Requirements specification for the ENTICE environment 2

29/09/2016

dEcentralized repositories for traNsparent and efficienT vIrtual machine opErations

Requirements specification for the ENTICE environment 2

Responsible author(s): **Jonathan Becedas**

Co-author(s): Jose Julio Ramos, David Pérez, Rubén Pérez, Gabor Kecskemeti, Attila Kertesz, Attila Marosi, Zsolt Nemeth, Thomas Fahringer, Radu Prodan, Simon Osttermann, Ennio Torre, Vlado Stankovski, Salman Taherizadeh, Jernej Trnkoczy, Carlos Rubia, Jorge Pérez, Ignacio Campos, Craig Sheridan, Darren Whigham.

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 644179.

Revision history

Administration and summary		
Project acronym: ENTICE		
Document identifier:	ENTICE D2.2	
Leading partner: Deimos		
Report version: 1.0		
Report preparation date: 29.09.2016		
Classification: PP (Restricted to other programme participants)		
Nature: Other **(**Internal Document)		
Author(s) and contributors: Jonathan Becedas et. al		
Status:	-	Plan
	-	Draft
	-	Working
	X	Final
	-	Submitted
	-	Approved

The ENTICE Consortium has addressed all comments received, making changes as necessary. Changes to this document are detailed in the change log table below.

Date	Edited by	Status	Changes made
19/09/2016	Jonathan Becedas	Working	General Review
26/09/2016	Carlos Rubia	Working	Update of CSP pilot description
28/09/2016	Jonathan Becedas	Working	Review of the CSP pilot update
29/09/2016	Nishant Saurabh	Working	Official review of the document
29/09/2016	Polona Štefanič	Working	Official review of the document
30/09/2016	Jonathan Becedas	Working	Integration of the official review

Notice that other documents may supersede this document. A list of latest *public* ENTICE deliverables can be found at the ENTICE Web page at http://www.entice-project.eu/.

Copyright

Citation

Jose Julio Ramos, David Pérez, Rubén Pérez, Gabor Kecskemeti, Attila Kertesz, Attila Marosi, Zsolt Nemeth, Thomas Fahringer, Radu Prodan, Simon Osttermann, Ennio Torre, Vlado Stankovski, Salman Taherizadeh, Jernej Trnkoczy, Carlos Rubia, Jorge Pérez, Ignacio Campos,

Craig Sheridan, Darren Whigham. (2016). Requirements Specification for ENTICE Environment. ENTICE Consortium, c/o Elecnor Deimos, http://www.entice-project.eu.

Acknowledgements

The work presented in this document has been conducted in the context of the EU Horizon 2020. ENTICE is a 36-month project that started on February 1st, 2015 and is funded by the European Commission.

The partners in the project are UNIVERSITAET INNSBRUCK (UIBK), MAGYAR TUDOMANYOS AKADEMIA SZAMITASTECHNIKAI ES AUTOMATIZALASI KUTATOINTEZET (SZTAKI), UNIVERZA V LJUBLJANI (UL), FlexiOps (FLEX), WELLNESS TELECOM SL (WTELECOM) and Deimos CASTILLA LA MANCHA SL (Elecnor Deimos Satellite Systems). The content of this document is the result of extensive discussions within the ENTICE © Consortium as a whole.

More information

Public ENTICE reports and other information pertaining to the project are available through ENTICE public Web site under http://www.entice-project.eu.

List of figures

List of tables

1 Introduction

This deliverable specifies the requirements for the ENTICE environment. For that purpose, an analysis of the three ENTICE's Pilot Use Cases in relation with the ENTICE environment has been included. From that analysis a list of requirements has been generated for each of the pilots. Finally, the state of the art of the technologies involved to develop the ENTICE environment has been updated.

In this version of the document there are some changes from the previous version: Deliverable D2.1 Requirements specification for the ENTICE environment 1. These changes only affect to the CSP pilot, described in Section 4. The update with respect to the previous version is the following:

- The architecture of the Unified communications use case and its scenarios have been updated.
- The architecture of the Alfresco use case and its scenarios have been updated.
- The architecture of the Redmine use case and its scenarios have been updated.
- The metrics of the CSP pilot have been updated.

The user requirements of this pilot use case remain unchanged. The rest of the document has not been modified but has been proof read.

The document is structured as follows:

Section 2 describes the EOD pilot case from Deimos and its requirements for ENTICE; Section 3 explains the FCO pilot from FlexiOps company (FLEX) and its requirements for ENTICE; Section 4 describes the CSP Wellness Telecom's pilot and its requirements for ENTICE; Section 5 manifests the current state of the art of technologies covering every feature of ENTICE's components. It was prepared by UIBK, SZTAKI and UL.

2 EOD Pilot Case

2.1 Introduction

Processing and distribution of big space data still presents a critical challenge: the treatment of massive and large-sized data obtained from Earth Observation satellite recordings. Remote sensing industries implement on-site conventional infrastructures to acquire, store, process and distribute the geo-information generated. However, these solutions do not cover sudden changes in the demand of services and the access to the information presents large latencies. Deimos' research focuses in the development of future internet technologies in order to improve Earth Observation (EO) services and to highly reduce the costs associated with on premises deployment.

The ENTICE environment consists of a ubiquitous repository-based technology which provides optimised Virtual Machine image creation, assembly, migration and storage. The Earth Observation data processing and distribution pilot case (EOD) will be implemented in the ENTICE environment to test, validate and demonstrate that ENTICE can support and improve the performance of commercial Earth Observation platforms computed in cloud.

The EOD pilot case mainly consists of the implementation in cloud of the already commercial gs4EO suit, commercialized by Deimos. It is currently operational within the Deimos-2 satellite mission, and it is performing on premises. The current infrastructure is deployed in the Satellite Systems Centre in Puertollano, Spain (see Figure 1).

Figure 1: On-site infrastructure to acquire process, store and distribute satellite imagery of Deimos-2 satellite

Within the ENTICE project, Deimos intends to implement the system in a cloud computing infrastructure to provide Deimos-2 satellite imagery to end users and carry out a pilot demonstration. The ENTICE environment will be used to optimize the implementation and performance of the EOD pilot. It is expected that the optimization of the VMs highly contribute to provide auto-scaling and flexibility to the ingestion of satellite imagery, its processing and distribution to end users with variable demands.

2.2 Global context analysis

Earth Observation commercial data sales have increased to 550% in the last decade. The field is considered a key element in the European Research Road Map and an opportunity market for the next years. The forecast for this decade is $4 billion in commercial data sales at the end of 2019. This makes EO a field of new business opportunities and work.

Although the proven importance of EO in society, the access to the information obtained from satellites follows traditional and expensive paths to cover on demand services for different potential customers: conventional data centres and conventional distribution of services. This presents several drawbacks: i) the cost of acquiring recent images of the Earth is very high, which limits the access of potential users to this technology, ii) clients cannot access directly neither fast to the information they need because this has to be processed and ad-hoc distributed, iii) the service is not flexible to be adapted to sudden changes in the demand.

Cloud computing is presented as a possible solution to improve common services and create new market opportunities because it is elastic, scalable and it works on demand through virtualization of resources.

Satellite and Earth Observation applications are then clear use cases for deployment on the cloud for the following reasons:

- The global nature of EO data, with ground stations and users geographically located all over the world, mandates to deploy a worldwide infrastructure connecting all the stakeholders. Ground stations, ground control centres and data processing centres would take advantage of a rapid, agile, resilient and secure interconnected computer system for sharing the bulk of EO data. Final users would also take advantage of having data access points as close as possible to them to minimize the delay.
- The massive size of EO data generated by today´s sensors, in the order of daily Terabytes, suggests the use of a cost-effective procurement of the computer infrastructure for archiving and processing. Thus, it seems a good idea to pay for computer resources only as you need them, as is the usual cost model of services on the cloud. Note that EO data is received 'in-batches' for each receiving ground station on each 'contact', when a satellite downloads all perceived data since the previous communication. Then the processing, archive and dissemination process of that received data is triggered and executed on the provided infrastructure.
- The amount of resources needed to optimally process and distribute EO data to the global user community is large. Data processing would be greatly enhanced by using a massive number of computing nodes working on parallel with techniques related to High Performance Computing (HPC) and High-Throughput Computing (HTC) techniques for, respectively, generating results as fast as possible, and for processing as many jobs as possible in a given time.

Earth Observation systems are ideal use cases to be deployed on a cloud infrastructure with a dedicated middleware for automatic and intelligent provisioning and deployment to run EO applications only when and where they are needed.

2.3 Use case analysis in the context of ENTICE

The EOD Cloud architecture is based on the following components: (i) GSmonitor, which ingests in the system the raw data acquired by the satellites and downloaded in the antennas; ii) Processing chain, conformed by product processors representing the Instrument Processing Facility for the Deimos-2 mission; (iii) Archive and Catalogue, which represents the data catalogue subsystem already used in several EO missions; (iv) A front-end layer for end users representing the data distribution portal used for both data access and ordering (out of the scope of this project); and v) an Orchestrator, which manages the modules of the whole architecture.

With ENTICE, we expect to optimize the deployment of virtual machines, which will reduce the time of auto scaling the processing and distribution of satellite imagery. First, VM images and their storage will be optimised for the needed QoS requirements. For the optimization, the necessary details will be described with the ENTICE ontology and will be included in the knowledge base. Provided details will include functional descriptors of EOD software components and also quantitative information about their computational, storage and communication needs. This will allow for an automatic generation of possible VM configurations at runtime. An optimal configuration may be found, for example in cases when some VMs are not permanently running, as the process of creating VMs is triggered by the satellite revisit plan occurring once every three days over each region/receiving station (for the Deimos-2 mission).

A major optimization is expected in the auto scaling tasks both for processing of ingested raw data in the cloud and for distribution of processed products to end users. In the case of the auto scaling of processing instances it is interesting to notice that current VMIs include the products processors software developed by Deimos, but also the operative system and local storage associated to the VM. In a hypothetical auto scaling task of the processing chains, notice that the VMI is replicated as many times as required with all the redundant information associated to the operative system and the local storage. An optimization of the processing chain VMs would highly simplify the VM and for instance its replication in auto scaling tasks, mainly if there is variability in the ingestion of raw data.

In the case of the distribution of final products to end users intrinsically has variable demand. Although we can estimate the number of users that we will have in a near future, is unpredictable under some circumstances in specific applications, for example in emergency situation scenarios. In this case, we have the same issue; we have to autoscale the service with non-optimized VMs. This provides higher latencies in the deployment of VMs and it implies a higher cost for deployment and storage, since we are storing redundant information.

Also in this area, the ENTICE solution will be used to dynamically optimise the runtime environment. Each time the satellite downloads its data on a receiving station, all processing chain tasks need to be recreated and start the data processing, catalogue and writing on the archive requiring fast deployment of appropriate VMs close to the receiving station. Meanwhile, the user services and archive and catalogue tasks need to continuously serve read queries from their users and must be quickly scaled-out to cope with likely bursts in user demand. The expected lifetime of product processors instances depends on the amount of received data and the overall system's processing speed.

This project will try to find the most cost-effective configuration that could cope with the hard and soft real-time constraints on data processing while ensuring a good performance, cost and storage optimisation trade-off. The overall process may also necessitate new SLAs, automatically prepared by ENTICE.

2.4 Impact

2.4.1 Scientific and technological impact

The pilot case EOD in ENTICE will contribute to provide a worldwide service in the Earth Observation Industry. It will test if the architecture for Earth Observation implemented on-premises by Elecnor Deimos can be implemented by using cloud computing technologies and facilities, providing viable solutions for the complex EO market and improving the quality of the services that are currently offered.

It will test the viability and ease of use close to market. In addition, it will define which parts of a complete EO system are likely of being virtualized in cloud, and which ones are more interested to keep in a private infrastructure depending on the variability of the demand and also in the complexity of the flight segment, i.e. the heterogeneity of satellites providing images to the EOD system computed in cloud.

The pilot case will also be a tool to analyse the performance of cloud computing with regard to the service offered. The implementation and demonstration scenario with the EOD pilot will facilitate a specific analysis of deployment costs, expected performances in real implementation and comparison with conventional infrastructures.

The impact that ENTICE will have in the EOD pilot is the following:

- Costs reduction:
  - Reduced storage of optimized VMIs
  - Reduced reservation and deployment time
  - Reduced runtime of the overall system
- Performance increase:

- Reduced reservation and deployment time
- Reduced runtime of the overall system

- Independence from a specific infrastructure provider:
  - Facilitation of the distribution of VMs in distributed infrastructures
  - Independence from a specific infrastructure provider

2.4.2 Impact in ENTICE

EOD will provide to ENTICE environment, functional and non-functional requirements to implement industry driven applications in cloud.

EOD will provide information to improve the QoS, optimization of the Virtual Machines, fast deployment of VMs and costs reduction.

The deployment time of the Virtual Machines and the optimization at storing them will be mainly investigated in order to improve the feasibility of the parallelization of the Earth Observation system's capabilities such as processing and geo-images distribution mainly. Furthermore the data rate in communications between Virtual Machines and the elastic resource provisioning will be further investigated.

Through the EOD pilot, ENTICE will be tested in a market that is increasing: modernization of Earth Observation and space system by using future internet technologies.

The EOD pilot will provide ENTICE a use case to optimize VMIs with high range to be optimized, especially the VMIs of the processing chain module.

Furthermore, ENTICE will be tested under a big data environment in which high size images and large amounts of data have to be processed.

In case of EOD pilot case, the main aspects that ENTICE has to optimize are: i) the size that each VMI needs to be hosted; ii) the data rate of communications between VMIs; iii) the deployment time that VMIs need to be instantiated and to be fully operative; iv) the level of customization that VMIs can support; v) the elastic resource provisioning offered; and vi) to reduce the difficulties on the VMI creation and specification.

2.4.3 Social impact

The reduction of the processing, storage, communications and distribution costs of EO services will facilitate the access to the remote sensing technology of common end users, but also of a more general public. EOD pilot case will contribute defining the basis to advance in the use of geospatial information of the nine "Societal Benefit Areas" defined in GEO: disasters, health, energy, climate, water, weather, ecosystems, agriculture and biodiversity; by demonstrating

whether or not cloud computing offers technologically and economically viable solutions to offer highly demanding services.

The results obtained with the implementation of the EOD pilot will be used by Deimos to offer new services to the general public, current end users and future potential users. The rest of the EO industry will be beneficiated since we will define a benchmark that relates the demand with the technology to offer quality services.

2.5 Description

The Earth Observation data processing and distribution consists of the implementation of the Elecnor Deimos' geo-data processing, storage and distribution platform of Deimos-2 satellite by using cloud technologies. The main functionalities of the system are the following:

- **Acquisition of raw data**: when the imagery data is ingested from the satellite into the ground station, the system is notified and the ingestion component automatically ingests the raw data into the cloud for processing.
- **Processing of the raw data**: once the data is ingested, it is processed in the product processors. There are several processing levels.
- **Archiving and cataloguing geo-images:** the different products obtained from the processing of raw data are archived and catalogued in order to provide these images to end users or to provide high-added value services.
- **Offering user services:** this is the front-end of the system. It allows end users to select which product they want to see or to download. Furthermore, it provides high-added values in the imagery.

2.6 Architecture

The EOD architecture is shown in Figure 2. It is composed by the following components:

- GSMonitor: It ingests the available new raw data from the ground stations to the Cloud system and notifies the action to the Orchestrator. When it finishes the task, it notifies the end of the task to Orchestrator by using a message.
- Orchestrator: The Orchestrator is the component that manages the tasks to be done by all the modules of the architecture computed in the cloud infrastructure. The Orchestrator has the following functions:
    - To identify which outputs shall be generated by the processors.
    - To generate the Job Orders. They contain all the necessary information that the processors need. Furthermore these eXtensive Markup Language (XML) files include the interfaces and addresses of the folders in which the input information to the processors is located and the folders in which the outputs of the processors have to be sent. They also include the format in which the processors generate their output.

- To look for raw data in the ground stations (pooling) to ingest such raw data in a shared storage unit in the cloud for its distribution to the processing chain.
- To control the processing chain by communicating with the product processors.
- To manage the archive and catalogue.

- Product Processors: they are modules which are in charge of the processing of the raw data and the products of previous levels to produce image products. The four most important operations are explained in Section 2.7.
- Archive and Catalogue: where the processed images are stored and catalogued for their distribution. It offers a Catalogue Service for the Web (CSW) interface with the catalogued products.
- User Services: end users can access to the products, request them or to obtain high added values services by using web services.

The following section explains the components of the pilot's architecture depicted in Figure 2.

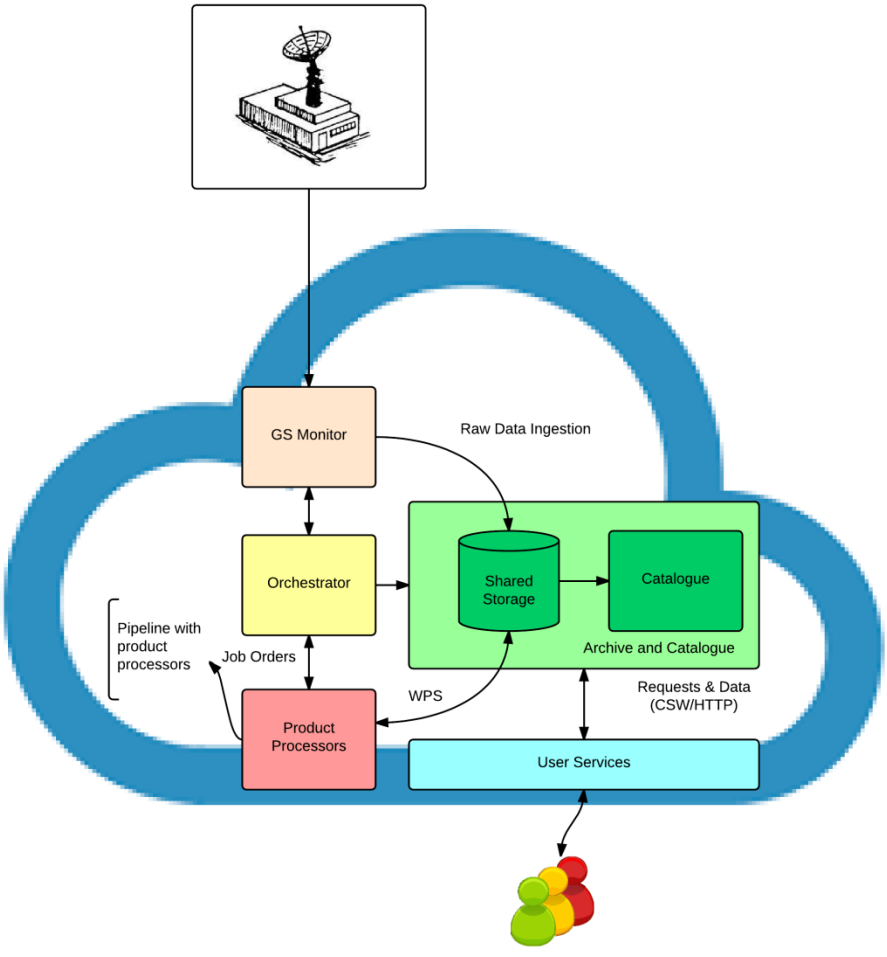

Figure 2: EOD's architecture

It shall be noticed that each component of the EOD's architecture will be attached in an individual Virtual Machine (VM) or container provided by the ENTICE environment.

2.7 Components of the pilot

2.7.1 EO data processing on demand

The Deimos-2 Product Processing Chain is an element of the Payload Data Generation Segment (Ground Systems) that is in charge of the processing of the payload raw data from a satellite recording to produce imagery products (images and associated post processing and analysis). The four, most important operations that the products processors perform on the input data are the following:

- **Calibration** - to convert the pixel elements from instrument digital counts into radiance units.
- **Geometric correction** - to eliminate distortions due to misalignments of the sensors in the focal plane geometry.
- **Geolocation** - to compute the geodetic coordinates of the input pixels.
- **Ortho-rectification** - to produce ortho-photos with vertical projection, free of distortions.

Moreover, the product processors generate metadata, in line with industry standards (such as OGC), to facilitate the cataloguing, filtering and browsing of the product image collection.

The output image products are classified into different levels, according to the degree of processing that they have been subjected to. The Deimos-2 Processing Chain contains four Processing Levels (L0, L1A, L1B and L1C) and associated Products (L1B-R1, L1B-G, L1C-R2, L1C-T1, L1C-R3 and L1C-T2) that are executed automatically or semi-automatically upon ingestion of new satellite data as depicted in Figure 3.

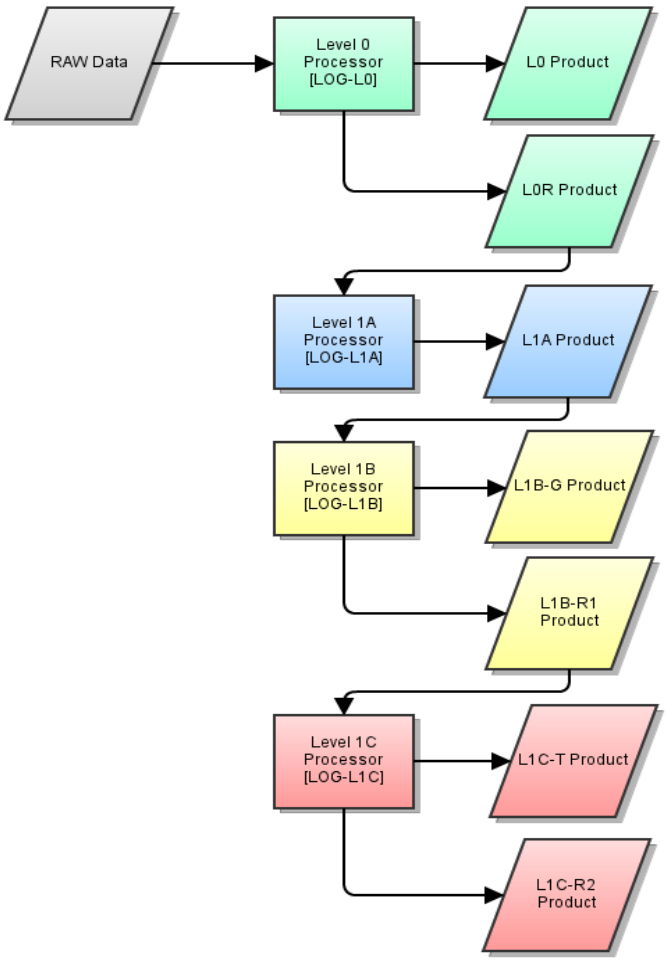

Figure 3: Processing chain Process4EO

Each of the processors levels depicted in different colours in Figure 3 could be deployed in one or several computing nodes. A central 'Orchestrator' would be the responsible to manage the complete system and to assign the execution of a certain processing level job on an available processing node. Extra computing nodes with Graphical User Interfaces could be used by human operators for some manual interactions with higher-level processors. Optionally, an external server could be used for authentication purposes in case the processing chain is using real Deimos-2 data.

2.7.2 Archive, Catalogue and Distribution

This is a multi-mission component in charge of the archiving, storage and inventory function of mission data and products. It is responsible for the permanent storage of all the data and products received from the satellite(s) for an extended period of time in the Archive. The Catalogue allows the other user Ground Segment (GS) facilities to access to the mission metadata and products.

It gathers these products interacting with other GS components and catalogues the products according to several useful metadata as well as physically storing the products, to be able to distribute them on demand (e.g. to the users).

- The **Archive** is composed by an optimized storage structure allowing managing a huge amount of data, efficient storage and retrieval of any kind of file. The Archive shall be organized in hierarchical levels of storage in order to provide a cost effective storage solution:
    - Short Term Archive (STA): Fast response for storing and retrieving most recent used data, high cost per gigabyte. This level provides redundancy in online storage.
    - Online Rolling Archive (ORA): On-line storage but slower than Data-cache. In principle, for Deimos-2 satellite, it will have all the data of the mission.
    - Backup: for safely maintaining an archive copy of all data in disk. Alternatively, media could be used. It is slower but it is a more cost effective storage solution for huge amounts of data.

  The Archive component is responsible for handling those archive levels, STA and ORA, which automatically moves data between archive levels. The Archive stores the minimum information of the archived files to provide quick access to the files.

- The **Catalogue** stores an inventory database with the metadata of archive files.
- The **Catalogue Web services** (CSW) includes an application and web servers to allow users accessing via HTTP interfaces to the Archive & Catalogue through Description, Discovery & Publication ops, following OGC standards.

2.8 User services

2.8.1 Description

The User Services tool has four main different objectives:

- **Generation of new image requests** (linked to a given processing level and possible distribution).
  The visualization is possible via 2D and 3D World Maps, with multiple configurable layers, or using Gantt Charts. Figure 4 shows a representation of the image requests interface.

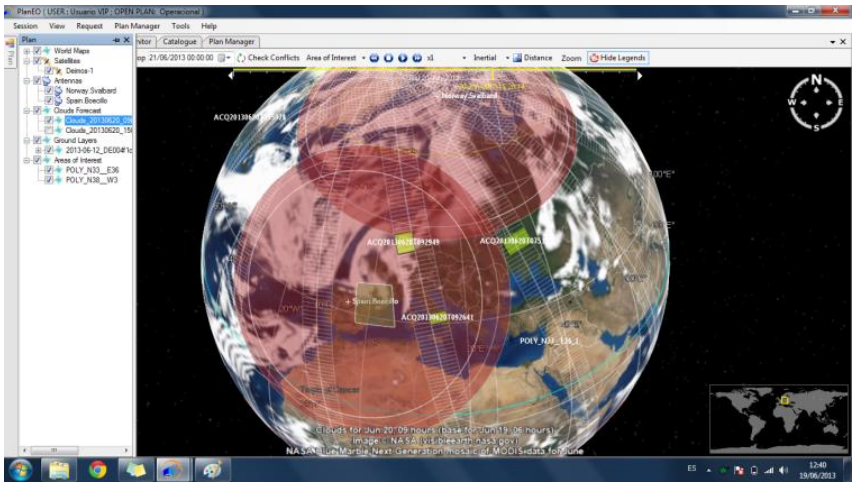

Figure 4: Representation of the image requests on user4EO

- **Chain monitor:** Follow the status of an acquisition, from accepted, to uplinked, acquired, downlinked, processed and distributed.
- **Access to the mission catalogue**, using different filters, and allowing to downlink quick-look images that can be placed onto the map (e.g. for supporting new image requests). Figure 5 depicts the catalogue interface.

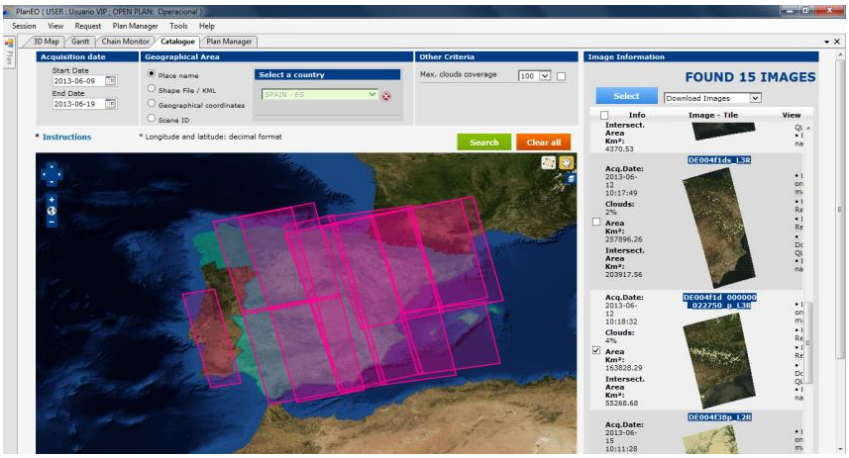

Figure 5: Access to mission catalogue on user4EO

- **Requesting Processing Orders** and/or **product dissemination** from the data already stored in the archive.

For the purposes of this project, the feature mentioned in second bullet “chain monitor”, that provides an interface with the Mission Planning facility, will not be available, since it allows the user to control the satellite.

2.8.2 Architecture

This component is actually deployed on machines of the final user and it is used as interface for the rest of the archive and data distribution services.

2.8.3 Compliancy with requirements

Regarding the expressed requirements for this system, this application presents the following features:

- Scalable and distributed – it can be deployed on several virtual machines, at least one node per component.
- Heterogeneous – it is formed by a collection of different interacting software components with computer resources:
    - Data-intensive: Archive, Data Acquisition and Data Dissemination
    - Compute-intensive: Catalogue and Catalogue Web Services
    - Network-intensive: Catalogue Web Services and Data Dissemination.

The user services could be deployed on the final user´s computers with GPU for a better data visualisation.

2.9 Markets and users

These products are oriented, to specific communities such as scientists, governmental institutions and private companies: a business-to-business (B2B) model more than a B2C. However, with the incorporation of our applications in cloud we expect to make wider the users spectrum to a more general market.

Currently we do not have more than two or three instances of our products being beta tested on selected customers but this situation will change in the near future after the launch of our new Earth Observation satellite, the Deimos-2.

We think that Satellite and Earth Observation applications are clear use cases for deployment on the cloud because of the global nature of EO data (with users geographically located all over the world), the size of that data (in the order of Terabytes, daily) and the amount of resources needed to optimally process and distribute this data to those final users.

2.10 Problems

To understand the problems that the Earth Observation Distribution (EOD) pilot case is facing and how the ENTICE system would try to solve them, first we need to describe the EOD system and how it works.

The EOD is a “Space data system”, seen as a big “black box” with inputs and outputs. The EOD acquires data, processes it with a series of functional steps and generates output products.

As simple as this definition sounds, there is a series of added complexities that makes this pilot case. Those complexities lie in the following characteristics of the system:

- The "input" (or data supply) consists of images obtained from satellites. These images are obtained by a constellation of satellites flying in orbit around the Earth, hundreds of kilometres above the surface, which can only image a certain area while they are directly above the objective (NADIR pointing, no agile platforms). Satellites can only work during a certain time per day as they operate with a limited set of resources (duty cycle) and can only send their acquired data once flying over a defined list of locations on ground, where a fixed (or temporary) Ground Station antenna is programmed to make a communication link with the overpassing satellite.
- The "system" shall be distributed throughout several geographically dispersed areas in the world but it always starts with a network of ground station antennas placed strategically to minimize the time (and allocated resources) needed from the time the satellite acquired the image and downloaded it into a ground station. After the data download there exists a series of processing, storage and management functionalities, aiming to refine the data acquired and extract useful information from them to finally distribute the outputs to the external users.
- The "output" (or data demand) is driven by a global community of users, placed anywhere in the world, with different latency requirements, request frequencies and asking different products and combinations of them.

The system can be described according to its "system effectiveness", i.e. the extent to which a system may be expected to achieve its objectives within its specified environment. The effectiveness of a Space data system is a function of all the following parameters:

- What is the system producing? What are the expected results of the system?
    - Input: different image types (coming from several sensor types, i.e. optical, SAR, LIDAR, radiometers, sounders, etc.); spatial resolution; spectral resolution; etc.
    - System: processing complexity; etc.
    - Output: product levels; etc.
- How much data is the system able to ingest? How much data is the system producing? How much data is the system storing?
    - Input: number and size of images acquired by a number of satellites, obtained during a number of contacts with ground stations, with a certain time of contacts and a contact bandwidth, etc.
    - System: data retention policy, data storing policy, data backup policy, etc.
    - Output: number and size of data products processed for a number of users on a number of requests per user, with different product levels, etc.
- How, how often and how fast is the system acquiring, producing and delivering the data?

- Input: time between contacts; number of ground station antennas; location of antennas; antenna bandwidth; antenna availability; etc.
- System: system size requirements; maximum cost requirements; latency requirements (RT, NRT, best effort), performance of components, throughput of the network, etc.
- Output: request frequency (on-demand, periodic, etc.), output bandwidth per user; etc.

There are simply too many variables to attend for having a complete characterisation of a Space data system and, in fact, this problem is solved by a well-known discipline in satellite engineering and mission analysis. This pilot case would use a simplified model of a hypothetical Space data system focused in only a subset of those parameters:

- System size: how much data gets in, needs to be kept in the system, and goes out.
- System performance: how often data gets in and gets out (data frequency) and how much time does it take from acquisition to delivery (product latency);
- System cost: derived from the other two, how much would it cost to comply with minimum requirements on size and performance.

For a more exhaustive and complete description of the pilot case, please see the document [ENTWP2TEV003-EOD-TechnicalApproach].

2.10.1 System definition

The system functionalities include the following steps, which are assumed to be done in serial (for simplicity):

1. Basic Processing - Acquisition and processing of the information, to derive the required mission data products. This functionality is further subdivided in these steps:
   1.1. Inward network, including the acquisition of raw data by the ground stations and the data movement to the processing and storage subsystems.
   1.2. VM management
   1.3. Orchestration and Processing of the raw data into increasingly refined levels of data products.
2. Storage - Storage of instrument output and all other information relevant to the correct interpretation of this output, including metadata and ancillary files. This functionality is further subdivided in these steps:
   2.1. Middle network, which is the movement of data products from the processors to the archive and catalogue subsystems and vice versa.
   2.2. VM management
   2.3. Archive and catalogue of the data products (including RAW data) to be easily accessible to users.

3. Distribution - Interfacing with consumers, receiving requests and delivering mission products. This functionality is further subdivided in these steps:
   3.1. Outward network, which is the movement of data products from the archive and catalogue subsystems to the user services and the final users.
   3.2. VM management
   3.3. User Services, Dissemination and Delivery of desired data products to final users either on a time synchronous way (after a defined period of time) or asynchronously, on-demand, after a request by the user.

All steps named as "VM management" are related to the preparation of templates, creation of VM images, and access to those images within the repositories, delivery, configuration and set up of the VM hosting. Those are the ones which will be most clearly affected by the ENTICE capabilities. Let's assume the rest of steps are provided subsystems, i.e. software running on standard platforms and existing network links which cannot be modified but which performance and cost would be tangentially improved by ENTICE as described in sections 2.10.3 and 2.10.4.

