Horizon 2020 Program (2014-2020)

Big data PPP

Research addressing main technology challenges of the data economy

Industrial-Driven Big Data as a Self-Service Solution

D5.1: Federated Resource Management for Data Analytics v.1[†]

Abstract: The content of this report is mainly focused on the setup of the infrastructure layer which includes the selected underlying storage and processing infrastructure of the I-BiDaaS solution. The document also describes the preliminary work carried out on the distributed large-scale layer which is responsible for the orchestration and management of the underlying physical computational and storage infrastructure.

Contractual Date of Delivery	31/12/2018
Actual Date of Delivery	31/12/2018
Deliverable Security Class	Public
Editor	*Enric Pages (ATOS)*
Contributors	ATOS, UNIMAN, BSC, ITML, FORTH
Quality Assurance	*Vassilis Chatzigiannakis (ITML)* *Giorgos Vasiliadis (FORTH)* *Kostas Lampropoulos (FORTH)*

[†] *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780787.*

The *I-BiDaaS* Consortium

Foundation for Research and Technology – Hellas (FORTH)	Coordinator	Greece
Barcelona Supercomputing Center (BSC)	Principal Contractor	Spain
IBM Israel – Science and Technology LTD (IBM)	Principal Contractor	Israel
Centro Ricerche FIAT (FCA/CRF)	Principal Contractor	Italy
Software AG (SAG)	Principal Contractor	Germany
Caixabank S.A. (CAIXA)	Principal Contractor	Spain
University of Manchester (UNIMAN)	Principal Contractor	United Kingdom
Ecole Nationale des Ponts et Chaussees (ENPC)	Principal Contractor	France
ATOS Spain S.A. (ATOS)	Principal Contractor	Spain
Aegis IT Research LTD (AEGIS)	Principal Contractor	United Kingdom
Information Technology for Market Leadership (ITML)	Principal Contractor	Greece
University of Novi Sad Faculty of Sciences (UNSPMF)	Principal Contractor	Serbia
Telefonica Investigation y Desarrollo S.A. (TID)	Principal Contractor	Spain

Document Revisions & Quality Assurance

Internal Reviewers

1. *Vassilis Chatzigiannakis (ITML)*
2. *Giorgos Vasiliadis, Kostas Lampropoulos (FORTH)*

Revisions

Version	Date	By	Overview
0.1	29/10/2018	ATOS	Initial ToC
0.2	15/11/2018	ALL	Minor changes after brainstorming
0.3	29/11/2018	FORTH	GPU Commodity Cluster
0.4	30/11/2018	ITML	Integration process inputs
0.5	14/12/2018	UNIMAN	Runtime environment inputs
0.6	21/12/2018	BSC	COMPs/Hecuba programming models
0.7	21/12/2018	ATOS	RMO contributions
0.8	21/12/2018	ATOS	Contributions merge
0.9	28/12/2018	Internal Reviewers	Document for review
1.0	28/12/2018	ATOS	Final document

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API: Application Programming Interface
CI: Continuous Integration
CPU: Central Processing Unit
CSP: Cloud Service Provider
DNA: DeoxyriboNucleic Acid
GPGPUs: General Purpose Graphics Processing Unit
GPU: Graphics Processing Unit
HPC: High Performance Computing
IT: Information Technologies
ITER: Instituto Tecnológico de Energías Renovables
KVM: Kernel-based Virtual Machine
MVP: Minimum Viable Product
OASIS: Organization for the Advancement of Structured Information Standards
PoC: Proof of Concept
RMO: Resource Management and Orchestration
SQL: Structured Query Language
TOSCA: Topology and Orchestration Specification for Cloud Applications
VM: Virtual Machine
VPN: Virtual Private Network
WP: Work Package

Executive Summary

The content of this report is mainly focused on the setup of the infrastructure layer which includes the selected underlying storage and processing infrastructure of the I-BiDaaS solution. The document also describes the preliminary work carried out on the distributed large-scale layer which is responsible for the orchestration and management of the underlying physical computational and storage infrastructure.

The first part of this document (Section 2) focuses on the provisioning and the configuration of infrastructure resources. This section details the cloud testbed's and the commodity cluster's architecture, resources and configuration.

The following section (Section 3) focuses on the design and the specification Resource Management and Orchestration software module (RMO), which is responsible for interfacing with the infrastructure layer in order to programmatically interact with the virtual and physical resources.

Section 4 presents the runtime environment which is the execution environment for data processing applications as well as the interface with the data storage. The section describes the need for a simplified environment for writing parallel and distributed applications to improve programming productivity and how I-BiDaaS plans to address this need.

Finally, the document concludes with a summary of the work done within this period as well as listing the challenges that were present within the first 12 months of the project and how they have been addressed.

1 Introduction

1.1 Overview and Objectives

One of the project main aims is to create new opportunities for self-service analytics towards a complete paradigm tailored to big data analytics. As such, I-BiDaaS framework aims to improve performance and advance capabilities of handling large data volumes. To this end, I-BiDaaS will offer a unified Big Data as a self-service solution that will empower non-expert Big-Data users to easily take advantages of the Big-Data technologies while at the same time increasing the speed of data analytics. The list of software modules that the final platform involves, their respective layer and role within the platform, as well as the initial modules designs are further described in the confidential I-BiDaaS report "D.1.2: Architecture Definition [1]".

"D5.1. Federated Resource Management for Data Analytics v1" mainly describes the provisioning and configuration work carried out in the scope of "Task 5.1 Provision and configuration of infrastructure resources" which is active from M6. Additionally, the document also briefly covers "Task 5.2 Resource management and optimized automatic usage of computational and storage resources" including the work achieved from M8.

Both tasks belong to *WP5 Distributed large scale framework and Integration* whose main objective is to provide a distributed large-scale framework that permits powerful Data processing on top of heterogeneous resources.

1.2 Relation to other Tasks and Work Packages

The content of this report covers the work carried out in the scope of Tasks 5.1 and 5.2. The systems and software modules presented in this document have a direct relationship with the deliverable D5.2 submitted in M12, as well as with the rest of the tasks within the WP5.

Due to the fact that provisioning and management of computational resources plays an important role within the project solution, this report has links with other technical reports across WP2, WP3, and WP4.

As stated above, initial module specification and designs have been covered in the I-BiDaaS architecture report, named D.1.2.

Before going through the content of this report, it is recommended to read the public deliverable "D.1.3 Positioning of I-BiDaaS" [2] which provides an overview on the industrial challenges of the data economy as well as setting the scene for the realisation of the I-BiDaaS platform.

1.3 Target Audiences

The primary targets of the document are internal I-BiDaaS technicians from WP2 to WP5 involved in the prototyping and implementation of the platform. Additionally, this document can also be interesting for external technical personnel that is willing to adopt I-BiDaaS solution or/and Cloud Service Provider (CSP) willing to be incorporated as infrastructure and data providers within our solution.

1.4 Structure of the Document

The outline of this document is as follows: The first chapter introduces the document and its objectives. The second chapter describes the provisioning and configuration processes followed within the project to setup the infrastructure resources required. The third chapter presents the preliminary work carried out in the scope of Task 5.2, where the Resource Management and Orchestration software module was prototyped. The runtime environment providing the execution environment for data analytics is introduced in section 4, together with the approach selected for the integration task of the project towards the I-BiDaaS Minimum Viable Product. Finally, the last section provides the conclusions and next steps (Section 5).

