities. It is envisioned that the Intercloud root is of course physically not a single entity, a global replicating and hierarchical system similar to DNS [23] would be utilized. All elements in the Intercloud topology contain some gateway capability analogous to an Internet Router, implementing Intercloud protocols in order to participate in Intercloud interoperability. We call these Intercloud Gateways. The entire topology is detailed in Figure 2.

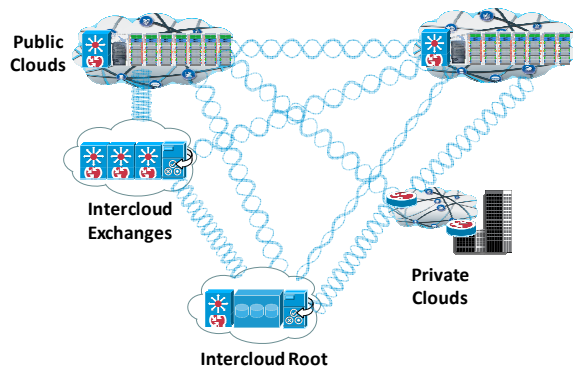


Figure 2. Reference Intercloud Topology

The Intercloud Gateways would provide mechanism for supporting the entire profile of Intercloud protocols and standards. The Intercloud Root and Intercloud Exchanges would facilitate and mediate the initial Intercloud negotiating process among Clouds. Once the initial negotiating process is completed, each of these Cloud instance would collaborate directly with each other via a protocol and transport appropriate for the interoperability action at hand; for example, a reliable protocol might be needed for transaction integrity, or a high speed streaming protocol might be needed optimized for data movement over a particular link.

3. Intercloud Root, Exchanges, and Catalog

Various providers will emerge in the enablement of the Intercloud. We first envision a community governed set of Intercloud Root providers who will act as brokers and host the Cloud Computing Resource Catalogs for the Intercloud computing resources. They would be governed in a similar way in which DNS, Top Level Domains [24] or Certificate Authorities [25] are, by an organization such as ISOC [26] or ICANN [27]. They would also be responsible for mediating the trust based federated security among disparate clouds by acting as Security Trust Service providers using standards such as SASL [28] and SAML [29].

The Intercloud Root instances will work with Intercloud Exchanges to solve the n^2 problem by facilitating as mediators for enabling connectivity among disparate cloud environments. This is a much preferred alternative to each cloud vendor establishing connectivity and collaboration among themselves (point-to-point), which would not scale physically or in a business sense.

Intercloud Exchange providers will facilitate the negotiation dialog and collaboration among disparate

heterogeneous cloud environments, working in concert with Intercloud Root instances as described previously [22]. Intercloud Root instances will host the root XMPP servers containing all presence information for Intercloud Root instances, Intercloud Exchange Instances, and Internet visible Intercloud capable Cloud instances. Intercloud Exchanges will host second-tier XMPP servers. Individual Intercloud capable Clouds will communicate with each other, as XMPP clients, via XMPP server environment hosted by Intercloud Roots and Intercloud Exchanges.

In order for the Intercloud capable Cloud instances to federate or otherwise interoperate resources, a Cloud Computing Resources Catalog system is necessary infrastructure. This catalog is the holistic and abstracted view of the computing resources across disparate cloud environments. Individual clouds will, in turn, will utilize this catalog in order to identify matching cloud resources by applying certain Preferences and Constraints to the resources in the computing resources catalog.

The technologies to use for this are based on the Semantic Web which provides for a way to add “meaning and relatedness” to objects on the Web. To accomplish this, one defines a system for normalizing meaning across terminology, or Properties. This normalization is called an Ontology.

4. Ontology based Cloud Computing Resources Catalog

The Intercloud system not only focuses on the provisioning of computing resources inside a single cloud; it provides a holistic and abstracted view of the computing resources across disparate cloud environments. Participating cloud providers will advertise their resource capabilities within the cloud computing resources catalog hosted by Intercloud Root Providers. Management of the thousands of resources and configurations requires careful control and planning to achieve business objectives and avoid errors. The chief objectives of the planned configuration are to provide cost effective use of computing resources and to meet the business objectives of the enterprise.

In order to automate an environment whereby software agents versus traditional human users discover and consume services, intelligent ontology based service registries are needed for dynamically discovering and provisioning computing resources across various computing cloud environments (Amazon, Azure etc. etc.).

