INDIGO-DataCloud

SPECIFICATIONS OF USE CASES FOR TESTING AND VALIDATION PURPOSES

EU DELIVERABLE: D2.3

Document identifier:	INDIGO-WP2-D2.3-V20
Date:	**5/Feb/2016**
Activity:	**WP2**
Lead Partner:	**U. Utrecht**
Document Status:	**FINAL**
Dissemination Level:	**PUBLIC**
Document Link:	

Abstract

This report summarizes the progress in the definition and partial implementation of use cases **for test and validation of INDIGO solutions** explored in Task 2.3. According to the needs and requirements from various user communities in WP2, the requirements on functionalities and resources of the scenarios for the test and validation activities in collaboration with WP3 and WP6 are proposed, and the first steps towards the implementation of uses cases are described.

I. COPYRIGHT NOTICE

II. DELIVERY SLIP

	Name	Partner/Activity	Date
From	Z. Kurkcuoglu, A.M.J.J. Bonvin	U.Utrecht/WP2	22 Dec 2015
Reviewed by	**Moderator:** J.Marco **Reviewers:** I.Campos, L.Dutka, R.Barbera	CSIC/WP2	21 Jan 2016
Approved by	**PMB**		5 Feb 2016

III. DOCUMENT LOG

Issue	Date	Comment	Author/Partner
1	25-nov-2015	First draft, including WeNMR sections	Z. Kurkcuoglu, A.M.J.J. Bonvin / U.Utrecht
2	11-Dec-2015	Updated draft including WP3 collaboration	J.Marco / CSIC
3	13-Dec-2015	Updated draft with new intro sections and LW sections	F.Aguilar, J.Marco / CSIC
4	14-Dec-2015	Updated draft including INSTRUCT sections	A. Rosato / CIRMMP
5	14-Dec-2015	Updated draft including EuroBioImaging sections	I. Blanquer / UPV
6	15-Dec-2015	Updated draft including GALAXY sections	M. A. Tangaro, F.Zembelli / CNR
7	18-Dec-2015	Updated draft including INAF/CTA and LBT sections and executive summary	L. A. Antonelli, S. Gallozzi / INAF-CTA, C. Knapic/INAF, J.Marco/CSIC
8	20-Dec-2015	Updated draft including WP6 section	M. Plociennik/PSNC
9	21-Dec-2015	Updated draft including ENES section	S.Fiore/CMCC
10	22-Dec-2015	Revised draft, ready for review	Z. Kurkcuoglu/ U.Utrecht J. Marco/CSIC
11	21-Jan-2016	Updated subtitles of Section 4 and WeNMR sections	Z. Kurkcuoglu/ U.Utrecht
12	26-Jan-2016	Updated INSTRUCT sections Updated GALAXY sections Updated EuroBioImaging sections Updated LifeWatch sections Updated ENES (Climate Model) sections Answers for INAF/CTA Updated INAF/LBT sections	A. Rosato / CIRMMP M. A. Tangaro / CNR I. Blanquer / UPV F.Aguilar, J.Marco / CSIC S.Fiore/CMCC S. Gallozzi / INAF-CTA C. Knapic/INAF
20	5-Feb-2016	Version ready for delivery	J.Marco/CSIC

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

This report introduces the use cases for test and validation of INDIGO solutions. It starts explaining the relevant role that they play in the project by providing the key components for this validation, trying to cover all the required functionality that will be provided by INDIGO using real examples of applications of interest for the different research communities. Furthermore, it introduces a methodology and first examples of the implementation and considers the complete definition of the tests.

The work presented here has been developed within the WP2 task 2.3, lead by U. Utrecht, and is based on the work reported in the previous deliverable D2.1. The additional work required to understand the details of the Case Studies to define the test use cases has been done mainly by the team of WP2 champions in each Research Community, and in coordination with the work done in tasks T2.1 and T2.2. Additionally, the first real implementation examples have been discussed with WP3, and also the final user perspective, including the practical implementation of the launching of different services, analysed with WP6.

The first test use cases presented in the report, and analysed in detail, including practical implementation points and also results of the execution in terms of scalability, or use of specialized hardware like GPGPU, corresponds to the WeNMR community (A worldwide e-Infrastructure for NMR and structural biology, see http://www.wenmr.eu), focused on the modelling of biomolecular complexes at atomistic and near-atomistic resolution using data from experimental techniques such as Nuclear Magnetic Resonance (NMR), Small Angle X-Ray scattering (SAXS) and Cryo-Electron Microscopy (Cryo-EM). The implementation of a local, self-contained version of HADDOCK portal, as well as two additional applications, DisVis and PowerFit, is proposed as test use cases. In addition, three other applications related to the structural biology field are proposed to be considered in the next iterations, including another one exploiting MPI-enabled HPC resources for parallelism.

