DELIVERABLE D3.1.2

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

D3.1.2

Version 1.0

Reference D3.1.2-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 :	2013-06-04
WP contributing to the deliverable :	*WP3*
Availability :	*Public*
Authors :	CNR
Approved by :	InGeoCloudS Steering Committee

Abstract

This document is the second 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. The document also includes feedback and return of experience about the use of cloud technologies. In line with the first issue of this deliverable, we assess the changes in the architecture technical requirements and software components.

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

DOCUMENT CHANGE LOG

Document Issue.	Date	Reasons for change
Version 1-Draft 1	2013-04-01	Creation of the document
Version 1-Draft 2	2013-05-22	Integration of partners' contributions. Delivered for internal review.
Version 1 – FinalDraft	2013-05-31	Steering Committee comments were addressed. Further contributions were integrated.
Version 1-Approved	2012-06-04	Final editorial changes – minor corrections in section2 – approval through 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] D3.2-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
AWS	Amazon Web Services cloud platform
CSP	Cloud Service Provider
GIS	Geographic Information System
SQL	Structured Query Language

Table 1: Acronyms and Definitions

1.2. OBJECTIVES OF THE DOCUMENT

This document is the second 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 D3.1.1[R1], and it also reports on the return of experience resulting by the exploitation of the Amazon Web Services (AWS) cloud infrastructure.

The main objectives of this document are:

- to provide a first and general return of experience discussing the strengths and weakness of chosen the AWS cloud platform in the context of geo-applications experiences so far;
- 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.

A last issue of this document will be delivered at M26 drawing final conclusions on the choices taken by the consortium, the evolution of cloud providers, and discussing the gap between InGeoCloudS applications' requirements and cloud providers experienced by the project consortium.

1.3. OVERVIEW OF THE DOCUMENT

Section 2 reports on the first return of experience on the use of cloud technologies, and in particular on the AWS cloud. Section 3 reports a few updates on the architectural requirements defined in the previous respective deliverable D3.1.1[R1] according to the more recent advancements in the project, and in particular it we shortly describe the software components of the platform not yet considered in D3.1.1. Section 4 provides an update on the market of cloud service providers, with particular focus on the cost factor. Finally Section 5 draws some final conclusions.

2. FIRST RETURN OF EXPERIENCE ABOUT CLOUD TECHNOLOGY FOR INGEOCLOUDS SERVICES

In the following we discuss the return of experience about cloud technologies gained at the current stage of the project, and in particular about the Amazon cloud platform, a.k.a. Amazon Web Services (AWS). For each of the basic services of the InGeoCloudS platform, we discuss strengths, difficulties and lessons learnt about AWS and about the cloud infrastructures in general. We include a description of our experience with specific AWS tools such as the Amazon Console, AWS API and AWS auto-scaling groups.

2.1. ELASTIC DATABASE SERVER

The objective is to propose an enterprise-level geo-database for InGeoCloudS services and applications. The services moved to the cloud in the frame of the InGeoCloudS project use traditional relational database systems for data storage and management. The support of geo-localized attributes, geographical queries and transactions is also a very important requirement among scientific data providers. As such, the PostgreSQL/PostGIS solution has become very much used in the domain of GIS. We also retained requirements for high availability and automated fail-over to a synchronously replicated secondary database in case of a failure. We also wanted to be able to scale out beyond the capacity of a single database deployment for read-heavy database workloads.

Amazon Relational Database Service (RDS) solution gives access to the capabilities of MySQL, Oracle or Microsoft SQL Server database engines and also focuses its technical and commercial offer on I/O intensive transactional (OLTP) database workloads. This is not in line with the core needs of InGeoCloudS based on the Postgres/Postgis geospatial database. Thus, InGeoCloudS Elastic Database Server includes:

- a PostgreSQL/PostGIS engine known by most developers and answering geo-data handling and requesting;
- a master/slave configuration with replication mechanisms for ensuring high-availability thanks to automated fail-over mechanisms;
- a load-balancer (PgPool-II[1]) in front-end.

The instantiation and deployment of this architecture on Amazon instances went smoothly. The main issues encountered are related to network management for correctly addressing instances (management of virtual IP addresses for communication in the DB cluster). Usage of Elastic IP Amazon services (fixing IP address of the PgPool -based load balancer) is a satisfactory design solution. Again, the management of this IP and of the deployed components on EC2 instances is relatively easy thanks to Amazon tools and APIs.