Comprehensive semantic descriptions of services are essential to exploit them in their full potential. That is discovering them dynamically, and enabling automated service negotiation, composition and monitoring. The semantic mechanisms currently available in service registries such as UDDI [30] are based on taxonomies called “tModel” [31]. tModel fails to provide the means to achieve this, as they do not support semantic discovery of services [32][33]. tModel supports a construct which serves two purposes: it can serve as a namespace for a taxonomy or as a proxy for a technical specification that lives outside the registry. Such a tModel construct has some intrinsic limitations, for example classifications for the Intercloud use case can also be defined for individual operations or their argument types. However, this requires searching mechanisms for services that are distinct from those for their argument types. Likewise, tModel’s reference to an external technical specification, as applied in UDDI also implies that a different mechanism is required for reasoning over service interfaces.

Although the terms “taxonomy” and “ontology” are sometimes used interchangeably, there is a critical difference. Taxonomy indicates only class/subclass relationship whereas Ontology describes a domain completely. The essential mechanisms that ontology languages provide include their formal specification (which allows them to be queried) and their ability to define properties of classes. Through these properties, very accurate descriptions of services can be defined and services can be related to other services or resources. We are proposing a new and improved service directory on the lines of UDDI but based on RDF/OWL [34] ontology framework instead of current tModel based taxonomy framework. This catalog captures the computing resources across all clouds in terms of “Capabilities”, “Structural Relationships” and Policies (Preferences and Constraints).

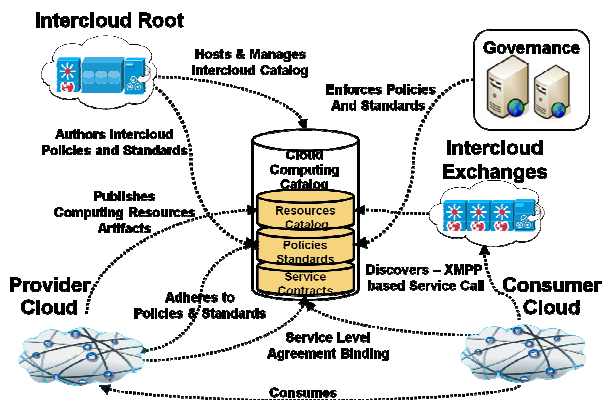


Figure 3. Cloud Computing Catalog

Effective cloud computing resources ontology information captured in the catalog provides the following advantages:

- Consolidated view of Computing Resources across clouds. Consistent way to expose Services Offered.
- Provide visibility and access to contractual information at any point in time.
- Provides the ability to protect sensitive information from unauthorized access. Configuration resources will have security restrictions applied against them.
- Governance Processes
- One-Stop/Consistency
- Time-to-Value
- Overall Effectiveness

5. Cloud Computing Resources Ontology

One of the paramount and key goals of an intercloud enabled cloud provider is not just to be able to offer vast computing resources but provide complete visibility and transparency to these resources, at the same time. Providing transparency and visibility in this manner ensures that the services and resources meet and are in compliance with the architectural, functional, policies and constraints requirements of other cloud providers.

In order to ensure that the requirements of an intercloud enabled cloud provider are correctly matched to the infrastructure capabilities in an automated fashion, there is a need for declarative semantic model that can capture both the requirements and constraints of computing resources.

Over the last several years, there has been ongoing effort around automation of datacenter/s by companies such as Elastra [35]. Elastra has defined a modeling language called EDML [36] for specifying the datacenter computing resources semantics in terms of XML based markup language.

We are proposing a similar ontology based semantic model that captures the features and capabilities available from a cloud provider’s infrastructure. These capabilities are logically grouped together and exposed as standardized units of provisioning and configuration to be consumed by another cloud provider/s. These capabilities are then associated with policies and constraints for ensuring compliance and access to the computing resources.

The proposed ontology based model not only consists of physical attributes but quantitative & qualitative attributes such as “Service Level Agreements (SLAs)”, “Disaster Recovery” policies, “Pricing” policies, “Security & Compliance” policies, and so on.

The following is a high level schematic of such ontology based semantic model.