2 Provision and Configuration of Infrastructure Resources

2.1 Cloud Testbed Overview

"D.5.1. Federated Resource Management for Data Analytics v.1" is focused on the setup of the infrastructure layer, as well as the software modules that will handle the management of the resources during the project lifetime. In order to achieve the project goals, the computational power is a fundamental element. High-performance computing facilities and data infrastructures for science and engineering are required to accommodate the exponential generation of digital assets and the necessity for tools to be generated to accommodate the ongoing transformation of the business models under the Big Data paradigm.

Among the different high-performance computing facilities and data infrastructures providers considered during the elicitation of requirements phase, the final candidate selected for I-BiDaaS is the "*Teide Supercomputer*", which is a general purpose High-Performance Computing infrastructure, housed in a D-Alix [3] datacentre which provides high available electrical and cooling infrastructure, and high-speed internet connectivity. The ***Teide-HPC*** supercomputer, as it is known, has been proved as a good infrastructure provider choice in more than one hundred national and international science and engineering research activities such as, among others, Helix Nebula [4] (The Science Cloud), the aeronautical simulation designs performed by Airbus company, research initiatives from the European Space Agency (ESA [5]) or even in previous Big Data research projects like IonGAP [6], where the high processing capacity of the infrastructure has been used to reduce the time and cost of the research that comprehensive analysis of data related to DNA bacteria require.

The Teide-HPC supercomputer is composed of 1100 Fujitsu computer servers, the core of the compute nodes are Fujitsu PRIMERGY CX250 S1 servers housed in a PRIMERGY CX400 chassis grouped in 4 nodes per chassis, and featured with the latest Intel Sandy Bridge processors, allowing to obtain not just the best performance but also a great energy efficiency. In terms of the overall capacity of the datacentre, it provides a total of 17800 computing cores and 36 TB of memory, including a high-performance network and parallel storage systems.

Atos' Infrastructure Services offered in I-BiDaaS are delivered on top of the hardware resources from *Instituto Tecnológico de Energías Renovables* (ITER) Data Centre located in the Canary Islands. The scope of this service comprises:

- Provision of bare-metal resources.
- Provision of server capacity on-demand.
- Provision of the storage capacity on-demand.
- Provision of network connection.
- Operation of the infrastructure.
- Online portal and/or API for service requests.
- Online portal and/or API providing standard reporting.
- Atos manages the targeted Cloud computing environment in accordance with ISO 9001 (quality), ISO14001 (environmental) and ISO 27001 (security) standards.

The table below (Table 1) describes the resource capacity planned to support the work during the project lifetime:

Table 1. Resource Capacity

	Total	Per Node
Computer Processing Capacity	48 cores (96 threads)	16 cores (32 threads)
Memory Capacity	192 GB	64 GB
Storage Capacity	36 TB	12 TB
Network Capacity	(2 x 1GbE) x 3	2 x 1GbE

2.2 Commodity Cluster Overview

FORTH's commodity cluster consists of a large pool of heterogeneous hardware architectures. Firstly, FORTH's commodity cluster contains tens of modern off-the-shelf commodity GPGPUs, such as the NVIDIA GeForce GTX 1080 Ti. Such GPGPUs offer extremely high processing throughput for parallelizable workloads with a low power cost. Some characteristics of GPUs are the following: (i) they contain thousands of cores, (ii) their architecture is highly parallel, (iii) their memory can reach up to 11GB with high bandwidth. These GPGPUs are programmable through frameworks, such as CUDA (supported only for NVIDIA GPGPUs) or OpenCL (offers uniform execution across different vendors and hardware architectures). In addition to this kind of GPUs, namely discrete or dedicated, there is another type of such processing unit, called integrated GPU. An integrated GPU is packed inside the CPU's die, such as the Intel HD Graphics. The main characteristic that differentiates these two kinds of GPUs is the memory space, where discrete GPUs have their own dedicated memory space, while integrated GPUs share the same memory space with the CPU. Apparently, integrated GPUs are even more power efficient than discrete GPUs. FORTH's commodity cluster contains also other types of hardware accelerators, such as the Intel Phi coprocessor. Finally, FORTH's cluster consists of powerful Intel processors (based on the Skylake micro-architecture or newer) enabled with the Intel Software Guard Extensions (Intel SGX), which is a set of CPU instruction codes that allows user-level code to allocate private regions of memory, called enclaves, that are protected from processes running at higher privilege levels. These processors can be used to preserve the privacy of users since SGX can guard both data and code in untrusted environments.

2.3 Cloud Testbed Set-up

This section covers the stages followed within the first period for setting up and maintaining the testbed provided by ATOS to the project for experimentation and validation of project outcomes.

2.3.1 Provisioning

First of all, in order to get access to the computing capacity of the existing HPC infrastructure, we followed the provisioning procedure defined by Teide-HPC [7]. After the acceptance of the offer, I-BiDaaS performed accessibility and performance tests over the infrastructure to verify that the infrastructure adapts to project requirements. Finally, after the verification process, on demand computation prices are set according to the final set of computing services selected.

Our nodes have been configured with Linux systems, more precisely based on Ubuntu Xenial Xerus (16.04) which is a long-term support (LTS) version which offers 5-year support for server releases; this fits with the need of the project.

2.3.2 Virtualization

Virtualization technology enables multiple operating systems to run on the same physical platform. This technique is used within I-BiDaaS to get the inherit advantages of decoupling the IT operations among isolated environments, making possible to maximize computer utilization while minimizing the associated overhead of management, maintenance and consumption of the computing resources.

The selected virtualization technology is Kernel-based Virtual Machine (KVM) which is an open source full virtualization solution for Linux on x86 hardware architectures containing Intel VT virtualization extension. This allows us to turn the Linux systems into type-1 (bare-metal) hypervisor managing every virtual machine as a regular Linux process with dedicated virtual hardware like CPUs, network cards, memory and disks.

2.3.3 Virtual Environments Set-up

2.3.3.1 VMs managed directly on the virtual environment

Virsh [8] has been used as the main interface for managing KVM guest domains during the setup of the virtual environment. The aforementioned tool is built on the libvirt management API, which is a library used to interface with different virtualization technologies.

The environment has been used to accomplish the following tasks:

- Configuration of the virtual layer, including the creation of the first virtual image template (based on Ubuntu Xenial) as well as for configuring and testing the network setup on both, physical and virtual servers.
- Perform proofs of concept with various cloud orchestrators to support the analysis of various candidates that need to be evaluated in the scope of Task 5.2.
- Creation of the first set of VMs to accommodate the software components and systems that will be used for the Minimum Viable Product (MVP) milestone (MS2). After installing and configuring all the software required for each component, the VMs will be backed-up and stored as a '*GOLD'* virtual image templates that can be used later to populate the Private CSP image repository (see section 4.3.2 for further details).

2.3.3.2 Private all-in-one Cloud Service Provider

The all-in-one Cloud Service Provider (CSP) selected for this environment is based on DevStack [9] distribution, which attempts to support the two latest LTS releases of Ubuntu, being Ubuntu 16.04 (our selected version) the most tested. DevStack offers tooling as well as a set of scripts that facilitates the process of bringing up a complete OpenStack ecosystem, which is an open source software for managing clouds, in a single server or VM. All the setup

for this distribution has been done on top of a KVM guest created for that purpose. The OpenStack services/components installed with this distribution are: keystone [10], glance [11], swift [12], nova [13], cinder [14], neutron [15] and horizon [16].