In terms of performance and reliability, we did not encounter any service disruption so far.

[1] http://www.pgpool.net/

2.2. ELASTIC FILE SERVER

Amazon AWS does not provide a "traditional" distributed file system. The paradigm for accessing a reliable and scalable storage is based on the key-value metaphor[2]: every object (*value*) has a name (*key*), which is used to store, update, or remove a piece of data. This paradigm, sometimes referred to as *NoSql*, allows easily designing transparent replication and scheduling mechanisms to achieve reliability, scalability and efficiency. Unfortunately, several components of the InGeoCloudS platform, such as the web server and the map server, require a standard file access protocol (e.g., NFS) and they are not designed to deal with a key-value storage. This makes it impossible to directly take advantage of the full potential of a cloud platform.

We investigated a few tools aimed at bridging this gap. Solutions like S3FS (https://code.google.com/p/s3fs/), S3QL (https://code.google.com/p/s3ql/), and S3BACKER (https://code.google.com/p/s3backer/), provide a standard file-based access interface where the application interacts with a virtual file system, which is implemented on top of the Amazon S3 key-value storage service. Unfortunately, all of these tools do not support simultaneous access by different users from different virtual machines, as it is the case of InGeoCloudS, so none of them could be adopted.

By Exploiting an Infrastructure-as-a-Service model, we deployed GlusterFS[3] on a pool of servers. GlusterFS project is sponsored by Red Hat, which uses it in its line of Red Hat Storage products. This is a more traditional solution that does not use the advanced services typically provided by a cloud platform. Nevertheless, this solution has two advantages: first, we reduced the lock-in effect in adopting a specific service of a specific cloud platform; second, we can easily support every InGeoCloudS component and application via an NFS-like protocol.

The current experience of GlusterFS installation within the InGeoCloudS platform is positive. GlusterFS allows easily for tuning the scalability and redundancy of the GlusterFS file system, providing very good performance. In fact, we measured experimentally a throughput of 730MB/s with 8 servers. The other components of the platform can seamlessly operate with the GlusterFS file system, and data application providers had no issues in using it.

We will continue monitoring cloud providers' offer by looking for new "adaptors" able to offer a file-based distributed and multi-user access to cloud key-value storage, or for new releases of the core software tools adopted by InGeoCloudS (e.g., map server) experimenting native cloud-storage services.

[2] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In OSDI, pages 205–218, 2006.
[3] http://www.gluster.org/

2.3. ELASTIC MAP SERVER

2.3.1. INSTALLATION AND CONFIGURATION OF THE MAP SERVER COMPONENT

Amazon WS does not provide any technical solution to publish and disseminate geospatial contents: no spatial database, no web mapping server or other spatial data API is available. After having analysed the technical solutions currently adopted by data providers in their in-premises infrastructure (e.g., MapServer, GeoServer, EsriServer), we decided to deploy an opensource solution to manage the geospatial dataset and the publication of the OGC services: MapServer and PostGIS extension to PostgreSQL database (see 2.1 for the latter).

In order to optimize the performance of the MapServer + Apache solution and to be able to satisfy all user requirements as described in the D2.1 annex II (i.e., KML, ArcGrid, etc.) we chose to install the solution with a direct compilation of all components in a dedicated EC2 server based on the Ubuntu Linux distribution. Based on the first results evidenced, there were no considerable issues concerning the installation and configuration of the map component in the Amazon AWS. However, some limitations regarding the Linux distribution did not allow installing all functionalities provided by MapServer, in particular for the management of specific file formats like KML. On the web server side, PhpMapScript[4] and other specific spatial libraries, like PhpOgr[5], were also installed from source code. These PHP extensions provide some interfaces to the InGeoCloudS web portal for more efficient manipulation of mapfiles (MapServer's configuration files) and other spatial data. We chose to compile all these spatial libraries and programs so as to be 100% tailored to the EC2 hardware, Linux operating system and InGeoCloudS specific software.