A second set of detailed test use cases is proposed by the research community of EuroBioImaging. The tests proposed could be considered as an example regarding the analysis of functional requirements, as it is explicitly detailed how they will be tested by the corresponding procedure. The detailed implementation plans and the first results will be provided in our next deliverable, as some of the components will be only available after the first release of INDIGO solutions.

An additional set of detailed test use cases corresponds to applications of interest from the LifeWatch ESFRI community: a large simulation suite (Delft3D), and a complete NGS pipeline implementation (TRUFA). Here the details of the required resources are provided, based on current solutions, but again the implementation details and the experience will be included once the INDIGO solutions start to be available. Similarly, test use cases are included for the Galaxy portal (of interest for ELIXIR research community), the astronomical pipeline for CTA big data processing and LBT data distribution and archiving (INAF research community), and for the Climate simulation and analysis packages of ENES.

As indicated there is a different degree of maturity in the implementation of these test use cases, and although it is clear that not all Cloud middleware is available, the idea followed in this analysis was to explore each case up to the point it was possible: the definition of the tests in most cases, the detailed specification of resources for a few of them, and even execution details for the most advanced in terms of implementation.

This deliverable will be used as a guide for the implementation of all test use cases in the different Research Communities: it provides a path to the collaboration with WP3 and WP6, and it will be updated

incrementally as the champions continue their work in task 2.3, towards the preparation of D2.8, *Test and validation suite and results.*

2 INTRODUCTION

The first and most relevant expected impact from the INDIGO-DataCloud project is an **increased access and usage of e-Infrastructures by scientific communities**. In fact, it was designed with a clear objective in mind, as stated in the DoW:

"The proposal is oriented to support the use of different e-infrastructures by ***a wide-range of scientific communities, and aims to address a wide range of challenging requirements posed by leading-edge research activities conducted by those communities.*** *Indeed, the force driving this proposal is the interest of these communities and the organizations supporting them, from many different fields in science, from biomedicine to astrophysics, from cultural heritage to climate, participating in very relevant initiatives at European level, such as INSTRUCT, ELIXIR, EMSO, DARIAH, LIFEWATCH, etc."*

Keeping this focus in a project that at the same time has complex innovative technical challenges, is one of the missions of the tasks within WP2/NA2, a work package structured into five different tasks, with a clear scheme. Tasks *T2.1* and *T2.2* are promoting the different Research Communities use cases to JRA work packages, assuring that the solutions being developed within INDIGO follow specific requirements of the Research Communities. Deliverables D2.1 (released in June 2015) and D2.4 (that will be released in December 2015) are providing this input, and establishing a common framework for this interaction through the use of different tools, in particular OpenProject.

The objective of task *T2.3* to help in this mission is to implement, in close cooperation with WP3 and WP6, a test and validation chain to assure the integration of INDIGO solutions with the final user applications, as stated in the DoW: *"selected, realistic use cases will be deployed and detailed feedback provided"*. The following subsection details the context of this task and the relationship with this deliverable.

2.1 The context, WP2/T2.3 - Application Test and Validation

The main objective of task T2.3 is to ensure that all the middleware and other INDIGO solutions developed in JRA work packages WP4-WP6 meet the needs and requirements of the various user communities. It is therefore crucial to properly test and validate them as well as demonstrate their applicability on real use cases. To meet this objective different subtasks within T2.3 were defined and are being implemented:

- *Definition of use cases*: we started considering a set of existing, well defined, use cases that could be considered for implementation from the start of the project. This set includes applications well known to the Research Communities, and some of them deployed in other distributed computing or HPC previous frameworks. In parallel, additional use cases have been defined based on the input of T2.1 through deliverable D2.1. **The definition and partial implementation of these use cases is reported in this deliverable D2.3.**

- ***Creation of VMs for each use case:*** the idea is that for each use case defined, a virtual machine (or container) will be provided, meeting all requirements for running on the INDIGO testing infrastructure. This work has already started in close collaboration with WP3, responsible for this testing infrastructure and also for the procedure for INDIGO software test and release. The context is described in the following subsection of this deliverable.

-Implementation of automatic probes: oriented to perform all tests on a regular basis, to validate and monitor in the future e-infrastructures where INDIGO solutions are deployed, again in coordination with WP3, and using tools similar to NAGIOS.