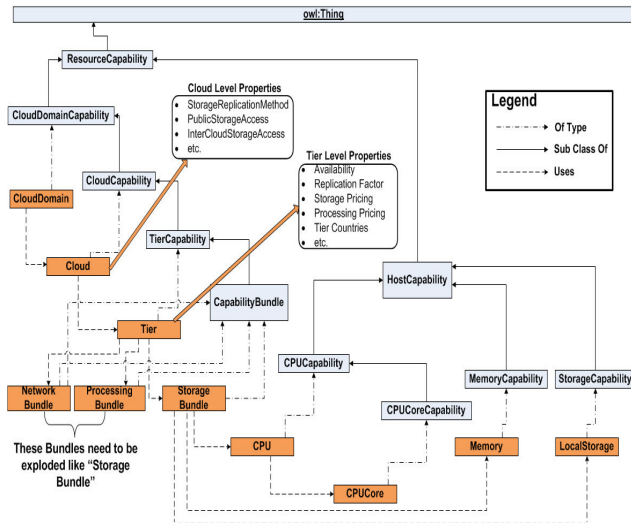


Figure 4. Cloud Computing Resources Ontology

At a very basic level, the RDF model is called a “triple” as it consists of three parts, Subject/Property/Object. It essentially contains one or more “descriptions” of resources. A “description” is a set of statements about a resource. It is structurally similar to entity/attribute/value. Essentially, a statement in RDF pulls resources, properties, and property values together. Statements are typically called triples because they include a subject (the resource), a predicate/verb (the property), and an object (the property value or another resource itself).

RDF allows you to define a group of things with common characteristics called “Classes”. “Classes” are allowed to inherit characteristics and behaviors from a parent class. Each user-defined class is implicitly a subclass of super class called “owl:Thing”.

The hierarchy of user-defined classes in our proposed ontology scheme are “ResourceCapability” → “CloudDomainCapability” → “CloudCapability” → “TierCapability” → “CapabilityBundle”.

In order to demonstrate a working example, the following is a code snippet of N-Triples [38] based ontology semantic model instead. N-Triples & Turtle [39] are a human-friendlier alternative to RDF/XML. N-Triples or Turtle code, in turn, can be easily converted to RDF/XML format using a converter tool.

The following sample shows the flow for semantic model for cloud computing resources. Due to the large

size of the proposed semantic model for cloud computing resources, we are unable to capture the sample RDF code snippet in this document. In order to demonstrate our working example, we are showing N-Triples [38] code snippet instead.

Step 1: In our ontology example, “CloudDomain” is an instance of class “CloudDomainCapability”. It consists of three resources “Cloud.1”, “Cloud.2” & “Cloud.3”:

```
<http://cloud/domain>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/#cloud.1>.

<http://cloud/domain>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/#cloud.2>.

<http://cloud/domain>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/#cloud.3>.

<http://cloud/domain> <http://www.w3.org/1999/02/22-
rdf-syntax-ns#type>
<http://www.csp/resOntology#CloudDomainCapability>.

<http://cloud/domain> <http://www.w3.org/2000/01/rdf-
schema#label> "Cloud Computing
domain"^^<http://www.w3.org/2001/XMLSchema#string>.
```

Step 2: “Cloud.1”, in turn, consists of tier instances “tier.1”, “tier.2” & “tier.3”:

```
<http://cloud/domain/#cloud.1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1#tier1>.

<http://cloud/domain/#cloud.1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1#tier2>.

<http://cloud/domain/#cloud.1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1#tier3>.
```

Step 3: Each of these cloud instances has associated properties such as “StorageReplicationMethod”, “InterCloudStorageAccess” etc. etc. These properties are, in turn, used for determining if the computing resources of a cloud provider meet the preferences & constraints of the requesting cloud’s interest and requirements:

```
<http://cloud/domain/#cloud.1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1#Storage-Replication-
Method>.

<http://cloud/domain/#cloud.1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1#Inter-Cloud-Storage-
Access>.

<http://cloud/domain/#cloud.1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1#Public-Storage-Access>.

<http://cloud/domain/#cloud.1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1#VPNGatewayAddress>.
```

```
<http://cloud/domain/#cloud.1>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.csp/resOntology#CloudCapability>.
```

```
<http://cloud/domain/#cloud.1>
<http://www.w3.org/2000/01/rdf-schema#label> "Cloud
1"^^<http://www.w3.org/2001/XMLSchema#string>.
```