Besides, the map API management was deployed in the global context of the API infrastructure defined in the InGeoCloudS platform without major difficulties. For development and deployment, temporary relaxation of some security rules provisioned for the production platform had to be carried out by the IT security manager. This could be handled without particular problems using AWS tools.

2.3.2. THE MAP SERVICE IN THE CLOUD

The server (and the "template" server defined as a AMI image) dedicated to the publication and dissemination of geo-data information is optimised to guarantee good performance and be very easily deployed in the cloud infrastructure. The AWS infrastructure does not generate specific constraints regarding the installation, configuration and maintenance of the map component.

The map component is based on HTTP and stateless exchanges to provide data and images (OGC principles). As a result, the scalability of this component will be provided through the replication of the server (using the "template" server with on demand management) and the use of a load balancer on top of the map service. The experimentation of scalability will be carried out during the next stage of the project. We will in particular test the capacity of the cloud infrastructure to fulfil the INSPIRE requirements for availability, performance indicators (See D2.1).

[4] http://www.maptools.org/php_mapscript/
[5] http://dl.maptools.org/dl/php_ogr/php_ogr_documentation.html

2.4. DATA INTEGRATION AND LINKING

2.4.1. INSTALLATION AND MANAGEMENT

Over the past months we worked with Amazon Web Services and especially with Amazon EC2 resources in order to launch the instances that host our Linked Data infrastructure. Such instances are used to host a large amount of Linked Data (~1B triples), which have to be accessed via SPARQL (or SPARUL) queries providing satisfactory response execution times.

We had to choose carefully the technical specifications of Amazon EC2 instances, which will host Virtuoso[6], the selected triple store, in order to cover the above preconditions for the Linked Data management for a reasonable cost. We decided to select the "standard large" instance which has sufficient storage space (~750GB), memory (~7.5GB) and computing power (4 EC2 Compute Units) to manage our Linked Data with respect to its cost. Amazon offers a web interface console, which allowed us to create our instances and choose all the appropriate parameters described above in a very user-friendly way. Moreover, the OS we used for our instances was Ubuntu 12.04 LTS which is fully compatible with Virtuoso. As a result, there were no considerable problems in the process of installing and configuring Virtuoso to fully exploit the capabilities of the created instances.

2.4.2. THE LINKED OPEN DATA MANAGEMENT API

Apart from Virtuoso, each created Amazon instance exploits an instance of the Linked Data Management API so as to enable advanced management functionality in the form of published web services, which is not available from the Virtuoso SPARQL end-point. To enable such exploitation the current version of Tomcat (Tomcat 7) was installed and the war file of the Linked Data Management API was deployed on it. As there was a small issue regarding the access rights (to a specific directory) for writing the RDF files to be exported, the Amazon EC2 instance was modified by granting such right to tomcat. Obviously, in order to enable the proper accessing of the API and connecting the API with Virtuoso, the respective instance ports had to be opened (i.e., 8080 for tomcat and 1111 for API-to-Virtuoso connection). Since these ports were open, it was quite easy to update the API when a new version was available by exploiting the Web interface of Tomcat.

[6] http://virtuoso.openlinksw.com/

2.4.3. THE LOAD BALANCING AND THE AMAZON AUTO-SCALE GROUP

Having selected the instance type, we had to consider that in many cases, the instance would receive many query requests (even simultaneously) from the Linked Data users. As a result, a single instance would be inadequate to serve and respond to such workload with the appropriate performance. For that purpose, we created a group of instances associated with an Elastic Load Balancer. Every instance of the group is a replicate of the initial instance, which was configured with Virtuoso, the Linked Data Management API and the corresponding already loaded datasets. The Elastic Load Balancing process automatically distributes incoming application traffic (i.e., Linked Data access) across the multiple Amazon EC2 instances of the group. However, due to the fact that the work load will not always be the same, we had to enable the group auto scale up or down with respect to the work load of the instances. In other words, on the one hand, when the number of incoming requests increases, the group scales up by launching new instances to serve them; on the other hand, when the number of requests decreases the group scales down by terminating the extra instances. All the above steps were facilitated using Amazon APIs and the management console in a straightforward way and without facing any significant problems.