-Creation of generic use case examples that can be used for dissemination and training purposes These examples are already being explored, taking into account the new solutions proposed by INDIGO in the technical deliverables D4.x, D5.x, D6.x.

In the context of task *T2.3*, this deliverable, D2.3, provides details of the use cases that have been originally defined and also of those newly identified from the feedback from user communities through deliverable D2.1 (in task 2.1), to test and monitor developments in WP3.

A forthcoming deliverable, D2.8, *Test and validation suite and results*, will describe the additional use cases implemented during the project to test and monitor the JRA developments, together with the corresponding monitoring tools and a report of their application.

Finally, the relationship with another WP2 task, *T2.4 Dissemination towards Research Communities*, must be mentioned: within this task the setup of the required e-infrastructure components to facilitate both training and demos at workshops, and on-line courses will be defined, and this work is being done in collaboration with task T2.3, that should provide the corresponding examples and experience.

2.2 Collaboration with WP3

As stated in WP3 deliverable D3.1 "*As part of software requirements verification WP3 interfaces with WP2 to enable user communities to preview, test and evaluate the software under adequate conditions thus gathering early feedback and promoting exploitation... A single helpdesk portal will be used both for support requests from external infrastructures and users, as well as for internal users from WP2...*"

Integration testing involves the deployment of the software component in a real scenario. Unlike the functional testing it MUST check the operation of the new functionalities with any other component with which it interacts. Along this line the collaboration with WP2 allows to explore the complexity of the many different, but real, scenarios that will be addressed, and in direct contact with the Research Communities that have defined the requirements and will exploit INDIGO solutions.

The test scheme adopted in WP3 will employ a Continuous Delivery approach (see Figure 10 in D3.1) to deal with the final steps required to deliver products ready for production: deploying on the Testing Infrastructure (also provided by T3.3), performing both automated and manual user acceptance tests and getting feedback from the user communities (WP2).

Task 2.3 in WP2 will benefit from this testing infrastructure as well to test the different use cases proposed, but to do so requirements should be transmitted to WP3, what is being done formally through this deliverable in section 5.

It is important to remark that regarding the Release Certification and Validation phase for INDIGO solutions, handled by WP3, during this phase, the software components are validated against the set of acceptance criteria defined by the customers, Technical Previews, alpha and beta releases, **are made available to user communities on the integration testbeds**. Then the final packaging and signing is performed. Components that do not pass the criteria are rejected and routed back to PTs for revision.

It is important also to remind that Continuous Delivery will be implemented using the Jenkins framework and that WP3 will deal also with the final steps required to deliver products ready for

production: deploying on the Testing Infrastructure and performing both automated and manual user acceptance tests and getting feedback from the user communities (WP2).

Finally, and regarding the collaboration between WP2 and WP3 on T2.3, it is also important to have in mind that task T3.4 will organize a number of workshops, tutorials and dissemination events to present the project results at popular yearly conferences as well as a number of solicited workshops at relevant organizations, and that for the purposes of the workshops, preconfigured infrastructure components will be set up and made available **in collaboration with task T2.3** to facilitate demos and self-taught courses.

2.3 Collaboration with WP6

WP6, by providing graphical user interfaces in the form of scientific gateways and workflows and means to access the INDIGO PaaS services and software stack, will allow the definition and set up of on-demand infrastructures to support WP2 test use cases.

There are already several integration scenarios identified during the requirements analysis and architecture design phases, and three of them are presented below.

From the WP6 perspective **the most common scenario** would see the administration of the end user service setup via the WP6 FutureGateway Portal. This would include all the services required by end users, e.g. a domain specific portal plus a batch system and data/storage end points. The administrator would be provided with ready-to-use recipes that he will be able to customize using a friendly graphical interface, in order to adjust parameters like the size of the resources required. Usually, many of the services will run for long periods, and the elasticity feature of the INDIGO PaaS will allow to increase/decrease the amount of required cloud resources to support them.

The final users of the domain specific services that are set up will receive the address of the services but will not need to be even aware of these INDIGO WP6 services.

The template for the recipes will be defined in close collaboration between WP2, WP6 and WP5.

A **second integration scenario** applies for the cases where the user communities have already their own Scientific Gateway setup, and would like to use the INDIGO features from inside of their Portals. For such cases WP6 provides an API set (the FutureGateway REST API) that allows using entire the INDIGO software stack. Such cases will require the development of domain specific plugins for domain specific tools that will allow integration of the corresponding INDIGO features. In this case, the final user again does not need to be aware of these WP6 services, and there will be ready to use recipes that could be instantiated via the API set.