Again, this system is too complex for trying to model it during the scope of this project, and for that, these steps would be adapted and simplified for a series of different scenarios using a series of "given" subsystems obtained from the Deimos' GS4EO (Ground Segment for Earth Observation) suite of products.

Finally, with regards to the problem at hand, the ENTICE system would focus on the "VM management" steps of the system, trying to optimize it in terms of the three parameters and metrics of the Space data system affecting the system effectiveness:

- System size - reducing the overall storage space needed for VM images and repositories;
- System performance (product latency -time), reducing the performance overheads due to an inefficient construction of the VMs and
- System cost (derived from the other two), which are also derived from all the others and co-dependant between each other. For instance, optimizing the up-time for each VM to avoid paying for idle processing times or reducing the overhead due to VM management.

ENTICE would see to reduce all three metrics on each of the "VM management" activities and therefore, improve the overall system effectiveness during a series of defined scenarios.

2.10.2 Scenarios

The defined scenarios for this system are:

- Sc1. Deimos-2 - Nominal operations. In which we obtain real performance metrics and present real system requirements for our operational system of one satellite. This scenario would be our metrics baseline.

- System. Our baseline system would be composed of the following nodes.

Table 1: System configuration for Scenario #1

	Product processors	Archive	Catalogue	Monitor	User Services
SC1	7	2	1	2	1

- Sc2. New Space. Hypothetical futuristic scenario, derived from the Deimos-2 scenario by applying a series of metrics multipliers, including several satellites, ground station antennas, more distributed processing centres, archives and catalogues and delivery networks all deployed on the cloud.
  - Sc2a. Land Management. Governments, planning offices and agricultural agencies need to characterize landscape and crops in vast rural regions. These users demand a multi-temporal product which allows analysing the land cover dynamics and detecting changes during a vast amount of time. Remote sensing capabilities are ideal for covering huge amounts of surface and to identify and classify vegetation, and to do it periodically. Figure 6 shows a sample land cover map and classification of Australia.

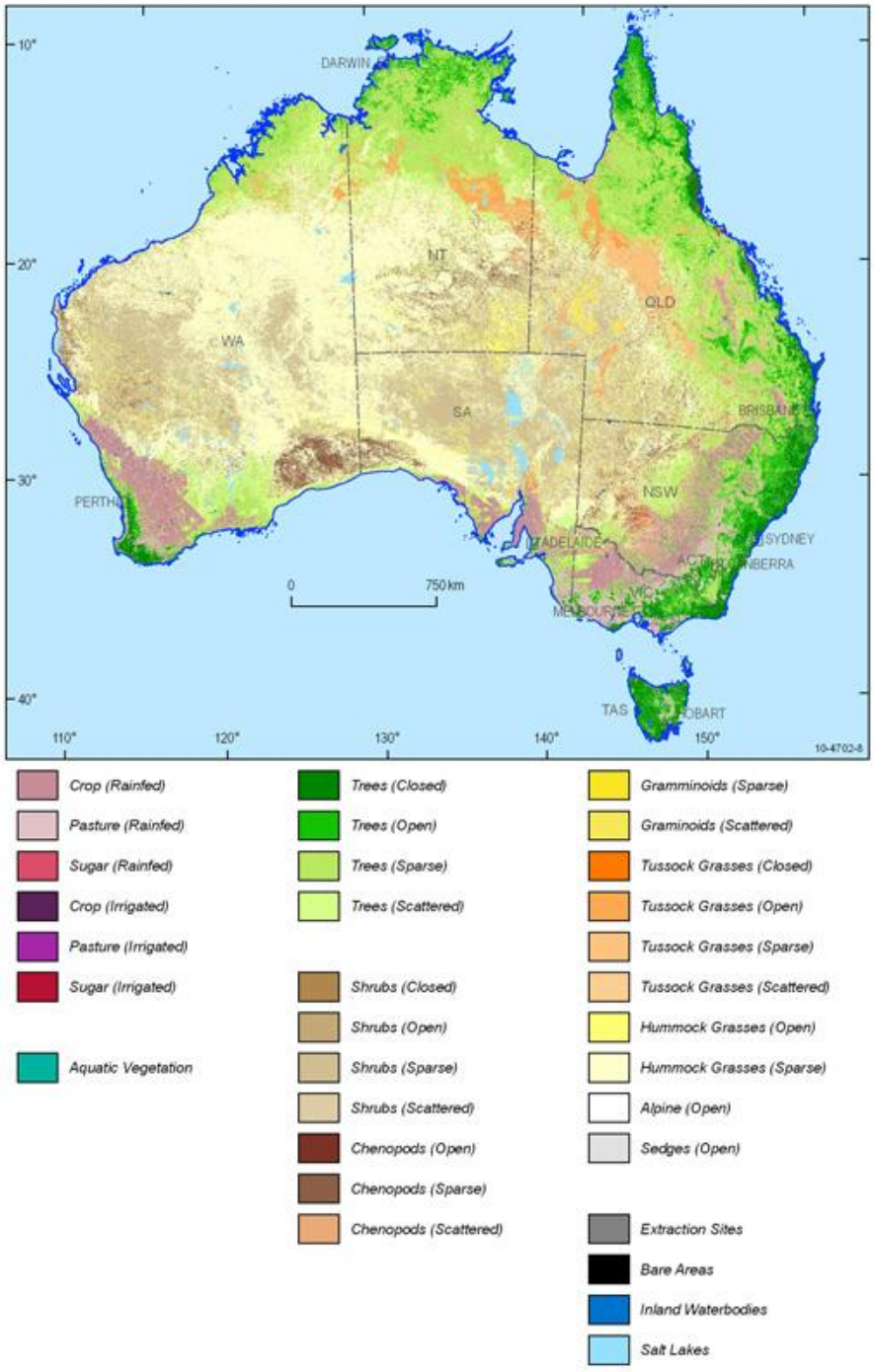

Figure 6: Land cover map of Austria (C) Australian Bureau of Agricultural and Resource Economics and Sciences

- Objective: archive optimization
- Uses: reprocessing campaigns; time-series analysis; etc.
- Description: The Space data system would be used to systematically acquire data over a big target area (1,000,000 sq. km), accumulate it on

an archive for a period of a year, perform a time-series analysis of the whole data sets and release the final derived product only once.

- ENTICE approach: The idea is to procure the VMs with acquisition, archive and cataloguing capabilities dynamically, only when needed, responding to the increasing data archive size requirements and optimizing costs.
- Requirements: Low performance needed on processing, high performance and size requirements on storage, low performance on delivery services. Variability is high but predictable.
- System. Our system for SC2a would be composed of the following nodes. The archive will scale up a 200% each month during the whole year of operations. The catalogue will scale up 50% monthly.

Table 2: System configuration for Scenario #2a

	Product processors	Archive	Catalogue	Monitor	User Services
SC2a	7	20	5	2	1

- Sc2b. Infrastructure monitoring. During the construction or operations of major infrastructures in remote, hazardous areas (like high performance railway lines in the desert, attacked by wind, moving sand dunes and changing temperature conditions), these infrastructures would need constant monitoring to minimize the impact of the ambient conditions. Remote sensing technologies can be applied to assess the risk of affection in the infrastructure and to minimize and prevent the impact: Sand movement and speed, risk maps, sandstorms monitoring, floods monitoring, etc. Figure 7 shows a sample of infrastructure which would benefit from space-based monitoring.

Figure 7: High Speed line Medina-La Meca in Saudi Arabia[1]

- Objective: processing optimization
- Uses: on-demand processing; etc.
- Description: The Space data system would be used to acquire images over a defined area (500 sq. km) with a fixed periodicity of at least-one image per day, process those images as fast as possible (optimizing costs vs latency) and deliver them to a fixed set of users daily.
- ENTICE approach: The idea is to procure the VMs with processing and distribution capabilities dynamically, only when needed -once per day- and de-commission them when they are not needed.
- Requirements: High performance needed on processing, low on storage, low performance on delivery services. Variability is high, but predictable.
- System. Our system for SC2c would be composed of the following nodes.

Table 3: System configuration for Scenario #2b

	Product processors	Archive	Catalogue	Monitor	User Services
SC2b	21	2	1	2	1

The product processors would be scaled-up 300% each daily, only during the time the processing is done.

- Sc2c. Response to disasters. Upon the event of a disaster, say an earthquake on a remote area, government and safety agencies would need to acquire, as soon as possible, images from the affected area before, during and after the event. Those

[1] Image from https://www.thalesgroup.com/sites/default/files/asset/document/Gonzalo%20Ferre.pdf

images would be later demanded by a huge number of users from all around the world, either to manage emergency services and to assess the damages in the infrastructure or simply out of curiosity for the damages. Figure 8 shows an example on how to monitor a disaster from space.

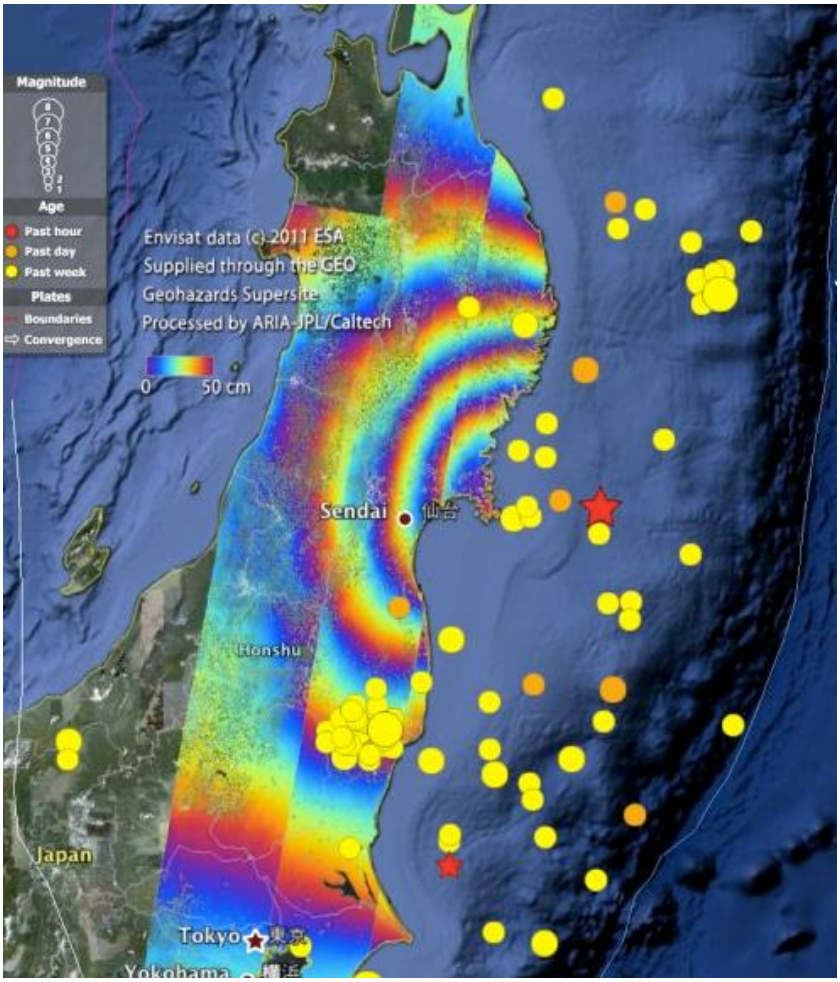

Figure 8: SAR image of the 2011 Tohoku-oki earthquake in Japan processed by JPL/Caltech ARIA project

- Objective: distribution optimization
- Uses: Content Distribution Network;
- Description: The Space data system would be used to acquire one image from a disaster area, process it on a standard infrastructure and to deliver it to as many users as possible, in the shortest possible time.
- ENTICE approach: The idea is to dynamically create the VMs with catalogue access and distribution capabilities, as “close” as possible to the final users, with copies of this “famous” product, optimizing costs.

- Requirements: Low performance needed on processing, low on storage, high performance on delivery services. Variability is high.
- System. Our system for SC2c would be composed of the following nodes.

Table 4: System configuration for Scenario #2c

	Product processors	Archive	Catalogue	Monitor	User Services
SC2c	7	2	1	2	10

The system would scale-up the user services nodes up to 10 times its size while the data is being distributed.

2.10.3 EOD problems and ENTICE objectives

The problems of the EOD pilot related with ENTICE objectives are described as follows:

- **Problem 1. Manual creation of VM templates** - Whenever there is a new subsystem version release engineers manually create VM templates with the required software packages and it could take an hour to complete. The ENTICE objective #1 is the creation of lightweight VM images through functional descriptions, making them highly-specific and highly-optimised tuned for minimal size and management overhead.
- **Problem 2. Monolithic VM images** - Our VM templates are all created on the top of a baseline Red Hat Enterprise Linux distribution including all software packages from the "desktop" suite. They are not tailored in any way to the target subsystems and this penalises the overall VM performance and size The ENTICE objective #2 is to create distributed lightweight VM image storages which will decompose VM images into smaller reusable parts bringing down the storage space and associated costs.
- **Problem 3. Proprietary un-optimized VM repositories** - Our systems uses a proprietary VM repository including all current and previous versions of the VM templates which is not optimized on size. The ENTICE objective #3 is to develop autonomous multi-objective repository optimisation processes which will ensure that commonly used VM image parts are replicated and stored more widely, and thus, they can be discovered and delivered from local repositories quicker, while user-provided parts come from a different external location.
- **Problem 4. Inelastic resource provisioning** - There simply isn't a dynamic resource provisioning system on place on our system premises. There is no automatic scalability procedure for responding to changes in the supply and demand of data or the latency requirements of our products and clients and manually shutting down VMs, assigning additional resources, and then powering up a new, better VM again could take 20 minutes of time causing an interruption on the service. The ENTICE objective #4 is to ensure elastic resource provisioning to improve the on-demand scaling of our systems in the cloud in response to changes in compute and storage requirements.

- **Problem 5. Automated preparation and deployment** - All VM preparation and deployment tasks are made manually, and they usually take an unacceptable time (longer than a complete processing pipeline) to finish.
  The ENTICE objective #5 is to create an information infrastructure for strategic and dynamic reasoning, to support automatic VM packaging of our applications based on criteria such as QoS functional properties, execution time, costs and storage and dynamic benchmark information about the underlying federated Cloud.

2.10.4 EOD problems' metrics

The following table gives some metrics figures for our current scenario (Sc1) and estimations for the future scenarios (Sc2) of the system.

Table 5: Current EOD metrics

Problem	Metrics	Units	Sc1	Sc2a	Sc2b	Sc2c
#1 - VM Synthesis. Manual creation of VM templates	VMI size	GB	111	111	111	111
	VMI delivery time	sec	180	180	180	180
#2 - VM Analysis. Monolithic VM images	Repository size	GB	2512	7729	5217	4251
#3 - Optimisation. Proprietary un-optimized VM repositories	VM storage	GB	1000	3077	2077	1692
#4 - Elasticity. Inelastic resource provisioning	Elasticity	BOOL	FALSE	FALSE	FALSE	FALSE
#5 - Semantics and SLA. Automated preparation and deployment	VMI preparation time	sec	600	600	600	600
	VM deployment time	sec	446	446	446	446

ENTICE objectives include optimizing the following metrics on each of these "VM management" activities:

- VM Synthesis
  - VMI size (60%)
  - VMI delivery time (30%)
- VM analysis
  - Repository size (80%)
- Optimisation
  - VM performance (30%)

- VM cost (25%)
- VM storage (25%)

- Elasticity
  - Scale time is less or equal to QoS improvement
- Semantics and SLA
  - VMI preparation time (25%)
  - VM deployment time (25%)

The mapping between those objectives and our use case scenarios is shown in Table 6.

Table 6: Objective EOD metrics

Activities	Metrics	Unit	Improvement	Sc1	Sc2a	Sc2b	Sc2c
VM Synthesis	VMI size	GB	60%	44	44	44	44
	VMI delivery time	sec	30%	126	126	126	126
VM Analysis	Repository size	GB	80%	502	1546	1043	850
Optimisation	VM performance		30%	0	TBD	TBD	TBD
	VM cost		25%	0	TBD	TBD	TBD
	VM storage	GB	25%	750	2308	1558	1269
Elasticity	Elasticity	BOOL	NO	TRUE			
Semantics and SLA	VMI preparation time	sec	25%	450	450	450	450
	VM deployment time	sec	25%	334	334	334	334

2.11 Metrics

This section defines the metrics that will be used to validate the EOD pilot and the proposed metrics to validate the ENTICE environment. First, we describe the common metrics for all EOD's components. Second, we specify the metrics for each individual component. Then, the pilot's metrics that will be obtained in the ENTICE environment are described.

2.11.1 Common metrics to all the components of the architecture

Table 7 shows the metrics that are common to all the components of the architecture.

Table 7: Common metrics to all components

Component	Resource	Metric	Description	Unit
All	Bandwidth	EffectiveBandwidth	The bandwidth between the EOD's components.	MB/s
	Latency	Latency	Latency between the components of the pilot case.	ms
	Virtual CPU load	VirtualCPU	The maximum and average CPU used by the component.	%
	Virtual RAM load	VirtualRAM	The maximum and average RAM used by the component.	%

2.11.2 Metrics of the GSMonitor Module

Table 8 shows the metrics of the GSMonitor module.

Table 8: Metrics of the GSMonitor module

Component	Resource	Metric	Description	Unit
GSMonitor	Concurrency	NumThreads	Number of ground stations that the monitor is able to manage.	Integer

2.11.3 Metrics of the Processing Chain Module

Table 9 shows the metrics of the Processing Chain, which includes different product processors modules. Notice that the metrics are applicable to all the product processors.

Table 9: Metrics of the Processing Chain module

Component	Resource	Metric	Description	Unit
Processing Chain	Virtual CPU load	CPUUse PerProcess	Percentage of allocated CPU currently in use by a certain process on the instance. This metric identifies the processing power required to run an application upon a selected instance.	%
	Disk	DiskRead Bytes	The amount of bytes read by a product processor. This metric identifies the processing power required to run an application upon a selected instance.	Bytes
	Disk	DiskWrite Bytes	The amount of bytes written by a product processor. This metric is used to determine the volume of data the application reads from the hard disk of the instance. This can be used to determine the speed of the application.	Bytes
	Network	NetworkIn	The number of bytes received on all network interfaces by the instance. This metric identifies the volume of incoming network traffic to an application on a single instance.	Bytes
	Network	Network Out	The number of bytes sent out on all network interfaces by the instance. This metric identifies the volume of outgoing network traffic to an application on a single instance.	Bytes
	Memory	Memory Use PerProcess	Total number of bytes used by each application on the instance. This metric identifies the size of each application on memory. This can be used to know if the assigned total memory of an instance is enough for the application needs.	Bytes
	Reliability	ErrorRate	Indicates the percentage of problem requests with respect to all requests, measuring “performance failure” in the system. This metric can be used to determine when the system no longer is reliable due to increased load.	%

2.11.4 Metrics of the User Services module

The User Service module is the software module that provides an interface between the end user with the Archive and Catalogue of products. Specific metrics to measure the quality of a service are provided in Table 10.

Table 10: Metrics of the User Services module

Component	Resource	Metric	Description	Unit
User Services	Virtual CPU load	CPUPer Process	Percentage of allocated CPU currently in use by a certain process on the instance. This metric identifies the processing power required to run an application on a selected instance.	%
	Disk	DiskRead Bytes	The number of bytes received on all network interfaces by the instance. This metric identifies the volume of incoming network traffic to an application on a single instance.	Bytes
	Disk	DiskWrite Bytes	The amount of bytes written by a product processor.	Bytes
	Network	NetworkIn	The number of bytes received on all network interfaces by the instance. This metric identifies the volume of incoming network traffic to an application on a single instance.	Bytes
		Network Out	The number of bytes sent out on all network interfaces by the instance. This metric identifies the volume of outgoing network traffic to an application on a single instance.	Bytes
	Memory	Memory Use PerProcess	Quantity of memory that the user services module needs to serve each user request.	MB

Component	Resource	Metric	Description	Unit
	Availability	Average Response Time	This metrics describes the responsiveness of the system from the users' perspective, including the delivery of HTML, images or any other resource. Response times can be measured as "time to first byte" or "time to last byte". The latter one is commonly the preferred Key Performance Indicator, because relates to the entire cycle of response. This metric has a strong influence on the user experience for a given service.	s
	Availability	Peak Response Time	Similar to the previous metric, PeakResponseTime, is a measure for the longest request/response cycle, locating potentially problematic resources: anomalies in the system, "expensive" database queries invoked by specific requests, large files, big libraries, etc.	s
	Reliability	ErrorRate	Indicates the percentage of problem requests with respect to all requests, measuring "performance failure" in the system. This metric can be used to determine when the system no longer is reliable due to increased load.	%
	Availability	Requests Second	Measures the number of requests sent to the target server, including requests for HTML pages, XML, JavaScript or images.	Integer
	Availability	Concurrent Users	Measures how many virtual users are active at any point in time. It is not the same as the request per second because one user can generate a high number of requests, and not every virtual user continuously generates request; real users behave as specified by the scenarios and steps defined in Section 2.10.2.	Integer

2.11.5 Metrics of the Orchestrator module

The orchestrator module is in charge of managing the whole system. Table 11 describes the metrics to evaluate its performance.

Table 11: Metrics of the Orchestrator module

Component	Resource	Metric	Description	Unit
Orchestrator	Virtual CPU	CPUUse	The CPU used by the Orchestrator module. Units:	%
	Availability	RequestAtTime	Number of concurrent process that the Orchestrator is able to manage.	Integer
	Network	NetworkIn	The number of bytes received on all network interfaces by the instance. This metric identifies the volume of incoming network traffic to an application on a single instance.	MB
	Network	NetworkOut	The number of bytes sent out on all network interfaces by the instance. This metric identifies the volume of outgoing network traffic to an application on a single instance.	MB
	Memory	MemoryUsePer Process	Quantity of memory that the Orchestrator needs to manage for the processing of each incoming raw data.	MB

2.11.6 Metrics of the Archive and Catalogue module

The following table shows the metrics of the Archive and Catalogue module. The main task of this module is the storage and listing of the products obtained from the Processing Chain module. Specific metrics are defined to value the functioning of the module.

Table 12: Metrics of the Archive and Catalogue module

Component	Resource	Metric	Description	Unit
Archive and Catalogue	Availability	Average Response Time	The most indicated indicator of how the system is performing from the users' perspective, including the delivery of HTML, images or any other resource. Response times can be measured as "time to first byte" or "time to last byte", being the second one the preferred as a Key Performance Indicator, because is more valuable to know the entire cycle of response; is what the user experiences.	s
Archive and Catalogue	Availability	Peak Response Time	Very similar to the previous one, indicates the longest request/response cycle, showing potentially problematic resources: anomalies in the system, "expensive" database queries involved in certain request, large files, big libraries, etc.	s
Archive and Catalogue	Reliability	Error Rate	Gives a percentage of problem requests to all requests, measuring "performance failure" in the system, being more evident as increases with higher loads. This quick rise error shows where the system is stressed beyond its ability to deliver adequate performance.	%
Archive and Catalogue	Disk	DiskRead Bytes	The number of Mbytes that the module has to serve on all network interfaces by the instance. This metric identifies the volume of out coming network traffic to an application on a single instance.	B
Archive and Catalogue	Disk	DiskWrite Bytes	The number of Mbytes that the module has to write on disk produced by the storing and cataloguing actions.	B

2.11.7 Metrics of ENTICE environment

Table 13 shows the metrics that ENTICE has to provide in order to validate the EOD pilot case.

Table 13: Metrics of the pilot for ENTICE environment

Resource	Metric	Description	Unit
Deployment time	Average Deployment	The average deployment time of VMs	s

	time of VMs	indicates how time is required to instantiate a Virtual Machine.	
Size of VMs	Average total size of VMs	The total size of the VMs required for being hosted on the cloud should be low in order to reduce the deployment time and the costs of storing the VM image on the cloud.	MB
MaxMemoryVM	Maximum available RAM memory that can be requested	The maximum RAM available for creating a VM is important in ENTICE because for the EOD pilot case, the VMs require specific RAM sizes.	MB
MaxDiskVM	Maximum available disk size that can be requested	The number of Mbytes disc space that can be acquired by the VMs.	MB
MaxCPU	MaxCPU	The number of virtual cores that the VMs can request.	Integer
MaxCPUFrecuency	CPU Frecuency	Minimum CPU frequency that ENTICE can offer for creating VMs.	Hz
EffectiveBandwidth	Bandwidth	The bandwidth between the EOD's VMs.	Mb
Latency	Latency	Latency between the VMs of the pilot case.	Ms
LoadBalancing Policies	LoadBalancing Policies	Load balancing policies or algorithms that the ENTICE environment can offer through its load balancers	Integer

2.12 Requirements of the EOD pilot

The following sections list the requirements of the EOD pilot case for the ENTICE environment. They were classified into three main groups: functional, non-functional and interface requirements.

2.12.1 Functional requirements

2.12.1.1 Behaviour

ENT.REQ.EOD.F.0100 The EOD pilot shall automatically ingest collections of satellite imagery from a network of distributed antennas around the world.

Rationale: Understand by a network of antennas a distributed system of ground stations receiving data from satellites. The ENTICE environment shall allow the EOD pilot to perform this behaviour (see Section 2.5).

Verification method: Test

Test type: Unitary test

ENT.REQ.EOD.F.0110 The EOD pilot shall automatically process raw data satellite imagery to provide L0, L0R, L1A, L1B and L1C products (see Section 2.7.1).

Rationale: The EOD pilot will have a software module named as Processing Chain. This component is in charge of the processing of the payload raw data from the satellite imagery to produce different kind of image products in different processing levels:

1. L0 level decodes the raw data providing L0 image products;
2. L0R level transforms the L0 image products in squared images;
3. L1A processor takes the L0R products as input and performs a geolocation and radiometric calibration in the images;
4. L1B level resamples the image and makes a more precise geolocation; and
5. L1C level ortho-rectifies the images of the L1B products by using ground control points.

Verification method: Test.

Test type: Unitary test

ENT.REQ.EOD.F.0120 The EOD pilot shall automatically catalogue satellite imagery.

Rationale: see Section 2.7.2.

Verification method: Test

Test type: Unitary test

ENT.REQ.EOD.F.0130 For the automatic cataloguing of the satellite imagery, the images shall be described with metadata.

Parent: ENT.REQ.EOD.F.0120

Rationale: Every image shall have a metadata file associated in XML format. The Metadata file contains additional information related to spacecraft attitude, spacecraft ephemeris, spacecraft temperature measurements, line imaging times, camera geometry, and radiometric calibration data.

Verification method: Inspection

ENT.REQ.EOD.F.0140 The EOD pilot shall archive satellite imagery organized in hierarchical levels of storage in order to provide a cost effective storage solution.

Rationale: The hierarchical levels of storage shall be the following (see Section2.7.2.):

1. Short Term Archive (STA): Fast response for storing and retrieving most recent used data, high cost per gigabyte. This level provides redundancy in online storage.
2. Online Rolling Archive (ORA): On-line storage but slower than Data-cache. In principle, for Deimos-2 satellite, it will have all the data of the mission.
3. Backup: for safely maintaining an archive copy of all data in disk. Alternatively, media could be used. It is slower but it is a more cost effective storage solution for large amounts of data.

Verification method: Test

Test type: Unitary test

ENT.REQ.EOD.F.0150 The EOD pilot shall provide automatically catalogued satellite imagery to end users through Internet.

Rationale: The main objective of the EOD pilot is to provide an online catalogue of satellite imagery accessible through the Internet (see Section 2.7.2.).

Verification method: Test

Test type: System test

2.12.1.2 Capabilities

ENT.REQ.EOD.F.0160 The ENTICE environment shall provide the EOD pilot Elasticity as a Service.

Rationale: The Elasticity is essential for the EOD pilot, which will perform in an elastic manner to optimize the use of resources to cover specific levels of demand in real time. This requirement is aligned with the EOD Problem 1: Manual creation of VM templates; and Problem 4: Inelastic resource provisioning, already defined. And for instance, they are closely related with ENTICE's Objective 1: Lightweight VM image creation; and Objective 4: Elastic resource provisioning (see Section 2.10.1).

Verification method: Test

Test type: System test

ENT.REQ.EOD.F.0170 The ENTICE environment shall provide the EOD pilot computing and storage resources scalability.

Parent: ENT.REQ.EOD.F.0160

Rationale: The scalability in the EOD pilot will allow the system to automatically auto scale computing and storage resources. This requirement is aligned with the EOD Problem 1: Manual creation of VM templates; and Problem 4: Inelastic resource provisioning, already defined. And for instance, they are closely related with ENTICE's Objective 1: Lightweight VM image creation; and Objective 4: Elastic resource provisioning (see Section 2.10.1.).

Verification method: Test

Test type: System test

ENT.REQ.EOD.F.0180 The ENTICE environment shall provide the EOD pilot resources flexibility.

Parent: ENT.REQ.EOD.F.0160

Rationale: The flexibility in the EOD pilot refers to guaranteeing the coverage of any level of demand in real time and automatically. This requirement is aligned with the EOD Problem 1: Manual creation of VM templates; and Problem 4: Inelastic resource provisioning, already defined. And for instance, they are closely related with ENTICE's Objective 1: Lightweight VM image creation; and Objective 4: Elastic resource provisioning (see Section 2.10.1).

Verification method: Test

Test type: System test

ENT.REQ.EOD.F.0190 The ENTICE environment shall allow the monitoring of the demand of services in the EOD pilot.

Rationale: The monitoring of the demand is essential to have metrics to measure the different levels of demand in different instants so the system architecture can be optimized. This requirement is aligned with the EOD Problem 1: Manual creation of VM templates; and Problem 4: Inelastic resource provisioning, already defined. And for instance, they are closely related with ENTICE's Objective 1: Lightweight VM image creation; and Objective 4: Elastic resource provisioning (see Section 2.10).

Verification method: Test

Test type: System test

ENT.REQ.EOD.F.0200 The ENTICE environment shall allow the monitoring of the EOD pilot system performance.

Rationale: The monitoring of the system performance is required to measure if the system is covering the demand of services in an optimized and automatic manner during both the instances deployment and runtime. This requirement is aligned with the EOD Problem 1: Manual creation of VM templates; and Problem 4: Inelastic resource provisioning, already defined. And for instance, they are closely related with ENTICE's Objective 1: Lightweight VM image creation; and Objective 4: Elastic resource provisioning (see Section 2.10.1).

Verification method: Test

Test type: System test

ENT.REQ.EOD.F.0210 The EOD pilot shall avoid queues in any of the data transfer between the different modules of the architecture as a consequence of the elasticity.

Parent: ENT.REQ.EOD.F.0160

Rationale: The EOD pilot has to be optimized to provide good quality of service. For that purpose queue avoidance has to be considered when using the elasticity characteristic of the system. This requirement is aligned with the EOD Problem 1: Manual creation of VM templates; and Problem 4: Inelastic resource provisioning, already defined. And for instance, they are closely related with ENTICE's Objective 1: Lightweight VM image creation; and Objective 4: Elastic resource provisioning (see Section 2.10.1).

Verification method: Test

Test type: System test

ENT.REQ.EOD.F.0220 The EOD system shall be automatically deployable without human intervention.

Rationale: The ENTICE environment shall facilitate the automatic deployment of the EOD pilot. This will optimize the system in the deployment, but also in the runtime when autoscaling of the system is performed. This requirement is aligned with the EOD Problem 5: Automatic preparation and deployment; and for instance, it is closely related with ENTICE's Objective 5: To create an information infrastructure for strategic and dynamic reasoning (see Section 2.10.1).

Verification method: Demonstration

Test type: System test

ENT.REQ.EOD.F.0230 The EOD system shall be compatible with Deimos-2 satellite.

Rationale: The EOD pilot shall be capable of processing raw data in form of satellite imagery coming from Deimos-2 satellite. For the tests a standard squared image (of 8 sectors) will be used.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.F.0240 In the EOD pilot all the deployed VMs shall be in use.

Rationale: This requirement is a metric which contributes to the optimization of the system and the manner to measure that it is optimized in use is that all the VMs deployed during the execution of the pilot shall be in use, i.e. the system cannot have a VM deployed if it is not being used. This requirement is aligned with the EOD Problem 2: Monolithic VM images; and for instance, it is closely related with ENTICE's Objective 2: Lightweight VM image creation (see Section 2.10.1).

Verification method: Demonstration

Test type: System test

ENT.REQ.EOD.F.0250 The VM repository of the EOD pilot shall be optimized.

Rationale: Understand by optimization that redundant components (VMIs) cannot be stored. This requirement is aligned with the EOD Problem 3: Proprietary un-optimized VM repositories; and for instance, it is closely related with ENTICE's Objective 3: To develop autonomous multi-objective repository optimisation.

Verification method: Test

Test type: System test

ENT.REQ.EOD.F.0260 The EOD system shall be deployable in different geographical locations.

Rationale: The EOD system has to be portable. It shall have the capability of being deployable in distributed nodes with different locations. This will allow the EOD system to be replicable in any location and configurable for specific ad hoc projects, such as for example if an end user asks for an EOD system to process images of his own satellite. This requirement is aligned with the EOD Problem 5: Automatic preparation and deployment; and for instance, it is closely related with ENTICE's Objective 5: To create an information infrastructure for strategic and dynamic reasoning (see Section 2.10.1).

Verification method: Demonstration

Test type: System test

ENT.REQ.EOD.F.0270 The EOD system shall be completely deployable in an IaaS with the exception of the monitoring and control terminal.

Rationale: The EOD system has to be completely deployable in a cloud computing infrastructure. However it is required that access to monitor and control the infrastructure to a local/private terminal is provided outside the cloud (see Section 2.6).

Verification method: Demonstration

Test type: Integration test

2.12.1.3 Knowledge model

Not Applicable

2.12.1.4 VMI images

ENT.REQ.EOD.F.0280 ENTICE shall allow the reservation of VMIs in Debian, CentOS and Ubuntu.

Rationale: The EOD pilot requires a Linux distribution to be deployed.

Verification method: Inspection

Test type: System test

ENT.REQ.EOD.F.0290 ENTICE shall allow the migration of EOD's VMIs.

