DELIVERABLE D3.1.3

Grant Agreement number : CIP-297300
Project acronym : InGeoCLOUDS
Project title : INspired GEOdata CLOUD Services

Funding Scheme : Pilot B

Analysis and Monitoring of Clouds for Geo-Data Services

D3.1.3

Version 1.0

Reference **D3.1.3-INGC**

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contract Number :	CIP-297300
Document Title :	Analysis and Monitoring of Clouds for Geo-Data Services
Document version :	1.0
Document status :	Approved
Date :	2014-05-14
WP contributing to the deliverable :	*WP3*
Availability :	*Public*
Authors :	CNR
Approved by :	InGeoCloudS Steering Committee

Abstract

This document is the third issue of a series of 3 documents. This document updates the previous review of cloud offer from both a technical and commercial point of view. To this end, the cost model of the InGeoCloudS platform is updated according to the actual resource consumption observed so far. This document also provides detailed analysis and considerations about moving the current architecture on a different cloud platform provider.

Keywords List *Cloud Service Providers, Cost Estimate, Cloud Service Providers Monitoring.*

DOCUMENT CHANGE LOG

Document Issue.	Date	Reasons for change
Version 1-Draft 1	2014-02-01	Creation of the document
Version 1-Draft 2	2014-04-08	Integration of first round of contributions
Version 1-Draft 3	2014-04-24	Integration of second round of contributions
Version 1-Draft 4	2014-05-02	Integration of a third round of contributions
Version 1-Approved	2014-05-14	Final editorial changes approved by steering committee

APPLICABLE AND REFERENCE DOCUMENTS (A/R)

A/R and Document Reference	Title
[A1] ICT PSP Grant Agreement N° CIP 297300	InGEOCloudS Grant Agreement and its annex (including the description of work)
[R1] D3.1.1-INGC	Analysis and Monitoring of clouds for geo-data services
[R2] D3.1.2-INGC	Analysis and Monitoring of clouds for geo-data services
[R3] D3.2-INGC	Cloud architecture, configuration and data access implementation

Table of Contents

List of Tables

1. INTRODUCTION

1.1. ACRONYMS AND DEFINITIONS

Term	Definition
N/A	Not Applicable
CSP	Cloud Service Provider
GIS	Geographic Information System
SEAM	An open source development platform for building rich Internet applications in Java
SPARQL	Sparql Protocol And RDF Query Language
SQL	Structured Query Language
WSDL	Web Service Description Language

Table 1: Acronyms and Definitions

1.2. OBJECTIVES OF THE DOCUMENT

This document is the last issue of a series of 3 documents. These documents survey existing cloud service providers (CSP) with the goal of choosing the best platform for hosting the InGeoCloudS infrastructure. This document reports on the activity conducted within task T3.5 “Monitoring of cloud costs and technologies”: it discusses and updates on the same topics covered by the previous deliverable D3.1.2 [R2], and in addition it reports on a detailed analysis of the impact on the InGeoCloudS infrastructure of moving to a different cloud service provider.

The main objectives of this document are:

- to update, if necessary, the requirements and the criteria driving the choice of a specific cloud platform;
- to review the state of the art of cloud service providers with the goal of corroborating or repudiate the previous choice of the Amazon cloud;
- to verify the economic convenience of the Amazon cloud and possible migration opportunities.
- to describe the impact and the development cost of moving the current infrastructure to a different cloud service provider.

1.3. OVERVIEW OF THE DOCUMENT

In Section 2 we report on the costs of the InGeoCLOUDS deployment on the Amazon AWS cloud service until February 2014, and in particular it reviews the resource used by Pilot 2 so as to define the InGeoCLOUDS deployment requirement needed to estimate the cost and the feasibility of deploying InGeoCLOUDS on a different cloud platform. A comparison among different providers is discussed. Section 3 discusses the technical issues and impact due to moving the InGeoCLOUDS platform to a different clouds service provider. Section 4 draws some final conclusions.

2. COST MODEL REVISION

The cost model described below is updated on the basis of the actual exploitation of the Amazon AWS platform after the deployment of Pilot 2.

We first recall the InGeoCLOUDS architecture deployment and report on the resources used. As described in D3.2-INGC [R3], the deployment of InGeoCLOUDS can be described in terms of layers:

- The ***Elastic Filer Server Layer***: we are using two running instances to manage the distributed file system of the platform.
- The ***Elastic Database Server Layer***: we are using three instances to support high-availability of the relational geo-database functionality.
- The ***Elastic Map Server Layer*** plus GeoPublication and GeoNetwork resources: we are using just one instance exposing map service functionalities, plus two instances respectively for the GeoPublication and GeoNetwork resources.
- The ***Elastic Linked Data Storage Layer***: also in this case we are currently using just one instance.
- The ***Geo-Computational Layer***: in this layer resources are allocated on demand, e.g., for the shake-map use case. When no service is running, there is only one instance running to offer the kriging geo-processing service.
- The ***InGeoCLOUDS Web Portal Layer***: two instances are hosting the Apache and the Tomcat based web servers respectively.
- The ***InGeoCLOUDS Backend***: uses one running instance to host the InGeoCLOUDS API server, and one AWS's Relational Database Service (RDS) instance for storing the monitoring and accounting data.

We used three kinds of instances, which are named *m1.small*, *m1.medium* and *m1.large*, according to the Amazon AWS nomenclature. They correspond to increasingly powerful instances. Their main features can be summarized in following table:

Table 2: Summary table of the kind of Amazon instances used.