Step 4: Computing resources are logically grouped together as bundles and exposed as standardized units of provisioning and configuration to be consumed by another cloud provider/s. These bundles are “StorageBundle”, “ProcessingBundle” & “NetworkBundle”. Each “Tier”, in turn, consists of instances of resource bundles such as “StorageBundle” etc. Each “Tier” also has its own associated properties depicting preferences and constraints:

```
<http://cloud/domain/cloud.1#tier1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1/bundle/#storage1>.
```

```
<http://cloud/domain/cloud.1#tier1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1/bundle/#processing1>.
```

```
<http://cloud/domain/cloud.1#tier1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1/bundle/#network1>.
```

```
<http://cloud/domain/cloud.1#tier1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1.tier.1#replicationfactor>
.
```

```
<http://cloud/domain/cloud.1#tier1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1.tier.1#availability>.
```

```
<http://cloud/domain/cloud.1#tier1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1.tier.1#storageprice>.
```

```
<http://cloud/domain/cloud.1#tier1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1.tier.1#processingprice>.
```

```
<http://cloud/domain/cloud.1#tier1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1.tier.1#countries>.
```

```
<http://cloud/domain/cloud.1#tier1>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.csp/resOntology#TierCapability>.
```

Step 5: “StorageBundle”, in turn, consists of resources such as “CPU”, “CPU Cores”, “Memory” & “LocalStorage”:

```
<http://cloud/domain/cloud.1/bundle/#storage1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1/bundle/storage1#CPU>.
```

```
<http://cloud/domain/cloud.1/bundle/#storage1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage0>.
```

```
<http://cloud/domain/cloud.1/bundle/#storage1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage1>.
```

```
<http://cloud/domain/cloud.1/bundle/#storage1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage2>.
```

```
<http://cloud/domain/cloud.1/bundle/#storage1>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1/bundle/storage1#Memory>.
```

```
<http://cloud/domain/cloud.1/bundle/#storage1>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.csp/resOntology#CapabilityBundle>.
```

```
<http://cloud/domain/cloud.1/bundle/#storage1>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.csp/resOntology#StorageCapabilityBundle>.
```

```
<http://cloud/domain/cloud.1/bundle/#storage1>
<http://www.w3.org/2000/01/rdf-schema#label> "EC2
Large"^^<http://www.w3.org/2001/XMLSchema#string>.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage1>
<http://www.csp/resOntology#quantity>
"450971566080"^^<http://www.w3.org/2001/XMLSchema#long>
.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage1>
<http://www.csp/resOntology#unit>
<http://www.csp/resOntology#Byte>.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage1>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.csp/resOntology#StorageCapability>.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage0>
<http://www.csp/resOntology#quantity>
"450971566080"^^<http://www.w3.org/2001/XMLSchema#long>
.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage0>
<http://www.csp/resOntology#unit>
<http://www.csp/resOntology#Byte>.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage0>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.csp/resOntology#StorageCapability>.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage2>
<http://www.csp/resOntology#quantity>
"10737418240"^^<http://www.w3.org/2001/XMLSchema#long>
.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage2>
<http://www.csp/resOntology#unit>
<http://www.csp/resOntology#Byte>.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#LocalStorage2>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.csp/resOntology#StorageCapability>.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#Memory>
<http://www.csp/resOntology#quantity>
"8053063680"^^<http://www.w3.org/2001/XMLSchema#long>.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#Memory>
<http://www.csp/resOntology#unit>
<http://www.csp/resOntology#Byte>.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#Memory>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.csp/resOntology#MemoryCapability>.
```

```
<http://cloud/domain/cloud.1/bundle/storage1#CPU>
<http://www.csp/resOntology#hasCapability>
<http://cloud/domain/cloud.1/bundle/storage1#CPUCore>.
```

```

<http://cloud/domain/cloud.1/bundle/storage1#CPU>
<http://www.csp/resOntology#hasCapability>
<http://www.csp/resOntology#X86-64Compatible>.

<http://cloud/domain/cloud.1/bundle/storage1#CPU>
<http://www.csp/resOntology#quantity>
"2200000000"^^<http://www.w3.org/2001/XMLSchema#long>.

<http://cloud/domain/cloud.1/bundle/storage1#CPU>
<http://www.csp/resOntology#unit>
<http://www.csp/resOntology#Hertz>.

<http://cloud/domain/cloud.1/bundle/storage1#CPU>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.csp/resOntology#CPUCapability>.

<http://cloud/domain/cloud.1/bundle/storage1#CPUCore>
<http://www.csp/resOntology#quantity>
"2"^^<http://www.w3.org/2001/XMLSchema#int>.

<http://cloud/domain/cloud.1/bundle/storage1#CPUCore>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.csp/resOntology#CPUCoreCapability>.