Rationale: The migration of EOD's VMIs is a feature that will facilitate portability independently of the infrastructure used. It is aligned with EOD Problem 3: Proprietary un-optimized VM repositories, and for instance with ENTICE's Objective 3: To develop autonomous multi-objective repository optimisation processes (see Section 2.10.1).

Verification method: Test

Test type: System test

ENT.REQ.EOD.F.0300 ENTICE shall allow the replication of EOD's VMIs.

Rationale: The replication of EOD's VMIs is a feature that will facilitate portability independently of the infrastructure used. It is aligned with EOD Problem 3: Proprietary un-optimized VM repositories, and for instance with ENTICE's Objective 3: To develop autonomous multi-objective repository optimisation processes (see Section 2.10.1).

Verification method: Test

Test type: System test

ENT.REQ.EOD.F.0310 ENTICE shall allow the download of EOD's VMIs.

Rationale: The download of EOD's VMIs to a local storage is a feature that will facilitate portability independently of the infrastructure used and will facilitate unitary tests. It is aligned with EOD Problem 3: Proprietary un-optimized VM repositories, and for instance with ENTICE's Objective 3: To develop autonomous multi-objective repository optimisation processes (see Section 2.10.1).

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.F.0320 ENTICE shall guarantee remote access to EOD VMIs.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.F.0330 ENTICE shall offer distributed shared storage.

Rationale: The EOD architecture is designed to use a distributed shared storage and file system to optimize the transfer of data between all the modules of the architecture (see Section 2.6).

Verification method: Test

Test type: System test

ENT.REQ.EOD.F.0340 ENTICE shall allow the system to perform with physical instances.

Rationale: It is important to guarantee a performance for IaaS and PaaS users. In the case of the EOD pilot, the software is operative in physical instances and it is desired to have the possibility of using dedicated physical instances which some IaaS commercial providers facilitate.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.F.0350 ENTICE shall offer multi and single tenancy.

Rationale: In a virtualized environment the resources are shared between all the users (multi-tenancy), however to guarantee sometimes specific levels of performance it is required to provide un-shared resources to specific users (single-tenancy). In the case of the EOD pilot it is very important to have stable input transfer bandwidth.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.F.0360 ENTICE shall allow the creation of customizable network topology.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.F.0370 ENTICE shall provide both IPv4 and IPV6 protocols.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.F.0380 ENTICE shall allow the automatic creation of VM templates.

Rationale: This requirement is aligned with EOD Problem 5: Automated preparation and deployment, and with the ENTICE objective 5: To create an information infrastructure for strategic and dynamic reasoning.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.F.0390 ENTICE shall allow the automatic deployment of VMIs.

Rationale: This requirement is aligned with EOD Problem 5: Automated preparation and deployment, and with the ENTICE objective 5: To create an information infrastructure for strategic and dynamic reasoning (Section 2.10.1).

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.F.0400 ENTICE shall offer VMs with configurable storage.

Rationale: The configurability of the storage provides elasticity and optimization to the EOD system. This requirement is aligned with EOD Problem 4: Inelastic resource provisioning and ENTICE Objective 4: Elastic resource provisioning (Section 2.10.1).

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.F.0410 ENTICE shall allow the attachment of configurable storage volumes to VMs.

Rationale: The configurability of the storage and the possibility to be attached to a computing instance provides elasticity and optimization to the EOD system. This requirement is aligned with EOD Problem 4: Inelastic resource provisioning and ENTICE Objective 4: Elastic resource provisioning (Section 2.10.1).

Verification method: Test

Test type: Unit test

2.12.1.5 Evaluation metric

Not Applicable.

2.12.2 Non Functional requirements

2.12.2.1 Performance

ENT.REQ.EOD.NF.0100 The time from the ingestion of raw data in the cloud infrastructure of the EOD pilot to the end of the cataloguing of an L1C image product shall be lower than 1 hour and 12 minutes for a Deimos-2 typical image of 8 sectors**,** which is the current value of this metric for the original system.

Rationale: This requirement provides a metric to measure the time to user of an L1C imagery product.

Verification method: Test

Test type: System test

ENT.REQ.EOD.NF.0110 In the EOD pilot the work load of a Processing Chain shall be higher than 95% in all processing stages.

Rationale: The processing stages are the ones defined in ENT.REQ.EOD.F.0110. This requirement is aligned with EOD Problem 5: Automated preparation and deployment; and with the ENTICE Objective 5: To create an information infrastructure for strategic and dynamic reasoning.

Verification method: Test

Test type: System test

ENT.REQ.EOD.NF.0120 The response time of the EOD system interface has to be lower than 1 second.

Rationale: The EOD system response highly depends on the system calls and the transfer of information between the modules of the architecture. It is necessary that the transfer of information is done as fast as possible to obtain a high performance computing solution. This requirement is aligned with EOD Problem 5: Automated preparation and deployment; and with the ENTICE Objective 5: To create an information infrastructure for strategic and dynamic reasoning.

Verification method: Test

Test type: System test

ENT.REQ.EOD.NF.0130 ENTICE shall guarantee that the VMI delivery time is less than 126 s, in average, for the EOD pilot.

Rationale: The previous value is the current value of this metric for the original system. The delivery of the VMIs has to be fast in order to make an optimized use of the system elasticity. This requirement is aligned with EOD Problem 4: Inelastic resource provisioning, Problem 5: Automated preparation and deployment; and with the ENTICE Objective 4: Elastic resource provisioning and Objective 5: Information infrastructure for strategic and dynamic reasoning.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.NF.0140 VM deployment time in the ENTICE environment shall be less than 334 s, in average, for the EOD pilot.

Rationale: The previous value is the current value of this metric for the original system. The deployment of the VMIs has to be fast in order to make an optimized use of the system elasticity. This requirement is aligned with EOD Problem 4: Inelastic resource provisioning, Problem 5: Automated preparation and deployment; and with the ENTICE Objective 4: Elastic resource provisioning and Objective 5: Information infrastructure for strategic and dynamic reasoning.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.NF.0150 VM preparation time in the ENTICE environment shall be less than 450 s, in average, for the EOD pilot.

Rationale: The previous value is the current value of this metric for the original system. The preparation time of the VMIs has to be fast in order to make an optimized use of the system

elasticity. This requirement is aligned with EOD Problem 4: Inelastic resource provisioning, Problem 5: Automated preparation and deployment; and with the ENTICE Objective 4: Elastic resource provisioning and Objective 5: Information infrastructure for strategic and dynamic reasoning (see Section 2.10.1).

Verification method: Test

Test type: Unit test

2.12.2.2 Storage

ENT.REQ.EOD.NF.0160 The VM repository size in ENTICE has to improve at least 80% for all the EOD pilot scenarios defined.

Rationale: The repository size for each of the EOD scenarios is defined in the following table:

Metrics	Units	Sc1	Sc2a	Sc2b	Sc2c
Repository size	GB	2512	7729	5217	4251

It has to improve to the following values:

Metrics	Units	Sc1	Sc2a	Sc2b	Sc2c	Improvement
Repository size	GB	502	1546	1043	850	80%

Notice that the repository will only include the VMI templates to create the VM afterwards. This requirement is aligned with EOD Problem 2: Monolithic VM images, and with ENTICE's objective 2: Distributed lightweight VM images storage (see Section 2.10.1).

Verification method: Test

Test type: System test

ENT.REQ.EOD.NF.0170 The VM storage size in ENTICE has to improve at least 25% for all the EOD pilot scenarios defined.

Rationale: The VM size for each of the EOD scenarios is defined in the following table:

Metrics	Units	Sc1	Sc2a	Sc2b	Sc2c
VM storage size	GB	1000	3077	2077	1692

It has to improve to the following values:

Metrics	Units	Sc1	Sc2a	Sc2b	Sc2c	Improvement
Repository size	GB	750	2308	1558	1269	25%

This requirement is aligned with EOD Problem 2: Monolithic VM images, and with ENTICE's objective 2: Distributed lightweight VM images storage (see Section 2.10.1).

Verification method: Test

Test type: System test

ENT.REQ.EOD.NF.0180 The VMI size in ENTICE has to improve at least 60% for all the EOD pilot scenarios defined.

Rationale: The VMI size for each of the EOD scenarios is defined in the following table:

Metrics	Units	Sc1	Sc2a	Sc2b	Sc2c
VMI size	GB	111	111	111	111

It has to improve to the following values:

Metrics	Units	Sc1	Sc2a	Sc2b	Sc2c	Improvement
VMI size	GB	44	44	44	44	60%

This requirement is aligned with EOD Problem 2: Monolithic VM images, and with ENTICE's objective 2: Distributed lightweight VM images storage (see Section 2.10.1).

Verification method: Test

Test type: System test

2.12.2.3 Cost model

Not applicable.

2.12.2.4 Technology

ENT.REQ.EOD.NF.0190 The ENTICE environment shall offer at least VMs with 1, 2, 4 and 8 cores.

Rationale: The previous values are supported by the original system.

Verification method: Inspection

Test type: Unit test

ENT.REQ.EOD.NF.0200 The ENTICE environment shall offer at least 10, 20, 40 and 80 GB selectable system storage in the computing instances.

Rationale: The previous values are supported by the original system.

Verification method: Inspection

Test type: Unit test

2.12.2.5 Privacy

ENT.REQ.EOD.NF.0210 The ENTICE environment shall ensure confidentiality to the EOD pilot.

Rationale: Understand by confidentiality to prevent the disclosure of information to unauthorized individuals or systems.

Verification method: Test

Test type: System test

2.12.2.6 Compliance

ENT.REQ.EOD.NF.0220 The ENTICE environment shall comply with the data protection laws of the country in which the data is located.

Verification method: Inspection

Test type: System test

2.12.2.7 Security

ENT.REQ.EOD.NF.0230 The EOD system shall provide data integrity, maintaining and assuring the accuracy and consistency of data over its entire lifecycle.

Verification method: Test

Test type: System test

ENT.REQ.EOD.NF.0240 The EOD system shall ensure authenticity.

Rationale: Understand by data authenticity that data, transactions and communications are genuine.

Verification method: Inspection

Test type: System test

ENT.REQ.EOD.NF.0250 ENTICE shall allow the user to select the location of the VMs provided by the IaaS provider (see Section 2.10).

Rationale: For the EOD pilot it is important that the location of the VMs and the hardware that supports it is selectable by the EOD system developers. In some specific applications a specific location has to be used, for example for applications for the Spanish Government.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.NF.0260 ENTICE shall allow the encryption of all the information transferred in the EOD pilot.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.NF.0270 The EOD pilot code shall be obfuscated.

Verification method: Test

Test type: Unit test

2.12.2.8 Reliability

ENT.REQ.EOD.NF.0280 The ENTICE environment shall guarantee the EOD pilot the ability to perform all the defined functionalities over the system lifetime.

Verification method: Demonstration

Test type: System test

2.12.2.9 Availability

ENT.REQ.EOD.NF.0290 The ENTICE environment shall guarantee EOD system availability at least 99.999% over the system lifetime.

Rationale: Availability is defined as the fraction of time the system has full functionality over a defined interval.

Verification method: Demonstration

Test type: System test

ENT.REQ.EOD.NF.0300 The ENTICE environment shall guarantee the EOD to restore functionality in less than 5 minutes.

Verification method: Test

Test type: System test

2.12.2.10 Maintainability

ENT.REQ.EOD.NF.0310 The EOD system shall ensure that future enhancements can be easily and quickly implemented.

Verification method: Demonstration

Test type: System test

ENT.REQ.EOD.NF.0320 The EOD system shall remain operational during all planned maintenance activities.

Verification method: Test

Test type: System test

2.12.2.11 Portability

ENT.REQ.EOD.NF.0330 The EOD system shall have deployment and runtime independence from the IaaS provider.

Rationale: This requirement is aligned with EOD Problem 5: Automated preparation and deployment, and with the ENTICE objective 5: To create an information infrastructure for strategic and dynamic reasoning.

Verification method: Test

Test type: System test

2.12.3 Interface requirements

2.12.3.1 Hardware

ENT.REQ.EOD.I.0100 ENTICE shall offer x86_64 Intel architecture to deploy the EOD pilot.

Verification method: Test

Test type: Unit test

2.12.3.2 Software

Not Applicable

2.12.3.3 Communications

ENT.REQ.EOD.I.0110 ENTICE shall provide (File Transfer Protocol) FTP communications protocol.

Rationale: The EOD pilot will communicate with the antennas through FTP to transfer data from the antennas to the EOD pilot computed in cloud.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.I.0120 ENTICE shall provide (HyperText Transfer Protocol) HTTP communications.

Rationale: The EOD pilot will communicate with end users through HTTP communications protocol.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.I.0130 ENTICE shall provide (Secure File Transfer Protocol) SFTP communications protocol.

Rationale: The EOD pilot will communicate with the antennas through SFTP to transfer data from the antennas to the EOD pilot computed in cloud.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.I.0140 ENTICE shall provide https communications.

Rationale: The EOD pilot will communicate with end users through https communications protocol.

Verification method: Test

Test type: Unit test

ENT.REQ.EOD.I.0150 ENTICE shall guarantee that the EOD's input bandwidth covers the supply of imagery data.

Verification method: Test

Test type: System test

ENT.REQ.EOD.I.0160 ENTICE shall guarantee that the EOD's output bandwidth covers the demand of imagery products in their distribution.

Verification method: Test

Test type: System test

ENT.REQ.EOD.I.0170 ENTICE shall guarantee that the communications between VMs in the EOD pilot do not present overhead in the system performance.

Verification method: Test

Test type: System test

3 FCO Pilot Case

3.1 Introduction

Image creation and storage is still a massive challenge for Cloud providers. As new multi cloud technologies arise more and more, images need to be pulled and pushed from one location to another with a greater focus on speed of transfer, image size and storage requirements.

To combat this need, the ENTICE environment will consist of a ubiquitous repository-based technology which provides optimised Virtual Machine image creation, assembly, migration and storage. The Image Repo pilot case will be implemented in the ENTICE environment to test, validate and demonstrate that ENTICE can support and improve the performance image transfer and storage with the cloud.

The Image Repo pilot case mainly consists of the implementation in cloud of Flexiant Cloud Orchestrator (FCO). It is currently operational within a FLEX Testbed hosted in Edinburgh, Scotland.

Within the ENTICE project, FLEX intends to implement the system in a cloud computing infrastructure to provide FLEX with smaller Images and faster deployment times. The ENTICE environment will be used to optimize the implementation and performance of images within the Image Repo pilot case.

3.2 Global Cloud context analysis

It is recognised that there are three different types of cloud models under the cloud stack, namely Infrastructure as a service (IaaS), Platform as a service (PaaS) and Software as a service (SaaS). ENTICE will tap into all of these opportunities. The growth of cloud computing as a worldwide phenomenon is extraordinary. Cloud computing promises unprecedented agility, flexibility and scalability together with cost-saving. According to market analysis by Gartner, worldwide spending on public cloud services will grow to Euro €179 billion in 2017 from today's market value of Euro €100 billion. IaaS will grow from Euro €4.4 bilion in 2012 to Euro €17.6 bilion in 2016, whilst the SaaS market will grow from Euro €10.3 billion in 2012 to €19.1 in 2016. During the same period spending on PaaS is to grow to €2.13 billion in 2016.

For the purpose of ENTICE we would focus on the overall relevant cloud business models – IaaS, PaaS and SaaS – as it is expected to have a great impact over the overall cloud modelling. ENTICE has a clear relevance to IaaS providers and could potentially be part of an interoperable ecosystem of clouds where images are easily optimised, traded and deployed. Indirectly, SaaS providers would benefit from the results so this group of providers as well will appear as potential relevance market objective. The relevant market details will be identified and

addressed as the project develops based on results and market research from the different use cases that will be implemented.

3.3 Use case analysis in the context of ENTICE

The Image Repo Cloud architecture is based on the following components: (i) Flexiant Cloud orchestrator and (ii) Ushare Repository;
(i) A front-end layer for end users to orchestrate their own cloud environment, which also manages the modules of the whole architecture.

With ENTICE, we expect to optimize the storage of images and deployment of virtual machines, which will reduce the overall storage required, transfer time of images and deployment time of a new VM with the images.

3.4 Impact

3.4.1 Business and technological impact

The Image Repo pilot case in ENTICE will contribute to provide smaller and more portable images for use in a multi cloud environment. It will test if the lightweight images can be deployed onto a cloud platform from different locations and improve performance. This will provide viable solutions for the difficulties faced by Cloud providers and improving the quality of the services that are currently offered.

The Image repo pilot will also be used to analyse the performance of cloud computing with regard to image service times and deployment times. The implementation and demonstration scenario within this pilot will show performances in a real cloud implementation and offering.

The impact that ENTICE will have in the Image repo pilot is the following:

- Costs reduction:
  - Reduced storage requirements of VMs and images.
  - Reduced deployment time
  - Reduced runtime of the overall system
- Performance increase:
  - Increase deployment speed of VMs
  - Reduced transfer time of images.
- Vender locking
  - Avoid vender lock-in with image transfer

3.4.2 Impact in ENTICE

Flex will provide to ENTICE environment, functional and non-functional requirements to implement industry driven requirements for image use with cloud.

Flex will provide information to improve the transfer speed of images and improve the deployment time of Virtual Machines and reduce the storage require for images and as a result a cost saving for the cloud operator.

Through the Image repo pilot, ENTICE will be tested within in a commercial cloud offering and the benefits detailed for all cloud providers to use.

The Image Repo pilot will provide ENTICE a use case to deploy smaller images within its cloud platform showcasing faster transfer speeds and a reduced storage requirement as well as investigating the performance gains this has on VM deployment times.

In case of Image repo pilot case, the main aspects that ENTICE has to optimize are: i) the size that each image needs to be stored; ii) the transfer time of images; iii) the deployment time that VMs need to be fully operative; iv) the possibility to avoid vender lock-in with images that are easily transferable.

3.4.3 Social impact

The 'image repo' pilot is linked to the Europe 2020 agenda and how it supports research and innovation in EU organizations to ensure the ENTICE image repo pilot ties in to European priorities. These initiatives are not part of the H2020 Work Programme but contain essential knowledge of EU strategy and goals essential for mapping to H2020. Europe 2020 is the EU growth strategy for the next 10 years.

The use case links to several initiatives that map to involvement in H2020 and realize we must tap into the full potential of the Single Market, its 500 million consumers and its 20 million entrepreneurs as identified in the Europe 2020 industrial policy.

We recognize that as entrepreneurial SMEs, tapping into valuable European research, allows us to benefit from expertise from others as well as sharing our own to help us continue with our technological innovation and tie well to the commission's initiative on Agenda for New Skills and Jobs to help the EU reach its employment target for 2020 which is the ultimate goal of the Horizon 2020 collaborative projects and SMEs who are innovating and differentiating in their sector with a well-organized plan for collaborating with relevant research specialists can benefit from the focus on creating highly skilled employment opportunities.

In a nutshell, the ENTICE image repo pilot in a Europe 2020 context is about delivering smart, sustainable and inclusive growth. It is hoped that through more effective investments in education, research and innovation Europe 2020 research can be considered "smart". ENTICE welcomes the initiative and the need for H2020 project output to make a tangible difference.

Our continuing plans to link research as closely as possible to the needs of the marketplace and service providers is a great example of this in action and we believe the opportunities that can be realized from ENTICE can have an impact on the EU employment and R&D targets.

3.5 Description

With ENTICE, we expect to optimize the creation, storage and deployment of virtual machines with smaller lightweight images. This allows a reduction in the required storage to host the different virtual machines and images.

3.5.1 Expected results

By using the ENTICE environment, the expected results of the implementation of the Flex use case are the following:

- To reduce the delivery and deployment time of the virtual machines and images.
- Decrease the total storage consumption.
- Portability of images to avoid vender lock-in

3.6 Architecture

The architecture of the Image Repo pilot case is shown in Figure 9 and made from the following components:

- *UshareSoft*: The UShareSoft Image repo is used by customers of FLEX to access a pre-configured range of images for use within their own environment. These images can be downloaded locally or pulled into the target cloud platform.
- *Flexiant Cloud Orchestrator:* Flexiant Cloud Orchestrator is a feature-rich software platform for cloud management. This allows management or "Orchestration" of a cloud infrastructure with a front end UI and access with an API's to perform actions such as creating and managing VMs and migrating images into the platform.
- *Image Storage:* Within FCO backend storage is used to storage the images used within the platform.

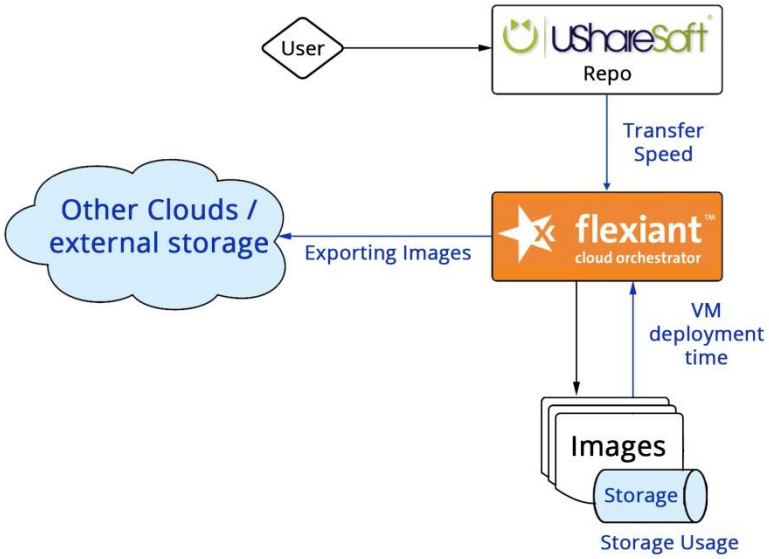

Figure 9: Flex architecture

The blue highlighted areas are sections that the ENTICE project will improve in our cloud offering.

3.7 Components of the pilot

3.7.1 FCO

Flexiant Cloud Orchestrator is a world-leading Cloud Orchestration Software solution. It enables service providers and enterprises to design, create and manage their own virtual public, private or hybrid cloud solutions.

Flexiant Cloud Orchestrator can manage the entire cloud solution, from hardware, network and storage management through to metering, billing & customer/end-user self-service. It can also be used to augment and orchestrate existing platforms such as VMware vSphere

A graphical outline of all the areas that Flexiant Cloud Orchestrator covers is shown within Figure 10 and the GUI is shown within Figure 11.

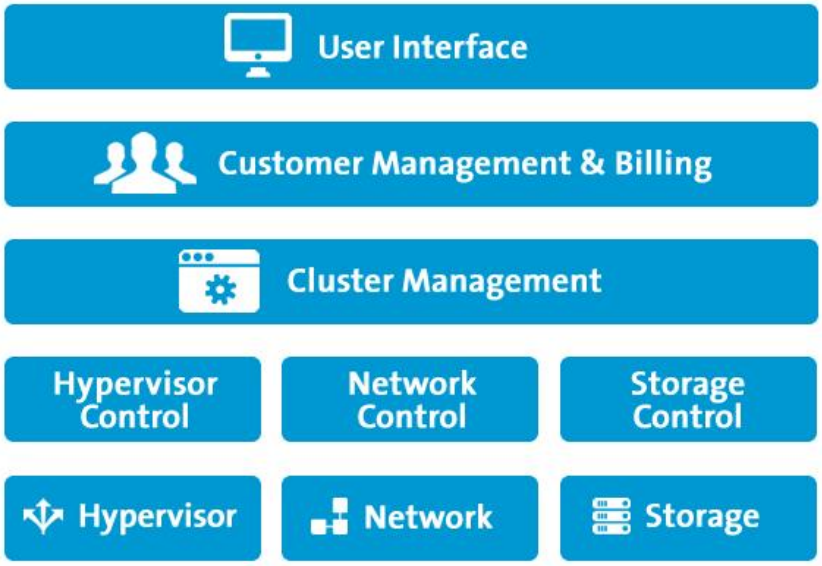

Figure 10: FCO overview

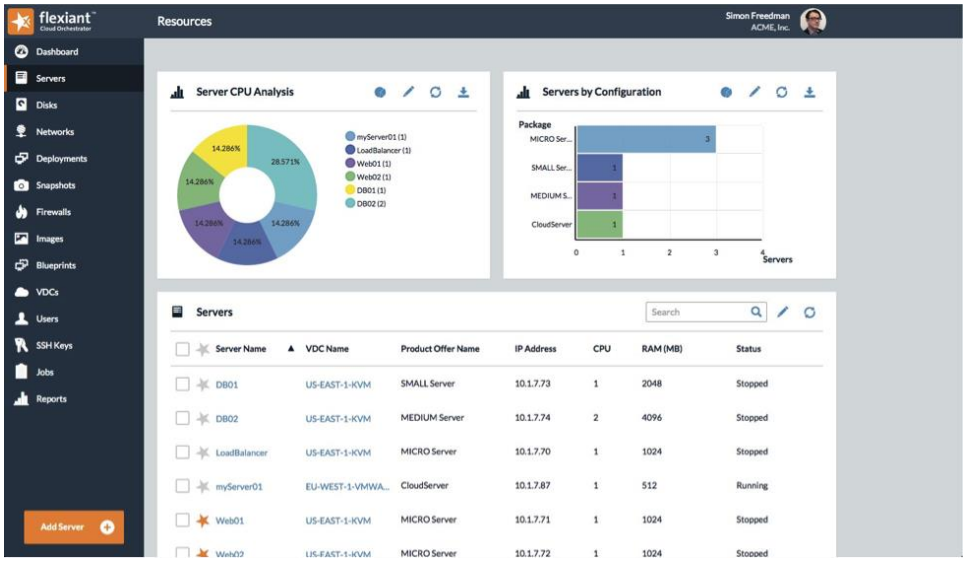

Figure 11: FCO GUI

3.7.2 Ushare

UShareSoft enables fast, easy, self-service on boarding of software by cloud users. A cloud server template that is independent of any hypervisor or IaaS platform. UForge then automatically generates and publishes a final machine image to the hypervisor or cloud of choice, in this case, Flexiant Cloud Orchestrator, ready for provisioning as a live VM. UForge also fully automates the full software delivery process, from build and generation to publish and maintenance. UShareSoft provides a number of benefits such as:

- The ability for users to create bespoke simple and complex software solutions with self-service provisioning to cloud;
- Migration services which extract information from VMs running in another environment, ready for instant repackaging to the provider's cloud;
- A cloud software marketplace of "ready-to-run" applications where developers and end users can create, share, clone, buy and customize cloud server templates; and
- The injection of "managed services" on-demand when a user purchases a cloud server template.

The GUI of the Ushare software is shown in Figure 11. The Ushare's repository can be seen in Figure 12.

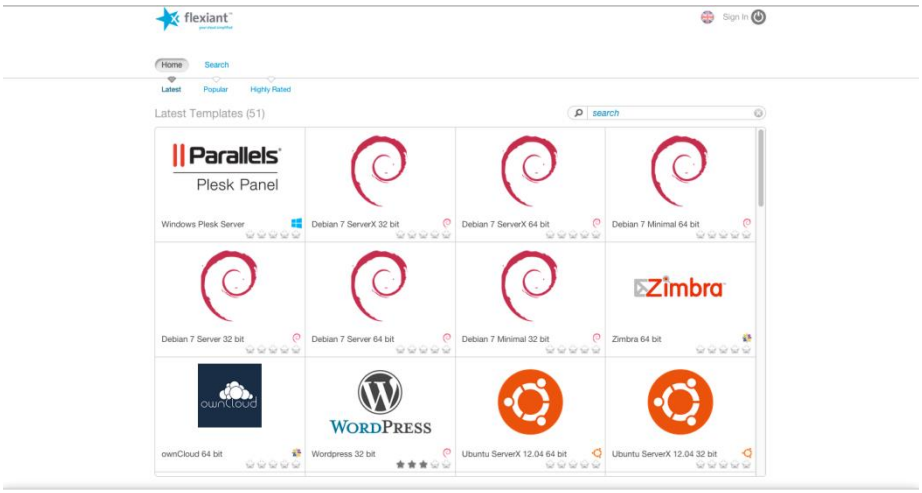

Figure 12: Ushare's repository

3.7.3 FlexiOps testbed architecture

The architecture of the current FlexiOps testbed is detailed in the following section and a full break down of the current testbed is detailed within Figure 13, Figure 14 and Figure 15.

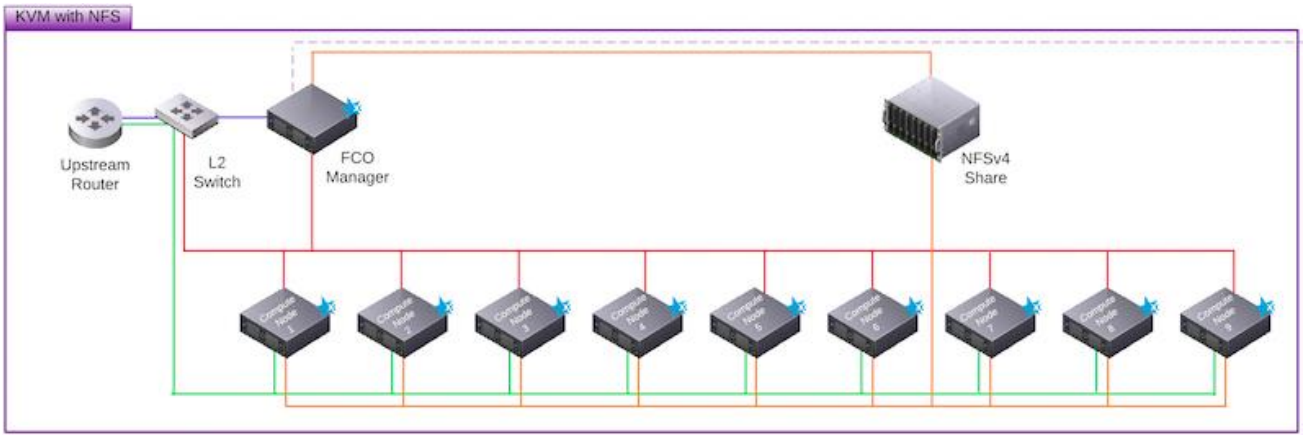

Figure 13: Flex Testbed Cluster 1

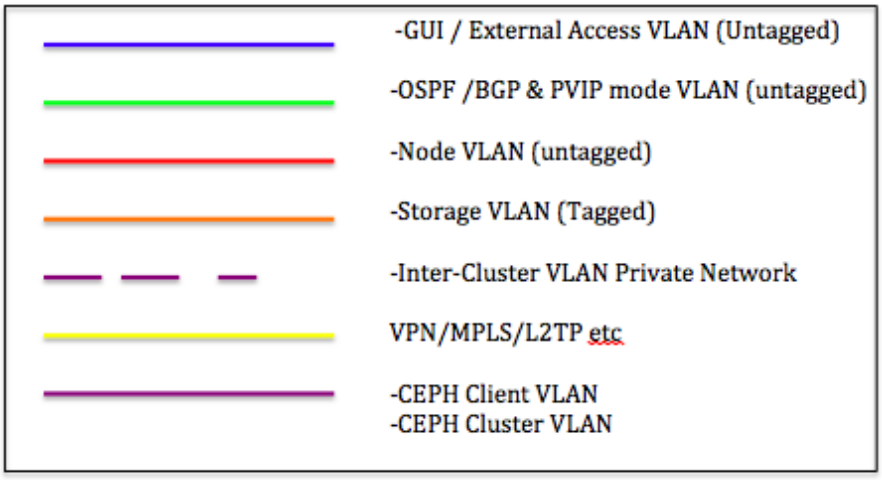

Figure 14: Key of different types of networks used in FCO

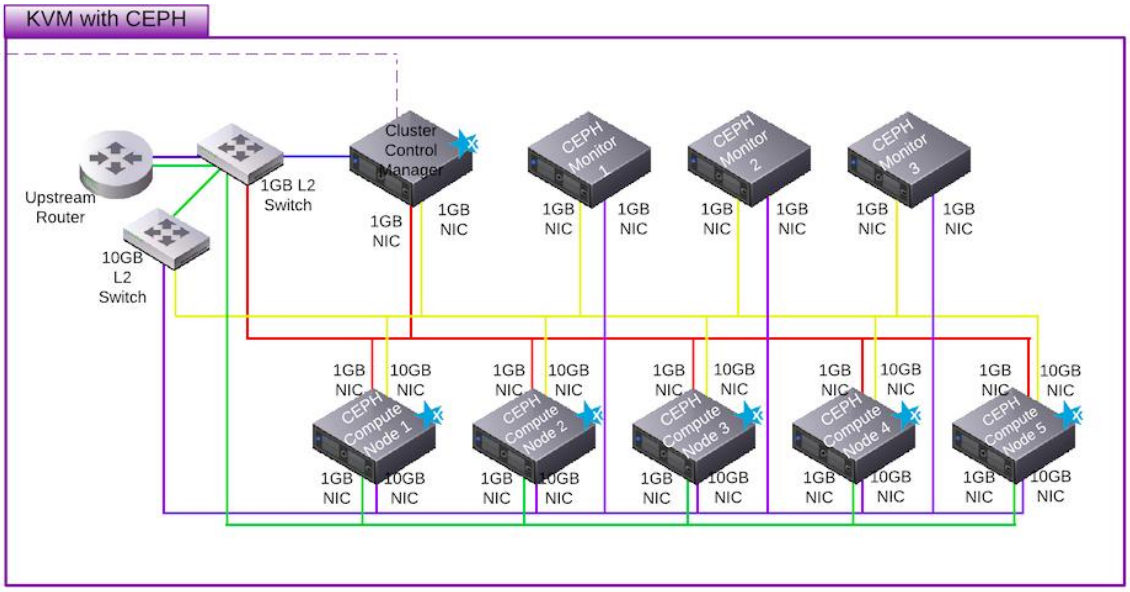

Figure 15: FlexiOps Testbed Cluster 2

Table 14: FlexiOps overview

	1 * FCO Manager	**Router Node**	**9 * Compute Node**
Cluster 1	16GB RAM 2*Quad Core CPUs 2*1GB/s NICs 2*500GB in RAID 1	8GB RAM 1*Quad Core CPU 2*1GB/s NICs with PXE capability	64GB RAM 2*6 Core CPUs 2*1GB/s NICs with PXE capability No local Storage NFSv4 Share 16TB Storage
Cluster 2	16GB RAM 2*Quad Core CPU 2*1TB (fast) Disks 2*1GB NIC 1*10GB NIC	32GB RAM 16 Core CPU 12*2TB Disk 2*1GB NIC with PXE capability 2*10GB NIC with PXE capability	64GB RAM 2*6 Core CPUs 2*1GB/s NICs with PXE capability No local Storage NFSv4 Share 16TB Storage

FLEX provides excellent storage capabilities to ENTICE (in cluster 2) using CEPH storage which solves many recent cloud storages related issues such as load balancing, CEPH uses adaptive load balancing whereby frequently accessed objects are replicated over more nodes.