AWS	*m1.small*	*m1.medium*	*m1.large*

Names			
References	S	M	L
CPU Cores	1	2	4
Memory	1.7 GB	3.75 GB	7.5 GB

Hereinafter, we simply refer to such instance types as *S* (small), *M* (medium) and *L* (large). The most powerful instance is used for the Elastic Linked Data Storage, since triple store processing is a compute-demanding task. The medium type of instance was used for the Map Server Layer and for the GeoPublication service. The less powerful instance was used for every other service. Note that the RDS features are the same as the m1.small instance.

The total number of required instances is thus 13, which is larger than the previous requirements discussed in D3.1.2-INGC [R2]. This is due to the decision of moving the storage of the monitoring and accounting data to a RDS instance, and to the introduction of a new computing instance for the kriging service.

In Table 3 we report the cost of the Amazon platform from May 2013 to February 2014. In the first column we recall the cost estimated in D3.1.2-INGC [R2] and in the second column we report the actual cost. At first glance the difference is not small. Indeed, the different is mainly due to the presence of additional platform, e.g., for development, with equal cost. In particular, from May to June, the cost of about €1,600 is due to the simultaneous presence of Pilot 1 and a development platform for Pilot 2. From August to September the cost increases to about €2,500 per month, due to the additional deployment of the actual Pilot 2.

The greatest cost occurs in October with a total of about €4,500. Such a large cost is due to the advance payment of reserved instances. Indeed, Amazon provides a billing method that guarantees a reduced hourly cost for one year with an upfront payment. If instances are use intensively, in InGeoCloudS most instances are up and running 24 hours a day, the total cost including the upfront payment is significantly smaller than the cost resulting from standard on-demand instance billing. In addition, from October 2013 Amazon allows to change the type of instance bought (e.g., switching two small instances to a medium one) which allows changing the deployment configuration over time without loosing the cost reduction of reserved instances. Since November 2013 the cost of the platform is about €800,00 and stable. This is the cost of Pilot 2 only.

In the last column of Table 3 we report the actual cost of the platform by splitting the advance reservation cost across one year in monthly fees. This is a correct figure of the monthly cost of the platform. We observe that the cost of Pilot 2 (i.e., from November to February) is about €100,00 higher than expected. This is not surprising since the requirements figures used in D3.1.2 were actually underestimating the resources eventually used in Pilot 2. On the other hand, we can claim that cost estimation of a cloud platform can be done with sufficiently good precision, ad we observed throughout the project. Of course, it is advisable to pay attention to the additional cost of the development period, which in our case required having multiple platforms running simultaneously on the cloud.

Table 3: Amazon AWS Costs.

Month	***D3.1.2 Estimated Cost (€)***	***Total Actual Cost (€)***	***Total Cost with Reserved Instances month split (€)***
2013 May	899.97	1,457.41	1,457.41
2013 June	899.97	1,483.95	1,483.95
2013 July	899.97	1,725.25	1,725.25
2013 August	899.97	2,493.80	2,493.80
2013 September	899.97	2,400.00	2,400.00
2013 October	899.97	4,510.61	1,852.28
2013 November	899.97	717.20	958.87
2013 December	899.97	714.10	955.77
2014 January	899.97	825.43	1067.10
2014 February	899.97	730.00	971.67

Our goal is to estimate the cost of the InGeoCloudS platform after pilot 2. To this end we proceed as follows:

- we fix the number of required instances and instance types to the production platform configuration described above;
- then we approximate the total running time of a single instance to 730 hours per month, i.e., 365 days evenly split in 12 months;
- finally, we approximate the other requirements, such as storage space, by using the actual amount billed by Amazon.

In Table 4 we compare the estimate produced in in D3.1.2-INGC [R2] with the actual resources used for Pilot 2 according the Amazon AWS billing. We recall that the previous estimated had to take into account that the cloud resources were used also for the development and testing activities. This time, since the amazon resources were used only for the InGeoCloudS Pilot 2 platform, we can more precisely measure the consumed resources.

We can observe a number of differences in all resources required by Pilot 2. As discussed above the total number of instances has changed from 11 to 13, due to the deployment of services in independent instances. In addition also the types of instances have changed, in particular we experimented that less powerful instances could be used for Pilot 2. On the other hand, the total storage space increased significantly. This is mainly due to the back-up service, which maintains multiple back-up copies of all the data in the platform (7 copies per day in the last week, a per week in the last month and 1 per month in the last semester). Also the number of I/O operations and the network traffic in Pilot 2 is different. In this case it is difficult to provide a meaningful explanation for such changes. We can say that the usage of the platform is more stable in Pilot 2. Some data is now stable and left unchanged, e.g. background maps. Some data synchronization and upload activities run periodically. There are no major application update or redeployment causing additional data transfers. Given the stability of the deployed Pilot 2, the information reported in Table 4 is probably more reliable than the measurements we had in previous deliverables.

Table 4: InGeoCloudS pilots resource requirements.

Resource	***D3.1.2 Estimate***			***Pilot 2***			
	S	***M***	***L***	***S***	***S (RDS)***	***M***	***L***
Number of Instances	*5*	*5*	*1*	*9*	*1*	*2*	*1*
CPU (hours/month)	*3650*	*3650*	*730*	6570	730	1460	730
Storage (GB)	*452*			2000			
I/O Requests (million req.s/month)	*15*			50			
Network Traffic (GB/month)	*36*			20			

In the remainder of this document, the resources listed in Table 4 for the Pilot 2 deployment will be used to estimate the cost of the various cloud platform providers.