```

6. SPARQL Query Language

SPARQL [39] (SPARQL Protocol And RDF Query Language) is a very powerful SQL-like language for querying and making semantic information machine process-able. The structure and example of a SPARQL Query is illustrated in Figure 5.

Structure:

```

PREFIX: Prefix definition (optional)
SELECT: Result form
FROM: Data sources (optional)
WHERE: Graph pattern (=path expression)
    •FILTER
    •OPTIONAL

```

Example:

```

PREFIX geo: <http://www.geography.org/schema.rdf#>
SELECT ?X ?Y
FROM <http://www.geography.org>
WHERE { ?X geo:hasCapital ?Y.
        ?Y geo:areacode ?Z }
ORDER BY ?X

```

Figure 5. Structure & Example of SPARQL Query

SPARQL provides a very powerful language for executing very complex queries into the RDF data which are often necessary. In our case, the following example query applies certain Preferences and Constraints to the resources in the computing semantics catalog for determining if the service description on another cloud meets the constraints of the first cloud's interest:

```

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?cld1 ?cld2 ?cld3 ?cld4 ?cld5
WHERE { ?cld1
<http://www.csp/resOntology#availabilityQuantity> ?avail
abilityQuantity .

```

```

?cld2
<http://www.csp/resOntology#replicationFactor> ?repl
icationFactor .
?cld3
<http://www.csp/resOntology#tierCountries> ?tierCountr
ies .
?cld4
<http://www.csp/resOntology#StorageReplicationMethod>
?StorageReplicationMethod .
?cld5 <http://www.csp/resOntology#
InterCloudStorageAccess > ?InterCloudStorageAccess .

FILTER ( ?availabilityQuantity = 99.999 )
FILTER ( ?replicationFactor = 5)
FILTER ( ?tierCountries = "Japan")
FILTER ( ?StorageReplicationMethod = "AMQP")
FILTER ( ?InterCloudStorageAccess = "NFS")

}

```

6.1. SPARQL Query over Hadoop

Due to very large size of “Cloud Ontology” set in the intercloud environment, we are expecting a very large RDF dataset. SPARQL queries against such a large RDF dataset would be highly inefficient and slow. We believe that such a large RDF dataset should be stored on a Distributed File System such as HDFS (Hadoop Distributed File System). By storing RDF dataset in HDFS and querying through Hadoop [40] “Map-Reduce” programming would make SPARQL queries highly efficient and faster.

We propose that the Intercloud Exchanges will leverage Hadoop based distributed processing for serving SPARQL request across federated resource catalogs hosted by Intercloud Root providers.

7. Conclusions and Future Work

We have gone into great detail to test the proposal that Intercloud Exchanges in conjunction with Ontology based Computing Resources Catalog and XMPP protocol are the key components for enablement of “Federated Cloud” environment.

As to continuing work, we are continuing to develop the suite of Intercloud protocols. With the proposed Intercloud Exchanges, XMPP protocol and RDF Ontology based Resources Catalog; we should be able to demonstrate an end-to-end comprehensive “Federated Cloud Storage” use case for Intercloud next.

8. References

- [1] Amazon Web Services at <http://aws.amazon.com/>
- [2] James Murty, *Programming Amazon Web Services; S3, EC2, SQS, FPS, and SimpleDB*, O’Reilly Press, 2008.