2.4.4. FUTURE ACTIONS AND EXPECTATIONS

We are currently performing stress tests in order to assess the load balancing capabilities of the Amazon Cloud as well as assess the query performance under realistic circumstances. It may well happen the case that additional parameters/metrics must be considered for load balancing. We expect that such parameters/metrics are not only available but can also be used in the Amazon's load balancing mechanism. The test results might also lead to migrating to an Amazon instance of a different type in order to appropriately handle the expected workload. We are also working on a geo-spatial extension of Virtuoso, which may require the use of the GIS-enabled database (PostgreSQL/PostGIS). We expect that such an extension will function properly in the Amazon Cloud and no additional tools are really required. Finally, we are working on a more automatic data set integration procedure, which maps and synchronizes relational descriptions of data sets to RDF-based ones. Similarly to the previous case, by assuming that such data sets exhibit a geo-spatial aspect, they will be imported in the GIS-enabled database by the data providers. Thus, we either should find mechanisms that either update the native database of Virtuoso with the relational data-set descriptions in the GIS-enabled database or exploit a commercial Virtuoso version that can immediately synchronize the relational and RDF data without requiring the relational ones to be imported to Virtuoso. The latter solution may also function even when data providers do not import their relational data sets to the GIS-enabled database. We expect that any of the above two solutions will properly operate in the Amazon Cloud without requiring any special tools or arrangements (apart from the opening of some ports).

2.5. GEO-COMPUTATIONAL SERVICES

Some advanced applications, that involve custom computations, require specific software in addition to the built-in services of the InGeoCloudS platform. As an example, consider the shake-maps use case, where specific software is required for the generation of shake-maps. For these applications, an InGeoCloudS "basic" instance was given to the data providers of the consortium with the possibility of installing and configuring any custom software. The distinctive feature of such basic instance is that it transparently provides access to the rest of InGeoCloudS services, thus supporting the effective development of new applications. In particular, it provides access to a private *Workspace*, which includes:

- a private storage space (residing in Elastic File System) accessible through a standard file access protocol (NFS);
- a private database (residing in Elastic Database System) accessible through standard API methods;
- a web-server accessible storage space for directly publishing data and application through the Elastic Web Server.

During the development / testing phase of an application deployment, application providers can have direct access to the EC virtual instance through the SSH port (22), thus having the option for console access and direct file transfer to the virtual instance. Once the "development" instance (i.e., his virtual image) is ready to be integrated in the "production" platform, and a most strict security policy is applied by disabling all external (not InGeoCloudS) access to the instance.

The first application deployed using this development flow was the Shake-maps use case. The instance was easily configured with the prerequisites and the application was deployed without any modifications. The application relies on the Elastic File System and Elastic Web Services provided by the InGeoCloudS platform to read/write and publish data respectively. Moreover, the InGeoCloudS API is utilized in order to upload new data or to trigger a new calculation. In order to decide on the resource characteristics of the EC virtual instance to be used for the Shake-maps use case, a series of performance tests were carried out. Since Shake-maps, in its current development stage, is not computationally intensive nor memory demanding the compute instance finally used was a "small" virtual machine with 1 CPU and 2GB of memory.

From the experience gained so far, the Elastic Compute instance integrates smoothly with the rest InGeoCloudS platform providing a stable Shake-maps computation service.

As this service evolves during the deployment of the Pilot 2 InGeoCloudS platform, the focus will be in implementing techniques for auto-scaling as computation becomes more demanding and for optimizing resource utilization, by consuming resources only as needed.

2.6. AMAZON TOOLS FOR DEVELOPING, DEPLOYING AND MONITORING APPLICATIONS

After one year of experience with the Amazon AWS cloud, we are able to provide some comments about the strengths and difficulties encountered in developing the InGeoCloudS platform, and thus in adapting existing geo-applications running on data providers' IT infrastructure. In this paragraph we consider practical administrative endeavours in most current usages of the cloud infrastructure management:

- Security Issues
- Monitoring resources usages
- Instances Management (stop / pause/ start /clone, AMIs…)
- Amazon EC2's API usage.

2.6.1. SECURITY ISSUES

While we were implementing the first version of the InGeoCloudS platform, we had to use EC2 instances for test and development. Early in the development phase, we changed the authorization/authentication data in all used instances in order not to be easily infected by malicious software. Although this is standard practice for any platform deployment, doing it on Amazon's Cloud requires particular attention since intrinsic openness to the Internet must be taken into account for each action. In a typical project, it is usual to consider security early but to have it fully implemented towards the end of the project when the system goes in production for example.