2.1. UPDATE ON EXISTING CLOUD COMPUTING PLATFORMS

2.1.1. EVALUATION CRITERIA

Below we shortly recall the evaluation criteria adopted in D3.1.1-INGC [R1]:

1. ***Functional requirements***: whether or not the platform can support the management and publication of geospatial data.

2. ***Software requirements***: whether or not the platform is able to accommodate the InGeoCLOUDS software requirements (applications, software modules, licensing, development environments and tools, web server, etc.).

3. ***Elasticity model***: whether or not does the platform provides sufficiently large (storage/computation/bandwidth) "facilities" so as to support scalability.

4. ***As-a-Service model***: which of the three could computing service paradigms is provided: IaaS, PaaS or both, and which API of interest are provided.

5. ***Maturity and diffusion levels***: whether or not there is a lively developers community, an ecosystem of useful libraries and software components.

6. ***Migration cost model***: whether or not the platform involves some lock-in effect.

7. ***Economic cost model***: estimate of the cost of the InGeoCLOUDS project.

We did not find the need to add new evaluation criteria, as the above one were sufficient to analyze the current cloud offer.

2.1.2. EVALUATION OF CLOUD SERVICE PROVIDERS

In the following we provide an update w.r.t. the cloud service providers considered in D3.1.2 [R2], with particular interest in the new estimated cost. The cost reported in Table 5 is computed on the basis of the estimated monthly requirements listed in Table 4. Our analysis includes the most representative CSP. First we discuss relevant updates, and finally we summarize the current cloud offerings.

2.1.3. AMAZON EC2

Amazon is still the technological leader. A number of interesting features were added in the last period. We mention a few below:

- Connection Draining in elastic load balancing allows that any instance to complete the current requests before this is terminated and removed from an elastic group (e.g., due to auto-scaling rules or failed health check).

- Cost Explorer in the Amazon AWS console provides a graphical interface that allows analyzing costs in a more detailed way, in particular split-by-tag is provided (recall that amazon resources can be associated with a textual tag). Note that we provided a more advanced interface within InGeoCLOUDS that uses tags and that in addition is able to exploit data providers' usage information.

- New and more powerful instances were made available, e.g., r3 memory optimized ones.

- Reserved instance type modification was allowed, that means that a given configuration of reserved instances can be changed over time according to arising requirements without incurring in any fee or loss.

One of the most important trends we observed is cost reduction. There were 2 significant cost reductions in the last 3 years, the last being effective from April 2014 and resulting in a price cut up to 40%. As an example, 1 small instance cost $490.30 per year in 2009, and now the same instance costs $210.00. As far as we know, the price reduction was not as aggressive as in the last years, probably due to the increasing "pressure" by important competitors such as Microsoft.

Amazon is still the leader, but we expect increasing competition in the near future.

2.1.4. CLOUDSIGMA

CloudSigma continues to improve its offering with a new CloudSigma 2.0 solution, including innovative features such as the private patching to public cloud or more advanced CPU options.

CloudSigma concluded partnership with different companies such as Equinix (in the USA) to improve global presence, or such as FileCatalyst to provide an accelerated file transfer solution.

The company offers standard subscription pricing but also have an on-demand burst pricing mechanism based on the utilization of their cloud. Prices are adjusted every five minutes based on the latest cloud utilization. Customers can use it in order to execute computing jobs according to price tolerances that the user sets.

Billing computing power depends on the sizes of instances with hourly rates based on the actual or virtual hardware reserved for the instance.

As an example, one small instance (comparable with the Amazon small instance) cost $598.41 per year (with 25% off), which is much over than Amazon.

2.1.5. GOGRID

GoGrid now has three data centers, two on either side of the USA and one in Europe in Amsterdam, and packed with lots of Intel hardware, a layer of Xen virtualization and a layer of automation tools for customers. Edgecast is behind the CDN, Salesforce is hooked into support functions, and Equinix provides some data center support. This combination of components seems to put GoGrid right in the middle of the IaaS field.

The company focuses now on BigData cloud services. The Iaas is available with many improvements like SSD servers, Raw disk Cloud servers, dedicated Servers, DC to DC connectivity (Cloud Link) and DC to World (Cloud Bridge).

It's now possible to setup a high Scalability application on GoGrid. Indeed, GoGrid offers Scaling "out", i.e., adding more servers to the infrastructure and scaling "up", i.e., adding resources (like RAM) to an existing cloud server.

Gogrid has considerably reduced its prices. For example, one small instance (comparable with the Amazon small instance) now costs $262,80.

2.1.6. GOOGLE APP ENGINE

Since December 2013, Google has a new IaaS solution named Google Compute Engine, comparable to the Amazon AWS solution. Google Compute Engine allows customers to deploy their applications in multiple zones with the same region (USA, EU, Asia). A zone is an isolated location within a region that ensures isolation for many types of infrastructure, hardware, and software failures.

Note however that Google does not guarantee that data is kept only in location chosen by the customer. So data can be kept in the USA even if customer selects the EU region to deploy its application (which is not the case by AWS for example). Google Compute Engine is compatible with many operating systems and distributions; however Ubuntu is not (yet) qualified by Google to work with Google Compute Engine.

Goggle also offers PaaS solutions such as App Engine, Cloud SQL (mysql), Cloud Datastore (NoSQL), BigQuery (Big Data), Cloud DNS, etc.

Google applies a per-minute billing for the compute resource usage and applies a discount depending on the real usage during the month.