- [3] Google AppEngine at <http://code.google.com/appengine/>
- [4] Eugene Ciurana, *Developing with Google App Engine*, Firstpress, 2009.
- [5] Microsoft Azure, at <http://www.microsoft.com/azure/default.aspx>
- [6] VMware VCloud Initiative at <http://www.vmware.com/technology/cloud-computing.html>
- [7] Nurmi D., Wolski R., Grzegorzczak C., Obertelli G., Soman S., Youseff L., Zagorodnov D., *The Eucalyptus Open-source Cloud-computing System*, Proceedings of Cloud Computing and Its Applications, Chicago, Illinois (October 2008)
- [8] Nurmi D., Wolski R., Grzegorzczak C., Obertelli G., Soman S., Youseff L., Zagorodnov D., *Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems*, UCSB Computer Science Technical Report Number 2008-10 (August 2008)
- [9] JSR 88: Java Enterprise Edition Application Deployment at <http://jcp.org/en/jsr/detail?id=88>
- [10] Microsoft .NET at <http://www.microsoft.com/net/>
- [11] Youseff, L. and Butrico, M. and Da Silva, D., *Toward a unified ontology of cloud computing*, GCE'08 Grid Computing Environments Workshop, 2008.
- [12] Lijun Mei, W.K. Chan, T.H. Tse, *A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues*, APSCC pp.464-469, 2008 IEEE Asia-Pacific Services Computing Conference, 2008
- [13] *Cloud Computing Use Cases Google Group (Public)*, at <http://groups.google.com/group/cloud-computing-use-cases>, <http://www.scribd.com/doc/18172802/Cloud-Computing-Use-Cases-Whitepaper>, accessed March 2010
- [14] Buyya, R. and Pandey, S. and Vecchiola, C., *Cloudbus toolkit for market-oriented cloud computing*, Proceeding of the 1st International Conference on Cloud Computing (CloudCom), 2009
- [15] Yildiz M, Abawajy J, Ercan T., Bernoth A., *A Layered Security Approach for Cloud Computing Infrastructure*, ISPAN, pp.763-767, 10th International Symposium on Pervasive Systems, Algorithms, and Networks, 2009
- [16] Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., and Morrow, M., *Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability*, ICIW '09. Fourth International Conference on Internet and Web Applications and Services, pp. 328-336, 2009
- [17] Bernstein, D., *Keynote 2: The Intercloud: Cloud Interoperability at Internet Scale*, NPC, pp.xiii, 2009 Sixth IFIP International Conference on Network and Parallel Computing, 2009
- [18] *Extensible Messaging and Presence Protocol (XMPP): Core*, and related other RFCs at <http://xmpp.org/rfcs/rfc3920.html>
- [19] XMPP Standards Foundation at <http://xmpp.org/>
- [20] W3C Semantic Web Activity, at <http://www.w3.org/2001/sw/>
- [21] *Resource Description Framework (RDF)*, at <http://www.w3.org/RDF/>
- [22] Bernstein, D., Vij, D., *Using XMPP as a transport in Intercloud Protocols*, submitted to 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10), for publication June 2010
- [23] *Domain Names – Concepts and Facilities*, and related other RFCs, at <http://www.ietf.org/rfc/rfc1034.txt>
- [24] *Domain Name System Structure and Delegation*, at <http://www.ietf.org/rfc/rfc1591.txt>
- [25] *Internet X.509 Public Key Infrastructure, Certificate Policy and Certification Practices Framework*, at <http://tools.ietf.org/html/rfc3647>
- [26] *The Internet Society*, at <http://www.isoc.org/>
- [27] *The Internet Corporation for Assigned Names and Numbers*, at <http://www.icann.org/>
- [28] *Simple Authentication and Security Layer (SASL)*, at <http://tools.ietf.org/html/rfc4422>
- [29] *Security Assertion Markup Language (SAML)*, at <http://saml.xml.org/saml-specifications>
- [30] *OASIS UDDI Specification TC*, at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec
- [31] *UDDI Registry tModels*, at http://www.uddi.org/taxonomies/UDDI_Registry_tModels.htm
- [32] Paolucci, M., Kawamura T., Payne T., and Sycara K., *Importing the Semantic Web in UDDI*, Web Services, E-Business and Semantic Web Workshop, 2002.
- [33] Moreau, L. and Miles, S. and Papay, J. and Decker, K. and Payne, T., *Publishing semantic descriptions of services*, First GGF Semantic Grid Workshop, held at the Ninth Global Grid Forum, Chicago IL, USA, 2003
- [34] *Web Ontology Language*, at <http://www.w3.org/TR/owl-features/>
- [35] Elastra, at <http://www.elastra.com>
- [36] EDML, at <http://www.elastra.com/technology/languages/edml>
- [37] N-Triples, at <http://www.w3.org/2001/sw/RDFCore/ntriples/>
- [38] Turtle – Terse RDF Triple Language, at <http://www.w3.org/TeamSubmission/turtle/#sec-diff-n3>
- [39] *SPARQL Query Language for RDF*, at <http://www.w3.org/TR/rdf-sparql-query/>
- [40] *Hadoop*, at <http://hadoop.apache.org/>