There are three main challenges a service provider building a cloud infrastructure must overcome:

- To be able to offer continuous availability to their end users even when faults occur to maintain quality of service and customer satisfaction.
- They need to ensure that their infrastructure is a help, not a hindrance to growing their cloud business over the course of time.
- They need to be able to achieve this while working under tight cost constraints.

Storage is well-known to be one of the major, if not the main infrastructure challenge and, for many years, service providers could choose only between two options: either local or centralized network storage.

Local Storage – With local storage, i.e. disks with the server node, the only way to ensure continuity of service was to replicate all data to another device that could be used in the event of a failure. This was not always possible for all types of workloads and most of legacy applications do not really cope with replication of data to different devices. Besides, local storage gave very little flexibility for service providers to manage their storage space.

Centralized Storage – If cloud service providers opted for the centralized network storage option, e.g. a Storage Area Network or a Network Attached Storage, this gave them the flexibility and control required to deliver storage services. However, resiliency was delivered through expensive hardware components, resulting in significant investments in CAPEX and OPEX that had a huge impact on budgets.

CEPH's distributed storage architecture enables service providers to overcome these issues. Because the data is partitioned, distributed and replicated across multiple nodes, service providers can save costs by only needing to use commodity hardware, with no single point of failure in the cluster. In addition, CEPH's self-managing and self-healing architecture means that the service provider can make savings on their operational overhead. Furthermore, they can quickly and seamlessly add storage capacity, scaling to petabytes if needed, and present distributed storage as one logical resource when required.

Driven by the same principles, Flexiant Cloud Orchestrator offers similar benefits around cost-effective resiliency and scalability at its core. By making use of the integrated solution between Flexiant Cloud Orchestrator and CEPH, service providers can base their cloud service on an

infrastructure that reflects these benefits both in terms of cloud orchestration and storage architecture.

CEPH cluster is deployed by FCO on top of the nodes running virtual machines in Cluster 2 of the testbed. Therefore, a service provider can share the same nodes to deliver both compute and storage as-a-service, reducing the number of components required and delivering full, modern “scale-out” architecture. Despite the automation introduced by FLEX for deploying Ceph in minutes, the service provider still retains the full ability to operate and configure Ceph to suit best the type of workload required.

Flexiant Cloud Orchestrator supports both Public Virtual IP (PVIP) and Virtual Local Area Network (VLAN) based networking modes. Flexiant Cloud Orchestrator also supports 802.1q & 802.1q-in-q/802.1af VLAN modes, as well as emulated technologies such as VMware's distributed switch.

The testbed provided is split across two clusters as detailed in 5 – 7. Each cluster consists of 9 compute nodes.

3.7.3.1 VLANs

Traffic is split across several logical VLANs as follows:

- GUI/External Access VLAN (Untagged)
- OSPF/BGP & PVIP mode VLAN (Untagged)
- Node VLAN (Untagged)
- Storage VLAN (Tagged)
- Inter-Cluster VLAN Private Network (VPN/MPLS/L2TP)
- CEPH Client VLAN
- CEPH Cluster VLAN

Within FCO we use a verity of tagged and untagged configurations. A "tagged" packet contains the VLAN information in the Ethernet frame while an "untagged" packet doesn't.

In addition, there is a network that is referred to as the 'External network'; this gives Flexiant Cloud Orchestrator external network access. It is in not connected to any part of Flexiant Cloud Orchestrator. Rather it provides connectivity to the DMZ (DeMitarized Zone) router (the other side of the relevant firewall) and the upstream router(s) (which are not part of Flexiant Cloud Orchestrator).

The external Access VLAN connects to all nodes within a cluster. This ensures access to all available nodes externally or via the GUI.

The PVIP network: This carries traffic between the compute nodes running PVIP and the upstream router, and between the router nodes and the upstream router.

The node VLAN: This is on a private IP range and is used to communicate between with the nodes (other than VMware compute nodes) and the management stack.

The storage network is used to move storage traffic between the compute nodes and the SAN, or (for image and disk fetch operations) the management servers and the SAN.

Within cluster 2 CEPH's distributed storage architecture is used. As the data is partitioned, distributed and replicated across multiple nodes, with no single point of failure in the cluster. Furthermore, storage capacity can be quickly and seamlessly added, scaling to petabytes if needed.

3.7.3.2 KVM

On KVM each node is a diskless server. Each node will be attached to 3 networks (for the node network, the public network and the storage network). These networks can be shared between NICs. The public network will not share a port with anything else, as this will increase the chance that a denial of service attack can take node out of service; in practice this produces a minimum requirement of two NICs (one for the Public network, and one for the Node Network.

3.8 Markets and users

Flexiant Cloud Orchestrator is a mature (now it's 5th version), reliable, feature-rich software platform for cloud management. As such, FCO's core users are Cloud Service Providers, providing them with a simple tool to build, launch and grow their cloud Infrastructure as a Service business

FCO allows service providers to generate more revenue and accelerate growth, compete more effectively and lead the market through innovation. Vendor agnostic and supporting multiple hypervisors, FCO offers a customisable platform, a flexible interface, integrated metering and billing, resell capabilities and application management.

3.9 Problems

Nowadays the implementation of the FCO pilot case in the actual cloud platforms there are several problems. By using the ENTICE environment, these problems are depicted in Table 15.

Table 15: FCO pilot case problems

Problem	Objective
Deployment time of VMs	The time for deploying the customized virtual machines shall be lower in order to the dynamically creation of virtual machines would be benefit.
FCO is limited to fetching disc/image	To provide a functionality to export

functionality and does provide means to export images	images from out of the FCO platform for use within other clouds and avoid vender lock-in
FCO suffers from high image storage consumption	New compression methods of stored images could have significant space/cost savings
FCO has limited support of different image types that can be fetched and converted	FCO to support new image types imported into the platform.

Problem 1: Deployment time of VMs

The deployment time of a VM is a key issue that will be addressed. FLEX will look to lowering the time taken to deploy a VM using the ENTICE framework and improve image deployment time by 33%. This will allow our customers to deploy faster and resolve issues/ meet demand quicker.

Problem 2: FCO export images

The ability to export images from FCO is a key limitation. This possibility of using the ENTICE framework to allow the exporting of images from the FLEX platform will, be investigated. This ability will be used to avoid vender lock-in and allow a multi cloud approach.

Problem 3: Image storage

As more and more images are used within Cloud platform, storage will become an issue. To combat this FLEX will use new image compression methods to reduce this overhead. This will allow our own customers to reduce storage requirements and costs, estimated to be 10% of total storage used.

Problem 4: Image types

FlexiOps will look to supporting new and improved images types within its cloud platform. With newly created image types within the ENTICE project, FLEX will look to supporting these versions within future versions of FCO. Table 16 gives some metrics figures for our current scenario (Sc1) within FCO.

Table 16: Current FCO metrics

Problem	Metrics	Units	Sc1

#1 - Deployment time of VMs	Image size	MB	228
	Image delivery time	sec	180
#2 - FCO export images	Image Size	MB	N/A
#3 - Image storage	VM storage	GB	2000
#4 - Image types	Image type	Types	.qcow2 .raw .ova .img .iso vmdk

Table 17 details the improvement that will be made with the use of the ENTICE framework for the FCO pilot.

Table 17: ENTICE environment metrics

Problem	**Metrics**	**Units**	**Sc1**	**Improvement**
#1 - Deployment time of VMs	Image size	MB	114	50%
	Image delivery time	sec	120	33%
#2 - FCO export images	Image Size	MB	228	100%
#3 - Image storage	VM storage	GB	1800	10%
#4 - Image types	Image type	Types	qcow2 .raw .ova .img .iso vmdk .Entice image format	.Entice image format supported

3.10 Metrics

This document defines the metrics that will be used to validate the FCO pilot (see Table 18) and the metrics that the ENTICE environment (see Table 19) has to provide to validate it. Those metrics are not intended to be actual requirements, but the parameters that will be measured during the testing and evaluation of the system when implemented by utilizing the ENTICE environment.

3.10.1 Metrics of the FCO pilot case

Table 18: FCO pilot case metrics

Resource	Metric	Component	Description
Network	NetworkIn	Network	The number of bytes received on all network interfaces by the instance. This metric identifies the volume of incoming network traffic to an application on a single instance. Units: Bytes
	NetworkOut	Network	The number of bytes sent out on all network interfaces by the instance. This metric identifies the volume of outgoing network traffic to an application on a single instance. Units: Bytes
Storage	Storage used	Storage	The amount of backend storage used to store the VM images within the target cloud platform.
Download Time	Time to complete download	Network	Time taken for image to be pulled into platform from an external source.
Authentication Time	Authentication time	Network	Time to authenticate to cloud platform when using API calls.
VM Provisioning times	VM provision time	Cloud provider	Time taken to provision new VM from image.
Image Size	Total Image size	Storage	Total size of image when compared to traditional non ENTICE image.
Config time of image applications	Time to for image with required applications to be created/deployed	Network	Configuration of images with required applications and placed into platform

3.11 Metrics of ENTICE environment

Table 19: ENTICE environment metrics

Resource	Metric	Description	Unit
Deployment time	Average Deployment time of VMs	The average deployment time of VMs indicates how time is required to instantiate a Virtual Machine.	Time/S

Size of VMs	Size of VMs	The size of VMs has to be as low as possible to reduce the deployment time and the recurrent costs	Time/S
Size of Image	Size of images used in platform.	The total size of images is import to me measured to ensure as little storage space is used.	MB
Bandwidth	Bandwidth	The bandwidth between the FCO VMs.	Mb
Latency	Latency	Latency between the VMs of the pilot case.	MS
Image creation time	Time for images to be created	Time for images to be created in FCO platform	Time/S

3.12 Requirements of FOC

The following sections list the requirements of the FCO pilot case for the ENTICE environment requirements.

The following section details the requirements of the FCO pilot case for the ENTICE project. These are broken down into three main groups: functional, non-functional and interface requirements.

3.12.1 Functional requirements

3.12.1.1 Behaviour

The FCO use case will look to using the ENTICE framework to allow faster image importing into FCO. This will be done by using smaller lightweight images that have been created by ENTICE. In addition the total storage will be reduced so the cloud provider and use require less storage for use images.

ENT.REQ.FCO.F.0100 Decrease of VM deployment time within FCO.

Rationale: Importing of lightweight images into the platform, that are smaller than standard image size. The ENTICE environment shall allow the FCO pilot to perform this behaviour. This comes from problem 1: Deployment time of VM problem 3: Image storage.

Verification method: Test

Test type: System testing

ENT.REQ.FCO.F.110 Reduction of storage required for images.

Rationale: Reduction in storage required for images. The new images when stored within FCO should be smaller than a non-ENTICE counterpart and should reduce the overall storage required. This comes from problem 3: Image storage.

Verification method: Test

Test type: System testing

ENT.REQ.FCO.F.0120 Reduction in transfer speed of images into FCO.

Rationale: Importing of light weight images into the platform. This requirement is aligned with the FCO Problem 2: FCO export images.

Verification method: Test

Test type: System testing

ENT.REQ.FCO.F.0140 Exporting of images from FCO.

Rationale: Exporting of images from FCO to external locations or additional cloud providers. This

Verification method: Test

Test type: System test

3.12.1.2 Capabilities

By the end of the ENTICE project, within FCO there should faster image transfer, reduced image storage, faster VM creation and the ability to transfer images between clouds. The ENTICE images must work with FCO and be imported using the FCO import functionality.

ENT.REQ.FCO.F.0150 The ENTICE environment shall provide the FCO pilot resources flexibility when required.

Rationale: The FCO pilot allows users to create VM, at varying sizes and configurations and change this as required. The ENTICE environment needs to ensure that this is supported. This requirement is aligned with the FCO Problem 1: Deployment time of VMs and with ENTICE's Objective 1: Lightweight VM image creation.

Verification method: Test

ENT.REQ.FCO.F.0160 Updated images for use within FCO system shall be automatically deployable without human intervention.

Rationale: The ENTICE environment should facilitate the automatic deployment of the FCO images required once they have been updated. These updated images should be faster to

deploy and as they located in the platform allow customers to quickly start new VMs. This requirement is aligned with the FCO Problem 1: Deployment time of VMs

Verification method: Demonstration

Test type: System test

ENT.REQ.FCO.F.0170 The ENTICE image repository shall be optimized and accusable by FCO pilot case.

Rationale: This requirement is aligned with the FCO, Problem 1 Deployment time of VMs Problem 3: Image Storage and Problem 4 Image types. It is closely related with ENTICE's Objective 3: To develop autonomous multi-objective repository optimisation.

Verification method: Test

Test type: System test

ENT.REQ.FCO.F.0180 The ENTICE Environment must support deployment between clusters that could be in different geographical locations.

Rationale: Within FCO hardware can be split into clusters. The ENTICE environment must understand this and ensure all images required are available for all clusters FCO system has to be portable. This requirement is aligned with the FCO Problem 1: Deployment time of VMs, Problem 2 FCO export images, Problem 3: Image storage and for instance, it is closely related with ENTICE's Objective 5: To create an information infrastructure for strategic and dynamic reasoning.

Verification method: Demonstration

Test type: Unit test

3.12.1.3 Knowledge model

Not applicable.

3.12.1.4 VMI images

ENT.REQ.FCO.F.0190 ENTICE shall allow the reservation of VMs in Debian, CentOS, Ubuntu and Windows.

Rationale: The FCO pilot requires a VM, to be deployed from a number of different OS

Verification method: Inspection

Test type: Unit test

ENT.REQ.FCO.F.0200 ENTICE shall allow the migration of FCO images.

Rationale: The migration of FCO images is a feature that will facilitate portability independently of the infrastructure used. It is aligned with FCO Problem 2: FCO export images and for instance with ENTICE's Objective 3: To develop autonomous multi-objective repository optimisation processes.

Verification method: Test

Test type: Unit test

ENT.REQ.FCO.F.0210 ENTICE shall allow the download of FCO's images.

Rationale: The download of FCOs image to local storage is a feature that will facilitate portability independently of the infrastructure used and will facilitate unitary tests. It is aligned with FCO Problem 2: FCO export images and Problem 3 Image storage, and for instance with ENTICE's Objective 3: To develop autonomous multi-objective repository optimisation processes.

Verification method: Test

Test type: Unit test

ENT.REQ.FCO.F.0220 ENTICE shall guarantee remote access to FCO VMs.

Verification method: Test

Test type: Unit test

ENT.REQ.FCO.F.0230 ENTICE shall allow support a verity of image types to be used in the cloud.

Rationale: FCO supports a verity of image types, to continue this ENTICE must also support this range of image types. It is aligned with FCO Problem 4 Image types.

Verification method: Test

Test type: Unit test

3.12.1.5 Evaluation metric

Not applicable.

3.12.2 Non Functional requirements

3.12.2.1 Performance

ENT.REQ.FCO.NF.0100 ENTICE shall guarantee that the VM delivery time is less than 240 s, on average for the FCO pilot.

Rationale: The delivery of the VMIs has to be fast in order in order for ENTICE to be useful from a business perspective. This requirement is aligned with FCO Problem 2: Deployment time of VM.

Verification method: Test

Test type: Unit test

ENT.REQ.FCO.NF.0120 Image import time in the ENTICE environment shall be less than 5m, in average, for the FCO pilot.

Rationale: The import time of images has to be fast in order to for ENTICE to be seen as useful for a customer and business sense. This requirement is aligned with FCO Problem 1: Deployment time of VM, Problem 3: Image Storage; and with the ENTICE Objective 4: Elastic resource provisioning.

Verification method: Test

Test type: Unit test

With this Pilot use case, we will see fast image transfer time into FCO. In addition with the use of smaller images there will be faster deployment time.

3.12.2.2 Storage

ENT.REQ.FCO.NF.0130 The VM storage size in ENTICE has to improve at least 25% for all the FCO pilot scenarios defined.

Rationale: The VM size for each of the FCO scenarios is needs to reduce the over image size and storage by at least 25% to be considered usable in a commercial deployment.

This requirement is aligned with FCO Problem 3: Image Storage and objective 2: Distributed lightweight VM images storage.

Verification method: Test

Test type: System test

3.12.2.3 Cost model

Cost for storage of images is set within the FCO level and will not be a consideration.

3.12.2.4 Technology

ENT.REQ.FCO.NF.0140 The ENTICE environment shall offer at least VMs with 1, 2, 4 and 8 cores.

Verification method: Inspection

Test type: Unit test

ENT.REQ.FCO.NF.0150 The ENTICE environment shall offer at a variable storage solution from 10 GB – 1TB selectable system storage in the computing instances.

Verification method: Inspection

Test type: Unit test

3.12.2.5 Privacy

ENT.REQ.FCO.NF.0160 The ENTICE environment shall ensure all data is secure and private for the FCO pilot.

Rationale: This is to prevent the disclosure of information to unauthorized individuals or systems.

Verification method: Test

Test type: System test

3.12.2.6 Compliance

ENT.REQ.FCO.NF.0170 The ENTICE environment shall comply with the data protection laws of the country in which the data is located.

Verification method: Inspection

Test type: System test

3.12.2.7 Security

ENT.REQ.FCO.NF.0180 The FCO system shall ensure authenticity by use of accounts or API Keys.

Rationale: to ensure that all access and actions genuine only authenticated users will be able to perform actions on the platform.

Verification method: Inspection

Test type: System test

3.12.2.8 Reliability

ENT.REQ.FCO.NF.0190 The ENTICE environment shall guarantee the FCO pilot the ability to perform all the functionalities over the lifetime of the ENTICE project and after.

Verification method: Demonstration

Test type: System test

3.12.2.9 Availability

ENT.REQ.FCO.NF.0200 The ENTICE environment shall guarantee FCO system availability at least 99.99% over the system lifetime.

Rationale: To ensure continue usage the ENTICE environment must have high availability.

Verification method: Demonstration

Test type: System test

ENT.REQ.FCO.NF.0210 The ENTICE environment shall guarantee the FCO to restore functionality in less than 5 minutes.

Verification method: Test

Test type: System test

3.12.2.10 Maintainability

ENT.REQ.FCO.NF.0220 The FCO system shall remain operational during all planned maintenance activities.

Verification method: Test

Test type: System test

3.12.2.11 Portability

ENT.REQ.FCO.NF.0230 FCO will remain independent from the ENTICE framework.

Rationale: To ensure no downtime, maintenance or future upgrade negatively affects FCO and its use, the ENTICE framework must be independent and will function independently from this.

Verification method: Test

Test type: System test

3.12.3 Interface requirements

3.12.3.1 Hardware

ENT.REQ.FCO.I.0100 ENTICE shall support all hardware within the FCO testbed.

Rationale: To ensure ENTICE will function will function with a verity of hardware, the ENTICE framework must work with the existing testbed provided by Flex as detailed within Table 14.

Verification method: Test

Test type: Unit test

3.12.3.2 Software

ENT.REQ.FCO.I.0120 ENTICE shall ensure images are full functional with a verity of hypervisors.

Rationale: FCO supports a number of different hypervisors, for ENTICE to be used so must ENTICE. These include KVM, VMware and Xen4.

Verification method: Test

Test type: Unit test

3.12.3.3 Communications

Not applicable.

4 CSP Pilot Case

4.1 Introduction

4.1.1 Global context analysis

The Cloud Service Provider (CSP) pilot case (see architecture in Figure 16) attempts to create a viable solution for both optimizing the operational and infrastructure resources used for providing proprietary and third party SaaS to their customers.

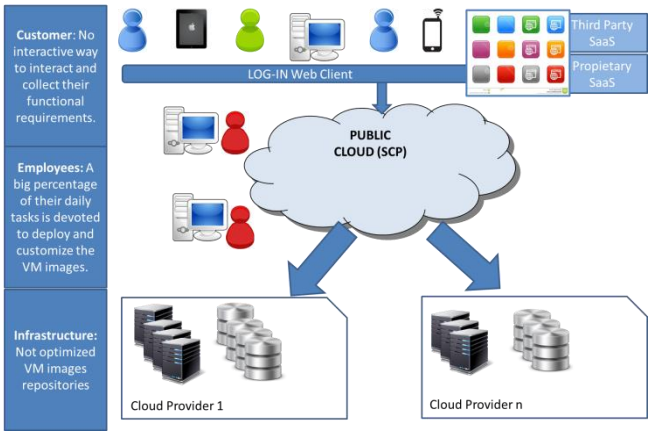

Figure 16: Cloud Provider architecture

4.2 Use case analysis in the context of ENTICE

The CSP pilot consists of the implementation of different services that WT offers to their customers, hence providing a clear perspective as how any CSP could benefit from the use of the ENTICE environment.

For example, in their daily work the cloud service provider deals with some of the major tasks that requires totally dedicated resources for the provisioning of services for the customers:

- Manual VM image creation. The time for deploying the customized virtual machines require gathering information from the costumer and time consumption for its creation.
- Service interruption to scale applications. Applications need to be manually scaled up or down based on the user load. Shutting down, manually assigning additional resources and then powering up a VM again consumes time and extra cloud resources.
- Proprietary un-optimised VM repositories. Cloud providers are highly interested in attracting new customers from other providers. Unfortunately, current users must be familiar with the actual repository interfaces in order to use them, which is an unsurpassable barrier in deploying new images and exploiting the resources of another cloud provider. While VM image format converters do exist, they are ad-hoc solutions that do not preserve the performance and optimizations.

By using ENTICE environment, it is expected not only to highly decrease the deployment and customization of VMIs but also to increase the quality of the service offered to their customers by offering a faster and elastic service. Thanks to the ENTICE environment CSPs will be able to offer more competitive services, reduce operational costs and avoid vendor-locking.

So finally, the use case aims to:

- Optimizing Virtual Machine Images (VMIs) creation and storage.
- Optimizing VMIs configuration procedures.
- Optimizing deployment of VMs (faster deployment).
- Eliminate dependencies with other cloud providers.
- Improved elasticity for on-demand scaling.

4.3 Impact

The ENTICE environment will provide a feasible solution to optimize the whole ecosystem of a CSP. By using ENTICE environment, it is expected not only to highly decrease the deployment and customization of VMIs but also to increase the quality of the service offered to their customers by offering a faster and elastic service. Thanks to the ENTICE environment CSPs will be able to offer more competitive services, reduce operational costs and avoid vendor-locking.

4.4 Description

The main services to be tested within the use case:

- Proprietary Software
    - Unified Communications open source framework for enterprises. The service requires elasticity, scalability and high availability. The main challenge is to optimize the deployment time and elasticity of the solution based on the demand. An important parameter to improve the deployment time is the

delivery time. This delivery time is the time that is needed to download a virtual machine image to a cloud where is going to be instantiated.

- Third Party Software
  - Alfresco is a free fully managed enterprise content management solution. The main challenge is to optimize the VMI configuration and dependencies with Cloud Providers, hence making it possible to perform on demand scaling within different clouds. Also the service will take advantage of images size reduction in order to decrease the resources required to store these images.
  - Redmine is a free, open-source web application for project and issue management. The main challenge is to optimize the VMI configuration procedures and resources needed to stores the images.

4.5 Unified Communications

During the last years, the communication technologies have evolved to a significant extent. There are new standards that are having a big impact in the sector. Nowadays, there are different available approaches for communication solutions:

- Traditional telephony: Analogy & VoIP (Voice over IP) telephony for internal use into company.
- In last years, some service providers are opening to SIP (Session Initiation Protocol) technology and offering their services to the customers.
- Nowadays:
  - Voice & Video Calls, IM (Instant Messaging) and Presence full supported in actual architectures.
  - WebRTC calls (calls supported on web browsers and without plugins) without thick client software.
  - Enterprise portal collaboration with poor support for voice and video tools.

Objective: The objective of this pilot is to use an open source communication framework to establish a full collaboration within the companies. The main challenging aspect of this pilot is that its architecture must be safe, elastic, and scalable and requires high availability.

The basic services of our use case are the unification between the telco and internet world:

- Traditional VoIP and Video calls
- WebRTC calls
- Voicemail
- Instant Messaging and Presence

The advanced services offered by our architecture are:

- SIMPLE IM and Presence, including XCAP and MSRP relay
- SIP calls Load Balancing

- Central Management and Provisioning
- Audio Call Recording
- IVR
- ASR / TTS
- SIP-to-PSTN calls
- Call Detail Recording
- Billing
- IPv6 support

4.6 Alfresco

Alfresco is an ECM platform based on open technology (open source and open standards). Alfresco allows consolidation to a single platform, significantly reducing the costs associated with traditional ECM.

Objective: The objective of Alfresco is to manage the critical content documents. The main challenge is to optimize the VMI configuration and dependencies within Cloud Providers, hence making it possible to perform on demand scaling within different clouds. Furthermore, lowering the amount of resources needed to store the images is also an objective for this service.

Some of its main features are:

- Flexible security document management system options
- Organization content into sites and folders
- Filters and tags to refine search criteria
- Powerful search
- It allows add custom types to help identify the purpose of documents
- Overlay property templates

4.7 Redmine

Redmine is a flexible project management web application. Written using the Ruby on Rails framework, it is cross-platform and cross-database. Redmine is open source and released under the terms of the GNU General Public License v2 (GPL).

Objective: The objective of this pilot is to use an open source web application to make easier the project and issue management. The main challenging aspect of this pilot is to optimize the different VMI configuration procedures and decrease the number of resources needed to store the VMI. The project should provide the necessary elasticity to our architecture to support the system dynamic load in every time.

Some of the main features of Redmine are:

- Multiple projects support

- Flexible role based access control
- Flexible issue tracking system
- Gantt chart and calendar
- News, documents & files management
- Feeds & email notifications
- Per project wiki
- Per project forums
- Time tracking
- Custom fields for issues, time-entries, projects and users
- SCM (Source Code Management) integration (SVN, CVS, Git, Mercurial, Bazaar and Darcs)
- Issue creation via email
- Multiple LDAP authentication support
- User self-registration support
- Multilanguage support
- Multiple databases support

4.8 Architecture

4.8.1 Unified Communications

The use case architecture diagram is presented in Figure 17.

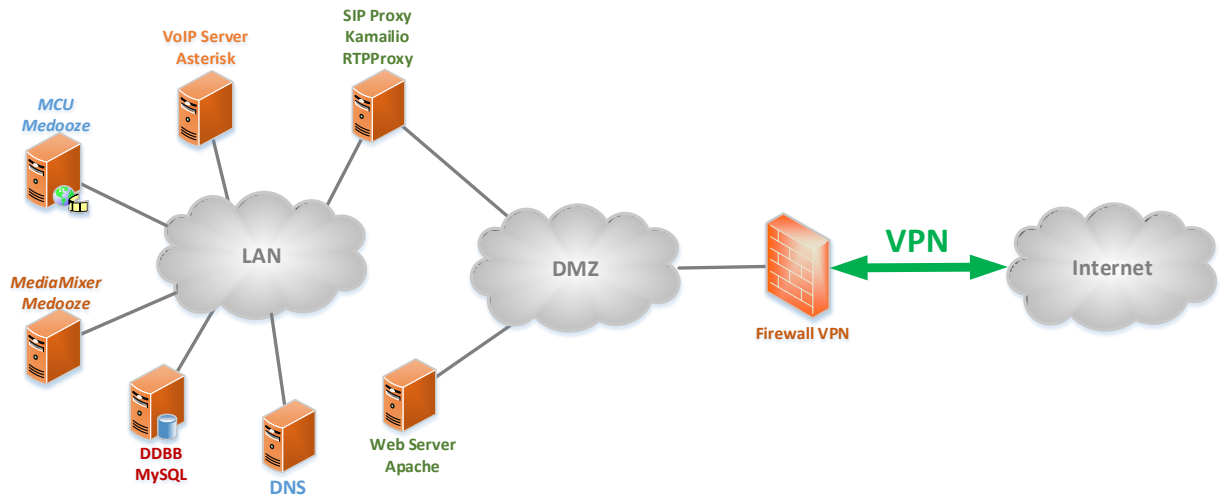

Figure 17: Unified communications use case architecture

Features of each functional block:

- SIP Proxy Kamailio and RTPproxy:
  - Secure SIP communication with TLS and call encryption (SRTP).

- WebSocket support for WebRTC.
- Gateway from WebSocket to TLS/TCP/UDP for all SIP messages.
- NAT (Network Address Translation) traversal support.
- Instant Message and Presence (user state) support.
- Call Admission Control.
- SIP proxy and SIP server.
- Registrar and Location server.
- Load Balancer server.

- VoIP Server:
  - Media Servers for voice services with Asterisk.
- Video Servers:
  - MCU and Media Mixer servers with Medooze service.
- DB :
  - User authentication.
  - User state for IM and Presence.
  - Configuration storage for all system.
- Firewall and VPN:
  - Network and Application layer security (L2 to L7).
  - VPN with certificates
- Public Web Portal
  - Collaboration Portal: WebRTC calls, IM and Presence

4.8.2 Alfresco

The use case architecture diagram is shown in Figure 18.

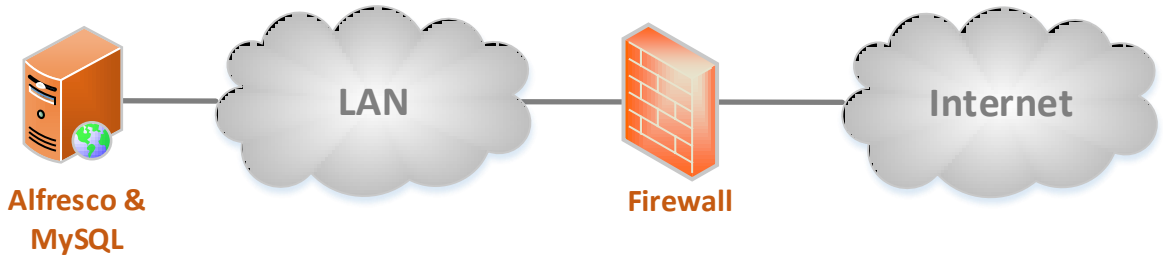

Figure 18: Alfresco use case architecture

Each functional block has the following features:

- Alfresco Server and MySQL
  - Enterprise content management.

- Web interface.
- User authentication.
- Configuration storage for all system.

- Firewall :
  - Network and Application layer security (L2 to L7).

4.8.3 Redmine

The use case architecture diagram is shown in the following image:

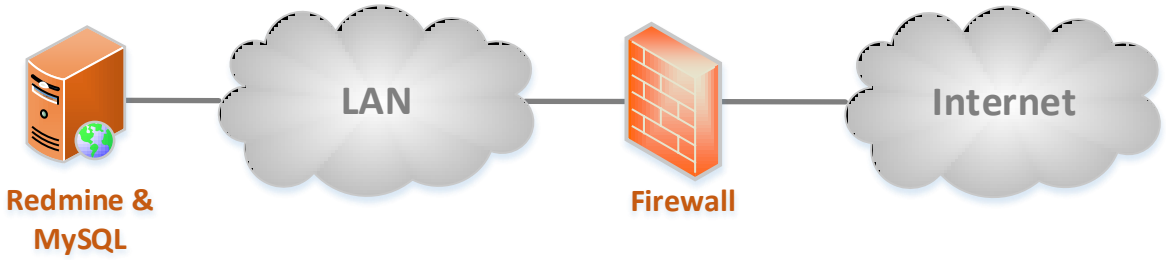

Figure 19: Redmine use case architecture

Features of each functional block:

- Redmine Server and MySQL
  - Web interface.
  - Multiple projects support.
  - User authentication.
  - Configuration storage for all system.
- Firewall :
  - Network and Application layer security (L2 to L7).

4.9 Components of the pilot

4.9.1 Unified Communications

4.9.1.1 Actors

- **System Administrator**: add and remove system users, deploy additional services and servers, enforce security polices, create call conferences, change firewall policies, etc.
- **System Operator**: view CDR and billing information, supervise server status, traffic trespassing firewall cluster, etc.
- **SIP service provider** (out of this use case): gateway to public switched telephone network (PSTN) and GSM (Global System for Mobile) networks.

- **End-user**: actor that uses the system: VoIP and video calls, Instant Messaging and Presence, Voicemail and WebRTC calls.

4.9.1.2 Pre-Conditions

- Final user must be connected to our architecture:
  - Connecting directly to a LAN (that depends on the firewall) or connecting to an infrastructure via WebRTC.
  - Additionally, the user must be register in the system and logged (on-line) in UC infrastructure in order to use available services.
- User can use UC services via the following options:
  - Heavy client (user must install softphone previously).
  - Thin client (WebRTC).
  - Calling from PSTN or GSM networks (out of the use case).
- All minimum services activated: At least one active server for every different service cluster (one VoIP server, one Firewall, etc.).
- Internet access if one of call leg comes from or goes to Internet. All the time two call legs must exist: The first call leg from source end-user to the UC architecture and the second call leg from the UC architecture to the destination end-user.
- Every communication (call signalling, instant message or presence change) is supported by SIP protocol. Media is supported by RTP protocol and security is supplied by protocols like TLS or SRTP.

4.9.1.3 Post Conditions

Success end condition

- The architecture can process voice and video calls from enterprise end-user from inside (corporative LAN) and from outside (WebRTC calls), also, the architecture offers added-value services (Voicemail, Call recording, IM, Presence, etc.).

Failure end condition

- Cloud hosts physical availability: failure covered by high availability architecture.
- Security attacks (DoS, SIP identity supply, toll fraud, etc.): risk-mitigated by techniques as DoS, password generation policies, ACLs, track systems, harden firewall policies, etc.
- Lack of resources: controlled by Connection Admission Control methods.