As an example, one small instance (comparable with the Amazon small instance) cost $233.52 per year.

Should Google change its policy on data location (Google does not guarantee data are not replicated worldwide) and if Ubuntu were supported, Google Compute Engine could be a very competitive public cloud provider alternative to Amazon for hosting InGeoCloudS.

2.1.7. MICROSOFT AZURE

The offer by Microsoft is becoming very competitive. On the one hand, the basic Infrastructure-as-a-service offer is now comparable with that of Amazon, also thank to the introduction of Linux-based instances. On the other hand, Microsoft is addressing directly higher level services such as support to Web and Mobile applications, and integration with non-cloud Microsoft solutions such as Microsoft SQL Database, Microsoft Visual Studio and Windows server back-up tools. So this is probably the best option for developers who are willing to stick with the Microsoft environment.

Also, Microsoft as significantly reduced its prices since 2014 in order to be competitive with Amazon. As an example, one small instance (comparable with the Amazon small instance) cost $788.40 per year in 2013 and now its price is $341,64 per year. This makes Microsoft much closer to Amazon. We observed that Microsoft is the only provider able to keep up with the price reduction policy run by Amazon. This will probably make Microsoft a good alternative to Amazon, but it will probably reduce the overall number of cloud providers.

2.1.8. OVH PUBLIC CLOUD

OVH offers two different IaaS solutions: the Public Cloud solution and the Dedicated Cloud solution. The Public Cloud solution is operated by RunAbove, an OVH company whereas OVH operate the Dedicated Cloud solution.

OVH and RunAbove site does not give detailed information on their solutions. However, the Public Cloud solution seems to rely on OpenStack for the compute and for the storage which is a good point in terms of openness and fitness with the market trends. OVH provides different hypervisors (vSphere or vCloud from VMWare and Hyper-V or Azure from Microsoft) to the customer to manage their Dedicated Cloud.

RunAbove claims to offer an extremely performing cloud computing solution, providing one virtual machine per physical host. This is a somehow strange choice that goes against cloud computing behavior that usually promotes the share of the hardware resources among virtual instances.

Under this limitation, one VM per host, the *RunAbove* offers only two types of instance (one optimized for the compute and one optimized for memory) at the same price: $0.24 / hour. So one instance costs $2102.40 / year, which is much more than Amazon costs.

2.1.9. CONSIDERATIONS ABOUT THE CLOUD PROVIDER MARKET IN EUROPE

By some estimates, the global cloud market is set to grow to $200 billion by 2020. Cloud providers are more and more to offers different Cloud solutions in order to take a piece of this huge market.

Big companies such as Microsoft, Goggle, Amazon, IBM invest a lot in their respective infrastructures and offer rather comparable IaaS and PaaS solutions. For example, IBM acquired SoftLayer for $2 billion last year and commits $1.2 billion to expand the SoftLayer cloud footprint. Smallest companies are compelled to innovate and offer pioneering or simple solution since they can hardly compete on the basic price side.

European agencies (as the American agencies) are very concerned with the security of their data. Lot of people in Europe (and in America) are afraid about their data that might not be kept in a data center located in their country. Moreover, the Patriot Act and the NSA Prism program are rightly viewed as scarecrow for the European people. That is why more and more European public and private companies think to move to privately hosted Cloud: a Cloud hosted on their own premises. Different Cloud solutions allow to build a private hosted Cloud: VMware vCloud Suite which is not for free (and even quite expensive), OpenStack which is the leading open-source cloud operating system, CloudStack which is another open-source cloud operating system supported notably by Apache.

2.1.10. SUMMARY

In Table 5 we report on the updates of previously reviewed cloud service providers, and in particular we report their estimated cost on the basis of the requirements listed in Table 4. We noted that the maturity of offers is growing: young comers from the traditional “infrastructure hosting and externalization” markets have now developed complete lines of services. A general comment regards the prices, which dropped significantly in the last year. This trend seems to be likely to continue in the future.

Since we are exploiting an Infrastructure-as-a-Service paradigm, there are not relevant updates w.r.t to the evaluation criteria mentioned above. The cost and the maturity of technical solutions and services are still the most important criteria for choosing a cloud service provider. In this respect, the Amazon AWS is still the most convenient. Therefore, the InGeoCloudS platform will continue to be hosted on the Amazon AWS cloud. Other solutions are becoming more and more interesting, and Microsoft Azure is probably the most relevant competitor.

Table 5: Summary of Cloud Service Providers

Cloud Provider	***Summary of updates***	***Estimated Monthly Cost (VAT included) (€)***
Amazon AWS	Amazon seems to be the technological leader, still introducing new advanced features. The prices have dropped again very significantly starting April 2014.	***469.98***
CloudSigma	CloudSigma offers innovative solutions.	891.00
Flexiant	No significant updates.	1606.54
GoGrid	GoGrid has a complete bigdata cloud solution. They also propose many IaaS offers, like amazon.	614.12
GoogleEngine	The offer is comparable with that of Amazon or Microsoft. However, lake of guarantee on data location disqualifies the solution.	493.58
Joyent	They are building a product being very well specialized for some specific technologies, such as MongoDB and NodeJS.	851.42
Microsoft Azure	The offer is now comparable with that of Amazon, with some specialized deployments for specific technologies such as .NET, NodeJS. Microsoft is the only one able to drop prices as Amazon is doing.	599.95
OVH	OVH offers very simple solution which does not fit with the INGC platform (instance types “too powerful”, i.e. too costly too)	2144.02
Opsource	No significant updates.	1645.81
Rackspace	Rackspace is now a mature solution, and it is starting to reduce its costs significantly. In our previous estimation its cost was ~€1,600.	1159.42