Amazon cannot ensure EC2 instances are secured; user agreement states that this is the EC2 customer's responsibility to ensure their running EC2 instances and applications are protected, e.g. against intrusion using technical account credentials (Linux, Tomcat, PostgreSQL, etc.). We identified security flaw potentialities and shield the EC2 instances from further compromise attacks.

Every developer is required to follow at least main security rules:

- Secure a running instance against Internet attack. This can be done using Amazon services like Security Groups (A Security Group allows defining the opened ports for an EC2 instance). Developers shall consider an EC2 instance as their personal computer: it must be secured with a firewall.
- Each piece of software installed in an EC2 instance must be secured. Popular software has vulnerabilities that are well known by the hacker's community. A default configuration or some features usually allows infecting the instance. Developers shall configure software to avoid well-known attacks and apply security updates as soon as they become available.
- Ensure the use of strong passwords that cannot be easily hacked.

2.6.2. MONITORING RESOURCE USAGE AND COSTS

An important point, both for cost and effectiveness, for the InGeoCloudS platform is the monitoring of resource usage.

On the one hand, it is very important to have a clear view of the system as a whole, and to identify the under and over-loaded instances of the platform. This is required to correctly plan the resources allocated for each service of the platform. Amazon provides some tools for per-instance monitoring. There exist several external tools, such as CloudVertical, Sensu, and Ylastic, etc., that provide a bird's eye view of a given cloud deployment and might have a non-trivial cost. Our choice is to develop a minimal ad-hoc solution satisfying the basic needs of the InGeoCloudS platform: status of each running service, number of instances allocated to each running service, and some aggregated statistics. Note that a number of statistics provided by Amazon AWS and they are already exploited apply rules of auto-scalability. Within the project, we planned to expose some of such statistics provided by Amazon AWS.

On the other hand, several applications are exploiting the services embedded in the platform, such as the GIS-enabled database, but it is very difficult to assess the share of usage for each of those applications. In particular, it is difficult to understand the impact of each use-case on the overall cost of the running platform. This problem requires an application level solution. It is useful to have both (a) some kind of distributed logging mechanisms to record each application activity and status over time, and (b) some correlation tool to match the application activity with the actual resource usage over time. Such tools are going to be integrated in Pilot 2.

2.6.3. AMAZON'S API USAGE

One of the main strengths of the Amazon AWS cloud dwells is its API. The API functions are very well documented and they are quite easy to use both through REST services and through provided shell tools. They are very stable and so far we did not experience any modification on API definition. Finally, Amazon AWS API can be considered the de-facto standard in cloud infrastructure API and it is very often directly supported via EC2 compatible interfaces by several other frameworks, such as OpenStack.[7] Regarding cloud computing standards, it is worth discussing Open Cloud Computing Interface (OCCI).[8] This is a proposed standard protocol defining a protocol for infrastructure-as-a-service cloud providers. This standard is receiving considerable attention, and it is partially implemented in OpenStack, and it can be used to manage Amazon AWS through a ruby library named rOCCI.[9] Still, the current implementations of this standard are not fully mature, and there is no public cloud provider (e.g., RackSpace, Microsoft) supporting them, but we will continue monitoring their development through the remainder of the project.

This ease of use of the Amazon API allowed us to automate the (simultaneous) deployment of configurable InGeoCloudS platforms, with the following advantages:

- Every developer was able to deploy a completely independent InGeoCloudS platform so as to test his own code in the true cloud environment.
- The running platform can be configured so as to have different capacity, i.e., number or type of machines, according to the type of experiments to be conducted.
- This process was transparent to geo-application developers since they were able to deploy a new platform without interacting directly with the Amazon API.

[7] http://www.openstack.org/
[8] http://occi-wg.org/
[9] http://occi-wg.org/2012/05/22/rocci-occi/

- The project converged to the common double deployment, where one deployment is considered as the production environment, and the other is shared for development and testing.