4.9.1.4 Dependencies

Based on network conditions and system load, the UC infrastructure might support a determined number of end-users. The larger bandwidth and VM resources, the more end-users are supported.

4.9.1.5 Trigger

The basic trigger condition is that end-user need to establish a communication with another end-user of UC infrastructure.

Other triggers:

- End-user presence change.
- End-user instant messaging event (send/receive).

4.9.1.6 Scenarios

4.9.1.6.1 Basic SIP VoIP call

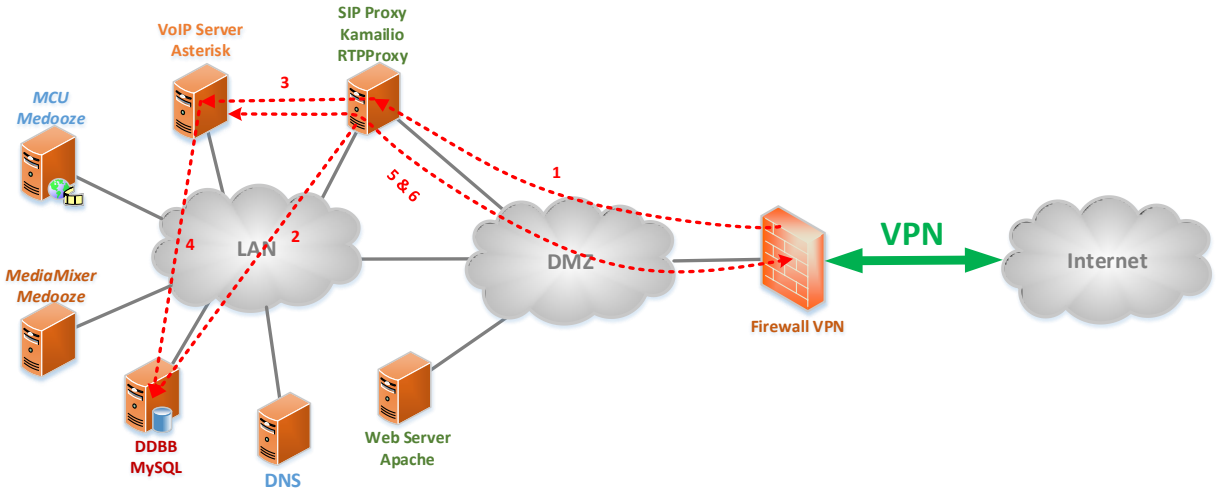

Figure 20: Basic SIP VoIP call scenario

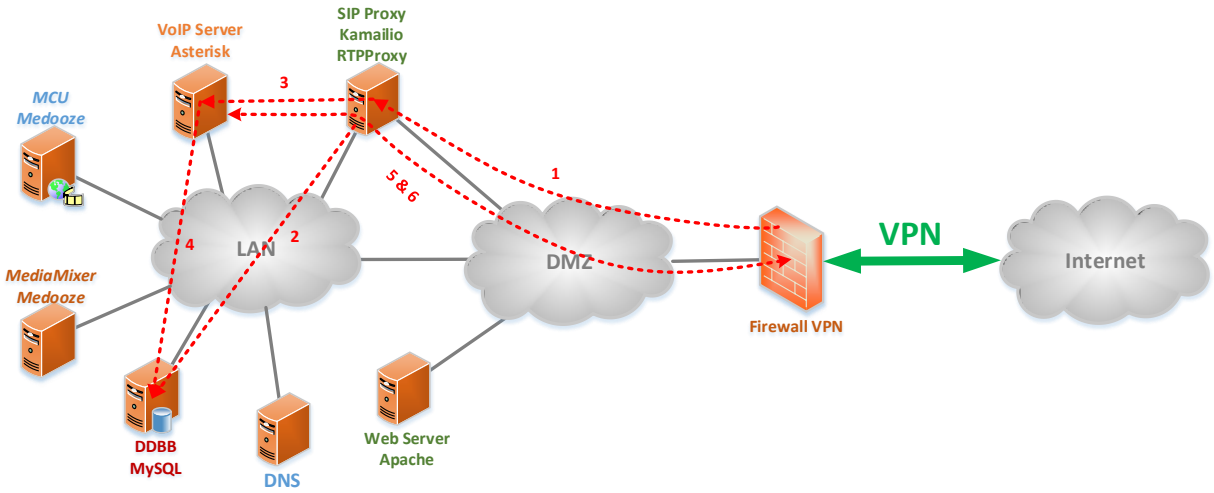

Figure 20 shows the actions that incoming VoIP calls take.

The steps that take place in the internal communication between the services and external users are the following (the role of the DNS has been ignored to simplify the diagram):

1) The original user of the call sends the connection request to Kamailio through the SIP signalling protocol.

2) Kamailio consults the database to check if the user is registered and balance the call on Asterisk.

3) Kamailio forwards the call to the Asterisk server using SIP selected.

4) Asterisk query the database to obtain the information needed to establish communication with the user call destination.

5) Asterisk server sets the second leg call with the user through Kamailio destination, using the SIP protocol.

6) The RTPproxy forwards RTP flows between the source and destination server and Asterisk users, supporting the media traffic between two users.

4.9.1.6.2 Basic SIP Videoconference call

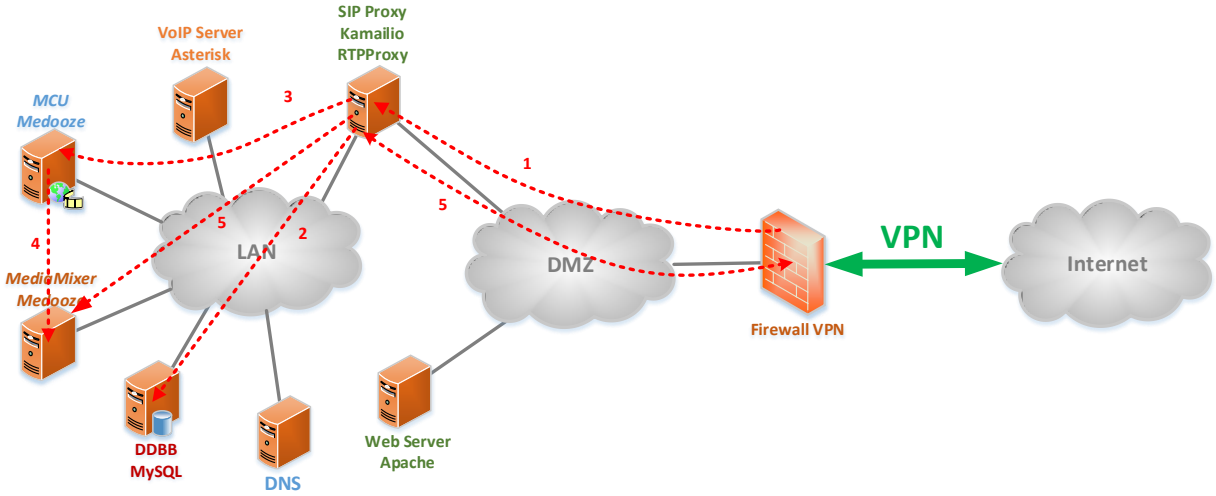

Figure 21: Basic SIP videoconference scenario

Again, the role of DNS has been ignored for simplicity of diagram in order to ease the understanding of the flow of internal communications.

The steps that take place in the video call are as follows:

1) A user sends the connection request to Kamailio by SIP.

2) Kamailio consults the database to check if the user is registered.

3) Kamailio forwards the video call to Medooze MCU using SIP, which create the conference room.

4) MCU Medooze assign video streams to a Medooze MediaMixer by XMLRPC, which is responsible for mixing the different streams into a single stream, distributed to each user connected to the room in question.

5) Medooze MediaMixer refers the RTP flow to rtpproxy who keep the RTP flows between calling user and called.

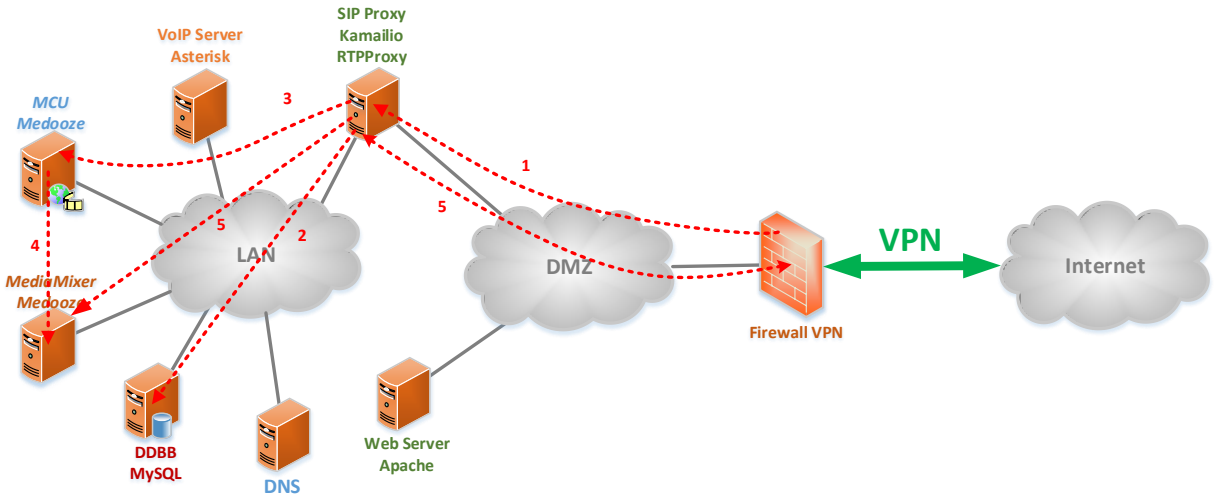

Figure 21 explains the flow diagram of this scenario.

4.9.1.6.3 WebRTC call

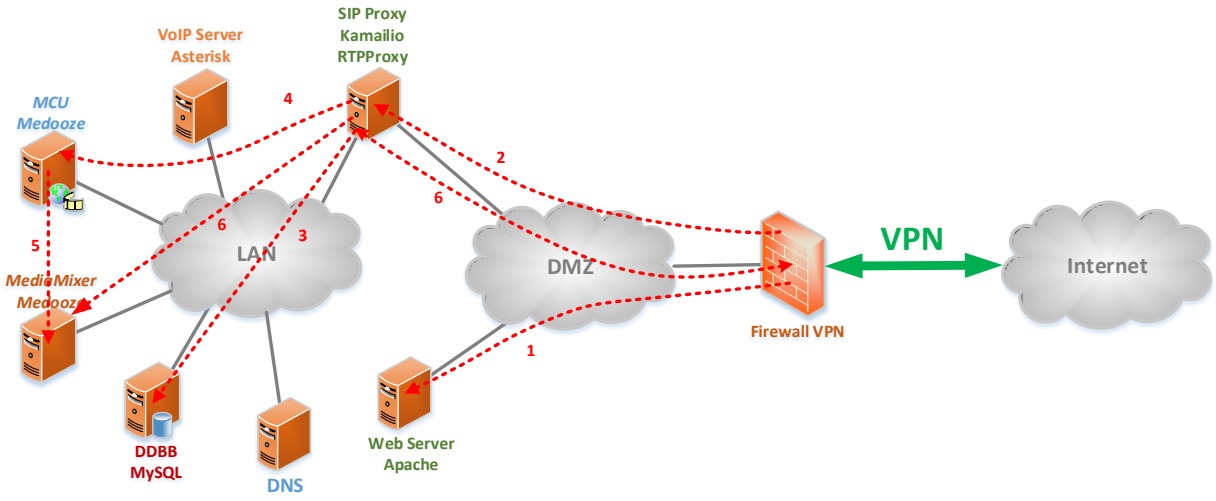

Figure 22: WebRTC call

In this case, the role of DNS has been ignored in order to ease the understanding of the diagram that shows the flow of internal communications.

1) WebRTC a user accesses the portal to start VoIP call or video, and your browser the library containing the SIP stack is discharged to make video calls.

1) From that moment, the user sends the connection request to Kamailio by SIP.

3) Kamailio consults the database to see if the user is registered.

4) Kamailio derives the video call to Medooze MCU using SIP, who created the conference room.

5) MCU Medooze assign video streams to a Medooze MediaMixer by XMLRPC, which is responsible for mixing the different streams into a single stream, distributed to each user connected to the room in question.

6) Medooze MediaMixer refers the RTP flow to rtpproxy who keep the RTP flows between calling user and called user.

4.9.1.6.4 IM sent and Presence change between end-users.

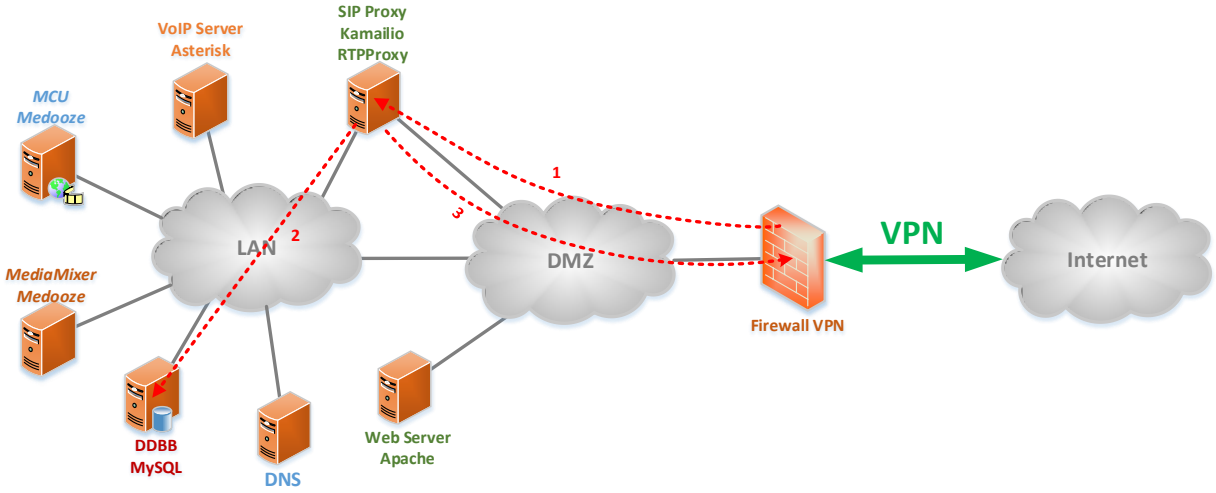

Figure 23. IM and Present service scenario

1) The original user sends the request to Kamailio through the SIP signalling protocol.

2) Kamailio consults the database to check if the user is registered.

3) Kamailio sends presence or IM.

4.9.1.7 Special Requirements

All main requirements are defined in the previous sections, but it must be emphasised the importance of network parameters: bandwidth and delay from end-user to UC infrastructure.

4.9.2 Alfresco

4.9.2.1 Actors

- System Alfresco Administrator: add and remove system users, deploy additional services and servers, change firewall policies, etc.
- End-user: actor that uses the Alfresco system.

4.9.2.2 Pre-Conditions

- Final user must be connected to Alfresco Server:
  - Connecting directly to a LAN (that depends on the firewall).
- Additionally, the user must be register in the system and logged (on-line) in the Alfresco Server via web interface.
- All minimum services activated: at least one active server for this service.
- The client needs a web browser.

4.9.2.3 Post Conditions

- Success end condition
  - A final user will be able to manage his critical content documents.
- Failure end condition
  - Security attacks: risk-mitigated by techniques as DoS, password generation policies, ACLs, track systems, harden firewall policies, etc.

4.9.2.4 Scenario

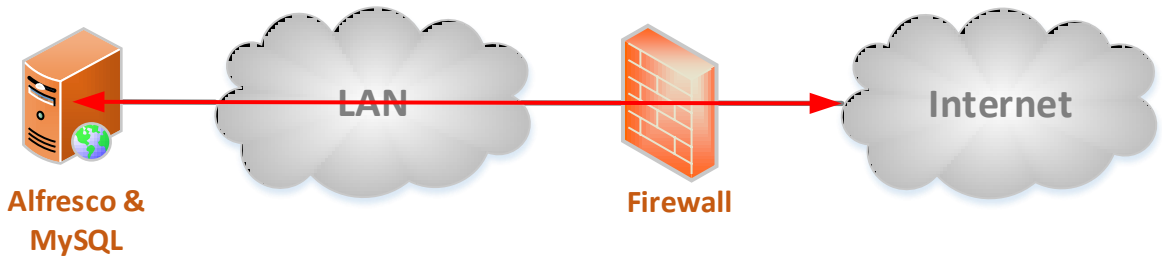

Figure 24: Alfresco scenario

The scenario exposed in Figure 24 explains how an external user can use the Alfresco's services form an internal server. The flow of actions is the following:

- A user access to Alfresco Server to consult a document.
- The server response to the user showing the document.

4.10 Redmine

4.10.1 Actors

- System Redmine Administrator: add and remove system users, deploy additional services and servers, change firewall policies, etc.
- End-user: actor that use the Redmine system.

4.10.2 Pre-Conditions

- Final user must be connected to Redmine Server:
  - Connecting directly to a LAN (that depends on the firewall).
  - Additionally, the user must be register in the system and logged (on-line) in the Redmine Server via web interface.
- All minimum services activated: at least one active server for this service.

- The client side needs one of the following browsers: Firefox, Chrome, Opera, Safari or Internet Explorer (10 and 11 versions).

4.10.3 Post Conditions

- Success end condition
  - The architecture allows that the VMI configuration procedures can be performed by a web interface. A final user will be able to:
  - Manage different tasks.
  - Establish priority and resources for the errors.
  - Create Gantt diagrams.
- Failure end condition
  - Security attacks: risk-mitigated by techniques as DoS, password generation policies, ACLs, track systems, harden firewall policies, etc.

4.10.4 Scenario

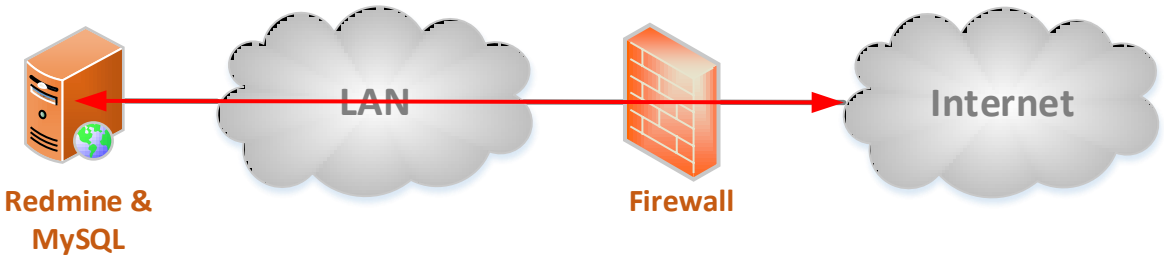

Figure 25: Redmine scenario

The Redmine scenario is quite simple. The user accesses from the Internet to a Firewall. Once the user is authenticated, it can connect to Redmine server. The diagram of this process is shown in Figure 25 and the workflow of this scenario is the following:

1. A user access to Redmine Server to manage a task.
2. The server response to the user showing the task.

4.11 Markets and users

The SaaS market has experienced tremendous growth during the past several years, which has accelerated the transition of on-premise hosted applications that are hosted in the Cloud and accessed over the Internet. The growth of the SaaS market has introduced significant new business and technical challenges.

This section shows some providers of this technology:

- **Akamai***:* Its cloud optimization offerings improve performance, increase availability, and boost security of applications and data delivered over the cloud.

- **AMAZON**: Amazon offers an infrastructure platform for their SaaS business (as the AWS usage based pricing model and scale on demand infrastructure aligns).
- *Cloud 9*: Cloud 9 is a simple application to help sales managers forecast. It offers real-time visibility to sale forecasts and pipeline management for line-of-business managers.
- **Abiquo**: Abiquo's software lets organizations use business policy to manage their entire computing infrastructure comprising unlimited physical and cloud resources through a single pane of glass, whether those clouds are private, public or hybrid.
- **Antenna Software**: It offers mobile SaaS software and Antenna Mobility Platform (AMP) for building, deploying and managing mobile applications.
- **IBM**: IBM offers three hardware platforms for cloud computing. These platforms offer built-in support for virtualization. For virtualization IBM offers IBM WebSphere application infrastructure solutions that support programming models and open standards for virtualization.
- **Cisco and Microsoft**: Cisco and Microsoft have teamed up to create a new product package that enables telecoms and other service providers to offer Azure-like cloud services. Microsoft brings its cloud infrastructure product and Cisco brings its networking devices and servers.

4.12 Problems

4.12.1 CSP Problems

The problems of CSP pilots related to ENTICE projects are listed as follows:

- **Problem 1: Manual VM image creation**. For an average technician at WT, the creation time of a new VM image takes at least approximately one hour. ENTICE environment can solve this problem providing previously configured and developed images, so its creation and deploying time can be highly reduced in order to attend properly the demand for new images.

- **Problem 2: Monolithic (non-decomposable) VM images.** WT offers a platform for custom Software-as-a-Service (SaaS) solutions for external parties which usually build on custom technologies and create monolithic VM images. These images are stored and managed by the company. When creating these images, in addition to requiring manual work (e.g. reconfiguration), at least one image per service is obtained. These images are not optimized in size, obtaining a monolithic (non-decomposable) VMI. As a result, the customer of the service has an extra cost (extra resources) due to the higher size of the images.

- **Problem 3: Proprietary unoptimised VM repositories**. Every Cloud provider is usually the responsible of managing and storing the images of the offered services. For that purpose a repository is provided in order to stock the images for all the services deployed. All the services that need to be deployed need to download an image from this repository to the corresponding Cloud where the service will be running. WT's resources cannot fulfill certain requirements, it is desired to transfer and deploy their VMs on not only in the same Cloud Infrastructure but also in a different Cloud environment. Thus to speed up the transfers of the image is needed. This can be done by optimizing the repository and using fragmentation of VM images in distributed repositories. This approach allows companies to transfer different parts of the images with parallel transfers, enhancing the delivery time.

- **Problem 4: Inelastic resource provisioning.** At WT, applications need to be manually scaled up or down based on the user load. Shutting down, manually assigning additional resources, and then powering up a VM again can take between 10−15 minutes causing a high service interruption during this time.

- **Problem 5: Lack of information to support automated preparation and optimized deployment.** WT needs to improve on the software engineering process and user acceptance of its Cloud platform through automated and optimized preparation and deployment of applications based on adequate user and resource-provided information, while taking full advantage of the federated Cloud.

4.12.2 CSP problems' metrics

Table 20 gives some metrics figures for our current scenario (Sc1) and estimations for the future scenarios (Sc2) of the system.

Table 20: Current CSP metrics

Problem	Metrics	Units	UC	Redmine	Alfresco
#1 - VM Synthesis. Manual creation of VM templates	VMI max size	MB	1000 (average per VM)	1100	2100
	VM max instantiation time (from image)	sec	480 (per VM)	300	420
	VMI preparation time	sec	36000	7200	18000
#2 - VM Analysis. Monolithic VM images	Minimum repository size	MB	1000 * no VM * no Services	1100 * no Services	2100 * no VM * no Services
#3 - Optimisation. Proprietary un-optimized VM repositories	VM storage	MB	1000 (average per VM)	1100	2100
	Transfer time	sec	1560	1820	3780
#4 - Elasticity. Inelastic resource provisioning	Elasticity	BOOL	FALSE	FALSE	FALSE
#5 - Semantics and SLA. Automated preparation and deployment	VM deployment time	sec	2158	2223	2495

The proposed CSP's targets for ENTICE objectives include optimizing the following metrics on each of these "VM management" activities:

- VM Synthesis
  - VMI size (60%)
  - VMI delivery time (30%)
  - VMI preparation time (25%)
- VM Analysis
  - Repository size (80%)
- Optimisation
  - VM performance (30%)
  - VM cost (25%)
  - VM storage (25%)

- Transfer time (50%)
- Elasticity
  - Scale time is less or equal to QoS improvement
- Semantics and SLA
  - VM deployment time (25%)

The mapping between those objectives and our use case scenarios is indicated in

Table 21.

Table 21: Objective CSP metrics

Activities	Metrics	Unit	Improvement	UC	Redmine	Alfresco
VM Synthesis	VMI size	MB	60%	400 (average per VM)	440	840
	VM max instantiation time (from image)	sec	30%	336	210	294
	VMI preparation time	sec	25%	27000	5400	13500
VM Analysis	Repository size	GB	80%	200 * no VM * no Services	220 * no Services	420 * no VM * no Services
Optimisation	VM performance		30%	TBD	TBD	TBD
	VM cost		25%	TBD	TBD	TBD
	VM storage	GB	25%	1500	300	1875
Elasticity	Elasticity	BOOL	NO	TRUE	TRUE	TRUE
Semantics and SLA	VM deployment time	sec	25%	1618,5	1667,25	1871,25

4.13 Metrics

This section defines the metrics that will be used to validate the UC architecture pilot and the metrics that the ENTICE environment has to provide to validate it. Those metrics are not intended to be actual requirements, but the parameters that will be measured during the testing and evaluation of the system when implemented by utilizing the ENTICE environment.

4.13.1 Metrics of the Unified Communication architecture pilot

4.13.1.1 Common metrics to all the components of the architecture

Table 22 show the metrics that are common to all the components of the architecture.

Table 22: Common metrics of the Unified Communication pilot

Component	Resource	Metric	Description	Unit
All	MOS	AvgQualityVoIP	Averaging the results of a set of standard: subjective tests with different listeners	Integer (1-5)
	Subjective Video Quality	AvgQualityVideo	Video quality as experienced by humans: how video is perceived by different viewers	Integer (1-5)
	Simultaneous calls	Calls	Number of simultaneous calls supported by the network	Integer
	Videoconference participants	Participants	Maximum number of participants allowed in a videoconference room	Integer
	System Load	SystemUsers	Maximum number participants allowed in the system	Integer

4.13.1.2 Metrics of the SIP Proxy-Kamailio

The Table 23 shows the metrics of the SIP Proxy.

Table 23: SIP Proxy-Core metrics

Component	Resource	Metric	Description	Unit
SIP Proxy-Core	VM	NumberVM	Number of Virtual Machines that will be necessary	Integer
	CPU	NumberCPU	Number of CPU (2GHz) that will be necessary	Integer
	RAM	RAMSize	Size of RAM that will be necessary	GigaBytes
	Storage	StorageSize	Size of Storage that will be necessary	GigaBytes
	Registered Users	AllocatedUsers	Location of the authenticated users	Integer
	Media Bandwidth	Throughput	Consumed Bandwidth by RTP flows	Kbps

4.13.1.3 Metrics of the VoIP Server

The VoIP Server will be located at the VoIP cluster. Table 24 shows its metrics.

Table 24: VoIP server metrics

Component	Resource	Metric	Description	Unit
VoIP Server	VM	NumberVM	Number of Virtual Machines that will be necessary	Integer
	CPU	NumberCPU	Number of CPUs (2GHz) that will be necessary	Integer
	RAM	RAMSize	Size of RAM that will be necessary	GigaBytes
	Storage	StorageSize	Size of Storage that will be necessary	GigaBytes
	Concurrent users	RegisteredUsers	Registered users in UC infrastructure	Integer

4.13.1.4 Metrics of the Video Cluster (MCU and Media Mixers)

The Video Cluster contains two different components: the MCU and the Media Mixers. Table 25 describes the metrics for evaluate them.

Table 25: Video Cluster metrics

Component	Resource	Metric	Description	Unit
Video Cluster	VM	NumberVM	Number of Virtual Machines that will be necessary for the Proxy-Core module	Integer
	CPU	NumberCPU	Number of CPU (2GHz) that will be necessary for the Proxy-Core module	Integer
	RAM	RAMSize	Size of RAM that will be necessary for the Proxy-Core module	GigaBytes
	Storage	StorageSize	Size of Storage that will be necessary for the Proxy-Core module	GigaBytes
	Concurrent users	UsersByRoom	Simultaneous users using UC infrastructure	Integer

4.13.1.5 Metrics of the DDBB

Table 26 shows the metrics of the DDBB.

Table 26: Metrics of the DDBB

Component	Resource	Metric	Description	Unit
	VM	NumberVM	Number of Virtual Machines that	Integer

DDBB Cluster			will be necessary	
	CPU	NumberCPU	Number of CPUs (2GHz) that will be necessary	Integer
	RAM	RAMSize	Size of RAM that will be necessary	GigaBytes
	Storage	StorageSize	Size of Storage that will be necessary	GigaBytes
	IOWait	Time	Destaging time to synchronize data between memory and disk	sec
	IOPS	AccessNumber	Number of access per user (input/output)	Operations by second

4.13.1.6 Metrics of the Web Portal

Table 27 shows the metrics of the Web Portal.

Table 27: Web Portal metrics

Component	Resource	Metric	Description	Unit
Web Portal	VM	NumberVM	Number of Virtual Machines that will be necessary	Integer
	CPU	NumberCPU	Number of CPUs (2GHz) that will be necessary	Integer
	RAM	RAMSize	Size of RAM that will be necessary	GigaBytes
	Storage	StorageSize	Size of Storage that will be necessary	GigaBytes
	ResponseTime	MaxResponseTime	Response time to serve a request user.	s

4.13.1.7 Metrics of the Firewall

Table 28 shows the metrics of the Firewall.

Table 28: Metrics of the Firewall

Component	Resource	Metric	Description	Unit
Firewall cluster	VM	NumberVM	Number of Virtual Machines that will be necessary	Integer
	CPU	NumberCPU	Number of CPUs (2GHz) that will be necessary	Integer
	RAM	RAMSize	Size of RAM that will be necessary	GigaBytes
	Storage	StorageSize	Size of Storage that will be necessary	GigaBytes

	Throughput	Bandwidth	Processed traffic by the firewall	Mbps

4.13.1.8 Metrics of the Network

Table 29 shows the metrics that the network needs to transport VoIP and Video calls. There are two different types of communications flows:

- Internal LAN (VoIP or video call)
- External (VoIP or video call)

Table 29: Metrics of the Network

Component	Resource	Metric	Description	Unit
All	Bandwidth end-to-end	EthernetBW	The bandwidth that will be used to transfer the call	Kbps
	One-way Delay end-to-end	MaxDelay	Maximum one-way delay between the source and destination users	ms
	One-Way Jitter end-to-end	MaxJitter	Maximum one-way jitter between the source and destination users	ms
	Packet Loss end-to-end	PacketLoss	Number of packets lost between the start and the finish of a call	%

Network metrics depend of codec type used by call and video calls.

4.13.2 Metrics of the Redmine Server

The Virtual Machine Server requires an Intel Xeon E540 Harpertown 2.5 GHz Processor. Redmine requires compute and storage-optimised. Table 30 shows the hardware metrics for the Server:

Table 30: Redmine metrics

Component	Resource	Metric	Description	Unit
Redmine Server	VM	NumberVM	Number of Virtual Machines that will be necessary	Integer
	Memory	MinMemorySize	Minimum size of memory that the Virtual Machine Server needs to work	Gigabytes
	Space DDBB	DiscSpace	Disc Space that the DDBB needs for the database files and the share documents	Gigabytes
	CPU	NumberCPU	Number of CPUs (2GHz) that will be necessary for the Proxy-Core module	Integer
	RAM	RAMSize	Size of RAM that will be necessary for the Proxy-Core module	MegaBytes
	Storage	StorageSize	Size of Storage that will be necessary for the Proxy-Core module	GigaBytes
	IOWait	Time	Destage time to synchronize data between memory and disk	sec
	IOPS	AccessNumber	Number of access per user (input/output)	Operations by second
	Reliability	ErrorRate	Gives a percentage of problem requests to all requests, measuring “performance failure” in the system, being more evident as increases with higher loads. This quick rise error shows where the system is stressed beyond its ability to deliver adequate performance.	%

4.13.3 Metrics of the Alfresco Server

Alfresco has a load balancing for high number of users. Redmine requires storage-optimised. Table 31 shows the metrics for the Alfresco Server:

Table 31: Alfresco server metrics

Component	Resource	Metric	Description	Unit
	VM	NumberVM	Number of Virtual Machines that will be necessary	Integer
	Memory	MinMemorySize	Minimum size of memory that the	Gigabytes

Alfresco Server			Virtual Machine Server needs to work	
	CPU	NumberCPU	Number of CPU (2GHz) that will be necessary	Integer
	RAM	RAMSize	Size of RAM that will be necessary	GigaBytes
	Storage	StorageSize	Size of Storage that will be necessary	GigaBytes
	IOWait	Time	Destage time to synchronize data between memory and disk	s
	IOPS	AccessNumber	Number of access per user (input/output)	Operations by second
	Availability	AvailabilityRate	Gives a percentage of how many requests to all requests are completed successfully.	%
	Reliability	ErrorRate	Gives a percentage of problem requests to all requests, measuring "performance failure" in the system, being more evident as increases with higher loads. This quick rise error shows where the system is stressed beyond its ability to deliver adequate performance.	%

4.14 Requirements of the CSP pilot

CSP (Cloud Service Provider) use case is oriented to offer optimized cloud services to potential customers. An administrator user from CSP is the responsible to customize the services offered to customers. CSP pilot considers three different use cases, related to offered services to customers: Unified Communications, Redmine and Alfresco. ENTICE environment shall help the CSP administrator to provide good service in terms of quality and adequate response time to customer requests.

Redmine and Alfresco use cases shall be provided with a static number of functional blocks (VMs). Unified Communications use case is more complex: it shall consist of some functional blocks with a static number of nodes (SIP-Proxies, Web Portal, DNS, vFirewall, Monitoring, Management, CDR and DDBB nodes), and other functional blocks with a dynamic number of nodes (VoIP, Video and File clusters).

4.14.1 Functional requirements

4.14.1.1 Behaviour

ENT.REQ.CSP.F.0100 The ENTICE environment shall facilitate the access to optimized VM repository to deploy the functional blocks in the CSP pilot.

Parent: ENT.REQ.CSP.F.0100

Rationale: the access to optimized VM repository to deploy services in an efficient way is a differential alternative from existing solutions. This requirement is aligned with the CSP Problem 3 and Problem 4. It is also related to ENTICE's Objective 3.