3. CONSIDERATIONS ABOUT MOVING TO A DIFFERENT CLOUD PROVIDER

The fundamental aim of InGeoCloudS project is to demonstrate that a cloud infrastructure can be used by public organizations to provide more efficient, scalable and flexible services for creating, sharing and disseminating spatial environmental data in innovative but also standardized ways. In D3.1.x documents, we reviewed several Cloud Service Providers by taking into account a number of criteria including migration costs, compliance with the InGeoCLOUDS infrastructure, etc. Given the information at hand, the Amazon platform was chosen as the one that best suits the requirement of InGeoCLOUDS and appeared to be also the cheapest one. The facilities for hosting data in an European data center (Ireland) was also appreciated.

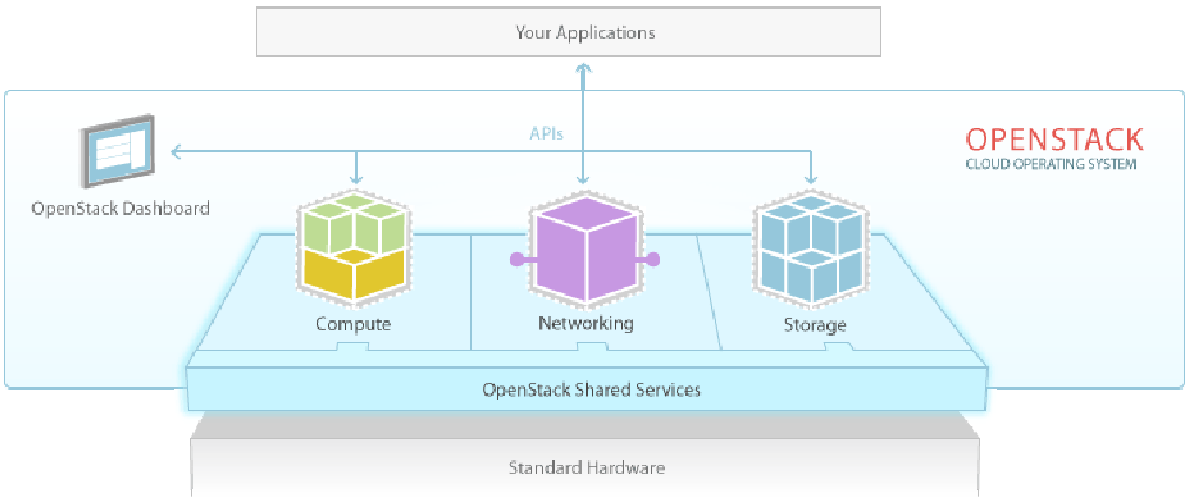
At the same time, the consortium agreed on a strong position of not being bound to a particular CSP. Technical choices and decisions taken during the implementation tasks have been taken in the light of a continuous assessment on the reversibility and level of integration of CSP-specific tools and languages (scripting for example).

This section describes how InGeoCloudS project could potentially migrate to a different CSP, by analyzing the list of actions we need to undertake, the limitations as well the problems that could arise in migrating to a different platform.

3.1. OPENSTACK

OpenStack is an open-source project that can be used to setup and run a cloud computing platform for public and private clouds. Released under the terms of the Apache License, the project is managed by the OpenStack Foundation, a non-profit corporate which promotes development, distribution and adoption of the OpenStack cloud operating system. The foundation has already attracted more than 16,000 individual members from 137 countries and more than 850 different organizations, including NASA, RackSapce, IBM, RedHat, HP, VMWare, etc.

OpenStack acts mainly as an Infrastructure as a Service (IaaS) platform and has a modular architecture consisting of a series of interrelated projects delivering various resources throughout a datacenter. The resources can be managed or provisioned through web dashboard, command-line tools or RESTful API.



The core components of the OpenStack operating system are the compute, networking and storage modules. The compute service (Nova) provides on-demand computing resources, by provisioning and managing large networks of virtual machines. It supports different hypervisors, and is designed to scale horizontally on standard hardware. The networking service (Neutron) provides facilities for managing networks, IP addresses, firewalls, load balancers and VPN. It ensures that the network will not be the bottleneck of the cloud environment. The storage service is split in two distinct modules: the object storage service (Swift) and the block storage service (Cinder). Object storage service (which is equivalent to Amazon S3) provides a highly available, distributed, eventually consistent object storage, which can be used to store lots of data efficiently, safely, and cheaply. The block storage service (Equivalent to Amazon EBS) on the other hand allows persistent block devices to be managed and connected to compute instances, in order to expand the storage or achieve better performance. It persist data independently from the running life of the compute instances, allowing to rapidly restart a compute instance in case of failure.

On top of the above described core components, the OpenStack operating system has several shared services that make it easier to implement and operate a cloud platform, as well as a web based dashboard (Horizon) for providing a graphical interface to manage the cloud resources. The shared services integrate the OpenStack components with each other, providing to the users a unified experience when interacting with the different cloud resources. The Identity service (Keystone) provides a common authentication system across the cloud operating system. The image service (Glance) provides utilities for managing disk and server images to use as templates. The telemetry service (Ceilometer) aggregates usage and performance data across the services deployed in the cloud environment. Finally, the orchestration service (Heat) is a template driven engine for describing and automatizing the deployment of the infrastructure.