3. UPDATE ON ARCHITECTURAL REQUIREMENTS

We reflect on the same chapter in D3.1.1 ([R1]) and assess whether new technical components introduced in our architecture imply new requirements with regards to the choice of Cloud Service Provider. There are only a few minor updates on requirements:

- ***PgPool-II v3.2***[10] (upgrade of the version used in Pilot1) is now used for integration in Pilot2 as load balancer in ElasticDB component. This version supports a watchdog mechanism for ensuring high-availability between a master PgPool and a spare one. This is the last version of the framework and it is not provided off-the-shelf from any CSP.
- ***Sitools2***[11] allows connecting at different data sources and to expose their contents in different ways: a REST API and/or graphical user interfaces based on AJAX. With its opened architecture, SITools2 is also a framework that permits developers to extend the server API, simply integrating their own applications as plug-ins. It serves as a simple portal framework in Pilot1 but its integration framework facilities are being used for Pilot2. No particular requirement is put on CSP solutions from the usage of this framework.
- ***GlusterFS***[12] is an open source, distributed file system capable of scaling to large volumes of data and clients. It clusters together storage building blocks over the network, aggregating disk and memory resources and managing data in a single global namespace. It is suitable for any Infrastructure-As-A-Service platform.

[10] http://www.pgpool.net/
[11] http://sitools2.sourceforge.net/
[12] http://www.gluster.org/

4. COST MODEL REVISION

The cost model described in deliverable D3.1.1-INGC [R1] was devised on the basis of the application requirements collected from the data and service providers from the project consortium. Below, we revise the cost model on the basis of the experience gained after the deployment of first Pilot on the Amazon AWS platform.

We recall the InGeoCLOUDS architecture deployment and report on the resources used. As described in D3.2-INGC [R2], 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 four instances to support the distributed file system functionality.
- The ***Elastic Map Server Layer***: we are using just one instance exposing map service functionalities.
- The ***Elastic Linked Data Storage Layer***: also in this case we are 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 are no resources allocated.
- 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.

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 Names	*m1.small*	*m1.medium*	*m1.large*
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 Elastic Database Server Layer, which manages most of the data of the platform, and for the Map Server Layer. The less powerful instance was used for every other service.

The total amount of running instances required by the platform is thus 11 (eleven). This is not very different from our initial estimate, where we expected three CPUs for each of the six use cases for a total of 18 running instances, compared to the eleven actually used on the Amazon AWS cloud. We can probably explain this by some partial economy of scale due to the sharing of resources and services across the data providers. We use these numbers to provide an approximate estimate of the developing and testing overhead. Amazon reported a total of 16,483 CPU hours consumed in April, against the 7,920 CPU hours corresponding to the eleven above-defined instances. By computing the ratio between the two, we can deduce that the total incurred cost (platform + developing + testing) is **2.1 times** the cost of the defined platform. We use this *development cost factor* to estimate the actual cost of the platform and the resource consumed by it where a direct monitoring is not possible, e.g., millions of I/O requests.

In Table 3 we report the costs of the Amazon AWS infrastructure relative to the first quarter of 2013. Within this timeframe, all of the InGeoCloudS services were already deployed and used by other project partners to publish their data and to run their services. The cost varies significantly over time. This is due to the continuous improvements in the platform architecture, and to the several tests (including some stress tests) that have been conducted on the various components. The last two months exhibit a more consistent behaviour. This last period probably reflects better a full and continuous usage of the stable operating platform, and we will use only these two months in the following considerations.

The total cost observed is larger than the cost of € 624.58 which was estimated in deliverable D3.1.1-INGC[R1]. However, the costs reported in the third column of Table 3 encompass both the costs of the operating platform and of the development and testing activities. The ratio between total cost and platform cost discussed above suggests that the cost of the platform itself is about half. Indeed, we worked on two independent platforms: the first is the exposed public one (the first Pilot), while the second is used for developing and testing new functionalities. Additional InGeoCloudS platforms were deployed for short periods of time according to specific developing needs. In conclusion, we can say that the average monthly cost of € *1696.75* resulting from activities conducted in the months of March and April corresponds to the cost of *two* operating deployments of InGeoCLOUDS.

We thus report in Table 3 both the estimated costs of the platform and of the developing activities according to the *development cost factor* defined 2.1. The resulting platform cost is thus *€* 826.38, which is not far from our early estimate. In the following we try to design an updated cost model to guarantee more accurate and well-grounded cost estimates.