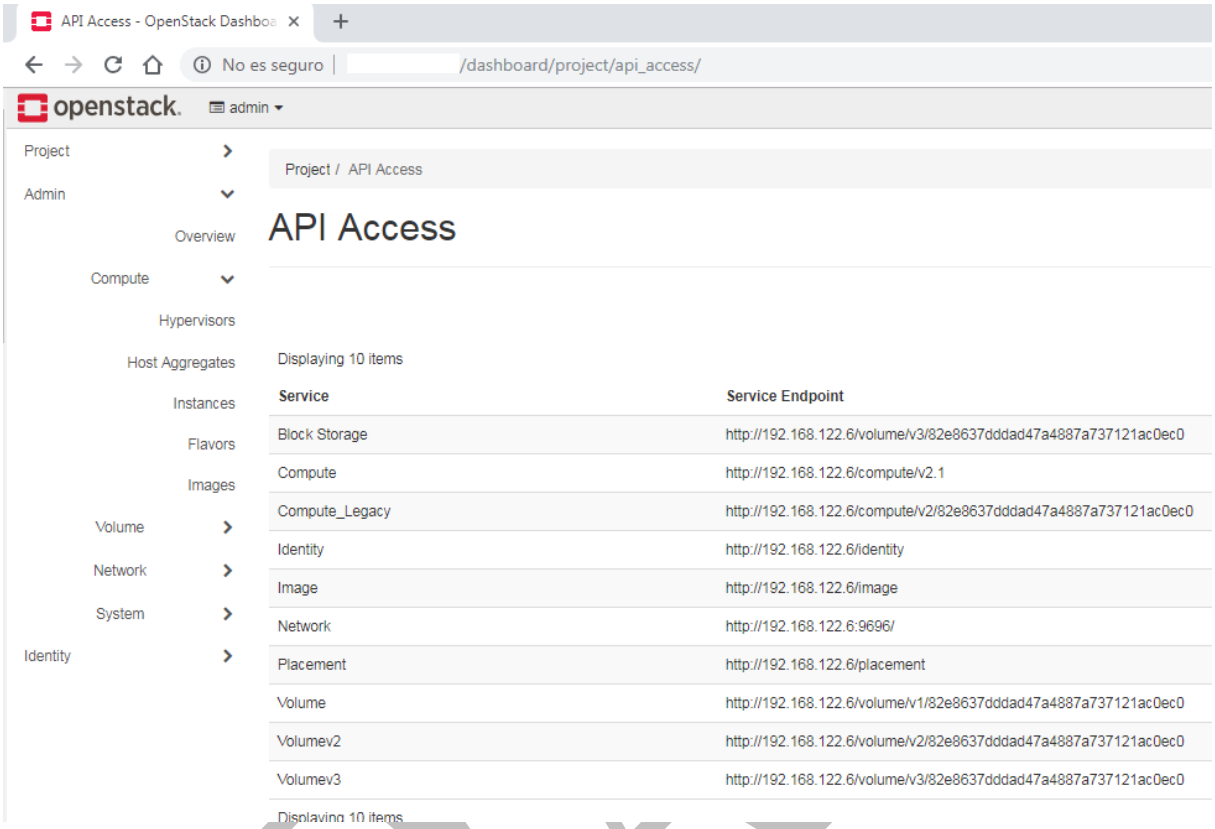

Figure 1. OpenStack all-in-one dashboard screenshot.

The all-in-one environment (see Figure 1) will be used in the project for two main purposes. On one hand it will be used to support the development of the Resource Management and Orchestration software modules of the I-BiDaaS architecture and on the other hand, it will be used as a testing environment to check the configuration changes that we would like to apply on the multi-node environment.

2.3.3.3 Private multi-node Cloud Service Provider

OpenStack multi-node setup provides the necessary compute, network and data storage services for building a cloud-based Big Data cluster to meet the needs of the project resource deployments. The Big Data architectures benefit from high-performance storage backends. The selected services allow us to create storage pools to offer block storage services to the virtual instances. Moreover, storage services are offered for storing images and performing volume back-ups (Figure 2).

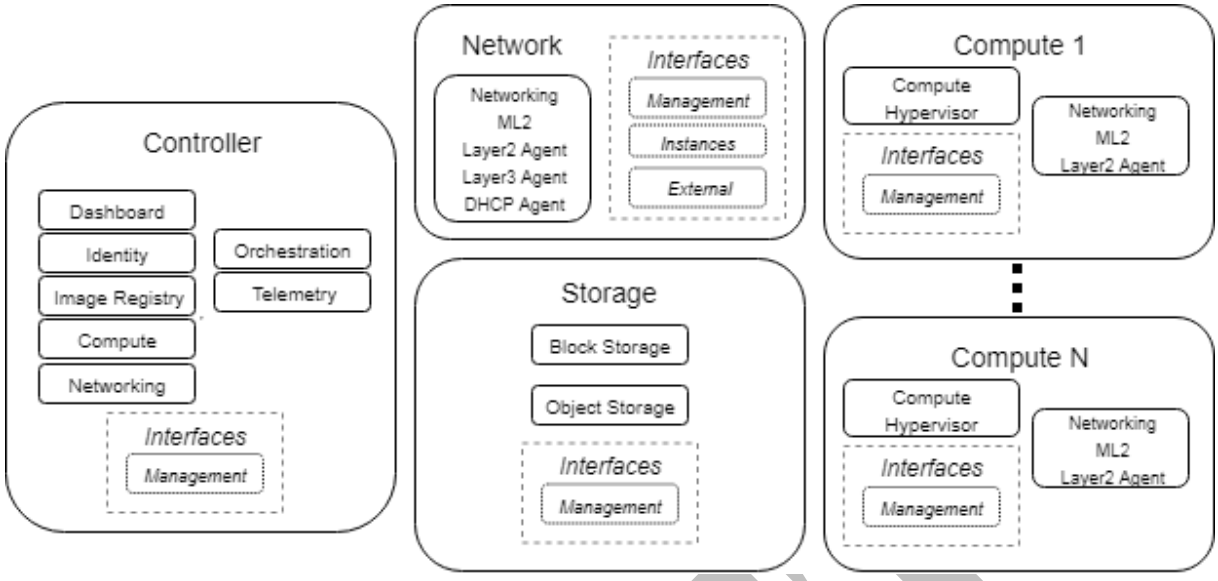

Figure 2. OpenStack multi-node building blocks.

2.4 Commodity Cluster Set-up

As already discussed, FORTH's commodity cluster consists of numerous types of hardware architectures, such as accelerators (e.g., discrete GPGPUs) or modern and powerful multi-core processors, enabled with the SGX technology. Each of these architectures has its own characteristics and requirements. More specifically, the basic configuration for the numerous machines of FORTH's commodity cluster is composed by a single (or dual) CPU socket paired with a discrete GPU (Figure 3 and Figure 4).