OpenStack claims its APIs are mostly compatible with the Amazon APIs, and thus client written for Amazon Web Services need minimal effort when moved to the OpenStack cloud environment. However, this is not fully the case. We tested last year the InGeoCLOUDS API on OpenStack and results were not successful. Some EC2 APIs that are used by the InGeoCLOUDS API are not supported or are bugged on the OpenStack environment.

3.2. JCLOUDS

Jclouds is an open source multi-cloud toolkit that allows provisioning and control of cloud resources from Java. Jclouds is licensed under the Apache License, Version 2.0.

The API allows developers to use both portable abstractions and cloud-specific features. Abstraction allows creating applications that are portable across clouds, while the provider-specific API gives developers full control to use cloud-specific features. Jclouds supports of dozens of cloud providers and cloud software stacks, including Amazon AWS, OpenStack, and vCloud. The APIs avoid dealing with REST-like APIs or Web Services and provide simple interfaces so that the programming model is familiar.

The runtime is portable so it is easy to switch Cloud provider. Jclouds drivers enable to operate in restricted environments like OpenStack or AWS. Best practice is to deploy the Java application with the Jcloud driver for the targeted cloud provider. Mocking complexity and brittleness of remote connections make difficult to write unit tests for cloud endpoints. Jclouds provides developers with Stub connections that simulate a cloud, which simplifies writing unit tests. Jclouds is fully configurable so developers can for example customize it to match the performance needs. Location feature provides location-aware abstractions. For example, you can get ISO-3166 codes to tell which country or province a cloud runs in. To ensure quality of Jclouds, every cloud provider and every cloud software stack is tested with live scenarios before each release. If it doesn't pass, it is out of the release.

The Compute API provides a basic abstraction across Compute APIs such as Amazon EC2 and VMware vCloud. Developers provision the compute resources needed by their infrastructure in any cloud provider. Tasks of managing machines in the cloud are simplified while developers are in control of the entire process: deployment configuration, provisioning and bootstrap.

The BlobStore API allows developers easily storing objects in a wide range of blob store providers such as OpenStack Swift and Amazon S3, regardless of how big the objects to manage are, or how many files are there. The API is dramatically simplified from the providers, yet still offering enough sophistication to perform most work in a portable manner.

The Load Balancer API provides an abstraction to configure the load balancers in any cloud that supports them. Developers just define the load balancer and the nodes that should join it.

Moreover, if cloud functionality is not available in the jclouds abstraction, developers have means to gain access to the provider-specific interface. Specific APIs such as DNS, firewall, storage, configuration management, image management, and much more are easily accessible to the developers.

3.2.1. MIGRATION FROM AMAZON AWS

INGEOCLOUDS API.

As already reported above, one of the InGeoCLOUDS design objectives is to ensure portability – as much as possible - to any cloud platform provider. This is achieved by implementing a specific middleware component, which is designed as the only access point to the facilities of the cloud service provider. The Elastic Compute component thus implements the basic operations regarding the management of compute and storage devices, virtual machines images and any other cloud related operations. These basic operations are used by the other components to achieve more complex tasks, e.g. run a pool of Web servers. In other words, other components of InGeoCLOUDS shall have no awareness of the specific cloud provider; they just take advantage of the services exposed by the Elastic Compute component. In order to port InGeoCLOUDS to a given cloud service provider (e.g. OpenStack), only the implementation of Elastic Compute should be updated. Currently the Elastic Compute module implements only a cloud interface towards the Amazon AWS cloud services.

The services exposed by the *Elastic Compute* module are the followings:

1. facilities for managing computing resources (virtual instances)
2. facilities for managing block storage devices

3. facilities for creating auto-scalable computing layer (auto-scaling and load balancing features)

4. facilities for tagging resources

The four services can be translated to use the OpenStack APIs because the related offering of OpenStack is very similar to the ones of Amazon. In fact regarding the computing resources we could switch from the Amazon EC2 service to the OpenStack Nova service, with a minimal effort. However, OpenStack does not provide a SDK for Java applications. This is a big issue because the InGeoCLOUDS API is written with Java language. Three options have to be considered to port the InGeoCLOUDS API to OpenStack:

1. Trust in EC2 compatibility of the OpenStack to move the InGeoCLOUDS API with minimal development efforts to the OpenStack platform. However, as mentioned previously, compatibility is not obvious. We have to test the compatibility of each EC2 API that is used in the InGeoCLOUDS API, method by method to ensure all the entire InGeoCLOUDS API will run perfectly on OpenStack. If necessary, some modifications should be done to fix some minor issues. Principal risk is to use an EC2 method that is not supported by OpenStack. Another risk is to use an EC2 method supported by one version of OpenStack but bugged on a new version of OpenStack.

2. Implement a version of the Elastic Compute component dedicated to OpenStack, calling the OpenStack API as REST services. As usual in cloud computing world, the OpenStack API is accessible as RESTful services. Many frameworks in Java world allow the developers to call REST services. Calling directly the OpenStack REST services avoid using an intermediate SDK that could be bugged. However, this solution implies big development efforts: calling a REST service involves writing several lines of code while a SDK allows calling a simple method.

3. Implement a cloud-agnostic version of the Elastic Compute component, using a cloud SDK such as **jCloud**s. Note that jClouds is the official Java SDK promoted by Openstack. As mentioned previously, jClouds project aims to provide a real cloud-agnostic API to write Java applications independent of the running cloud platform. This means the development efforts would be done to move the API not only to OpenStack, but also to any other cloud platform supported by jClouds. However, jClouds does not support (yet) all the required APIs: e.g. load balancer API does not support OpenStack. The development efforts should be done to implement the missing features either directly in the InGeoCLOUDS API, or as a contribution to the jClouds project.