Table 3: Amazon AWS Costs, estimated platform and development costs.

Month	***D3.1.1 Estimated Cost (€)***	***Total Actual Cost (€)***	***Est. Platform Cost[13] (€)***	***Est. Development And Testing Cost[13] (€)***
January	*624.58*	**994.47**	--	--
February	*624.58*	**782.03**	--	--
March	*624.58*	**1727.91**	*859.65*	*868.27*
April	*624.58*	**1665.58**	*793.13*	*872.45*
Last 2 months Average	***624.58***	***1696.75***	***826.39***	***870.36***

Our goal is to estimate the cost of the InGeoCloudS platform excluding the development and testing costs, which are going to decrease over time. Rather than estimating the platform cost on the basis of the typical and high peak requirements as we did in D3.1.1-INGC [R1], 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;
- finally, we approximate the other requirements, such as storage space, by dividing the actual amount billed by Amazon by the *development cost factor*.

The resulting estimated platform-only requirements listed in Table 4 are used to update the estimated cost of the platform for each cloud service provider under consideration.

Table 4: Monthly estimated platform requirements and Amazon resources actually used.

Resource	***Est. Platform-only***			***Amazon Report***		
	S	***M***	***L***	***S***	***M***	***L***
Number of Instances	*5*	*5*	*1*	*11.30*	*5.83*	*1.32*
CPU (hours/month)	*3650*	*3650*	*730*	8252	4265	966

[13] The costs are estimated on the basis of the Amazon AWS billed costs (column 3) and assuming a *development cost factor* (Total cost/Platform cost) of 2.1.

Storage (GB)	*452*	950
I/O Requests (million req.s/month)	*15*	31
Network Traffic (GB/month)	*36*	76

4.1. UPDATE ON EXISTING CLOUD COMPUTING PLATFORMS

4.1.1. 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.

4.1.2. EVALUATION OF CLOUD SERVICE PROVIDERS

In the following we provide an update w.r.t. the cloud service providers considered in D3.1.1 [R1], with particular interest in the new estimated cost. The cost reported in Table 7 is computed on the basis of the estimated monthly requirements listed in Table 4. We cannot provide here a comprehensive list of providers nor pretend the presented view is comprehensive: the market is highly dynamic and moving on a daily basis. Anyway, our analysis includes the most representative CSP. First we discuss relevant updates, and finally we summarize the current cloud offerings.

4.1.2.1. SIGMACLOUD

CloudSigma has four main products that are proposed independently from each other: CPU, RAM, data storage and bandwidth. Unlike many other IaaS providers, CloudSigma does not bundle these products together but allows its customers to finely tune exactly the combination of resources they require. As such billing is supposed to be more convenient and helpful for customers in order to manage their infrastructure.

4.1.2.2. GOGRID

GoGrid cloud infrastructure is always a pure IAAS solution without any major change in the service. From now on, a data center based in Europe (Amsterdam) is fully operational and could be used for European project and European citizen target in particular.

With regard to the initial analysis, the migration model has now been better defined by the company and some external solutions (in particular Racemi with http://www.racemi.com/cloud-path/) are now available to manage the migration of the servers in another cloud infrastructure.

However, the elasticity of the cloud is not natively proposed by GoGrid (some manual solutions are now available as described by the company, http://blog.gogrid.com/2013/04/23/how-to-create-an-auto-scaling-web-application-on-gogrid-part-1-theory/).

4.1.2.3. GOOGLE APP ENGINE (NOW GOOGLE COMPUTE ENGINE)

Differences in price between Google's newly available Google Compute Engine service and Amazon Web Services are minimal, especially for US hosted machines. The difference is higher in favour of Amazon for European-based deployments and even OVH proposes lower brute prices. Google public IaaS offering still comes in at a higher price than Amazon's cloud.

Table 5: Summary table of Amazon's comparable standard GoogleEngine instances.