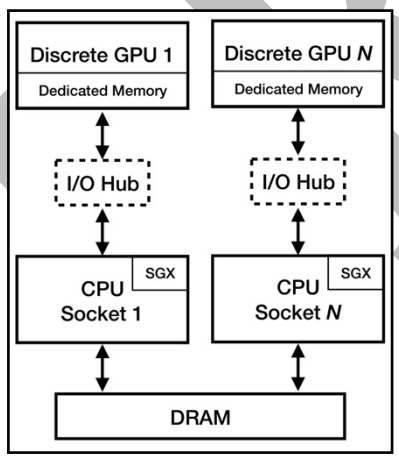

Figure 3. Architectural configuration with a single discrete GPU

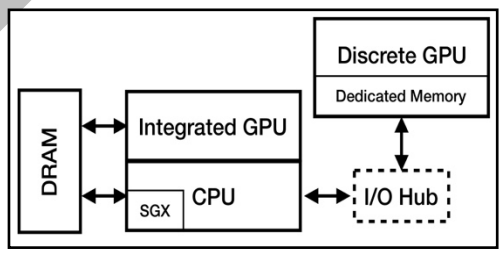

Figure 4. Architectural configuration with two CPU sockets and two discrete GPUs

When available, an integrated GPU is included in the configuration (Figure 5). Our systems run Linux (LTS version) with the latest drivers installed for CUDA and OpenCL.

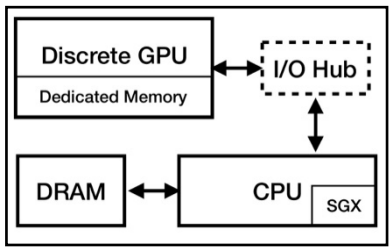

Figure 5. Architectural configuration with a single discrete GPU and an integrated GPU

3 Resource Management and Optimized Usage of Resources

3.1 Introduction

I-BiDaaS solution is based on three main layers: the infrastructure layer, the distributed large-scale layer, and the application layer. The software module presented in this section belongs to the distributed large-scale layer. The Resource Management and Orchestration software module (RMO) is responsible for interfacing with the infrastructure layer allowing to programmatically interact with the set of virtual and physical resources that the application requires. The component aims to support multiple providers and various computational resource types.

For orchestrating the resources and the workflow, various technologies have been evaluated in this period. The evaluation has been performed through the realisation of Proof of Concepts (PoC) willing to analyse the viability for each set of technology enablers for the I-BiDaaS solution.

The following sections present the design and specification of the envisaged software module, in addition the interactions with other software modules within the architecture are also considered. The specifications details described are part of an on-going work within Task 5.2, therefore they cannot be considered in a final state. The software modules and their interfaces are going to be prototyped and implemented in the following months. An enhanced version of the specifications will be included in the second period report "D5.3 Federated Resource Management for Data Analytics v2".

3.2 Positioning within the I-BiDaaS Architecture

The Resource Management and Orchestration module, which belongs to the distributed large-scale layer of the architecture, has the mission of handling the virtual/containerized resources on top of the infrastructure layer. This software system is the glue between the two layers, allowing the components within the same layer, to deploy the resources planned to accommodate the needs of the application layer.

The inputs coming from the runtime environment (Figure 6) will be consumed through a blueprint manifest describing the resource requirements and their relationships. The RMO module interprets the manifest to provide the best placement and offers adaptation mechanisms for managing elasticity at runtime. A monitoring system will be used to expose performance and availability information during the execution of the resources (Figure 7).

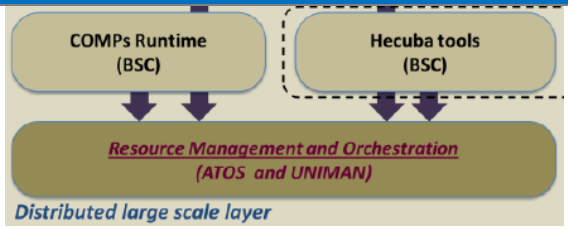

Figure 6. Inputs coming from Runtime environment.

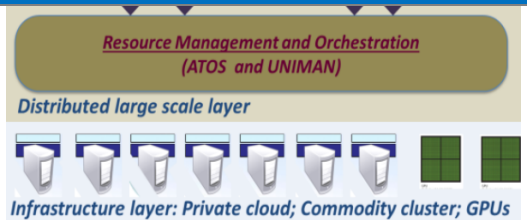

Figure 7. Interaction between RMO and the infrastructures layer

3.3 Baseline Concepts and Technologies

3.3.1 Baseline Technologies

This section presents the set of technologies and systems evaluated to suit the I-BiDaaS solution and use cases. Within I-BiDaaS we will take advantage of the current existing tools in the market to build the required solution. Task 5.2. *"Resource management and optimized automatic usage of computational and storage resources"* has been started in M8 with a knowledge-acquisition phase consisting of the realization of Proof of Concepts (*PoC*) on various technology enablers. The set of enablers evaluated includes multi-cloud providers, cloud orchestrator engines, container and cluster managers among other correlated technologies such as languages to describe cloud-based topologies and the relationships between the components that compose them.

The following table summarizes the main technologies involved in the PoCs:

Table 2. Technologies included in the PoC

Cloudify
Cloudify is a Cloud Orchestrator used to manage the interconnection and interaction among cloud-based entities. The orchestration refers to the automation of processes and workflows required to meet the application's performance goals, minimizing the associated deployment and operation costs while maximizing the application performance.
Apache Brooklyn
Apache Brooklyn is an open-source framework for modeling distributed applications defined using declarative blueprints. It is able to abstract the deployment of cloud services among different providers (or locations) using Apache jclouds. It supports locations such as amazon, IBM Softlayer, Google Compute Engine, Openstack or Apache CloudStack. In addition to his own native YAML format to describe the cloud-based applications, it also supports TOSCA specification.
Heat
Heat is the orchestration service within OpenStack. It is capable of managing the entire lifecycle of the infrastructure within Openstack Clouds. It uses a native template format (HOT-Heat Orchestration Template) while keeps compatibility with AWS CloudFormation format. The orchestration engine used to launch multiple cloud applications can be enhanced with an autoscaling service integrated with the OpenStack Telemetry service.
AWS CloudFormation
AWS CloudFormation enables developers to model and manage different AWS services through a common language, allowing to describe and provision all the parameters and conditions required to deploy and manage applications on top of this public cloud provider.
Kubernetes
It is a container provisioning software that enables automatic deployment, scaling and management of containerized applications. Kubernetes orchestrates containers, but it does not provide the container technology itself. The software to containerize like Dockers has to be installed and running on those machines where Kubernetes will orchestrate the execution.
Docker
Docker provides software components and tools for ship and run applications, it started using LXC which is an operating-system-level virtualization method. It becomes very useful during software development as it is a lightweight manner to deploy an application in comparison of using an entire VM. It also offers good support for CI/CD systems while simplifies deployment to different infrastructures.
Apache Mesos
Mesos is a platform for sharing commodity clusters between diverse cluster computing frameworks, such as Hadoop and MPI. Apache Mesos abstracts CPU, memory, storage, among other resources away from physical or virtual resources. Through this abstraction, it can run multiple instances of the same framework our build

specialized frameworks targeting particular problem domains.
TOSCA
TOSCA is an OASIS standard specifications used to describe cloud web services and their relationships. The language includes specifications to create or modify the web-services associated with a cloud-based topology.
Multi-Cloud for DAaaS asset
This is an internal Atos Research and Innovation asset under development. The asset aims to manage the deployment and life-cycle of Big Data clusters in a variety of Cloud infrastructures, currently targeting Hadoop clusters, the system relies on Apache Brooklyn and the Brooklyn Ambari plugin to manage the cluster nodes. The module includes an Adaptation Engine that is in charge of managing scalability of Big Data and the cluster nodes. It could be adapted to support I-BiDaaS solution.

3.3.2 Component Architecture

The Resource Management and Orchestration software system aims to achieve pre-configured and fully operational deployments across the infrastructure environments described in section 2.