The third scenario will mainly apply to the cases that already use, or would like to enrich, services based on the new tools and portals developed in WP6, including the FutureGateway Portal, Scientific Workflows Systems (like LONI, Taverna, Kepler), Big Data Analytics Frameworks (like Ophidia), or Mobile Applications. In this scenario the final users as well as domain administrators will use the WP6 GUI tools. The administrator will use them as described in the first case. In addition, domain specific users will be provided with specific portlets/workflows/apps that will allow graphical interaction with their applications run via the INDIGO software stack. This integration scenario requires the interaction between WP2 and WP6 to implement the specific parts of the interface, and this interaction has already started along the work presented in this deliverable.

3 DEFINITION OF USE CASES

This section describes the Use Cases that have been considered for the implementation on test and validation activities. Most of these use cases are derived from the Case Studies described in D2.1. However, the emphasis in this deliverable is put on their actual implementation and interest for test and validation, in particular considering if (i) their functional requirements cover a significant fraction of the expected INDIGO-DataCloud proposed solutions, and (ii) their deployment in the testing e-infrastructure allows an estimation of the interest for the Research Communities, in terms of improvement both in power and in ease of use, an important point also for WP6. The following subsections introduce the different applications, while the next section is devoted to a more detailed analysis of those that are closer to an initial implementation plan, describing the e-infrastructure resources requirements.

3.1 Local Version of HADDOCK

HADDOCK is an integrative, information driven approach for modelling macro-molecular assemblies. The software is available through a user-friendly web-interface, offered both on local resources and within the context of the WeNMR Virtual Research Community. Currently, some statistics about HADDOCK include >1000 local installations, with more than 6000 users worldwide, submitting in the order of 25000 molecular docking runs per year, which translate into >10 million single jobs per year submitted to HTC infrastructures like the European Grid Infrastructure (EGI). The Bonvin group at the University of Utrecht is developing and distributing the software (http://bonvinlab.org/software/haddock2.2/haddock.html), and operating the web portals. HADDOCK is a typical high-throughput application that has already been ported to the grid. For a more detailed description about HADDOCK, please refer to Deliverable 2.1 and references provided therein.

As already advanced in the proposal for Task 2.3, a local, self-contained version of the HADDOCK portal in a VM is a use case to be implemented from the start of the project. This virtualized HADDOCK portal should meet the entire required computational infrastructure together with computing resources to run on the INDIGO-DataCloud testing infrastructure. The functional and e-infrastructure requirements for this use case are described in Section 4.1 of this report. The inputs for HADDOCK are text files, which are described in detail, in reference [R1].

3.2 Multi-Threading and MPI-based Molecular Dynamics

Structural biology deals with the characterization of the structural (atomic coordinates) and dynamic (fluctuation of atomic coordinates over time) properties of biological macromolecules and adducts thereof. The dynamic properties of these systems are crucial to many aspects of their biological function, such as recognition of molecular partners or diffusion of small molecules (substrates, products, inhibitors) to/from the active site of catalytic machineries. These properties are hard to characterize experimentally in a direct and comprehensive (i.e. for all atoms) manner at the atomic level. Consequently, researchers in the field largely rely on computer simulations to tackle the dynamic aspect of structural biology. Simulations can be validated by comparison to different types of experimental data. CIRMMP and U. Utrecht, both partners of this project, have been involved in facilitating the adoption of molecular dynamics (MD) simulations by the broad biological community, through the implementation of dedicated web portals to run short simulations on a grid computational infrastructure. Such portals are available within the context of the WeNMR Virtual Research Community. For example, the AMPS-NMR portal developed by CIRMMP allows users to optimize the energetics and stereochemical properties of protein structures determined by NMR spectroscopy [R2].

Within the INDIGO-DataCloud project, a main objective is the implementation of interfaces that allow users to run multi-threading as well as MPI-based molecular dynamics simulations depending on specific needs, provide access to the simulated data (trajectories) and perform standardized analyses of the trajectories. For multi-threading simulations, users will exploit a number of cores in the range 20-40, which do need to reside on the same processor. Each simulation is independent of the others.