Verification method: Test and Demonstration

Test type: System test

ENT.REQ.CSP.F.0110 The ENTICE environment shall allow to customize the VM characteristics, based on user requirements.

Parent: ENT.REQ.CSP.F.0110

Rationale: end user should be able to define the expected system load (number of users using the use case application) and quality of communications (Unified Communications case). This requirement is aligned with the CSP Problem 5. It is also related to ENTICE's Objective 5.

Verification method: Test and Demonstration

Test type: System test

ENT.REQ.CSP.F.0120 The ENTICE environment shall allow CSP administrator user to define the advanced configuration for the services that will be running into VMI images.

Parent: ENT.REQ.CSP.F.0120

Rationale: service advanced configuration must be manually adapted by CSP administrator, based on end user requirements, e.g., to adapt VoIP service dial-plan for CSP customer. This requirement is partially aligned with the CSP Problem 5. It is also related to ENTICE's Objective 5.

Verification method: Test and Demonstration

Test type: System test

ENT.REQ.CSP.F.0130 The ENTICE environment shall allow to increase the number of nodes in each functional block, based on the system load (number of simultaneous users).

Parent: ENT.REQ.CSP.F.0130

Rationale: CSP pilot considers three different use cases, i.e., Unified Communications, Redmine and Alfresco. For instance, in the case of Unified Communications (the more complex one), ENTICE environment should increase or decrease the number of nodes in the VoIP and/or Video cluster based on the system load, i.e., based on the number of users which are using the Unified Communications infrastructure. Additionally, in the Unified Communication use case, ENTICE environment should inform Proxy-Core on the changes of number of nodes in a specific cluster. Thus, Proxy-Core would know how to balance the load over the VoIP or Video cluster. This requirement is aligned with the CSP Problem 3, 4 and 5. It is also related to ENTICE's Objectives 3, 4 and 5.

Verification method: Analysis, Test and Demonstration

Test type: System test

ENT.REQ.CSP.F.0140 The ENTICE environment shall provide the VM networking configuration for use case execution in federated cloud environments.

Parent: ENT.REQ.CSP.F.0140

Rationale: ENTICE environment should assure the communications between VMIs that compose a particular use case. For instance, ENTICE environment should define DMZ, LAN and WAN networks where VMIs will be connected in the Unified Communications use case, and allowing vFirewalls to control the communications between functional blocks (communications between clusters of each service, i.e., VoIP, Video, etc.). ENTICE should manage the communication between nodes in a federated environment and for each HA functional block: Proxy-Edge, Web portal, DNS service, etc. This requirement is inherent to the dynamic reasoning outlined in the CSP Problems 4 and 5. It is also related to ENTICE's Objectives 4 and 5.

Verification method: Analysis and Test

Test type: System test

ENT.REQ.CSP.F.0150 The ENTICE environment shall facilitate the monitoring of the demand of services in the CSP pilot.

Parent: ENT.REQ.CSP.F.0150

Rationale: The monitoring of the demand is essential to have metrics to measure the different levels of demand in different instants so the system architecture can be optimized. This requirement is aligned with the CSP Problems 1 and 4. It is also related to ENTICE's Objectives 1 and 4.

Verification method: Test

Test type: System test

4.14.1.2 Capabilities

ENT.REQ.CSP.F.0160 The ENTICE environment shall provide scalability and elasticity to CSP use case.

Parent: ENT.REQ.CSP.F.0160

Rationale: ENTICE environment should increase or decrease the number of necessary VMIs based on the system load, i.e., based on the number of users which are using the Unified Communications infrastructure, Redmine or Alfresco. This requirement is aligned with the CSP Problems 3, 4 and 5. It is also related to ENTICE's Objectives 3, 4 and 5.

Verification method: Test and Demonstration

Test type: System test

ENT.REQ.CSP.F.0170 The ENTICE environment shall allow the location redundancy for Proxy-Edge in the Unified Communication use case.

Parent: ENT.REQ.CSP.F.0170

Rationale: Proxy-Edge is the entry point for Unified Communication infrastructure. Federated cloud approach from ENTICE environment should allow the redundancy of the Proxy-Edge geo-location, so High Availability service will be assured. This requirement is aligned with the CSP Problem 4. It is also related to ENTICE's Objective 4.

Verification method: Test and Demonstration

Test type: System test

4.14.1.3 Knowledge model

ENT.REQ.CSP.F.0180 The ENTICE environment shall register the historical data of system usage.

Parent: ENT.REQ.CSP.F.0180

Rationale: ENTICE environment should register the historical data of system usage, i.e., the number of users that contribute to the system load, in order to predict the periods where ENTICE should increase or decrease the number of nodes in each cluster of services. This requirement is aligned with the CSP Problem 4. It is also related to ENTICE's Objective 4.

Verification method: Test and Demonstration

Test type: System test

4.14.1.4 VMI images

ENT.REQ.CSP.F.0190 The ENTICE environment shall distribute VMI images from each use case around federated cloud in an efficient way.

Parent: ENT.REQ.CSP.F.0190

Rationale: ENTICE environment shall distribute VMI images from each use case around federated cloud so that interrelated VMs will be grouped in the location that VMIs are delivered. This requirement is aligned with the CSP Problems 1, 2 and 4. It is also related to ENTICE's Objectives 1, 2 and 4.

Verification method: Test and Demonstration

Test type: System test

ENT.REQ.CSP.F.0200 The ENTICE environment shall be compatible the Operating System that supports the use case applications.

Parent: ENT.REQ.CSP.F.0200

Rationale: ENTICE environment shall be compatible with the Operating Systems used to build the functional blocks from each use case, i.e., O.S. from each VM. This requirement is aligned with the CSP Problems 1 to 4. It is also related to ENTICE's Objectives 1 to 4.

Verification method: Test and Demonstration

Test type: System test

ENT.REQ.CSP.F.0201 The ENTICE environment shall allow to map use case VMs to virtual containers.

Parent: ENT.REQ.CSP.F.0201

Rationale: ENTICE environment shall may use virtual containers, e.g. Docker technology, with the purpose of optimize VMs. This requirement is aligned with the CSP Problems 1 to 4. It is also related to ENTICE's Objectives 1 to 4.

Verification method: Test and Demonstration

Test type: System test

4.14.1.5 Evaluation metric

ENT.REQ.CSP.F.0202 The ENTICE environment shall evaluate the SLA to meet the QoS (Quality of Service) and QoE (Quality of Experience) requirements.

Parent: ENT.REQ.CSP.F.0202

Rationale: ENTICE environment shall monitor the appropriate parameters of deployed use case infrastructure in order to detect the degree of compliance with the defined SLA. This requirement is aligned with the CSP Problem 4. It is also related to ENTICE's Objective 4.

Verification method: Test and Demonstration

Test type: System test

4.14.2 Non Functional requirements

4.14.2.1 Performance

ENT.REQ.CSP.NF.0100 The ENTICE environment shall deliver good performance in reference to networking and systems.

Parent: ENT.REQ.CSP.NF.0100

Rationale: ENTICE environment shall deliver good performance in reference to networking and systems, in order to meet high demands of communications and data flow processing, particularly in the Unified Communications use case. This requirement is aligned with the CSP Problems 3 and 4. It is also related to ENTICE's Objectives 3 and 4.

Verification method: Test

Test type: System test

4.14.2.2 Storage

ENT.REQ.CSP.NF.0110 The ENTICE environment shall adapt the storage type to each use case functional block.

Parent: ENT.REQ.CSP.NF.0110

Rationale: ENTICE environment shall adapt the storage type to each use case functional block. For instance, regarding to Unified Communication, ENTICE shall deliver SATA storage to File cluster and SAS/SSD storage to others blocks like VoIP or Video cluster. This requirement is aligned with the CSP Problems 3 and 4. It is also related to ENTICE's Objectives 3 and 4.

Verification method: Test and Demonstration

Test type: System test

4.14.2.3 Cost model

Not Applicable.

4.14.2.4 Technology

ENT.REQ.CSP.NF.0120 The ENTICE environment shall adapt the storage type to each use case functional block.

Parent: ENT.REQ.CSP.NF.0120

Rationale: ENTICE environment shall adapt the storage type to each use case functional block. For instance, regarding to Unified Communication, ENTICE shall deliver SATA storage to File cluster and SAS/SSD storage to others blocks like VoIP or Video cluster. This requirement is aligned with the CSP Problems 3 and 4. It is also related to ENTICE's Objectives 3 and 4.

Verification method: Test and Demonstration

Test type: System test

ENT.REQ.CSP.NF.0130 The ENTICE environment shall be able to deploy use case VMs in any virtualization system.

Parent: ENT.REQ.CSP.NF.0130

Rationale: ENTICE environment shall deploy use case VMs in an agnostic way regarding to the subjacent virtualization system (VMware, OpenStack, etc.). This requirement is aligned with the CSP Problems 1 to 4. It is also related to ENTICE's Objectives 1 to 4.

Verification method: Test and Demonstration

Test type: System test

4.14.2.5 Privacy

Not Applicable.

4.14.2.6 Compliance

Not Applicable.

4.14.2.7 Security

ENT.REQ.CSP.NF.0140 The ENTICE environment shall control the access to VMI repository.

Parent: ENT.REQ.CSP.NF.0140

Rationale: ENTICE environment shall restrict the access to VMI repository to authorized users. This requirement is aligned with the CSP Problem 3. It is also related to ENTICE's Objective 3.

Verification method: Test and Demonstration

Test type: System test

4.14.2.8 Reliability

ENT.REQ.CSP.NF.0150 The ENTICE environment shall offer an elevated degree of reliability.

Parent: ENT.REQ.CSP.NF.0150

Rationale: ENTICE environment shall offer an elevated degree of reliability since the services provided by CSP will be critical to customers. This requirement is aligned with the CSP Problems 2 and 3. It is also related to ENTICE's Objectives 2 and 3.

Verification method: Test and Demonstration

Test type: System test

4.14.2.9 Availability

ENT.REQ.CSP.NF.0160 The ENTICE environment shall provide high availability in every functional block and in global system.

Parent: ENT.REQ.CSP.NF.0160

Rationale: ENTICE environment shall offer a degree of availability over 99.999%, due to operational criticality of provided services. This requirement is aligned with the CSP Problems 2 and 3. It is also related to ENTICE's Objectives 2 and 3.

Verification method: Test and Demonstration.

Test type: System test.

4.14.2.10 Maintainability

Not Applicable.

4.14.2.11 Portability

Not Applicable.

4.14.3 Interface requirements

4.14.3.1 Hardware

ENT.REQ.CSP.I.0100 The ENTICE environment shall build the VMIs considering x86_64 architecture.

Parent: ENT.REQ.CSP.I.0100Rationale: ENTICE environment shall build the VMIs considering x86_64 that is the architecture used by VMs from CSP use cases. This requirement is aligned with the CSP Problems 1 to 3. It is also related to ENTICE's Objectives 1 to 3.

Verification method: Test and Demonstration

Test type: System test

ENT.REQ.CSP.I.0110 The ENTICE environment shall consider the type and number of each physical resource (CPU, RAM, HDD) provided to VMIs.

Parent: ENT.REQ.CSP.I.0110

Rationale: ENTICE environment shall consider the type and number of each physical resource (CPU, RAM, HDD) provided to VMIs, based on requirements of each application of CSP use cases. This requirement is aligned with the CSP Problems 1 to 3. It is also related to ENTICE's Objectives 1 to 3.

Verification method: Test and Demonstration

Test type: System test

4.14.3.2 Software

ENT.REQ.CSP.I.0120 The ENTICE environment shall provide an appropriate user interface to build the use case that will run in the target cloud.

Parent: ENT.REQ.CSP.I.0120

Rationale: ENTICE environment shall provide an appropriate user interface to build the use case that will run in the target cloud. The interface will be intuitive and easy to operate. This requirement is aligned with the CSP Problem 5. It is also related to ENTICE's Objective 5.

Verification method: Test and Demonstration

Test type: System test

ENT.REQ.CSP.I.0130 The ENTICE environment shall provide an API to communicate CSP offered applications running into VMIs and ENTICE.

Parent: ENT.REQ.CSP.I.0130

Rationale: ENTICE environment shall provide an API to communicate CSP offered applications running into VMIs and ENTICE. The objective is to define an API by which applications notify to ENTICE on the system load, so ENTICE shall take the necessary actions. For instance, it shall be

possible to define 3 load levels (Low, Medium and High) mapped to a specific number of system and network resources. As the system load increases (/decreases), ENTICE shall increase (/decrease) the VMI assigned resources. This requirement is aligned with the CSP Problems 1 to 5. It is also related to ENTICE's Objectives 1 to 5.

Verification method: Test and Demonstration

Test type: System test

4.14.3.3 Communications

ENT.REQ.CSP.I.0140 The ENTICE environment shall consider inter-relation between VMs which compose CSP use cases.

Parent: ENT.REQ.CSP.I.0140

Rationale: ENTICE environment shall consider inter-relation between VMs which compose CSP use cases, in order to allow the proper communications between VMIs. This requirement is aligned with the CSP Problem 5. It is also related to ENTICE's Objective 5.

Verification method: Test and Demonstration

ENT.REQ.CSP.I.0150 The ENTICE environment shall consider the extremely demanding requirements in respect of network specifications.

Parent: ENT.REQ.CSP.I.0150

Rationale: ENTICE environment shall consider the extremely demanding requirements in respect of network specifications, in terms of bandwidth, delay and packet loss. This requirement is aligned with the CSP Problem 4. It is also related to ENTICE's Objective 4.

Verification method: Test and Demonstration

5 State of the Art

The following sections describe the State of the Art of the key technologies to be developed in the ENTICE project: Lightweight creation of VM images and storage, multi-objective techniques, cloud elasticity, and knowledge based modelling and reasoning.

5.1 Lightweight VM image creation and storage

This section gathers the existing VM image creation and storage techniques employed for the allocation of applications and resources in a distributed computing environment. The following sections present those techniques and evaluate their pro and cons. The analysis of the State of the Art presented here is required for the consecution of ENTICE's *objective 1: Light image*

creation and objective 2: Distributed lightweight VM image storage and it is closely related with the implementation of *WP 3 Interoperable and decentralized VM image synthesis, analysis and storage*. It is also related to *WP 4 Federated multi-objective VM repository optimisation* and WP5 Based acknowledge reasoning.

5.1.1 Lightweight VM image creation and storage state of the art

5.1.1.1 Image format and interoperability

Tang et al. [Tang11] present Fast Virtual Disk (FVD) as a new virtual machine (VM) image format and the corresponding block device driver developed for QEMU, an emulator for multiple hypervisors, including KVM, Xen-HVM, and VirtualBox. Its feature set includes flexible configurability, storage thin provisioning without a host file system, compact image, internal snapshot, encryption, copy-on-write, copy-on-read, and adaptive prefetching. As a principle, functions are intentionally orthogonal so that each feature can be configured independently yielding flexibility. The work is centred around VM mobility in a cloud focusing on subtle details of copy-on-read, and adaptive prefetching enabling instant VM creation and instant VM migration, even if the VM image is stored on direct-attached storage. ***Advantages***: Experiments show that the throughput of FVD is 249% higher than that of QCOW2 for file creation. **Disadvantages:** the proposed FVD needs heavy adoption by the IaaS providers to provide useful benefits for the other players in the cloud landscape.

The separation of application and the (virtual) infrastructure appears again in paper by Nguyen et al. [Nguyen13]. The work introduces an elastic instrument (called high-level Cloud Abstraction Layer – CAL) allowing easy development and deployment of services on resources of multiple infrastructure-as-a-service (IaaS) clouds simultaneously. Technically, the CAL provides an approach with emphasis on abstraction, inheritance and code reuse. **Advantages:** Cloud-based services can be developed easily by extending available classes provided by the CAL or other developers. Interoperability between different clouds is solved by the basic abstraction classes of the CAL and all services are inherited and benefited from the advantage. **Disadvantages:** Since this approach builds a higher level, object oriented layer on top of IaaS, and applications can be controlled by defined interfaces, it largely resembles an OO PaaS construction.

5.1.1.2 Image transfer, distribution and placement

Schmidt et al. [Schmidt10] analysed the distribution of VM images in a multi-cloud computing environment. They survey and analyse Network File System (NFS), unicast distribution, binary tree distribution, Fibonacci-tree distribution, peer-to-peer distribution (BitTorrent), multicast, cross-cloud and layered copy-on-write (UnionFS) distribution methods. Furthermore they take into consideration the security of VM image distribution such as encryption. **Advantages:** Their performance is compared experimentally and conclusions claim multicast as the one with best performance while a layered copy-on-write approach can save significant (up to 90%) data

traffic in cross-cloud scenario. **Disadvantages:** Their layering approach is more relevant for the runtime of the VMs as the layers are established once there are several executed VMs based on the same original virtual machine image.

Zhou et al. [Zhou13] investigate VM image migration techniques in case of a heterogeneous (both regarding type and performance) storage system focusing on multiple criteria, i.e. user experience, device wearing and manageability. Based on a migration cost metric three new storage migration strategies are put forward towards (i) least amount of redundant writes (ii) highest IO performance and (iii) balance between IO performance and write redundancy. The multi-criteria optimisation of transferring VM images is also an objective of ENTICE. However, the consideration of device heterogeneity and consequently, the differentiation between criteria accordingly, is a novel and outstanding dimension. **Advantages:** The prototyped system outperforms existing live storage migration by a significant margin that can be further improved by adaptive combination of the proposed schemes. **Disadvantages:** Since the approach is unique and probably orthogonal to other solutions, no real disadvantages can be specified.

Xu et al. [Xu14b] address the management of huge amount of VM images. Earlier approaches to optimize images usually focused on either improving performance or decreasing image size, which cannot satisfy the requirements of high IO performance, low storage consumption, and low management cost simultaneously. Typically, high IO performance requires images storing close to VMs, but this increases redundant data and consumes extra storage, and closer image means more data stored in local disks rather than a normal shared storage, which increases management cost as well. They propose a zone-based model to balance the requirements by partitioning computing nodes into many zones, and construct a shared storage in each zone to cache data. They also managed to improve the normal Copy-on-Write and cache mechanisms, providing new image types and cache functions to enhance the eventual effectiveness. **Advantages:** Their performed evaluations show that this solution improves IO performance by more than 100% in general and even 10 times while adopting a friendly VM placement strategy. **Disadvantages**: Though they improve IO performance and storage utilization with the propose solution, it also introduces some management overhead. To use this model, the whole image of a VM should be separated to a base image, a work image, and caches, which increase redundant data.

Bazarbayev et al. [Bazarbayev13] present a content-based scheduling algorithm for the placement of VMs in data centres. Their approach tries to find identical disk blocks in different VM disk images with similar operating systems, and schedule VMs with high content similarity to a previously deployed image on the same hosts. In this way they can reduce the amount of data transferred when deploying a VM on a destination host. Their analysis showed that content similarity between VMs with the same operating system and close version numbers can

be as high as 60%. It is also found that there is close to zero content similarity between VMs with different operating systems. They also designed a content-based scheduling algorithm that lowers the network traffic associated with transfer of VM disk images by skipping similar contents. **Advantages:** Their experimental results show that the amount of data transfer associated with deployment of VMs and transfer of virtual disk images can be lowered by more than 70%. **Disadvantages**: They have only simulation results; there is no information how it performs in a real datacentre.

Reich et al. [Reich12] propose a P2P streaming technique for storing VM images in a tool called vmTorrent. It is able to cope with read-intensive processes unlike network storage solutions. It uses a novel combination of block prioritization, profile-based execution prefetch, on-demand fetch and decoupling of VM image presentation from underlying data-stream. They also performed an experimental evaluation of vmTorrent, and found that it achieves comparable execution time to that achieved using local disks. **Advantages:** This solution is also able to maintain this performance while scaling to 100 instances, providing 11x speedup over current state-of-the-art and 30x over traditional network storage. **Disadvantages:** In order to use this approach, a custom file system server needs to be deployed that effectively virtualizes the VM images.

The work by Razavi et al. [Razavi13] is aimed at solving a common scenario when many VM images are to be transferred to many compute nodes simultaneously resulting slow VM startup times, negatively impacting both dynamic scaling of web applications and the startup of high-performance computing applications consisting of many VM nodes. The work is based on the observation that only a tiny part of the VM image is needed for the VM to be able to start up hence, small caches for VM images are able to overcome the VM startup bottlenecks. VM image caches are implemented as an extension to QCOW2, QEMU's implementation of copy-on-write. **Advantages:** Once caches are warm, a large amount of network traffic can be avoided. **Disadvantages**: Cold caches can cause significant performance degradation in certain environments. Current implementation is strongly tied to the structure of QEMU/QCOW2.

5.1.1.3 Duplication, de-duplication, chunks and image similarity

The core of the work by Peng et al. [Peng12] is an analysis of VM instance traces collected at six production data centres during four months with the aim of finding certain behavioural patterns that may enhance VM distribution. Results indicate that the number of instances created from the same VM image is relatively small at a given time and consequently conventional file based p2p sharing approaches may not be effective. Based on the understanding that different VM image files often have many common chunks of data, a chunk-level (as opposed to file-level) Virtual machine image Distribution Network (VDN) is proposed. Furthermore, the distribution scheme takes advantage of the hierarchical network topology of data centres to reduce the VM

instance provisioning time and to minimize the overhead of maintaining chunk location information. **Advantages:** Evaluation shows that VDN achieves as much as 30–80x speed up for large VM images under heavy traffic. The aims of this work are in strong relation to some objectives of ENTICE, the chunk based image distribution corresponding to the notion of small reusable parts of ENTICE where also the concept of image decomposition and reconstruction are put forward. **Disadvantages:** Though they use a hierarchical metadata server, it still represents a bottleneck.

The paper by Keren et al. [Keren09] is well aligned with Task T3.2 and shows that with simple block level de-duplication techniques, one can identify nearly 70% of identical parts in frequently used virtual machine images. In contrast to the paper's outcomes, the project will not aim on complete de-duplication as it would contradict the aims of T4.1-2, and in general might negatively impact the timely delivery of virtual machine images to their place of use. The paper shows several chunk identification techniques, but these chunks never reach over the level of file systems. In order to support online VM image assembly (T4.3), the project has a chance to investigate chunking over file systems as well (allowing the use of more metadata and easier identification of likely de-duplicable content in the images). Finally, the paper addresses the issue of the package management systems and their effect on the efficiency of de-duplication. The different behaviour and dependencies in the widely available package management systems lead to some variance in data chunks on the virtual machine images. This variance is expected to be unavoidable even if one processes the metadata offered by file systems, but with the help of T3.1 we expect that the variance in package management systems will disappear because of the functional optimization techniques applied by the project. **Advantages**: they have shown that with even simple chunk-level de-duplication methods the size of a single image file can be reduced up to 80%. **Disadvantages**: the block level handling of the images hides several important inter-relationships between the image parts.

Liquid is introduced as a lightweight distributed file system heavily building on ideas from the fields of de-duplication and peer to peer computing [Zhao14]. The new file system is especially targeted on storing virtual machine images. The authors consider de-duplicating both running and offline VM images, and they allow rapid cloning mechanisms with the help of copy on read. As Liquid is implemented as a file system it can be rather transparent for the IaaS system, on the other hand the authors require the IaaS systems to completely adopt Liquid in order to achieve the best performance. Collaboration within ENTICE WP3 and WP4 will allow a similarly performing system without the need to alter the IaaS system, thus allowing better performance on already available IaaSs without needing cooperation between their users and their providers. **Advantages**: Liquid provides good IO performance while doing deduplication in the background by caching frequently used datablocks and organizing them into chunks to reduce disk

operations. P2P technique provides good scalability. **Disadvantages**: On the other hand, an own file system should be used to track VM lifecycle with a metadata server.

Over several hundred VM images were analysed by Jayaram et al. [Jayaram11]. The analysis consisted of checking how efficiently 5 de-duplication techniques could work on the images, and also the authors introduced several similarity metrics. The used metrics and techniques were both checked in inter and intra VM image contexts. Unfortunately, this paper does not go further than point out that there is a chance of optimizing image storage or delivery with the help of de-duplication. Task 3.2 will continue with the analysis of other fingerprinting techniques, especially with techniques that are not restricted to block level, but also utilizing file system level information. **Advantages:** The authors have shown, in a real world scenario that choosing the right chunk-size for de-duplication is crucial, as the chunk size increases so the de-duplication factor decreases. **Disadvantages**: the analysed techniques were not checked for their capabilities of VM image size optimization.

The paper by Ng et al. [Ng11] goes further than the regular de-duplication techniques applied in past research. First of all, it analyses if the de-duplication actually reduces the performance of VM operations. Next, they consider how a changing ecosystem of VM images could be followed with a more dynamic de-duplication technique (which does not assume immutable VM image contents). Finally, the paper aims at improving the performance of running VM images by changing how de-duplication is applied to the constantly updating images. The authors propose the LiveDFS file system to answer the previously discussed 3 challenges. The authors even show the applicability of their file system in an OpenStack based cloud environment where a single LiveDFS instance is shared amongst the physical machines in the cloud and their storage systems. **Disadvantages:** Unfortunately, the levels of intrusion they introduce make their file system hard to apply in the context of ENTICE. **Advantages**: On the other hand the combination of such de-duplicating file system and the ENTICE VMMT concept might introduce a transparent technique for VM image delivery in clouds that did not incorporate de-duplication over ENTICE managed VM images.

Lei et al. [Lei14] present their method for de-duplicating data in existing image files for virtual machines. Their work is based on two methods: first the entire image is checked whether it exists in a repository, if not the image is chunked into fix sized blocks and the blocks are checked individually (block level de-duplication or fixed-size partition [FSP] method). The authors used different OS images and images that are based on these images to test their method. They used 4, 8, 16 and 32KB blocks. They found that as the block size increases the de-duplication decreases, since there is a higher possibility of changed data within a block. **Advantages:** Their proposed technique can significantly reduce the transmission time of image files that have already existed in storage. **Disadvantages**: the calculation of fingerprints of blocks is compute

intensive that could be improved. Lowering the computational cost would allow to use smaller block sizes.

Xu et al. [Xu14a] propose a VM image de-duplication method to speed up the cloud operation. Since different images have a large amount of same data segments, these duplicated data can lead to serious waste of storage resource. Although previous solutions could achieve a good result in removing duplicate copies, they are not very suitable for virtual machine image de-duplication in a cloud environment, since it could lead to serious performance interference to the hosting virtual machines. The paper proposes a local de-duplication method which can speed up the operation progress of virtual machine image de-duplication and reduce the operation time. The proposed method is based on an improved k-means clustering algorithm, which could classify the metadata of backup image to reduce the search space of lookups for the de-duplication. **Advantages** The experiments show that this approach can significantly reduce the performance interference to the hosting virtual machine with an acceptable increase in disk space usage. **Disadvantages**: re-duplication is still done during runtime – hence the interference – in ENTICE we will aim at reducing the interference caused by our techniques by exploiting periods of underutilisation.

Deshpenade et al. [Deshpande13] introduce the so called gang migration for the simultaneous live migration of multiple Virtual Machines from one set of physical machines to another. This process generates a large volume of network traffic and can overload the core network links and switches in a datacentre. To reduce this network overhead, gang migration uses global de-duplication (GMGD), which identifies and eliminates the retransmission of duplicate memory pages among VMs running on multiple physical machines in the cluster. They designed a GMGD prototype using QEMU/KVM VMs. **Advantages** The performed evaluations on a 30-node Gigabit Ethernet cluster having 10 GigE core links show that GMGD can reduce the network traffic on core links by up to 65% and the total migration time of VMs by up to 42%, when compared to the default migration technique in QEMU/KVM. **Disadvantages**: Some overheads, especially in worst-case scenarios.

Kochut et al. [Kochut12] leverage similarity in virtual machine images to reduce the data volume transferred from the storage server to the hypervisor on which the virtual machine is being instantiated. Really, the virtual machine instances are created using copy-on-write technology and based on read-only images. Therefore read-only images, originally obtained from the storage server, can be used to create other images. They presented an analytical model of such a provisioning process using image similarity, and validated it using a discrete event simulator. **Advantages** They found that the proposed provisioning scheme can achieve significant (up to 80%) reduction in data transfer between storage server and hypervisors. **Disadvantages** They use image redundancy information to supplement capacity based placement, which is only

effective when overlaps across images present on direct attached storage for reconstituting a virtual image.

This work by Nicolae et al. [Nicolae15] focuses again on a common pattern, collective on-demand read, accessing the same image or dataset from a large number of VM instances concurrently. The work is based on the collaboration of unrelated VM instances belonging to different groups to and exchanging common data in order to reduce the I/O pressure on the storage system. A mirror is a local view of the virtual disk image stored remotely on the VM repository to the hypervisor. From the perspective of the hypervisor, the local mirror appears to have already fetched and created a copy of all necessary content, however, the mirror gets populated with content only as needed during runtime. Mirrors advertise chunks to each other in a peer-to-peer collaborative scheme and prefetch any missing chunk soon as an advertisement about it has been received, in anticipation of future read requests. Nevertheless, the exact details (selection of peers, prefetching vs. conserving bandwidth) is said to be out of scope in this work. Detection of content similarity is realized by a low-overhead fingerprint based approach (a strong hash function) and duplicated content can be detected and exchanged on-the-fly. **Advantages:** The authors observed a speedup of completion time up to 11x compared to a naïve solution and a speed-up of 1.88-2x compared with collaborative schemes. Even with a single group the authors have measured a 4% overhead compared with an approach that is specifically aimed at a single group. **Disadvantages**: the proposed mirroring technique is mostly efficient only when the mirrors are placed close to the VMs' hosting locations.

5.1.1.4 Virtual Machines analysis

Jebessa et al. [Jebessa13] discuss a case study of analysing the dependency network of a virtual machine with few software and it discusses that security threats can be minimized in optimized systems by just eliminating unnecessary kernel functionalities. In this respect the work identifies some even bigger challenges than ENTICE faces (namely open source systems can be tightly optimized by analysing their entire source code). Also the paper discusses some aspects of describing the purpose of a virtual machine. These discussions and related works might be of direct input to WP5 activities. **Advantages**: The proposed solution is a pre-requisite for any VM analysis and transformation procedures. **Disadvantages** In order to use this approach, VMs should be declaratively described in a domain-specific language by providing descriptions such as used set of packages, kernel features and system configurations for software, and so on.

Razavi et al. [Razavi14] investigate the effects of VM image size and contents on VM startup time in Amazon EC2. They developed an approach (analysing the dependency graph, creating a pruned dependency graph of the needed components and moving these components into a blank image) for consolidating size and contents of VMIs within the ConPaaS project.

Advantages Their proposed approach applied to the VMIs resulted in four times reduction of the disk size, three times speedup for the VM start-up time, and three times reduction of storage costs, compared to an unmodified VMI. **Disadvantages**: The authors do not take into account that VM start-up times are not constant (especially on Amazon).

Menzel et al. [Menzel13] provide an analysis of one of the regions of Amazon's cloud. Through this analysis they show that only a few users dare to publish virtual machine images (because of the technical issues faced while constructing one). And they highlight that even just focusing on a single region one can find over 10 thousand images stored which are often based on similar software thus these images would be ideal for identifying their smaller building blocks. The paper also provides insight on how to identify configuration setups of virtual machine images. The approach relies on starting up each and every tracked virtual machine image. And the authors require the deployment of custom configuration management software (if not already installed there) on the virtual machine created from the image, before they can extract configuration information. In work package 3, we expect to not just analyse the configuration of the images but also other similarities amongst the virtual machine images. **Advantages**: the paper shows that one can achieve significant performance improvements even in commercial clouds by just adhering some image content requirements. **Disadvantages**: Requires specific components to be pre-installed on every VM. Since creating a virtual machine for every new image could be costly for analysis the Task 3.2 will mostly focus on techniques that do not require instantiating the image under analysis.

Luo et al. [Luo13] introduce S-CAVE, an extension to contemporary virtual machine monitors (VMM) so they can efficiently manage caching and access to the more and more widespread SSD drives that represent the ephemeral storage for VMs. **Disadvantages:** Although this paper shows significant improvements for VMs in environments with multi tenancy, the VMs must be running and they also expect the cloud provider to upgrade their VMMs so they also support the S-CAVE extension. As the objectives of ENTICE are mainly targeted on pre-utilization behaviour of the VMs and their images, S-CAVE cannot have significant effects on the VM instantiation process. On the other hand, the VMMT concept of WP4 could exploit S-CAVE extensions if multiple ENTICE created VM images are instantiated on a single host. Thus, ENTICE should investigate when it is possible to detect both being on a single host and having a host that supports S-CAVE. If one can detect these circumstances then the VMMTs running on the single host should synchronize their installation and configuration activities allowing the VMs to exploit S-CAVE based caching

Malhotra et al. [Malhotra13] propose VMCloner to reduce time needed for cloning virtual machine images by using multi-threaded cloning. The designed framework improves the

behaviour of Virtual Box by reducing the time taken by clonevdi API to create multiple copies of the virtual disk. The clonevdi function simply reads a bunch of blocks from the source drive and writes them to the destination drive. By sequential block-by-block copying the cloning time increases with the increase in virtual disk size and can be optimized in the range of 32 KB to 16 MB as buffer sizes. VMCloner implements multi-threaded VM cloning with an optimum buffer size to decrease cloning time. They observed that the time taken to clone a virtual machine decreases with the increase in cluster size. For small sized virtual disk, no significant time decrease is seen as the context switching is high as compared to the size of the disk. **Advantages:** The multi-threaded cloning mechanism developed by the authors is efficient for larger sized virtual disk cloning. **Disadvantages:** the proposed technique is restricted to single hosts as well as to the proprietary VM image format of Virtual Box.