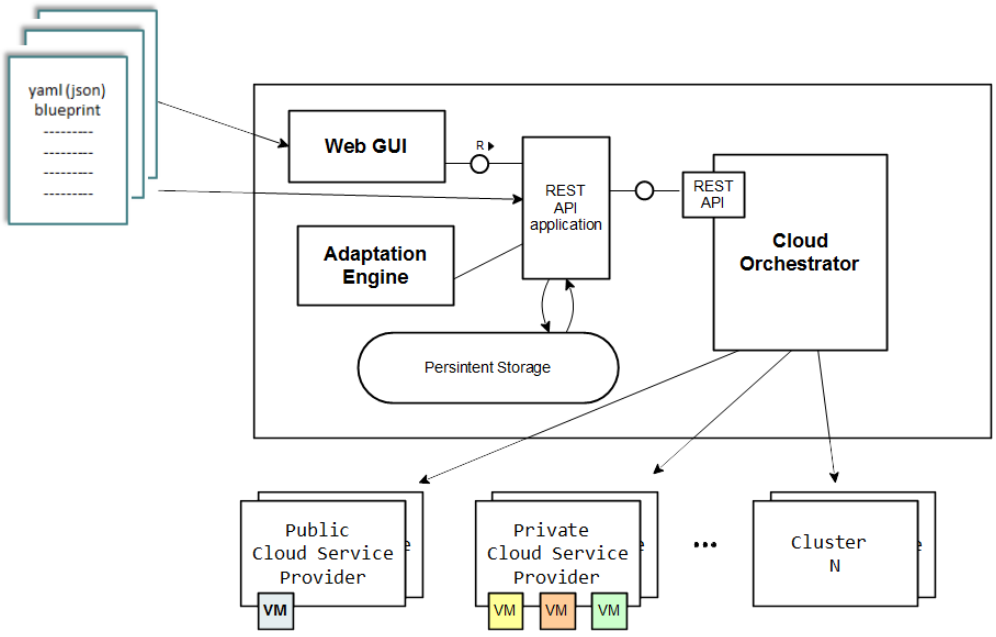

Figure 8. RMO sofware module architecture.

The architecture is depicted in Figure 8, and the software module is composed of the following components:

- A REST API acts a northbound interface enacting the required calls through the Cloud Orchestrator module to perform the deployment of virtual resources. It consumes deployment templates describing the requirements of the service and the conditions to meet at runtime.
- The Adaptation Engine should guarantee that the service conditions are fulfilled, this is evaluated through the information exposed by the Cloud Orchestrator module, the module will manage the lifecycle of the service applying pre-defined elasticity rules at runtime.

- A SQL-like database stores application states.

The following figure (Figure 9) maps each of the targeted software modules with one of the technology enablers evaluated during the PoCs. There are also other combinations of technologies which may be feasible. The final setup is going to be finalised during the second period.

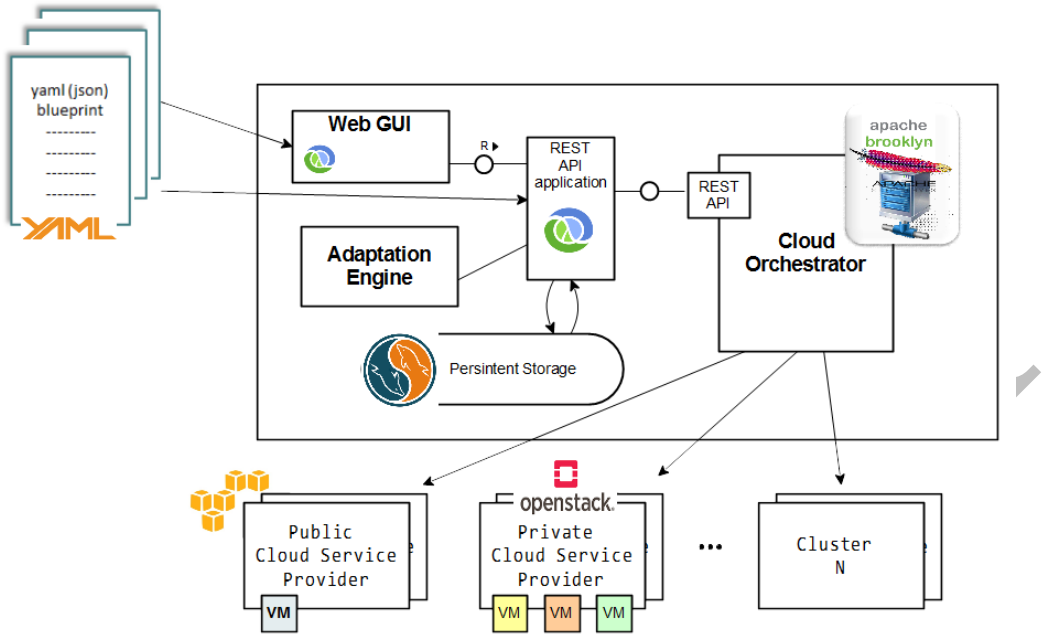

Figure 9. RMO sofware module technology map.

3.3.3 Sequence Diagrams

The figure below (Figure 10) describes the interactions with the runtime environment.

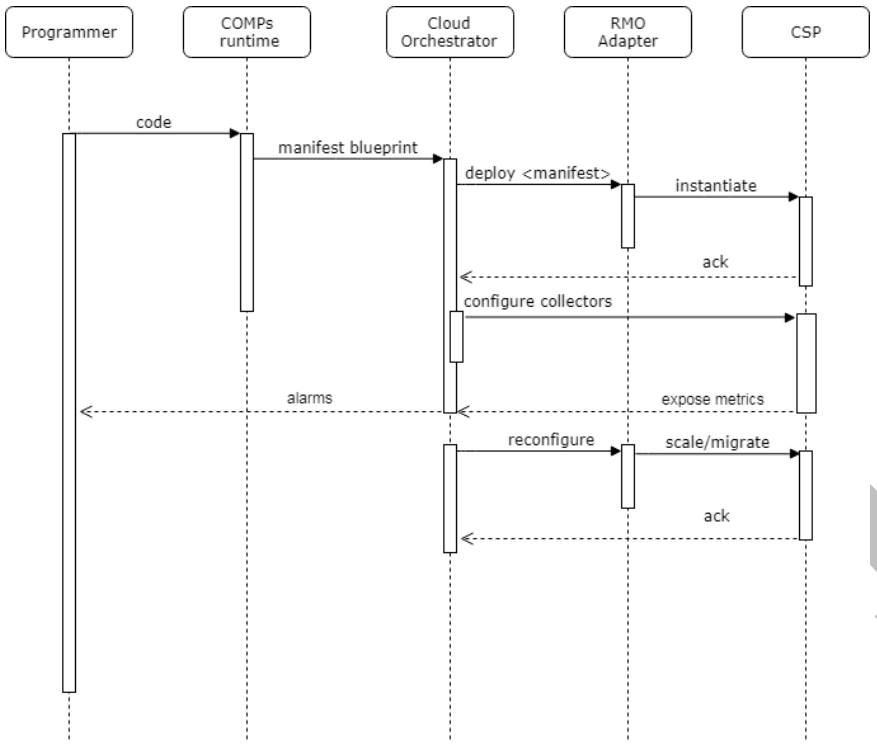

Figure 10. D5.1 RMO Sequence diagram

3.4 Roadmap and Features

3.4.1 Tasks Roadmap until M12

Month	Summary of Main Actions
M6	• Organisation of the work plan through teleconferences and face to face meetings. • Contribute to requirements and architecture for I-BiDaaS solution.
M7	• Infrastructures provisioning and setup • Knowledge acquisition
M8	• Training • Build PoC selected stacks
M9	• Alignment with I-BiDaaS architecture • Feedback across technical WPs
M10	• Infrastructure setup • First ToC of D5.1 • Orchestration management evaluation • MVP time plan
M11	• Initial infrastructure setup • Orchestration prototype • Runtime environment integration analysis.
M12	• Deliverable and Quality Review Process