GoogleEngine Instance Type	RAM	vCores	DiskSpace (non persistent)	Hour-based price (€ VAT excluded) – hosted in Europe
n1-standard-1-d	3.75 GB	1	420 GB	0.145
n1-standard-2-d	7.5 GB	2	870 GB	0.29
n1-standard-4-d	15 GB	4	1.77 TB	0.580

4.1.2.4. OVH PUBLIC CLOUD

Since last year OVH dramatically extended its offer and can now be considered as a plausible alternative to American giants (this is at least what they are targeting at). In particular documentation, user support and APIs (e.g., http://www.ovh.com/fr/cloud/2012/api/) have been enhanced significantly (e.g., instance images and templates management, SSH keys management…). A major upgrade is also expected in July2013. Moreover new data centers in Strasbourg (but also in Canada) enrich original Roubaix's one.

Basic prices for OVH instances (granularity is "up-and-running or not?" and does not consider CPU usage) are the following[14]:

Table 6: Summary table of Amazon's comparable standard OVH instances.

Instance Type	RAM	CPU	vCores	DiskSpace (non persistent)	Hour-based price (€ VAT excluded)
XS	0.6 GB	0.8 GHz	1	5 GB	0.01
S	1.75 GB	1 GHz	1	150 GB	0.05
L	7 GB	4 GHz	2	600 GB	0.24
XL	15 GB	8 GHz	4	1.2 TB	0.48

No traffic limits, persistence of disks is billed 0.0005 € per GB / hour. I/O flows on persistent disks are included in this price. Although these prices are much more competitive than last year, they remain far from what the market leaders (e.g. Amazon) can propose. The "fully-European" sell argumentation is put on the front by OVH.

4.1.2.5. SUMMARY

In Table 7 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 (e.g. OVH). 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 and Infrastructure-as-a-Service paradigm, there are not relevant updates w.r.t to the evaluation criteria mentioned above. The cost and the maturity 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.

[14] May2013 status

Table 7: Summary of Cloud Service Providers

Cloud Provider	***Summary of updates***	***Estimated Monthly Cost (VAT included) (€)***
Amazon AWS	There are no significant updates regarding services of interest to InGeoCloudS. The costs of compute services have been significantly reduced from 5% to 30% depending on the instance type.	***899.97***
Atlantic.NET	The platform seems not yet mature in terms of API for large deployment: for instance, server scale up is provided via written requests. It seems more targeting users needing a single server deployed on the cloud. Some services needed by the platform seem not to be available, e.g. load balancing. We exclude this provider from further consideration.	***--***
Flexiant	Important features, such as load balancing, have not been made available and the cost still seems quite high.	***1 577.11***
GoGrid	No significant updates.	***1 550.00***
GoogleEngine	Europe-hosted machines are in particular more expensive than Amazon's one. The price war just starts and it is expected that both provider prices will further decrease in following months. Incoming traffic is not charged by Google.	***2 090.00***
Joyent	Developer support has improved with the introduction of a set of API. Still, only very basic services are provided, and advanced ones are sold separately. They suggest to use ZFS as distributed file system.	***1 720.49***
Microsoft Azure	It now supports also linux instances. Interestingly, it provides a cloud deployment of Microsoft SQL Server but not supporting GIS extensions and full ArcGIS compatibility. The price is still an issue.	***1 219.46***
OVH	We could notice significant improvements in the documentation and in the completeness of APIs that have been release during 2012. New version with major upgrades is yet expected in July2013. New data centres are available.	***1 105.00***
Opsource	No significant updates.	***1 362.16***

Rackspace	Rackspace is now a mature solution and it is considered among the best cloud service providers.	***1 632.62***
SigmaCloud	This Swiss CSP is also becoming a reference in Europe and proposes very aggressive prices. The API (last release v2.0 dated May 22nd, 2013) is now quite rich and complete. Latest version of the Web console is also very attractive in terms of ergonomy and set of functions.	***1 473.00***

5. CONCLUSIONS

In this document we reported on the first return of experience in using the Amazon AWS cloud, and then we updated on the status of the cloud service providers market. The outcome of this analysis is twofold.

First, we found that no significant change in the requirements was needed w.r.t. deliverable D3.1.1-INGC, meaning that the deployment of geo-services to the Amazon cloud went smoothly. In particular we also reported on the first return of experience after a one-year progress of the project. This feedback will contribute to the guidelines and blueprints that the project will output at the end of the project.

Second, we reviewed again the cloud service provider offer. We observed a general decrease in the costs, and an increased competitiveness. The choice of the Amazon AWS is confirmed to have, currently, the lower cost and the greater maturity.