INGEOCLOUDS INFRASTRUCTURE.

Of course, as usual in cloud environments, the Elastic Compute module uses virtual machine images being stored on the cloud service provider platform. Thus, one of the first things to do is to recreate those images on the new cloud provider (reproducing software installations, configuration and file dependencies). Regarding the block device, we could switch to use the OpenStack Cinder service, and regarding the elasticity feature we could use the LBaaS service (Load Balancer as a Service) recently introduced in OpenStack. It offers all the interesting features we are currently using in the Amazon cloud environment: the management of an auto-scaling group of instances, a load balancer for sorting requests among the instances belonging to a specific elastic group, the session persistence at the load balancer level as well as the health monitoring of the instances of those elastic group. The server metadata feature can be used to tag every resource, as this feature is massively used inside the IGC platform.

Moving to OpenStack also requires managing the network infrastructure for the InGeoCLOUDS platform. Network infrastructure concerns all the resources necessary to build the network (network, subnets, router, gateways, NAT, DNS, Load Balancers, etc.) and implement the security (firewall, VPN). In AWS, the platform relies on the default network infrastructure provided by EC2. This was useful and minimized implementation efforts. All the cloud providers do not provide a default infrastructure like Amazon does with EC2. This means we might have to create a network infrastructure for the InGeoCLOUDS platform using solutions provided by the cloud platform. We could modify our API to integrate the management of the network infrastructure. However, this is not a good idea because of the specificities of the network solutions provided by each cloud provider.

Clearly, the network solutions provided by a cloud provider are close to the physical infrastructure. The challenge for cloud providers is not to create/stop virtual machines, but it is more how to integrate VMs in a network infrastructure and how to seamlessly connect them, with security, accessibility, and bandwidth efficiency. Each cloud provider also provides specific solutions to help customers to fully control the integration of their applications in the cloud platform. E.g. Amazon provides the VPC solution that allows customers to create their private network and associated resources (route table, VPN, etc.). VPC is a pure Amazon solution that cannot be ported to other cloud platforms like OpenStack (VPC is close to the Amazon physical infrastructure and relies on specific technical solutions that Amazon implemented to manage its data centers). Note however that in order to improve security, high availability and simplify some technical points (e.g. instances network addresses/DNS management) in the InGeoCLOUDS platform, we might use the VPC feature on the existing infrastructure. On an OpenStack implementation, we could use OpenStack Neutron service, which relies on industry best technical solutions (e.g. possibility to create VLAN) and facilitate integration of network devices in the infrastructure.

Another solution could be to create the network infrastructure using the dedicated tools provided by the cloud providers. In order to facilitate the deployment, some cloud providers provide an orchestration service. Amazon provides CloudFormation and Openstack provides the Heat service. The two solutions allow customers to describe their platform with a template that can be executed at any time in order to deploy all the defined resources: network infrastructure (private network, sub networks, routers, Internet gateways, firewalls, load balancers, VPNs) and servers (starting virtual machines, applying loads balancing policy).

Defining an infrastructure using templates is easier, faster and more flexible than using API. Moreover, using the dedicated orchestration service allow to create the most suitable network infrastructure for the InGeoCLOUDS platform taking into account the cloud platform specificities.

We could use the orchestration service to create the network infrastructure of the InGeoCLOUDS platform (private network, subnets/VLAN, router firewall, and still use the API to fully control the IGC services lifecycle (start/stop/monitoring/accounting).

OTHER INGEOCLOUDS COMPONENTS.

Unfortunately, some IGC components had the need to interact with specific services offered by the cloud provider without using the Elastic compute module. We need to pay particular attention to those dependencies because we have to patch them one by one. The services involved are:

1. The backup/restore service, regarding the S3 dependency,
2. The monitoring and accounting services, regarding the Amazon RDS dependency,
3. The elasticDB, regarding the EC2 dependencies (ElasticIP and instance tags),
4. The ElasticMapServer, regarding the S3 dependency (e.g. tiles buffering)
5. The accounting service, regarding the Amazon detailed usage report

For backup and restore purposes, we have chosen to rely on S3. This choice is the best choice on Amazon because backup concerns the data disaster recovery. And the only serious solution on Amazon is to use S3 to store critical data. Fortunately, backup and restore scripts uses S3 utility commands that could be easily replaced. OpenStack offers a similar Object Storage service to S3, and provides command line tools to interact with. So moving the backup/restore scripts should not be a big effort.

The problem with the third point is that, for backup and restore purposes, we have chosen to use a software (S3FS) that mount an Amazon S3 bucket as a local file system (by transparently translating I/O calls to HTTP request). The results is that we are using the benefit of the Object Storage solution offered by Amazon with the ability to use the service as a tradition file system. OpenStack offers a similar Object Storage service, but unfortunately the s3fs software, as the name suggest, supports only the Amazon service. A solution to the problem could be to use CloudFUSE[1] software similar to s3fs) to spoof Object Storage as a normal file system.