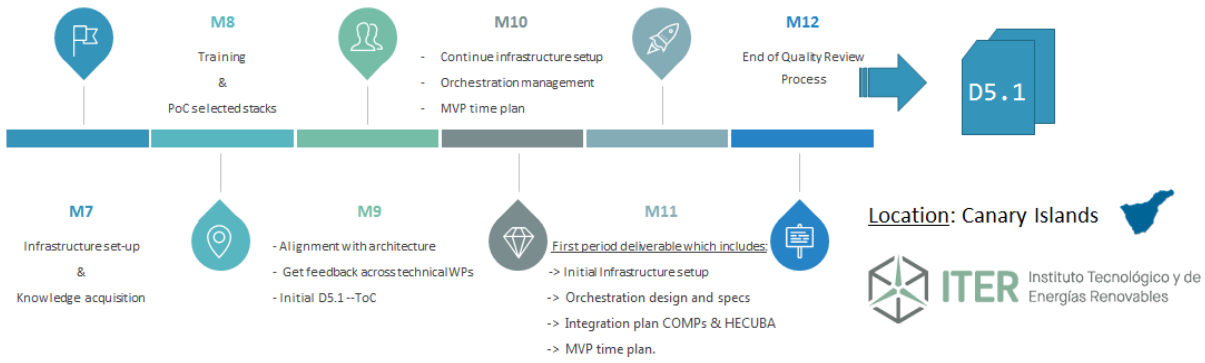

Figure 11. D5.1 Time plan & RoadMap

3.4.2 Code Repository

The source code of the software modules is going to be stored in the source control system of the project based on GitHub. Further information is documented in the deliverable "D5.2. Big-Data-as-a-Self-Service Test and Integration Report".

4 Runtime Environment

4.1 Introduction

The runtime environment provides the execution environment for data processing applications as well as the interface with the data storage.

Based on the analysis of the user requirements (reported in D1.3 [2]) the I-BiDaaS platform *should support diversified, analytic processing, machine learning and decision support techniques to support multiple stages of analysis (FR4).*

This implies the ability to process huge volumes of data and therefore, programming productivity is crucial. To this end, distributed processing capability is required so that different processing algorithms can be implemented and executed in parallel.

This, however, poses significant programming challenges, inherent to concurrent and distributed programming, e.g., dealing with threading, messaging, data partitioning and transfer. Additional challenges relate to the complexity of the underlying distributed computing infrastructure, which enables the execution of multiple tasks in distributed resources.

Thus, the runtime environment should provide IT developers with a simple means for writing parallel and distributed applications that process big amounts of data, in order to improve programming productivity without sacrificing performance. At the same time, it should provide transparent interoperability with the resource management and orchestration component thus freeing IT developers from having to deal with the details of the specific resource infrastructure.

Furthermore, as non-relational databases are the most common solution when dealing with the huge data sets and massive query workloads relevant to big data, the runtime environment should provide a set of tools and interfaces, which facilitate programmers with an efficient and easy interaction with non-relational data base technologies.

4.2 COMPs

COMPSs [17], is a task–based programming model designed to facilitate the development of applications for distributed infrastructures, such as Clusters, Grids and Clouds.

Starting from the original sequential application, written using different programming languages (Python, Java, C++), the COMPSs runtime exploits the inherent parallelism of the code execution by detecting and taking care of the data dependencies that may exist between invocation of different parts of the code (tasks). One of the most relevant features of this programming model is the transparency with regards to the computational infrastructure that allows to completely isolating the application from the execution logic.

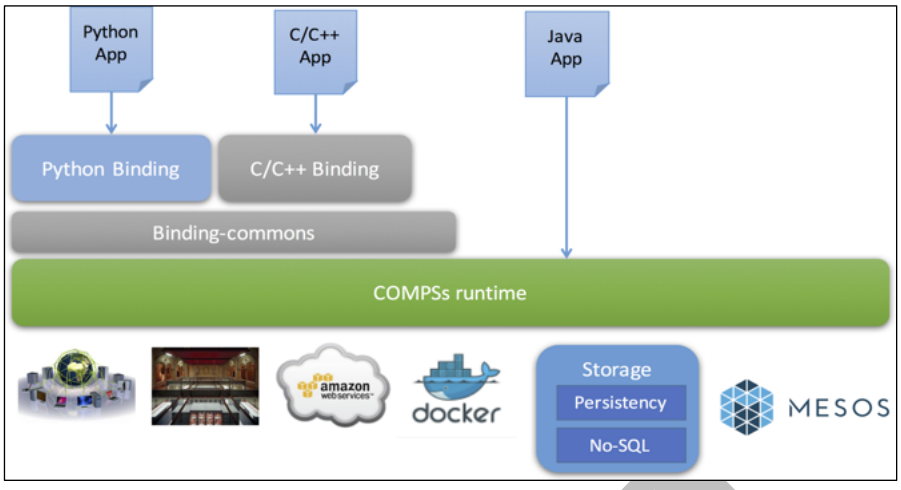

Figure 12. COMPSs architecture

As depicted in Figure 12, COMPSs has native support for Java applications with bindings for C/C++ code and Python scripts. In the model, the developer is mainly responsible for:

- Identifying the functions to be executed as asynchronous parallel tasks;
- Annotating them with a standard Python decorator or Java interface.

Then, the runtime [18] is in charge of exploiting the inherent concurrency of the execution of the functions, automatically detecting the data dependencies between tasks and spawning those tasks to the available resources, which can be nodes in a cluster, cloud or containers. A key aspect of providing an infrastructure-unaware programming model is that programs can be developed once and run on multiple backends, without having to change the implementation.

In COMPSs, the programmer is freed from having to deal with the details of the specific platform, since these details are handled transparently by the runtime. The availability of different connectors, each implementing the specific provider API (e.g., Cloud providers), makes it possible to run computational loads on multiple backend environments without the need for code adaptation. In cloud environments, COMPSs provides scaling and elasticity features that allow the number of utilised resources to be dynamically adapted to the actual execution needs.

Calls to annotated functions are wrapped by a function of the Python binding [19], which forwards the function name and parameters to the Java runtime. With that information, the Java runtime creates a task and adds it to the data dependency graph, immediately returning the control to Python. At this point, the main program can continue executing right after the task invocation, possibly invoking more tasks. Therefore, the Java runtime executes concurrently with the main program of the application, and as the latter issues new task creation requests, the former dynamically builds a task dependency graph. Such a graph represents the inherent concurrency of the program and determines what can be run in parallel. When a task is free of dependencies, it is scheduled by the run-time system on one of the available resources, specified in XML configuration files.

The default scheduling policy of the runtime is locality-aware. When scheduling a task, the runtime system computes a score for all the available resources and chooses the one with the highest score. This score is the number of task input parameters that are already present on that resource, and consequently, they do not need to be transferred.