5.1.1.5 Image reconstruction

The solution proposed by Zhang et al. [Zhang13] is focusing on providing software on demand, i.e. deploying VMs without software installation and adding the necessary program components on the fly. The solution is based on a "double isolation" mechanism, introducing user level virtualization of the on-demand software and a central distribution system in order to decouple application software from VM to improve the deployment flexibility. The user-level virtualization isolates applications from the OS (and then the lower-level VM); so that a user can choose which software will be used after setting the virtual machines' configuration. Moreover, the chosen software is not pre-installed (or pre-stored) in the VM image; instead, it can be streamed from the application depository on demand when the user launches it in a running VM to save the storage overhead. **Advantages**: During the whole process, no software installation is needed. Further, the enormous existing desktop software can be converted into such on-demand versions without any modification of source code. **Disadvantages:** The current concept is based on centralised storage servers that hinder scalability and deployment is realised for applications but not for VM images.

5.1.1.6 Use cases

Shiraz et al. [Shiraz13] analyse how VM deployment and management influences execution time specifically in mobile cloud environment when applications are offloaded to data centres in order to augment computing potentials of smart mobile devices. VM is one of the prominent approaches for offloading computational load to cloud servers where the utilisation of additional computing resources related to deployment and management of VM on Smartphone represent a challenge. The objective of this work is to ensure that VM deployment and management requires additional computing resources on mobile device for application offloading. This paper analyses the impact of VM deployment and management on the execution time of application in different experiments in simulation environment by using CloudSim. The analysis concludes that VM deployment and management require additional

resources on the computing host. Therefore, VM deployment is a heavyweight approach for process offloading on smart mobile devices. **Advantages**: They showed that VM deployment and management do have an impact on the execution time of applications in different experiments, which can be taken into account for ENTICE investigations. **Disadvantages**: As the authors state VM-based process offloading is a heavyweight approach. Such an approach is not necessarily the only or best solution, considering time, and computational resources needed and overhead. Rather simpler approach, e.g., container-based one, could be more suited and should be investigated.

Blomer et al. [Blomer14] describe an approach, how the HEP community tries to optimize image creation. They propose a so-called CernVM File System to separate scientific software used for experiments and the execution environment. This solution allows the use of a small bootable CD image that comprises only a Linux kernel while the rest of the operating system and the actual application is provided on demand by CernVM-FS. **Advantages**: This approach speeds up the initial instantiation time and reduces virtual machine image sizes by an order of magnitude. Security updates can also be distributed instantaneously through CernVM-FS. It also supports versioning and snapshot creation. **Disadvantages**: the approach necessitates the VMs to be always constructed during runtime, which might introduce some delays for frequently requested but barely changing images.

Karve et al. [Karve13] introduce how clouds are used in agile development, and how images evolve during the development process of several software systems maintained by a single company. **Advantages**: As the evolution is rather iterative, the authors advise not to transfer images between development and production systems, instead they suggest a method to identify portions of the image that are already present on the production system, and therefore they reduce the transfer between the production and development image repositories to the minimum possible. **Disadvantages:** Although this technique is very promising, it is limited to the above mentioned two repository scenario (the mechanics of distributed repository management and information propagation about image contents are not addressed) while they also use their proprietary techniques to resolve the identified issues thus reduce the applicability of their solutions in commercial cloud environments where there is little chance to alter virtual machine image repositories.

Mao et al. [Mao12] investigate the start-up time of cloud VMs across three cloud providers: Amazon, Rackspace and Microsoft Azure (for Windows based VMs). The authors analyse the relationship between the VM start-up time and time of the day, OS image size, instance type, data centre location and multiple instances started. They also investigate the VM start-up time of Amazon EC2 Spot instances. They found that Spot instances show a longer waiting time and greater variance compared to on-demand instances.

The authors use the duration from the time of issuing VM acquisition requests to the time that the acquired instances can be logged in remotely as the VM start-up time. They argue that it is a more precise and direct measurement than relying on the status tags provided by the infrastructures. They authors found that instantiating Azure Windows images takes more time than Linux images on Rackspace or Amazon EC2 since the Windows images tend to be larger and for some reason the transfer speed between instance and blob storage is low. Within each provider VM start-up time seem not to be affected by much by time of day, datacentre location. However the instantiation time is dependent on the image size. Compared to a then two year old study Azure's instantiation speed increased by 200 seconds, but Amazon EC2 remained unchanged.

VM image creation and storage technologies comparison

Table 32 summarizes a comparison of the technologies described above.

Table 32: SoA related with the project objectives, WP objectives and tasks

Reference	T3.1	T3.2	T3.3	T4.1	T4.2	T4.3	T5.1	T6.2	Obj 1. Creation of lightweight VM images through functional descriptions	Ob. 2. Distributed lightweight image storage	
[Menzel13]	X								1 Automated image synthesis		
[Zhang13]						X					
[Blomer14]								X		6 Opt.	
[Zhao14]		X			X				2 Image analysis, extraction of common patterns		
[Jayaram11]		X									
[Ng11]			X		X						
[Lei14]			X								
[Jebessa13]	X						X				
[Xu14a]			X								
[Kochut12]		X			X	X				4 Image distribution	
[Nicolae15]		X			X						
[Peng12]		X			X						
[Keren09]		X									
[Karve13]								X		6 Opt.	
[Tang11]			X						3 Image interoperability		
[Nguyen13]		X						X			
[Schmidt10]										4 Image distribution	5 Migration, deployment
[Xu14b]						X					6 Opt.
[Razavi13]					X	X					
[Malhotra13]					X						
[Deshpande13]		X								5 Image migration, deployment	
[Zhou13]					X						6 Optimisation
[Mao12]								X			
[Razavi14]	X										
[Bazarbayev13]		X			X						
[Shiraz13]								X			
[Luo13]							X				
[Reich12]					X						

5.1.2 VM image creation and storage technology challenges for ENTICE

Existing methods for VM image creation do not provide optimisation features other than dependency management. This sort of dependency management is based on predefined dependency trees produced by third-party software maintainers. If complex software is not annotated with dependency information, it requires manual dependency analysis upon VM image creation based on worst case assumptions and consequently, is far from optimal in most cases. The resulted VM image sizes hinder any operations related to transferring VM images.

Regarding with existing methods for VM image storage, to manage repositories is an important part of an IaaS Cloud provider. Today, commercial providers rarely expose the technologies they use to store the images for their users. Some providers allow sharing user-created images in a rudimentary way, e.g. using Cloud storage systems like S3 in case of Amazon EC2. Open source systems seldom allow the involvement of third-party repository services, but provide some simple centralised proprietary service instead. The limited sharing capabilities lead to non-optimal movement of VM images resulting in a slow Cloud VM instantiation across Clouds hindering the proliferation of Cloud usage.

5.1.3 VM image creation and storage technology roadmap

The process to be followed to achieve the ENTICE's objectives is described in Table 33. This table shows the relation between the scientific literature previously described and the tasks of the project.

Table 33: Lightweight VM image creation/ Lightweight VM image storage and Technology Roadmap

	Know-why	**Know-what**	**Know-how**
Science and technology roadmap	Identify the problem domain and the scope of lightweight VM image creation and storage.	Analyse the related works and determine the techniques currently used. Understand the research challenges Confront the techniques characteristics and their applicability to the problem. Define the objectives of the optimization for lightweight VM image creation and storage.	Determine the feasible development frameworks and programming languages that will be used to implement the solution.

5.1.4 Technical constraints for ENTICE in lightweight VM creation and storage

The multi-criteria optimisation of transferring VM images is an objective of ENTICE. However, the consideration of device heterogeneity and consequently, the differentiation between

criteria accordingly, is a novel and outstanding dimension. Research in this novel area has to provide an intermediate solution with a significant margin that can be further improved by adaptive combination of the image creation and image storage technologies previously defined.

The creation of a lightweight distributed file system for peer to peer computing based on storing of virtual machine images of the federated system will present a technical constraint which can only be overcome through a close collaboration within ENTICE WP3 and WP4. From this collaboration it is expected a substantial improvement of the IaaS performance with respect to existing methods.

5.2 Multi-objective techniques

The vast majority of existing scheduling research in distributed computing environment, at application and resource management levels, deal with the optimization of more than one objective. Most of the works focus indeed their attention on the trade-off between performance (i.e. execution time, throughput, response time) and resources utilization/wastage, reliability, cost or energy consumption. The employed optimization methodologies differ in the formulations and techniques adopted to solve the problem. We observe two main research directions: problem-based heuristics and bio-inspired meta-heuristics.

This section gathers the existing multi objectives scheduling techniques employed for the allocation of applications and resources in a distributed computing environment. The following sections present those techniques and evaluate their pro and cons in a table. The analysis of the State of the Art presented here is required for the consecution of ENTICE's *objective 3: Autonomous multi-objective repository optimisation* and it is closely related with the implementation of *WP 4 Federated multi-objective VM image repository optimisation*.

5.2.1 Multi-objective Techniques State of the Art

5.2.1.1 Problem-based heuristics

Those solution methodologies exploit heuristics to guide the search towards the space of solutions. The heuristics are problem-specific, they usually are adapted to the problem at hand and they try to take full advantage of the characteristic of this problem [Doğan05]. Most problem-based heuristics are lightweight techniques that cannot be considered as pure multi-objective methods since they only compute a single trade off solution instead of the Pareto front. They can be classified in three groups.

The **aggregation methods (AM)** simplify the multi-objective problem to a single objective optimisation by weighting the objective functions and combining them into one single analytical function [Assayad04] [Doğan05] [Hakem07] [Garg09]. Assayad al. [Assayad04] defined a bi-criteria compromise function which minimization minimizes the schedule length and maximizes

the system reliability. Garg at al. [Garg09] presented a heuristics for scheduling parallel applications on Utility Grids that optimize the trade-off between time and cost minimizing their composite function. Hakem at al. [Hakem07] defined a bi-criteria function like in [Assayad04] and the heuristic, BSA, which produces a schedule that optimises the trade-off between execution time and reliability.

Their main drawback is that the weights are decided a-priori with no knowledge about the application, the infrastructure, and the problem to be solved. Additionally, the weights may represent unfeasible solutions if the relationship between objectives is unknown.

The **constraint programming methods (CSP)** limit the solution search space providing certain constraints. Those methods mainly differ by the way the objective optimization is performed.

A first method is to optimise the objectives in a predefined order while keeping them within certain constraint ranges.

Benoit et al. [Benoit09] proposed a scheduling heuristic which aims at minimizing execution time under a fixed number of tolerated failures and then determines a maximum number of tolerated failures in correspondence of the computed execution time. Sakellariou et al. [Sakellariou07] investigate heuristics that allow scheduling the tasks into resources in a way that satisfies a budget constraint and the execution time. Like in [Benoit09] the optimization is done in a 2-step fashion: firstly the minimum cost is calculated and then for this cost is evaluated the task reassignment with the minimum execution time.

A second method is based on the use of a single objective function which is optimised in a Constraint satisfaction problem. Van et al. [Van10] solved the provisioning and placement problem optimizing the trade-off between energy consumption and resource utilization such as to achieve high server consolidation. The objective trade-off is defined as a weighted function which is maximized through the use of a constraint solver.

The constraint programming methods include some offline preferential information on the system. However, reasonable a-priori constraints are often unknown until the first schedule is computed.

The **mixed integer linear programming methods (MIP)**, similarly to the constraint programming methods, optimise a single object function within certain constraints. Li et al. [Li11] used mixed integer linear programming to define a multi-optimization application placement problem as a sole objective function and a set of constraints. Their heuristic aims to optimize several conflicting objects such as power consumption, response time and resource utilization in Cloud. Calcavecchia et al. [Calcavecchia12] and Stillwell at al.[Stillwell10] use MIP to formulate the VM

placement problem and a heuristics to find the optimal solution in terms of resources utilization and quality of service.

The mixed linear programming methods, like the constraints ones, imply an a priori knowledge of the constraints. Furthermore, the problem formulation is often complex and the constraints cannot be expressed without having to be translated into linear inequalities.

5.2.1.2 Bio-inspired meta-heuristics

Recently meta-heuristic techniques have been successfully used to address the scheduling optimization problems. They can provide better global solutions than problem-based heuristic which are easy to fall into the local optimal solution due to their greedy nature [Blum03]. They can be classified in Genetic and Ant Colony techniques.

The Genetic techniques (GA) are evolutionary inspired techniques that mimic the biological evolution. They are largely used in the optimization problems due to their effectiveness. Garg et al. [Garg11a] proposed a scheduling algorithm based on the use of NSGA-II. The proposed scheduling algorithm aims at optimize two conflicting objects like execution time and cost. Talukder et al. [Talukder07] design MODE, a multi-objective Differential Evolution algorithm, to produce a Pareto set of scheduling solution as trade-off between cost and execution time. Garg et al. [Garg11b] use NSGAII to generate a set of trade-off scheduling solutions according to the users QoS requirements. Lama et al. [Lama11] formulate a multi-objective optimization problem that is to minimize the number of physical machines used, the average response time and the total number of virtual servers allocated for multi-tier applications. The Pareto front is then obtained by using NSGA-II.

As said above, Genetic algorithms can provide better solutions than problem-based heuristics at the cost of a higher computation. Although these algorithms have been applied to simultaneously optimise three objectives, it is known that the performance of multi-objective genetic algorithms seriously degrades beyond three objectives [Khare03].

Lately, ant colony optimization (ACO) techniques have been employed as a solution to the slow convergence speed of genetic algorithms [Gao13] [Ma12]. Their working principle is based on the observation of ant foraging behaviour. Gao et al. [Gao13] use ACO to determine the optimal VM placement that simultaneously minimizes total resource wastage and power consumption. Ma et al. [Ma12] provide a VM placement technique based on ACO to find a balance in multiple conflict objectives, it attempts to reduce the resource wastage, power consumption, and minimize violation of SLA.

5.2.2 Multi-objective techniques comparison

Table 34 summarizes a comparison of the technologies described above.

Table 34: Multi-objective techniques comparison table

Technique	Advantages	Disadvantages	ENTICE's objective	WP related
AM	Relatively fast convergence speed. Can be able to provide a good solution for the optimization problem	Reduce the multi-objective problem to a single objective one Need an a priori definition of the objective weights Can get stuck in a local optimal solution due to the use of a problem based heuristic Does not provide a Pareto set of solutions	Project Objective 3: Autonomous multi-*objective* repository optimization	WP4 objectives: Research heuristics for multi-objective distribution and placement of VM images across the decentralised repository that optimizes multiple conflicting objectives.
CSP	A constraint solver can be used to determine a solution The constraint are able to reduce the search space Can be able to provide a good solution for the optimization problem	Problem formulation is often complex Need an a priori offline preferential information on the system Can get stuck in a local optimal solution due to the use of a problem based heuristic Does not provide a Pareto set of solutions	Project Objective 3: Autonomous multi-objective repository optimization	WP4 objectives: Research heuristics for multi-objective distribution and placement of VM images across the decentralized repository that optimizes multiple conflicting objectives.

Technique	Advantages	Disadvantages	ENTICE's objective	WP related
MIP	MIP solver can be used to determine a solution The constraint are able to reduce the search space Can be able to provide a good solution for the optimization problem	Problem formulation is often complex Need an a priori offline preferential information on the system Can get stuck in a local optimal solution due to the use of a problem based heuristic Does not provide a Pareto set of solutions	Project Objective 3: Autonomous multi-objective repository optimisation	WP4 objectives: Research heuristics for multi-objective distribution and placement of VM images across the decentralized repository that optimizes multiple conflicting objectives.
GA	Problem-independent Techniques Easy to use Can be adapted to many problem Can be more effective than problem based heuristics Provide a Pareto front of solutions	Slow convergence speed	Project Objective 3: Autonomous multi-objective repository optimization	WP4 objectives: Research heuristics for multi-objective distribution and placement of VM images across the decentralized repository that optimizes multiple conflicting objectives.
ACO	Problem-independent Techniques Easy to use Can be adapted to many problem Can be more effective than problem based heuristics Provide a Pareto front of solutions Can converge faster than GA	Needs a definition of intrinsic parameters in order to adapt the technique to the problem at hand	Project Objective 3: Autonomous multi-objective repository optimization	WP4 objectives: Research heuristics for multi-objective distribution and placement of VM images across the decentralized repository that optimizes multiple conflicting objectives.

5.2.3 Multi-objective technology challenges for ENTICE

Optimising parallel and distributed applications in heterogeneous environments, including placement of ready-made VMs for their execution IaaS Clouds, has been shown to be an NP-complete problem [Ullman75], hence no polynomial algorithm for solving it exists (assuming P ≠ NP). While this has been for decades the only parameter of interest, modern heterogeneous federated Clouds are raising today other questions regarding metrics of equal importance such as execution costs and storage requirements. Real world scenarios are therefore confronted with a multi-objective optimisation problem where many of these objectives are conflicting. In these scenarios, there is no single solution that optimises all objectives, but a set of trade-off solutions known as Pareto front. A Pareto front is an essential tool for decision support and preference discovery, whose shape provides new insights and allows scientists to explore the space of non-dominated solutions, possibly revealing regions of interest that are impossible to see otherwise. Concretely, a Pareto front can answer questions such as “are my optimisation goals conflicting?", “can I simultaneously improve execution time and energy efficiency?" and “how large is the trade-off between my optimisation criteria?", or it can ease the selection of the best fitting solution to the user's requirements. For example, it may happen that by conceding 1% in execution time, the energy consumption halves. The Pareto front will reflect this kind of information, which is often not visible without computing it.

5.2.4 Multi-objective technology roadmap

Existing research is mostly limited to two objectives and employs a-priori methods that do not approximate the Pareto front, but aim to find a single compromise solution through aggregation or constrained programming. Moreover, most existing works target scheduling of applications to resources and do not target optimised placement of VM fragments in a repository distributed across several Cloud sites. To the best of our knowledge, our work will be the first truly multi-objective approach able to optimise the set of trade-off solutions with respect to distributed VM placement in Cloud systems and involving three conflicting objectives: time, cost and storage.

The process to be followed is described in Table 35.

Table 35: Multi-objective Science and Technology Roadmap

	Know-why	Know-what	Know-how
Science and technology roadmap	Identify the problem domain and the scope of the optimization.	Analyse the related works and determine the optimization techniques currently used. Understand the research challenges. Confront the techniques characteristics and their applicability to the problem. Define the objectives of the optimization.	Determine the feasible development frameworks and programming languages that will be used to implement the solution.

5.2.5 Multi-objective technical constraints for ENTICE

The VM images distribution has to consider network and storage constraints in any new placement. The Image composition time is determined by number and size of the different image parts as well as by their repository. Images parts hosted on the same local repository are more quickly available than remote ones. However not all the images parts can be locally stored, their high replication is limited by the storage capacity of each repository. On the other hand a low replication increases the number of transfers over the network with a resulting latency which impacts on the image composition time.

The design of the ENTICE architecture will help to have a clear definition of those and possibly even more technical constraints.

5.3 Cloud Elasticity

Existing definitions of cloud elasticity are inconsistent and unspecific. Furthermore, a clear distinction between the related two terms scalability and efficiency is not defined. Herbst et al. [Herbst13] propose a precise definition of elasticity, identify main differences to scalability/efficiency and present a set of appropriate elasticity metrics. Islam et al. [Islam12] define a numeric score that reflects the financial penalty of under-/over-provisioning to a particular consumer.

This section present the analysis of the state of the art and the identification of challenges for ENTICE. The analysis of the State of the Art presented here is directly linked with ENTICE's

objective 4: Elastic resource provisioning and it is closely related with the implementation of *WP 4 Knowledge based modelling and reasoning*.

5.3.1 Cloud Elasticity State of the Art

5.3.1.1 Simple-Yet-Beautiful Language

Dustdar and Copil et al. [Dustdar12, Copil13] developed programming directives for elastic computing, called SYBL[2] (Simple-Yet-Beautiful Language) which specifies possible Java annotations and runtime functions to control elasticity in cloud-based applications. Based on SYBL/rSYBL and ReSpecT[3], Mariani and Dustdar et al. [Mariani14] propose a novel approach, called coordination-aware elasticity.

5.3.1.2 MELA

Moldovan et al. [Moldovan15] defined two novel concepts, called elasticity space and elasticity pathway. Based on these concepts, the authors develop an elasticity analytics as a service framework, called MELA, for real-time monitoring and multi-level analysing the elasticity of cloud services in multi-cloud environments. MELA categorizes the monitoring data in three dimensions, which were already defined in Dustdar et al. (2011). The three dimensions are cost (metrics that influence the cost), quality (metrics characterising QoS) and resource (metrics for resource usage and allocation information).

5.3.1.3 Control, Monitoring and Testing (CoMoT)

Truong and Dustdar et al. [Truong14] describe a novel PaaS for elasticity, namely CoMoT (Control, Monitoring and Testing). The elasticity monitoring and analysis module of CoMoT is based on MELA [Moldovan15], which sends the elasticity information to the elasticity control, based on SYBL [Copil13].

5.3.1.4 PRedictive Elastic ReSource Scaling (PRESS)

Gong et al. [Gong10] present a novel PRedictive Elastic ReSource Scaling (PRESS) scheme for cloud computing. PRESS combines signal processing (Fast Fourier Transformation) and statistical learning algorithms (Markov Chains) to achieve real-time predictions of dynamic application resource demand.

5.3.1.5 TIRAMOLA

TIRAMOLA is a similar approach to PRESS. It was used by Tsoumakos et al. [Tsoumakos13]. It is an open-source framework that performs automatic resource resizing of NoSQL clusters.

[2] http://www.infosys.tuwien.ac.at/research/viecom/SYBL/

[3] http://apice.unibo.it/xwiki/bin/view/ReSpecT/

5.3.1.6 ADVISE

Copil et al. [Copil14] present a framework, called ADVISE, for estimating and evaluating the elasticity behaviour of cloud services. To estimate the elasticity behaviour, ADVISE applies clustering techniques on previous gathered information like service structure, service runtime and control processes.

5.3.2 Cloud Elasticity techniques comparison

Table 36 summarizes a comparison of the cloud elasticity technologies described above.

Table 36: Cloud Elasticity techniques comparison table

Technique	Advantages	Disadvantages	ENTICE's objective	WP related
SYBL/rSYBL	Defines 3 directive class names: MONITORING, CONSTRAINT and STRATEGY	Need to modify the user application using a set of preprocessing directives and to recompile it. Does not define elasticity metrics. Metric.	Project objective 4: Elastic resource provisioning	WP5 objectives: - introduce a definition of elasticity metric for cloud applications and infrastructures. - use it as benchmark to improve the elasticity
ReSpecT	Supports: Implementation of new primitives, modification of existing primitive semantics, change of coordination laws at run-time Couple of reaction specification tuples	Does not define elasticity metrics.	Project objective 4: Elastic resource provisioning	WP5 objectives: - introduce a definition of elasticity metric for cloud applications and infrastructures. - use it as benchmark to improve the elasticity

Technique	Advantages	Disadvantages	ENTICE's objective	WP related
PRESS	Implemented on Xen Fast Fourier Transformation for signature-driven resource demand prediction Markov chain model for state-driven resource demand model Examines a window of recent resource demands to predict future resource demand Does not assume the workload is cyclic Provides resource prediction for both repeating and non-repeating patterns. Integrates online resource demand prediction with dynamic VM resource scaling Avoids the need to profile offline by deriving resource demands as the application runs Is application and platform agnostic	Implemented only on the Xen hypervisor by exploiting its virtualization mechanisms. Does not define elasticity metrics.	Project objective 4: Elastic resource provisioning	WP5 objectives: - introduce a definition of elasticity metric for cloud applications and infrastructures. - use it as benchmark to improve the elasticity
TIRAMOLA	Can employ any cost or optimization model to its functionality Open-source framework for NoSQL clusters Multi-grained monitoring (using Ganglia) that uses a rich set of metrics to realize any resize policy Resize decisions are modelled as Markov Decision Process Strives to be customizable and robust	Elasticity measurement framework for only NoSQL data store.	Project objective 4: Elastic resource provisioning	WP5 objectives: - introduce a definition of elasticity metric for cloud applications and infrastructures. - use it as benchmark to improve the elasticity

Technique	Advantages	Disadvantages	ENTICE's objective	WP related
MELA	Cloud services Is an elasticity analytics as a service Provides features for real-time multi-level analysis of elastic Adopts the cloud user perspective and provides a customisable mechanism for mapping system level monitoring data to the running service structure Not only focuses on monitoring and relies on data from existing solutions, structures and enriches it, to finally analyse the elasticity of the cloud service Introduces concepts and techniques for extracting elasticity boundaries	Does not define elasticity metrics.	Project objective 4: Elastic resource provisioning	WP5 objectives: - introduce a definition of elasticity metric for cloud applications and infrastructures. - use it as benchmark to improve the elasticity
CoMoT	Is a PaaS for elasticity Uses MELA for elasticity monitoring and analysis Uses SYBL for elasticity control Supports native software-defined elastic systems (SES)	It is a PaaS with elasticity guarantees for its running applications.	Project objective 4: Elastic resource provisioning	WP5 objectives: - introduce a definition of elasticity metric for cloud applications and infrastructures. - use it as benchmark to improve the elasticity

Technique	Advantages	Disadvantages	ENTICE's objective	WP related
ADVISE	Framework for estimating and evaluating cloud service elasticity behaviour Gathers service structure, cloud infrastructure information deployment, service runtime and control processes Uses clustering techniques to identify elasticity behaviour Useful tool for elasticity controllers for improving the quality of control decisions		Project objective 4: Elastic resource provisioning	WP5 objectives: - introduce a definition of elasticity metric for cloud applications and infrastructures. - use it as benchmark to improve the elasticity

5.3.3 Cloud elasticity technology challenges for ENTICE

The development of performance-oriented applications involves many cycles of code editing, execution, testing, performance analysis, and tuning. Some performance metrics, such as scalability, efficiency, or speedup, normally require manual testing of numerous VM configurations and sizes for different target architectures. In modern Cloud environment, an important emerging metric is the elasticity of an application and its capability to adapt to different resource sizes and configurations, requiring advanced multi-experimental optimisation and analysis techniques. To this date, there is little understanding of the term elasticity in Cloud computing and tool support to improve applications and tune middleware infrastructures towards this metric. Table 36 shows different frameworks to monitor and analyse elasticity, as well as approaches to control the elasticity behaviour. However, a uniform elasticity metric is still missing. The challenge is to determine all related elasticity measures and their relations, to finally define a uniform elasticity metric.

Moreover, different pattern matching techniques have to be analysed to find the best technique for a well suited elasticity prediction. Furthermore, a feasible granularity of elasticity measurements has to be researched. The challenge is to improve the elasticity of the ENTICE use cases and to minimize the overhead, caused by the measurements, the analysis and the storage among others.

5.3.4 Cloud elasticity technology roadmap

As part of ENTICE we will formally introduce a definition of an elasticity metric for Cloud applications and infrastructures and use it as a benchmark in quantifying and improving the elasticity of the ENTICE use cases and middleware, as presented in the project objective 4. In the

process to obtain the elasticity formal definition the research topics defined in Table 37 will be investigated.

Table 37: Cloud Elasticity Science and Technology Roadmap

	Know-why	**Know-what**	**Know-how**
Science and technology roadmap	Identify the ENTICE use cases and the scope of the elasticity metric	Analyse the related works and determine elasticity measures Formally introduce a definition of an elasticity metric Use the new metric as benchmark in quantifying and improving the elasticity of the ENTICE use cases	Determine the set of meaningful elasticity measures Observe the relations between the measures and define a metric Determine the feasible frameworks that will be used to measure elasticity

5.3.5 Cloud elasticity technical constraints for ENTICE

The main technical constraint that can be envisaged at this early stage is to find an appropriate and generic metric that can be used to quantify cloud elasticity in the federated framework proposed. This occurs because federation has specific requirements that common non-federated cloud do not have.

Another technical constraint identified is the generalization of the metric considering the implementation the three ENTICES pilot cases. It is possible that a major contribution can be obtained with the implementation of the FCO pilot, which is an orchestrator which will manage the elasticity of the cloud infrastructure.

5.4 Knowledge based modelling and reasoning

Non-traditional triple-store databases provide high performance on federated infrastructure to support research in a wide range of disciplines. These databases are the backbone of many companies (e.g., Google Big Table [Chang08], Amazon Dynamo [DeCandia07], Facebook Cassandra[Lakshman10], and Apache HBase [George11]). Actually, they can be built simply and provide high performance on federated hardware infrastructure. There exist many non-traditional database tools which have different performance currently available and are widely used for government applications [Accumulo2015]. Non-traditional databases allow data to be ingested and indexed with very little a priori knowledge. This allows new classes of data and queries to be added to the existing tables without modifying the schema of the database. Hence, it can be used to merge unstructured, semi-structured data and share it across the network. A very important point here is how to use an efficient storage system to store

structured, semi-structured and unstructured data that can be fed by different feeders and can be used to serve many client requests concurrently.

RDF (Resource Description Framework) is a standard model for data interchange. Using this simple model, it allows all kinds of data to be mixed, exposed, and shared across different applications. Producers can produce the RDF data and share on the network where as consumers can crawl it to use in their applications. This improves the reusability of existing information without having to create new one. Moreover, "SPARQL Protocol and RDF Query Language" (SPARQL) [Prud'hommeaux15] is the RDF query languages that can be used to query RDF data.

5.4.1 Technologies and Techniques

5.4.1.1 Resource Description Framework

RDF [Klyne15] is a framework or model that was proposed by the W3C, to represent linked data on the web. It is used to describe a model that makes statements to connect two resources or to connect resource to its value. RDF does not have pre-defined schemas to represent data. However, RDF has its own schema language called Resource Description Framework Schema (RDFS). RDF statements are represented as subject-predicate-object. It is called a triple. It is similar to the object, relationship, and value in the object oriented paradigm, but RDF is not actually object oriented. Here, the subject can be mapped to the object, the predicate can be mapped to the relationship between the object and its value, and the object can be mapped to the value. For example, RDF statement like ":CloudProvider :hasCloudServer :CloudServer1" represents :CloudProvider is a subject, :CloudServer1 is an object and :hasCloudServer is a predicate to connect both. All such RDF statements form a RDF graph, where nodes represents subject, object and links define the relationship among them. For example, an RDF graph for Cloud Service Level Agreement (SLA) is represented as shown in the Figure 26 below:

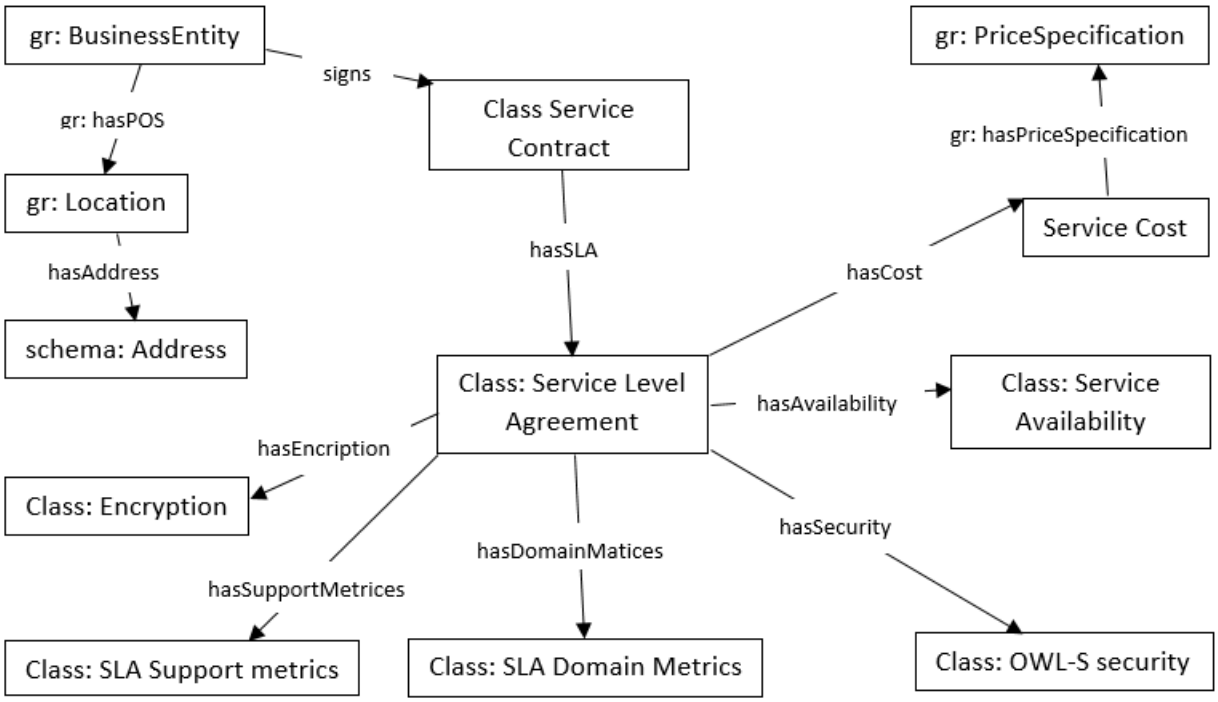

Figure 26: Example of RDF graph for Cloud service level agreement [Joshi15]]

Regarding the Figure 26 above, the SLA class consists of properties that are common across all Cloud applications. These include service fields like service delivery mode, and service availability; and security related attributes like Cloud location, data encryption and data deletion. The SLA class consists of hasDomainMetrics property whose domain is the SLA Domain Metrics class which consists of properties that are specific to the service domain. For our ontology we have included properties of Cloud storage domain like storage size, backup, etc. The hasSupportMetrics property in the SLA class has domain class SLA Support Metrics, which consists of service support properties like outage notification, resolution time, response times, etc. These properties are used to determine the performance of the service, and so can be used in the consumption phase to automate service monitoring.

RDF data is represented as directed graphs. So, we can infer basic semantics from the information when no RDF schema is available. RDF has also the ability to merge two different data sources without having defined schemas. Hence, it can be used to merge unstructured, semi-structured data and share it across the network. Producers can produce the RDF data and share on the network where as consumers can crawl it to use in their applications. This improves the reusability of existing information without having to create new one.

RDF has several serialization formats, each having its own syntax and specifications to encode the data. Some of the common serialization formats that RDF supports are Turtle, N-Triples, N-Quads, JSON-LD, N3 and RDF/XML. The following Table 38 shows a small section of a sample

RDF data in which a metric named “MetricPacketLoss” is one of properties defined for the class named “QoSMetrics”. Actually, “rdfs:domain” says for what class the property is defined. That means instances of this particular class (“QoSMetrics”) can have that property (“MetricPacketLoss”). “rdfs:range” says what type of values can the property take. That means the metric “MetricPacketLoss” can take “double” values. Also, “rdfs:label” is used to provide a human-readable version of a resource's name.