[1] See https://github.com/redbo/cloudfuse
and https://lists.launchpad.net/openstack/msg06891.html

The monitoring and accounting services on the other hand have the problem that we have chosen to use the Amazon RDS feature for permanently storing the values of the indicators and of the costs. Of course we are not locked-in with this service, but it was something that helped us to provide an efficient and rapid solution to the problem of permanently storing some data in database without risking losing them when the platform is stopped. A potential solution could be to use the OpenStack Trove service (DataBase as a Service, DBaaS) introduced very recently into the OpenStack operating system. This service provides facilities for managing a single instance database service, thus without guarantying any kind of high availability (but provides methods for backup/restore the database content). Alternatively, we could adopt the solution to internally manage a Mysql database engine. Another problem of the monitoring service is that for collecting some very specific indicators (e.g. the average response time of an elastic group) we use the CloudWatch feature of Amazon. These indicators should be collected at a lower level (e.g. by analyzing the response time of each instance inside an elastic group).

The ElasticDB service uses the ec2-api-tools to implement high availability. The tool is a client interface to the Amazon API for managing the computing resources. We should switch to use the OpenStack command line tools here to solve the dependency.

The ElasticMapServer infrastructure requires a common and shared storage between all map servers to store cartographic tiles. These data are not critical; they can be regenerated at any time. The amazon S3 cloud service suits perfectly, because it's inexpensive, shareable and available. We have chosen to use (like backup and restore service) S3FS software to mount S3 bucket has local drive. The migration of Elastic Mapserver service to a different Cloud provider requires the use of a similar storage service (e.g., Cloudfuse for OpenStack) or implement of a specific solution by using the NFS technologies or GlusterFS.

The Linked Data Management API was designed in such a way that is independent of any cloud service provider as it does not actually exploit any particular API of Amazon. However, the architecture of the Linked Data Management Service comprises a Load Balancer, which is used to distribute the load on the different Linked Data Management Service instances. This Load Balancer is created through Elastic Compute, so once Elastic Compute is modified to run to another cloud provider, the Linked Data Management Service will also be able to be executed in this cloud provider..

Last but not least, the accounting service need for the most part to be redesigned and implemented in a different way because of the dependency with the detailed accounting report Amazon provide us, report that we use in order to know the cost of each service as well as the cost to account to each provider.

OPENSTACK MIGRATION.

In the following table we summarize the mapping between the service used directly be each InGeoCLOUDS module and the related solution we could adopt inside the OpenStack environment. Modules that use the ElasticCompute functionalities will not be mentioned here because the module acts like an interface between the platform and the cloud provider, and so will mask the difference in its implementation. Of course, the ElasticCompute module itself has some dependencies that are summarized below.

IGC Module	Amazon services used	OpenStack mapping

ElasticCompute	Elastic Compute Cloud (EC2) 1. Virtual Instance 2. Elastic Load Balancing 3. Auto Scaling 4. Elastic Block Store (EBS) 5. Resource tagging 6. Amazon Machine Images	Compute (Nova) 1. Virtual Instance 2. Load Balancer as a service (LBaas) 3. Auto Scaling/AWSAutoScaling 4. Block Storage (Cinder) 5. Resource Metadata (Nova)Virtual Machine Images
ElasticFS	1. Simple Storage Service (S3)	1. Object Storage (Swift)
ElasticDB	1. Simple Storage Service (S3) 2. EC2 client tools	1. Object Storage (Swift) 2. OpenStack client tools
ElasticMapServer	1. Simple Storage Service (S3) 2. S3FS tool	1. Object Storage (Swift) 2. CloudFuse Or ad hoc solution
ElasticWebServer	No direct dependencies.	No Changes.
DataImport	No direct dependencies.	No Changes.
DataPublication	No direct dependencies.	No Changes.
LinkedData	No direct dependencies.	No Changes.
Monitoring	1. Relational Database System (RDS) 2. CloudWatch	1. DataBase as a System service (Trove) 2. Ceilometer
Accounting	1. Relational Database System (RDS) 2. Detailed usage report	1. DataBase as a System service (Trove) 2. n.d. (ad hoc solution)

4. CONCLUSIONS

The InGeoCLOUDS's best choice for running on a public cloud infrastructure still is AWS form at least both economical and technical viewpoints. Nevertheless, the competition is getting even harder with big names such as Microsoft, IBM Softlayer, Google or even Rackspace. But the fact that InGeoCloudS is already operational on AWS leads us to prefer that solution yet.

On the other hand, as seen in other context, the fact that AWS is an American company and thus falling under the Patriot Act prerogatives is a worry for new adopters, even if we can guarantee that – a priori- data are kept in an European data centre (in Ireland) and even if we are dealing for now with public open data. These are legal and political issues that endanger the overall objective of the project of becoming a popular and most-used infrastructure for the couple of European public services we address. For the rest of the world, the fact of relying on a US-based company is a priori much less an issue.

Several European actors are now proposing more and more advanced cloud services and somehow more competitive prices (but still far more expensive than AWS for similar ones). Nevertheless, - as with CloudSigma and its incomplete technical documentation and catalogue of services– there are always serious limitations that would make a move to a European public cloud rather expensive. Intermediate results of users' surveys, exploitation/customer prospects and some consortium partners' preferences are pushing the consortium to also contemplate the use of a privately-hosted cloud, in particular to get more controlled traceability for data-residency issues.

As a matter of fact, open source cloud management system have recently gained in maturity, reliability and popularity thus making the scenario of hosting and managing a public cloud on own infrastructure with a professional-level SLA and without licensing fees more realistic. Of course, legacy solutions like VMware's vCloud Suite still offer best-of-the-class capabilities but free and Open Source solutions like Cloudware, OpenStack are both supported by big names of the IT world (HP, Rackspace, RedHat, IBM, Oracle...) and are adopted by many European players. Such platforms might be considered for post-project exploitation scenarios.

**** End of the document ****