4.2.1 Hecuba and Qbeast

Hecuba [20] is a set of tools and interfaces developed at BSC, that aims to facilitate programmers an efficient and natural interaction with a non-relational database. Using Hecuba, the applications can access data like regular objects stored in memory and Hecuba translates the code at runtime into the proper code, according to the backing storage used in each scenario. Hecuba integrates with the COMPSs programming model [18]. This integration provides the user with automatic and mostly transparent parallelization of the applications. Using both tools together enables data-driven parallelization, as the COMPSs runtime can enhance data locality by scheduling each task on the node that holds the target data.

The current implementation of Hecuba supports Python applications that use data stored in memory, in Apache Cassandra database or in ScyllaDB.

Hecuba supports the interaction between Apache Cassandra and both the advanced machine learning module and the TDF.

Qbeast is a multidimensional indexing system integrated with Hecuba that is provided also as part of the software stack. Qbeast implements a novel indexing algorithm based on D8tree algorithm [21], which is designed to support both analytical and data-thinning queries on multidimensional data while aiming to lower latency and achieving linear scalability. The main idea of the indexing algorithm is to merge multidimensional indexing, data sampling and denormalization techniques to achieve high scalability, availability, and performance.

4.2.2 Key-value database

I-BiDaaS platform uses Apache Cassandra [22] as the common database to keep all the data. Apache Cassandra is a distributed and highly scalable key-value database. Cassandra implements a non-centralized architecture, based on peer-to-peer communication, in which all nodes of the cluster can receive and serve queries. Data in Cassandra is stored on tables by rows, which are identified by a key chosen by the user. This key can be composed of one or several attributes, and the user has to specify which of them compound the partition key and which of them the clustering key. A partitioning function uses the partition key to decide how rows distribute among the nodes.

ScyllaDB [23] is a drop-in replacement for Apache Cassandra that targets applications with real-time requirements. The primary goal of ScyllaDB is to reduce the latency of the database operations by increasing the concurrency degree using a thread-per-core event-driven approach thus lowering locks contentions and better exploiting the modern multi-core architectures. The programmer interface and the data model are the same with Apache Cassandra. The peer-to-peer architecture that provides scalability and availability is also inherited from Cassandra.

In I-BiDaaS, we plan to perform some experiments replacing Cassandra by ScyllaDB. This change will not affect the implementation of the algorithms nor the architecture of the platform (because Cassandra and ScyllaDB are fully compatible) and will allow us to evaluate which data management solution fits better with the requirements of the I-BiDaaS use cases.

4.3 Integration Towards Minimum Viable Product (MVP)

4.3.1 Integration Methods and Tools

The integration approach followed in I-BiDaaS is based on the concept of Continuous Integration (CI) that is a cornerstone of modern software development. In fact, it is a real

game changer—when Continuous Integration is introduced into a project, it radically alters the way teams think about the whole development process. It has the potential to enable and trigger a series of incremental process improvements, going from a simple scheduled automated build right through to continuous delivery into production. A good CI infrastructure can streamline the development process right through to deployment, help detect and fix bugs faster, provide a useful project dashboard for both developers and non-developers, and ultimately, help teams deliver more real business value to the end user [24]. CI is a software development practice where team members/developers integrate their work and involves testing each modification a developer introduces to the codebase automatically, immediately after they occur [25]. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible [26] and to help software developers to complete tasks in a timely and predictable manner. In this context, it is an essential step that should be adopted in the frame of any new project as it serves as a practice of merging all developer working copies, thus allowing team members to develop cohesive software. More details on the CI approach and its benefits can be found in deliverable 5.2 "Big-Data-as-a-Self-Service Test and Integration Report (first version)".

In regard to the I-BiDaaS envisioned integrated solution, there are three major releases planned:

- A proof of concept demonstration (Minimum Viable Product (MVP) - M12
- A 1st complete prototype – M18
- A 2nd prototype – M30

CI essentially starts with the delivery of the MVP and is supposed to last until the delivery of the final prototype. Part of this deliverable is to identify and prepare the infrastructure and procedures needed for supporting CI from M12 to M30.

The software tools that will be used in I-BiDaaS Integration methodology (for integration testing also) are summarized in Figure 13.

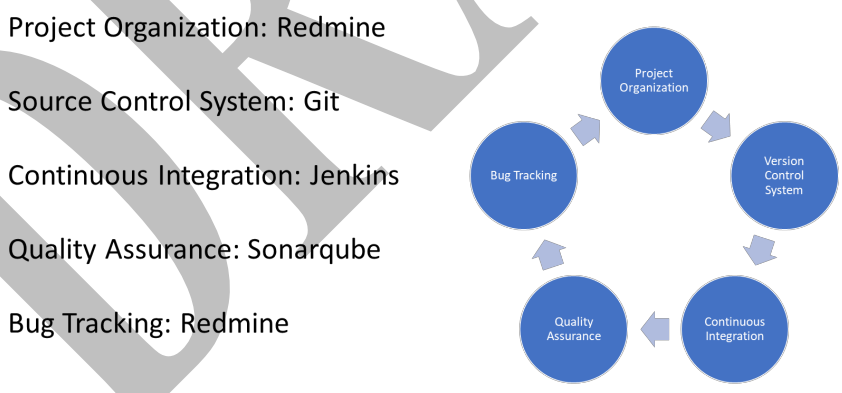

Figure 13. Software development

Regarding the basic requirements that have been identified for the integration towards the MVP, Hardware requirements are briefly presented in Table 3, while the technologies that will be used are presented in Table 4. More details on the specific requirements for the MVP can be found in deliverable 5.2 "Big-Data-as-a-Self-Service Test and Integration Report (first version)".

Table 3. Requirements

Component(s)	Platform	Cores	RAM	Storage
IBM InfoSphere® Optim Test Data Fabrication	Linux server (x86_64)	4	8GB	20GB
Hecuba (2 nodes)	Linux server (x86_64)	4 per node	4GB per node	20GB per node
Software AG Apama Server	Linux server (x86_64)	4	8GB	50GB
Software AG Universal Messaging	Linux server (x86_64)	4	8GB	50GB
Visualisation Toolkit / I-BiDaaS Dashboard	Linux server (x86_64)	2	4GB	30GB
Database (Cassandra)	Linux server (x86_64)	4	4GB	100GB
Integration Tools	Linux server (x86_64)	4	16GB	100GB
	TOTAL	30	56 GB	400 GB

Table 4. Technologies

Technology	Minimum Version
SonarQube	7.4
Redmine	3.4.6
Jenkins	2
Cassandra Database	3.11
Mysql server	5.6
Python modules: Numpy and cassandra-driver	--
Apache Tomcat	8
Apache Web server	2.4
Java	8
Ruby on Rails	2.0
Python	3.5
Node.js	6.14
Centos Linux	7.x
Ubuntu Linux	16.04

4.3.2 MVP Components

The CAIXA use case considered for the MVP specifies that data is fabricated according to the schemas defined by CAIXA with IBM's TDF. The data is written to a database and optionally simultaneously to a MQTT broker. In our case, this is SAG's Universal Messaging component.

4.3.2.1 Batch Processing

The Batch Processing components included on the MVP are one application derived from the use case and the technological components that this application required with the corresponding extensions: Hecuba, Apache Cassandra and the TDF tool. The goal of the application is to find relations between people, given a set of connections to IP addresses; more details are provided in deliverable D5.2 "Big-Data-as-a-Self-Service Test and Integration Report (first version)" [27].