Table 38: Sample RDF Data

```
<?xml version="1.0"?> <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntaxns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="http://webprotege.stanford.edu/MetricPacketLoss">
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
   <rdfs:label>MetricPacketLoss</rdfs:label>
   <rdfs:domain rdf:resource="http://webprotege.stanford.edu/QoSMetrics"/>
   <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
</rdf:Description>
</rdf:RDF>
```

5.4.1.2 SPARQL Protocol and RDF Query Language

SPARQL [Prud’hommeaux15] also stands for SPARQL Protocol and RDF Query Language. SPARQL, as its name suggests, is a RDF query language, which is used to get the information stored in the RDF format. It was recommended by W3C. A simple SPARQL query consists of triple patterns, which represents subject, predicate, and object. Complex SPARQL queries also consist of conjunctions, disjunctions, and optional patterns. There are many tools available to construct SPARQL queries, to translate SPARQL queries to other query languages such as SQL, to run SPARQL queries on NoSQL databases such as MongoDB, Cassandra.

Table 39 illustrates a simple SPARQL query. More specifically, it should return the data set of all the person names. It has two query patterns. One is to find the triples of type person and other is to find the persons who have name associated with them.

Table 39: SPARQL Query Example

```
SELECT ?p ?o
WHERE {
        ?s <http://webprotege.stanford.edu#MetricTime> ?p .
        ?s <http://webprotege.stanford.edu#MetricPacketLoss> ?o
}
```

This is the simplest query, as it needs to match only two triple patterns. “SELECT” identifies the variables to appear in the query results. Within the “WHERE” clause, we have two triple patterns. Two patterns are jointly used together in order to retrieve all values of both properties called MetricTime and MetricPacketLoss. SPARQL should join the results from pattern 1 and pattern 2 and output the combined result dataset. A placeholder (“?”) in front of a variable indicates binding the variable to corresponding parts of each triple. SPARQL also follows a procedure called federated query, where it will distribute the query to end points, computed, and results submitted.

SPARQL supports four different query forms in order to read RDF data from the databases. SELECT Query is used to get information stored in the database and results are stored in tabular format. Construct query is used to get information from the repositories and results will be converted to RDF format. ASK query is used to get simple true or false information. DESCRIBE query is used to get the RDF graph from the SPARQL endpoints.

5.4.1.3 Web Ontology Language (OWL)

Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them. They encode knowledge in a domain and also knowledge that spans domains. In this way, knowledge reusability is promoted by ontologies. The availability of machine-readable metadata would enable automated agents and other software to access the Cloud environment more intelligently. The agents would be able to perform tasks and locate related information automatically on behalf of the human. Cloud ontology is able to provide a common definition of concepts related to Cloud domains and to describe Cloud components like infrastructures, platforms and also services. There are several languages available to develop ontology like Developing Ontology-Grounded Methods and Applications (DOGMA) [Spyns08], Frame Logic (F-Logic), LOOM, Web Ontology Language. The OWL is designed for the purpose of those functionalities that need to process the content of information instead of just presenting information to the humans. OWL [Dean15] is a part of semantic frameworks, used to represent rich, complex information (knowledge) about individuals, and relation between individuals. OWL is part of semantic web technologies. The documents represented in OWL are called ontologies. Each ontology consists of complete knowledge about a specific domain. OWL language is more useful to represent semantics. Documents represented in OWL can be interpreted and verified by computers to ensure their consistency with respect to knowledge.

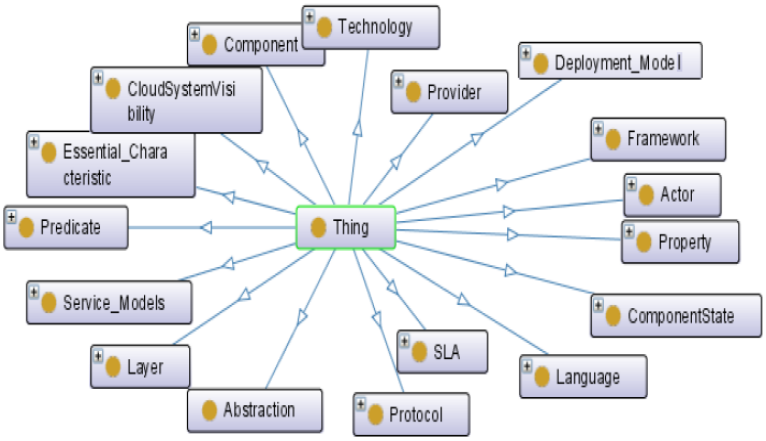

Figure 27: Cloud Ontology

Based on integrated infrastructure of resource sharing and computing in distributed environment, Cloud computing involves the provision of dynamically scalable and provides virtualized resources as services over the Internet. These services, especially in terms of IaaS, also bring a large scale heterogeneous and distributed information which poses a great challenge in terms of the semantic ambiguity. It is critical for users in Cloud computing environment to provide intelligent service and precise information; because the process of developing, deploying, executing Cloud applications is strongly influenced by the specifics of the Cloud providers.

There are several tools available for developing the ontologies like OntoEdit [Sure02], WebODE [Arpirez03], Protégé [Protege15] etc. Protégé fully supports the latest OWL and RDF specifications from the W3C. Protégé is supported by a strong community of academic, government, and corporate users, who use Protégé to build knowledge-based solutions in areas as diverse as biomedicine, e-commerce, and organizational modelling. Protégé's plug-in architecture can be adapted to build both simple and complex ontology-based applications. Developers can integrate the output of Protégé with rule systems or other problem solvers to construct a wide range of intelligent systems. It is based on Java, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development. Figure 28 shows a part of Protégé developing environment which depicts the desired network-based QoS monitoring system; the owl:topObjectProperty is the class of all object properties, and hence is a binary predicate relating all individuals in the model to all individuals in the model. The owl:topDataProperty is the class of all data properties, and hence is a binary predicate relating all individuals to all literals. In other words, the role of owl:topObjectProperty is that it connects all possible pairs of individuals. The role of owl:topDataProperty is that it connects all possible individuals with all literals. Annotation properties are binary predicates that provide informal documentation annotations about

ontologies, statements, or IRIs. For instance, the rdfs:comment annotation property is used to provide a comment.

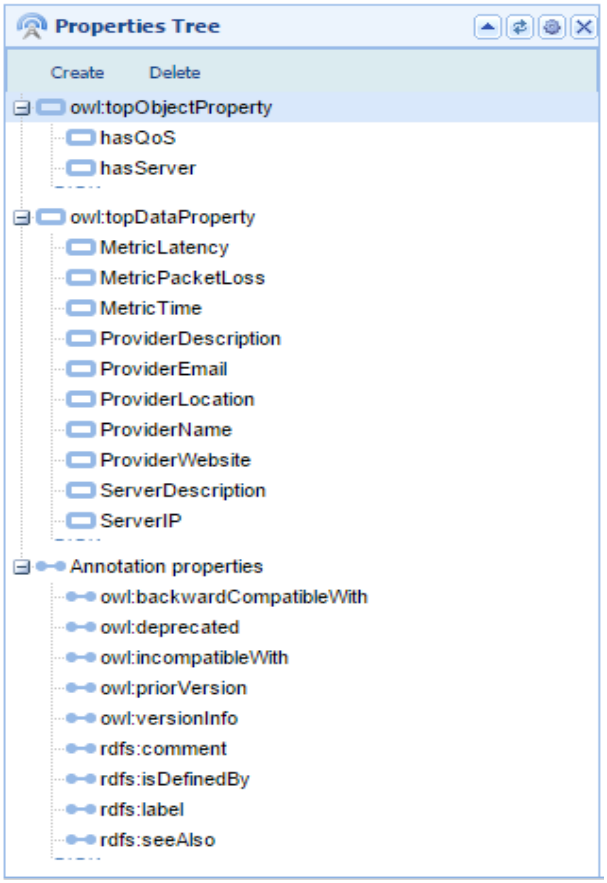

Figure 28: Developing environment for ontology

5.4.2 Comparison among RDF Database Tools

An overview of frameworks and applications with the ability to store and to query RDF data is provided in the following Table 40.

Table 40: Overview of some available RDF Triple Stores

Name	Programming Language	Supported Query Language	Evaluation	License
AllegroGraph	Lisp	SPARQL	Yes	Commercial
Virtuoso	Java	SPARQL	No	Commercial – Open Source
Sesame	Java	SPARQL - SeRQL	Yes	BSD style
Open Anzo	Java	SPARQL	Yes	Eclipse public
Big Data	Java	SPARQL	No	GPL
OWLIM-Lite	Java	SPARQL - SeRQL	No	GNU
Jena Fuseki	Java	SPARQL - RDQL	Yes	Open Source

5.4.3 Technology Challenges

An important challenge confronting potential adopters is the complexity of Linked Data technologies such as RDF/XML, RDFS, OWL and SPARQL. There is an apparent lack of tools and applications for creating Linked Data. The human interaction with Linked Data is not as user friendly and intuitive as Web 1.0. HTML presents data in a friendly manner for users to view; however, data formatting is missing in Linked Data as it is intended for machines to understand. The technology itself is still somewhat young. It's rather complicated and it's not widely understood and appreciated, so there is a social dimension to it in a sense of helping the community to understand and appreciate what it can do with it.

One of the major barriers in the utilisation of linked data is the relative lack of tools for practitioners. Many agreed that there is a great need for more user-friendly tools that will allow archivists to use and understand linked data, and that can demonstrate the potential uses and benefits of linked data.

In addition, applications face many challenges in order to search, integrate, and present the data in a unified view. There are thousands of ontologies (third party and user-generated) used to describe the data. Data fusion requires data integration and aggregation from different sources written in different languages thus requires data cleansing (including deduplication and removal of irrelevant or untrustworthy data) and mapping of the schemas used to describe the data.

Another challenge in Linked Data is related to the core technology that it uses. OWL has significant expressivity limitations. OWL 2.0 was introduced to resolve some of the shortcomings of OWL 1.1 such as expressing qualified cardinality restrictions, using keys to uniquely identify data as well as the partOf relations (asymmetric, reflexive, and disjoint). In addition, OWL is written in RDF (triples) which makes it complex to express relations such as

"class X is the union of class Y and Z." Essentially, OWL is a description language based on first order logic and it is unable to describe integrity constraints or perform closed-world querying [Dean15].

Datasets in the Linked Data Cloud are very heterogeneous and of varying quality. There is neither a quality assurance nor any kind of approval process for deciding if a dataset is acceptable. Linked Open Data represents basically a grass root movement where everybody can just publish their data using RDF and link it to other existing datasets. Hence, there are various problems when dealing with diverse schemas, sparse metadata, and limitations of query interfaces.

5.4.4 Technology Roadmap

Cloud computing is a computing paradigm which allows access of computing elements and storages on-demand over the Internet. Virtual Appliances, pre-configured, ready-to-run applications are emerging as a breakthrough technology to solve the complexities of service deployment on Cloud infrastructure. However, an automated approach to deploy required appliances on the most suitable Cloud infrastructure is neglected by previous works which is the focus of the ENTICE project. Ontologies offer the means of explicit representation of the meaning of different terms or concepts, together with their relationships. They are directed to represent semantic information, instead of content. With OWL complex relationships and constraints can be represented in ontologies. In the ENTICE project, a main part of the ontology should represent an IaaS infrastructure model. Using the OWL [Dean15] it is possible to add create richer vocabularies to describe the classes, relationships between classes, comparison between classes, cardinality constraints and characteristics of the properties. An OWL class represents an element in the domain, for example, an instance of the class "CloudServer" would represent a machine in a Cloud provider. At first stage of the project, Jena Fuseki server will be used as a store of data related to the Cloud environment. From literature point of view[KJoshi15], Fuseki can be easily queried for continuous QoS monitoring since it is in a machine-readable format. Additionally, we will go with Fuseki as it is an Apache product that provides all its own features as open sourced. Actually, the use of the Fuseki module enables the exposure of RDF triples as a SPARQL end-point which is accessible through a REST API.

The following Table 41 shows the Science and Technology Roadmap.

Table 41: Science and Technology Roadmap

	Know-why	Know-what	Know-how
Science and Technology Roadmap	Defining the challenges and problem domain and also the area of knowledge base (KB)	Analyze the related works and determine the KB techniques currently used Understand the research challenges related to KB Confront the techniques characteristics and their applicability to the problem	Identifying the useful tools and programming languages that will be used to implement the KB

5.4.5 Technical Constraints

The Resource Description Framework pioneered by the W3C is increasingly being adopted to model data in a variety of scenarios, in particular data to be published or exchanged on the Web. Managing large volumes of RDF data, the sheer size, the heterogeneity and the further complexity brought by RDF reasoning are the most challenging constraints. As evolved from the semantic web, to work with RDF and Linked Data, there are plenty of tools, engines and languages to create datasets and ontologies, to visualise them, to store them in special databases, provide them with or without restrictions, to retrieve and query them. Almost every tool has its unique features and we cannot simply decide which one is better. The evaluation of RDF databases is based on three categories:

- Technologies
  - Software producer provides details about the company implementing the framework.
  - Associated licenses shed light on the usage of the frameworks, whether it can be used in business applications or not.
  - Project documentation should be rather complete. Furthermore, tutorials should be available supporting the work with these systems especially in the period of vocational adjustment.
  - Support is the last basic criteria. Support should be covered for example by an active forum or a newsgroup.
- Architectural facets of the considered frameworks
  - Extensibility is a very important criterion for the integration of new features, e.g., to optimize the existing working process. One of these features could be the implementation of new indices, which accelerate the performance and advance the efficiency of the entire system.
  - Architectural overview provides an insight into the structure of the framework and the used programming language.

- OWL should be supported by the databases, because it enlarges the semantic expressiveness of RDF especially as far as reasoning is concerned.
- Available query languages is another point of interest, is there support for other RDF addressing query languages in addition to SPARQL.
- Interpretable RDF data formats are not part of central focus. The most important formats should be covered by the frameworks from the point of completeness.

- Expressiveness of SPARQL queries and the performance of the frameworks / applications

5.5 Discussion and conclusions

Existing research mostly focuses on pre-optimising algorithms, which are not applicable to already available VM images. With its VM synthesiser, ENTICE will extend pre-optimising approaches so that image dependency descriptions are mostly automatically generated. The project will also introduce new comprehensive post-optimising algorithms so that existing VM images can be automatically adapted to dynamic Cloud environments.

Research in the area of image storage management targets only single Cloud environments. Even within a single Cloud, the proposed techniques are not mature enough to be applied in a commercial scenario. ENTICE will research and develop reliable and efficient VM image management method that reduces transfer costs, fine-tuned to federation-wide scenarios.

Multi objective optimization techniques aim to find a solution or a set of solutions as a trade-off between conflicting objectives. In the previous section several techniques have been introduced outlining their pros and cons and their working principles.

The Bio-inspired meta-heuristics, compared with the Problem based heuristics, have more chances to provide a better solution. However their convergence is a time costly process due to the vastness of the search space explored. On the other hand, the Heuristic methods exploit a local search approach which can easily get stuck in local optimal minimum. Furthermore, they do not provide a Pareto front but a single solution. However, the determination of a set of solutions, rather than a sole one, could be essential in a multi-criteria decision problem where one could be interested in the compromise between several objectives.

The elasticity metric is a fundamental concept for modern cloud environments. In the previous section several approaches have been introduced. Some approaches address programming directives [Copil13] for controlling the elasticity behaviour or monitor the current resource usage, analyse it and finally try to predict the resource usage in the future [Gong10,Tsoumakos13,Copil14,Moldovan15, Truong14].

The current state of the art shows that there is no formal definition of an elasticity metric and no single best solution to monitor and predict elasticity. Therefore, ENTICE will formally introduce a definition of an elasticity metric for cloud applications and infrastructures and use it

as a benchmark in quantifying and improving the elasticity of the ENTICE use cases and middleware.

Linked-data and RDF triple-stores offer the following advantages over RDMS (Relational Database Management System):

- No need to create schemas: RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed.
- No need to link tables because you can have one to many relationships directly.
- Universal, not local (more flexibility and easy to be integrated into new data): By using HTTP URIs as globally unique identifiers for data items as well as for vocabulary terms, the RDF data model is inherently designed for being used at global scale and enables anybody to refer to anything. Clients can look up any URI in an RDF graph over the Web to retrieve additional information. Thus each RDF triple is part of the global Web of Data and each RDF triple can be used as a starting point to explore this data space.
- More powerful to write complex queries: Using RDF to represent statistical data makes complex queries feasible using SPARQL, the standard RDF querying language.
- Best option to handle unstructured and semi-structured data: Applied data is mainly semi-structured and unstructured data, particularly for data generated by Cloud machines and also statistical and geo-spatial information which is closely related to real life such as satellite imagery, air status, population, weather, etc.
- Easy-to-use tools for data management: Various domains have been attempting to use Linked data technique such as public area, broadcast, publication, and library. In public area, Linked Data has been adopted to bridge the gap between responsible data publishing and easy data use for the diverse needs of data consumers.

6 Bibliography and references

The bibliography referenced throughout the text is described in the following table.

[Accumulo2015] http://accumulo.apache.org/, Retrieved on 15.5.2015
[Arpirez03] Arpirez, Cesar, Julio, Corcho, Oscar, Fernandez-Lopez, Mariano, Gomez-Perez, Asuncion “WEBODE in a Nutshell”, AI Magazine Volume 24 Number 3, 2003
[Assayad04] Assayad, Ismail, Alain Girault, and Hamoudi Kalla. "A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints." Dependable Systems and Networks, 2004 International Conference on. IEEE, 2004.

[Bazarbayev13] Bazarbayev, Sobir, et al. "Content-based scheduling of virtual machines (VMs) in the cloud." *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*. IEEE, 2013.

[Benoit09] Benoit, Anne, Mourad Hakem, and Yves Robert. "Multi-criteria scheduling of precedence task graphs on heterogeneous platforms." The computer journal (2009): bxp067.

[Blomer14] Blomer, Jakob, et al. "Micro-CernVM: slashing the cost of building and deploying virtual machines." *Journal of Physics: Conference Series*. Vol. 513. No. 3. IOP Publishing, 2014.

[Blum03] Blum, Christian, and Andrea Roli. "Metaheuristics in combinatorial optimization: Overview and conceptual comparison." ACM Computing Surveys (CSUR) 35.3 (2003): 268-308.

[Calcavecchia12] Calcavecchia, Nicolò Maria, et al. "VM placement strategies for cloud scenarios." Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, 2012.

[Chang08] Chang, Fay et al. “Bigtable: A Distributed Storage System for Structured Data”, ACM Transactions on Computer Systems (TOCS), Volume 26 Issue 2, June 2008.

[Copil13] Copil, Georgiana, Daniel Moldovan, Hong-Linh Truong, Schahram Dustdar, "SYBL: an Extensible Language for Controlling Elasticity in Cloud Applications" (Submitted PDF, presentation slides), 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), May 14-16, 2013, Delft, the Netherlands.

[Copil14] Copil, Georgiana, Demetris Trihinas, Hong-Linh Truong, Daniel Moldovan, George Pallis, Schahram Dustdar, Marios Dikaiakos. "ADVISE - a Framework for Evaluating Cloud Service Elasticity Behavior" (submitted PDF, given presentation) the 12th International Conference on Service Oriented Computing. Paris, France, 3-6 November, 2014.

[Dean15] Dean, Mike, Guus Schreiber, November 2009, http://www.w3.org/TR/owl-ref/, retrieved on 19.5.2015.

[DeCandia07] DeCandia, Giuseppe et al. “Dynamo: amazon's highly available key-value store”, Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, 2007

[Deshpande13] Deshpande, Umesh, et al. "Gang migration of virtual machines using cluster-wide deduplication." *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE, 2013.

[Doğan05] Doğan, Atakan, and Füsun Özgüner. "Biobjective scheduling algorithms for execution time–reliability trade-off in heterogeneous computing systems." The Computer Journal 48.3 (2005): 300-314.

[Dustdar12] Dustdar, Schahram ; Guo, Yike ; Han, Rui ; Satzger, Benjamin ; Truong, Hong Linh: Programming Directives for Elastic Computing.. In: IEEE Internet Computing, 16 (2012), Nr. 6, S. 72-77.

[Gao13] Gao, Yongqiang, et al. "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing." Journal of Computer and System Sciences 79.8 (2013): 1230-1242.

[Garg09] Garg, Saurabh Kumar, Rajkumar Buyya, and Howard Jay Siegel. "Scheduling parallel applications on utility grids: time and cost trade-off management." Proceedings of the Thirty-Second Australasian Conference on Computer Science-Volume 91. Australian Computer Society, Inc., 2009.

[Garg11a] Garg, Ritu, and Awadhesh Kumar Singh. "Multi-objective optimization to workflow grid scheduling using reference point based evolutionary algorithm." Int J Comput Appl Technol 22.6 (2011): 44-49.

[Garg11b] Garg, Ritu, and Darshan Singh. "ε–Pareto Dominance Based Multi-objective Optimization to Workflow Grid Scheduling." Contemporary Computing. Springer Berlin Heidelberg, 2011. 29-40.

[George11] George, Lars “HBase: The Definitive Guide”, O'Reilly Media, September 2011

[Gong10] Gong, Zhenhuan ; Gu, Xiaohui ; Wilkes, John: PRESS: PRedictive Elastic ReSource Scaling for cloud systems.. In: CNSM : IEEE, 2010, S. 9-16.

[Hakem07] Hakem, Mourad, and Franck Butelle. "Reliability and scheduling on systems subject to failures." Parallel Processing, 2007. ICPP 2007. International Conference on. IEEE, 2007.

[Herbst13] Herbst, Nikolas Roman ; Kounev, Samuel ; Reussner, Ralf: {Elasticity in Cloud Computing: What it is, and What it is Not}. In: Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013) : USENIX, 2013.

[Islam12] Islam, Sadeka ; Lee, Kevin ; Fekete, Alan ; Liu, Anna ; Kaeli, David R. (Bearb.) ; Rolia, Jerry (Bearb.) ; John, Lizy K. (Bearb.) ; Krishnamurthy, Diwakar (Bearb.): How a consumer can measure elasticity for cloud platforms.. In: ICPE : ACM, 2012. - ISBN 978-1-4503-1202-8, S. 85-96
[WordNet] a lexical database for the English language, Princeton University, at http://wordnet.princeton.edu

[Jayaram11] Jayaram, K. R., et al. "An empirical analysis of similarity in virtual machine images." *Proceedings of the Middleware 2011 Industry Track Workshop*. ACM, 2011.

[Jebessa13] Jebessa, Naod Duga, Guido van't Noordende, and Cees de Laat. "Towards Purpose-Driven Virtual Machines." *ESSoS Doctoral Symposium*. 2013.

[Joshi15] Joshi, Karuna Pande, Pearce, Claudia "Automating Cloud Service Level Agreements using Semantic Technologies", Proceedings of CLaw Workshop, IEEE International Conference on Cloud Engineering (IC2E), March 2015.

[Karve13] Karve, Alexei, and Andrzej Kochut. "Image transfer optimization for agile development." *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013.

[Keren09] Jin, Keren, and Ethan L. Miller. "The effectiveness of deduplication on virtual machine disk images." *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*. ACM, 2009.

[Khare03] Khare, Vineet, Xin Yao, and Kalyanmoy Deb. "Performance scaling of multi-objective evolutionary algorithms." Evolutionary Multi-Criterion Optimization. Springer Berlin Heidelberg, 2003.

[Klyne15] Klyne, Graham, Carrol Jeremy J., February 2014, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, retrieved on 19.5.2015.

[Kochut12] Kochut, Andrzej, and Alexei Karve. "Leveraging local image redundancy for efficient virtual machine provisioning." *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012.

[Lakshman10] Lakshman, Avinash, Malik Prashant "Cassandra: a decentralized structured storage system", ACM SIGOPS Operating Systems Review, Volume 44 Issue 2, April 2010.

[Lama11] Lama, Palden, and Xiaobo Zhou. "aMOSS: Automated multi-objective server provisioning with stress-strain curving." Parallel Processing (ICPP), 2011 International Conference on. IEEE, 2011.

[Lei14] Lei, Zhou, et al. "An Improved Image File Storage Method Using Data Deduplication." *Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on*. IEEE, 2014.

[Li11] Li, Jim Zw, et al. "CloudOpt: multi-goal optimization of application deployments across a cloud." Proceedings of the 7th International Conference on Network and Services Management. International Federation for Information Processing, 2011.

[Luo13] Luo, Tian, et al. "S-cave: Effective ssd caching to improve virtual machine storage performance." *Proceedings of the 22nd international conference on Parallel architectures and compilation techniques*. IEEE Press, 2013.

[Ma12] Ma, F., F. Liu, and Z. Liu. "Multi-objective optimization for initial virtual machine placement in cloud data centre." J. Infor. and Computational Science 9.16 (2012).

[Malhotra13] Malhotra, Akshita, and Gaurav Somani. "VMCloner: A Fast and Flexible Virtual Machine Cloner." *Cloud and Green Computing (CGC), 2013 Third International Conference on*. IEEE, 2013.

[Mao12] Mao, Ming, and Marty Humphrey. "A performance study on the vm startup time in the cloud." *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012.

[Mariani14] Mariani, Stefano, Hong-Linh Truong, Georgiana Copil, Andrea Omicini, Schahram Dustdar, "Coordination-aware Elasticity"(PDF), 7th IEEE/ACM International Conference on Utility and Cloud Computing, 8-11 December, London, 2014.

[Menzel13] Menzel, Michael, et al. "A configuration crawler for virtual appliances in compute clouds." *Cloud Engineering (IC2E), 2013 IEEE International Conference on*. IEEE, 2013.

[Moldovan15] Moldovan, Daniel, Georgiana Copil, Hong-Linh Truong, Schahram Dustdar, "MELA: Elasticity Analytics for Cloud Services", International Journal of Big Data Intelligence, 2015, Vol. 2, No. 1.

[Ng11] Ng, Chun-Ho, et al. "Live deduplication storage of virtual machine images in an open-source cloud." *Proceedings of the 12th International Middleware Conference*. International Federation for Information Processing, 2011.

[Nguyen13] Nguyen, Minh Binh, Viet Tran, and Ladislav Hluchy. "A generic development and deployment framework for cloud computing and distributed applications." *Computing and Informatics* 32.3 (2013): 461-485.

[Nicolae15] Bogdan Nicolae, Andrzej Kochut, Alexei Karve. Discovering and Leveraging Content Similarity to Optimize Collective On-Demand Data Access to IaaS Cloud Storage. *CCGrid'15: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2015, Shenzhen, China.

[Peng12] Peng, Chunyi, et al. "VDN: Virtual machine image distribution network for cloud data centers." *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012.

[Protege15] http://protege.stanford.edu, retrieved on 19.5.2015.

[Prud'hommeaux15] Prud'hommeaux, Eric, Seaborne, Andy "SPARQL Query Language for RDF", January 2008, http://www.w3.org/TR/rdf-sparql-query/, retrieved on 15.5.2015.

[Razavi13] Razavi, Kaveh, and Thilo Kielmann. "Scalable virtual machine deployment using VM image caches." *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. ACM, 2013.

[Razavi14] Razavi, Kaveh, Liviu Mihai Razorea, and Thilo Kielmann. "Reducing VM Startup Time and Storage Costs by VM Image Content Consolidation." *Euro-Par 2013: Parallel Processing Workshops*. Springer Berlin Heidelberg, 2014.

[RDF15] RDF Core working group, 25 February 2014, http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/, retrieved on 15.5.2015.

[Reich12] Reich, Joshua, et al. "VMTorrent: scalable P2P virtual machine streaming." *CoNEXT*. 2012.

[Sakellariou07] Sakellariou, Rizos, et al. "Scheduling workflows with budget constraints." Integrated Research in GRID Computing. Springer US, 2007. 189-202.

[Schmidt10] Schmidt, Matthias, et al. "Efficient distribution of virtual machines for cloud computing." *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*. IEEE, 2010.

[Shiraz13] Shiraz, Muhammad, et al. "A study on virtual machine deployment for application outsourcing in mobile cloud computing." *The Journal of Supercomputing* 63.3 (2013): 946-964.

[Spyns08] Spyns, Peter, Tang, Yan, Meersman, Robert, "An Ontology Engineering Methodology for DOGMA", Journal of Applied Ontology, special issue on "Ontological Foundations for Conceptual Modeling", Volume 3, Issue 1-2, p.13-39, 2008.

[Steve15] Steve Harris, Garlik, Andy Seaborne, March 2013, http://www.w3.org/TR/sparql11-query/, retrieved on 15.5.2015.

[Stillwell10] Stillwell, Mark, et al. "Resource allocation algorithms for virtualized service hosting platforms." Journal of Parallel and Distributed Computing 70.9 (2010): 962-974.

[Sure02] Sure, York, Staab, Steffen, Angele, Jürgen "OntoEdit: Guiding Ontology Development by Methodology and Inferencing", Proceedings of the International Conference on Ontologies, Databases and Applications of SEmantics ODBASE, October 2002.

[Talukder07] Talukder, A. K. M., Michael Kirley, and Rajkumar Buyya. "Multiobjective differential evolution for workflow execution on grids." Proceedings of the 5th international workshop on Middleware for grid computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference. ACM, 2007.

[Tang11] Tang, Chunqiang. "FVD: A High-Performance Virtual Machine Image Format for Cloud." *USENIX Annual Technical Conference*. 2011.

[Truong14] Truong, Hong-Linh, Schahram Dustdar, Georgiana Copil, Alessio Gambi, Waldemar Hummer, Duc-Hung Le, Daniel Moldovan, "CoMoT – A Platform-as-a-Service for Elasticity in the Cloud" (slides), IEEE International Workshop on the Future of PaaS, IEEE International Conference on Cloud Engineering (IC2E 2014), Boston, Massachusetts, USA, 10-14 March 2014.

[Tsoumakos13] Tsoumakos, Dimitrios ; Konstantinou, Ioannis ; Boumpouka, Christina ; Sioutas, Spyros ; Koziris, Nectarios: Automated, Elastic Resource Provisioning for NoSQL Clusters Using TIRAMOLA.. In: CCGRID : IEEE Computer Society, 2013. - ISBN 978-1-4673-6465-2, S. 34-41.

[Ullman75] Ullman, J. Np-complete scheduling problems. Journal of Computer and System sciences, 10(3):384-393, 1975.

[Van10] Van, Hien Nguyen, Frédéric Dang Tran, and J-M. Menaud. "Performance and power management for cloud infrastructures." Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. IEEE, 2010.

[Xu14a] Xu, Jiwei, et al. "A lightweight virtual machine image deduplication backup approach in cloud environment." *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual*. IEEE, 2014.

[Xu14b] Xu, Xiaolin, et al. "Rethink the storage of virtual machine images in clouds." *Future Generation Computer Systems* (2014).

[Zhang13] Zhang, Youhui, Yanhua Li, and Weimin Zheng. "Automatic software deployment using user-level virtualization for cloud-computing." *Future Generation Computer Systems* 29.1 (2013): 323-329.

[Zhao14] Zhao, Xun, et al. "Liquid: A scalable deduplication file system for virtual machine images." *Parallel and Distributed Systems, IEEE Transactions on* 25.5 (2014): 1257-1266.

[Zhou13] Zhou, Ruijin, et al. "Optimizing virtual machine live storage migration in heterogeneous storage environment." *ACM SIGPLAN Notices*. Vol. 48. No. 7. ACM, 2013.

7 Abbreviations/ Glossary

AB – Advisory Board

ACO – Ant Colony Optimization

AM – Aggregation Methods

API - Application Programming Interface

B2B - Business-to-Business

BSA – Bethesda Softworks Archive

CAL – Cloud Abstraction Layer

CSP – Cloud Service Provider

CSP – Constraint programming methods

CSW – Web Catalogue Service

DL – Deliverable responsible

DMZ – DeMilitarized Zone

DOGMA - Developing Ontology Grounded Methods and Applications

DoS – Denial of Service

EC - European Commission

EO - Earth Observation

FCO – Flexiant Cloud Orchestrator

FTP – File Transfer Protocol

FVD – Fast Virtual Disk

GA – Genetic Algorithms

GMGD – Gang Migration uses Global De-duplication

GS – Ground Station

GSM – Global System for Mobile

GUI - Graphical User Interface

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

IaaS - Infrastructure as a service

IM - Instant Messaging

IO - Input/Output

KB - Knowledge Base

KVM - Kernel-based Virtual Machine

MIP - Mixed Integer linear Programming methods

NAT - Network Address Translation

NFS - Network File System

NSGA - Multi-objective Optimization Algorithm

OGC - Open Geospatial Consortium

OO - Object Oriented

OS - Operative System

OWL - Web Ontology Language

P2P - Peer to Peer

PaaS - Platform as a Service

PC - Project Coordinator

PRESS - Predictive Elastic ReSource

PSTN - Public Switched Telephone Network

PVIP - Public Virtual IP

QM - Quality manager

QoS - Quality of Service

RDF - Resource Description Framework

RDFS - Resource Description Framework Schema

REST - Representational State Transfer

RTP - Real Time Protocol

SaaS - Software as a Service

SC - Steering Committee

SciC - Scientific coordinator

SCM - Source Code Management

SFTP - Secure File Transfer Protocol

SIP - Session Initiation Protocol

SLA - Service Level Agreement

SPARQL - SPARQL Protocol and RDF Query Language

SRTP - Secured Real Time Protocol

STA - Short Term Archive

SYBL - Simple Yet Beautiful Language

TB -Technical board

TCP - Transfer Control Protocol

TL - Task leader

TLS – Transport Layer Security

UDP - User Datagram Protocol

URI - Uniform Resource Identifier

VDN - Virtual machine Distribution Network

VLAN - Virtual Local Area Network

VM - Virtual image

VMI - Virtual Machine Instance

W3C - World Wide Web Consortium

WL - Work package leader

WP - Work package

XML - EXtensible Markup Language