We have developed three different versions for this application: the sequential version, the parallel version using COMPSs and the parallel version with Cassandra, using COMPSs and Hecuba. All these codes are available in a GitHub repository. For all three versions, the input data, describing all the connections, is assumed to be stored in Apache Cassandra and is accessed through the interface implemented by Hecuba. We plan to explore two different choices for TDF to provide data to the Batch layer of the I-BiDaaS platform: one is TDF to storing data directly on the database and the other one is TDF to communicating with Hecuba through Universal Messaging. For the MVP, we are considering only the first option.

4.3.2.2 Streaming analytics based on APAMA software

When a streaming use case is implemented, fabricated data is immediately grabbed by Apama from UM and analyzed. More details are given in the Apama related deliverables D4.1 [28] and D4.3 [29], the former is due at the same time as this deliverable, namely M12.

4.3.2.3 Visualization

The visualisation needs of the MVP are based on the Multipurpose Interface and interactive visualisation framework which are reported analytically in D2.3 [30]. Building upon the extensible visualisation framework, the MVP includes visualisations of the data that are generated in the context of the MVP use cases as described in the previous paragraph. More specifically, for the first use case, the goal of the visualisations is to display the various clusters of relationships and let the end-users quickly get an overview of the sizes of these groups as well as identify in what kind of relationships a specific transaction might be included. For the second use case, a visualisation of the generated alarms/events deriving from the stream processing component is available to let users know about detected events in real time.

5 Conclusions

According to the DoA, for WP5 the "*the primary objective is to provide the distributed large-scale framework that permits powerful and scalable Data processing on top of heterogeneous and federated infrastructures.*"

The actions performed to overcome the WP challenges are summarized in the table below:

Table 5. Challenges to Overcome

Challenge	Summary of actions to M12
Management of diverse infrastructure resources for Data Analytics (including cloud and GPU resources)	Provisioning and setup of a cloud infrastructure Setup of commodity cluster including GPU resources (see Section 2)
Orchestration of infrastructure resource management across diverse resource providers	Design and specification of the Resource Management and orchestration software modules and their interactions. (see Section 3)
Seamless integration and exploitation of infrastructure elasticity capabilities by the runtime environments	The integration methods and tools selected toward the MVP I-BiDaaS platform have been analysed during this period. Specification of the runtime environment which provides the execution environment for data processing applications as well as interface with the data storage pools. (see Section 4)

The work carried out within this period allows us to have an operational infrastructure environment supporting the realisation of the I-BiDaaS MVP prototype as well as the deployment of the software modules which are part of the I-BiDaaS solution.

The first period has been focused on the provisioning and management of the infrastructure while the next steps will be focused on the orchestration of computational resources for Big Data applications and I-BiDaaS use cases.

6 References

[1] "I-BiDaaS Consortium, D1.2: Architecture Definition," 2018.

[2] "I-BiDaaS Consortium D1.3: Positioning of I-BiDaaS," 2018.

[3] "D-Alix," 2018. [Online]. Available: http://www.d-alix.com/. [Accessed November 2018].

[4] "Helix Nebula," 2018. [Online]. Available: http://www.helix-nebula.eu/. [Accessed November 2018].

[5] "European Space Agency," November 2018. [Online]. Available: https://www.esa.int/ESA. [Accessed November 2018].

[6] "IonGAP," 2018. [Online]. Available: http://iongap.hpc.iter.es/. [Accessed November 2018].

[7] "Teide HPC," 2018. [Online]. Available: http://teidehpc.iter.es/en/services/. [Accessed November 2018].

[8] "Virsh," 2018. [Online]. Available: https://libvirt.org/virshcmdref.html. [Accessed November 2018].

[9] "Devstack," 2018. [Online]. Available: https://wiki.openstack.org/wiki/DevStack. [Accessed November 2018].

[10] "OpenStack Keystone," 2018. [Online]. Available: https://docs.openstack.org/keystone/latest/. [Accessed November 2018].

[11] "OpenStack Glance," 2018. [Online]. Available: https://docs.openstack.org/glance/latest/. [Accessed November 2018].

[12] "OpenStack Swift," 2018. [Online]. Available: https://docs.openstack.org/swift/latest/. [Accessed November 2018].

[13] "OpenStack Nova," 2018. [Online]. Available: https://docs.openstack.org/nova/latest/. [Accessed November 2018].

[14] "OpenStack Cinder," 2018. [Online]. Available: https://docs.openstack.org/cinder/latest/. [Accessed November 2018].

[15] "OpenStack Neutron," 2018. [Online]. Available: https://docs.openstack.org/neutron/latest/. [Accessed November 2018].

[16] "OpenStack Horizon," 2018. [Online]. Available: https://docs.openstack.org/horizon/latest/. [Accessed November 2018].

[17] Lordan, F., E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, and R. M. Badia, "ServiceSs: an interoperable programming framework for the Cloud," *Journal of Grid Computing, Volume 12, Issue 1,* pp. 67-91, 2014.

[18] Badia, R. M., J. Conejero, C. Diaz, J. Ejarque, D. Lezzi, F. Lordan, C. Ramon-Cortes, and R. Sirvent, "COMP Superscalar, an interoperable programming framework," *Software X, Volumes 3-4,* pp. 32-36, 2015.

[19] Enric Tejedor, Yolanda Becerra, Guillem Alomar, Anna Queralt, Rosa M. Badia, Jordi Torres, Toni Cortes, Jesús Labarta, "PyCOMPSs: Parallel computational workflows in Python," *IJHPCA 31 (1),* pp. 66-82, 2017.

[20] "Hecuba," 2018. [Online]. Available: https://github.com/bsc-dd/hecuba. [Accessed November 2018].

[21] C. Cugnasco, Y. Becerra, J. Torres, E. Ayguade, "D8-tree: a de-normalized approach for multidimensional data analysis on key-value databases," in *ICDCN '16: Proceedings of the 17th International Conference on Distributed Computing and Networking*, 2016.

[22] "Apache Cassandra," 2018. [Online]. Available: http://cassandra.apache.org/. [Accessed November 2018].

[23] "Scylla DB," 2018. [Online]. Available: https://www.scylladb.com/. [Accessed November 2018].

[24] "Jenkins Definitive Guide," 2018. [Online]. Available: https://www.bogotobogo.com/DevOps/Jenkins/images/Intro_install/jenkins-the-definitive-guide.pdf. [Accessed November 2018].

[25] "Continuous Integration on Software Development," 2018. [Online]. Available: https://blog.getty.io/importance-of-continuous-integration-on-software-development-30ab74c61c1. [Accessed November 2018].

[26] M. Fowler, "Continuous Integration," 2018. [Online]. Available: https://martinfowler.com/articles/continuousIntegration.html. [Accessed November 2018].

[27] "I-BiDaaS Consortium. D5.2: Big-Data-as-a-Self-Service Test and Integration Report," 2018.

[28] "I-BiDaaS Consortium. D4.1, Real Time Complex Event Processing Engine – Design And Approach," 2018.

[29] "I-BiDaaS Consortium. D4.3, Streaming analytics and predictions," 2018.

[30] "I-BiDaaS Consortium. D2.3, I-BiDaaS visualization and monitoring framework, and a multipurpose interface," 2018.

[31] Badia, R. M., J. Conejero, C. Diaz, J. Ejarque, D. Lezzi, F. Lordan, C. Ramon-Cortes, and R. Sirvent, "Comp superscalar, an interoperable programming framework," *SoftwareX,* pp. 32-36, 2015.