3.3 The Maximum Occurrence Approach for the Characterization of Internal Dynamics in Multi-Domain Proteins

Protein mobility occurs over an extensive range of timescales and structural levels. Local, fast mobility has been already extensively addressed in the scientific literature. NMR spectroscopy allows the direct measurement of protein flexibility through so-called nuclear relaxation measurements. These data can be interpreted using different physical models and report on the motions affecting individual chemical groups within each amino acid of the protein. The most common framework of analysis is the model-free formalism. On the other hand, motions that involve larger portions of the overall protein structure are harder to characterize experimentally and there is no consensus in the community on the physical model(s) for their interpretation. Combination of experimental data from different structural techniques are required for the determination of dynamic features based on the combined analysis of independent data types. Within this context, CIRMMP has developed an original approach to estimate the probability of various conformations in multi-domain proteins, through the integration of NMR and SAXS (Small-Angle X-ray Scattering) data. This approach is called the Maximum Occurrence (MaxOcc) approach. In this method, we compute the maximum occurrence of each and every conformation in a large pool of randomly generated conformations, which can be assumed to exhaustively sample the entire conformational space of the protein. The MaxOcc of any given conformation is defined as the maximum weight that it can obtain when part of a conformational ensemble without violating the constraints of the experimental data. No restriction is posed on the number of conformations to be included in the ensemble. The MaxOcc value can be interpreted as the maximum fraction of time that a conformation can exist, when taken together with any ensemble of conformations with optimized weights.

The MaxOcc approach has been implemented also through a web portal within the WeNMR Virtual Research Community [R3]. A local, self-contained version of this web portal has similar requirements to the HADDOCK portal already mentioned in previous section, and thus can be used as a VM in the context of INDIGO.

3.4 Climate Model Intercomparison Analysis

The case study on **Climate models intercomparison data analysis** relates to the scientific community organized within the European Network for Earth System modelling (ENES). It is directly connected to the Coupled Model Intercomparison Project (CMIP), one of the most internationally relevant and large climate experiments as well as to the Earth System Grid Federation (ESGF) infrastructure in terms of existing eco-system/services.
In the last three years, ESGF has been serving the Coupled Model Intercomparison Project Phase 5 (CMIP5) experiment, providing access to 2.5PB of data for the IPCC AR5. The proposed case study will consider input datasets from this federated, global data archive (CMIP5 data archive). In this regard CMCC provides about 100TB of CMIP5 climate simulations datasets (through the ESGF data node running at the CMCC SuperComputing Centre) related to three different CMCC models.
With specific regard to the CMIP context, the proposed test case study focuses on the following use case on **"large scale climate model intercomparison data analysis"**.

The test case provides a common approach useful for three different scientific data analysis classes:

- *anomalies analysis*: anomalies are defined as an incidence or occurrence when the actual result under a given set of assumptions is different from the expected result. An anomaly provides evidence that a given assumption or model does not hold in practice;
- *trend analysis*: analyzing data trends is an age old and powerful tactic, which is used to measure the performance of marketing campaigns over time and to predict future outcomes. Trend Analysis is the practice of collecting information and attempting to spot a pattern, or trend, in the information. Although trend analysis is often used to predict future events, it could be used to estimate uncertain events in the past;
- *climate change signal analysis*: the treatment of "signal" and "noise" in constructing climate scenarios is of great importance in interpreting the results of impact assessments that make use of these scenarios. If climate scenarios contain an unspecified combination of signal plus noise, then it is important to recognise that the impact response to such scenarios will only partly be a response to anthropogenic climate change; an unspecified part of the impact response will be related to natural internal climate variability.

A step-wise description of this test case is reported below:

1. The user selects a data analysis experiment (anomalies analysis, trend analysis, climate change signal analysis) through a specific interface provided by a scientific gateway.
2. The experiment is defined in terms of a set of input (e.g. variables, models, scenarios, timeframe, bounding box, etc.) and a workflow for data analysis. The workflows will be also published in a workflow repository for further re-use, community-based feedback, link to the scientific gateway, etc. To this end, existing market-place tools, will be used (in particular MyExperiment[12]).
3. The experiment will involve datasets at a single site or at multiple sites. To this end, two different levels of Workflow Management System (WfMS) would be deployed: one coarse

[1] http://www.myexperiment.org/home

[2] Goble, C.A., Bhagat, J., Aleksejevs, S., Cruickshank, D., Michaelides, D., Newman, D., Borkum, M., Bechhofer, S., Roos, M., Li, P., and De Roure, D. (2010): **myExperiment: a repository and social network for the sharing of bioinformatics workflows**. *Nucl. Acids Res.* **38** (suppl 2): W677-W682. . http://dx.doi.org/10.1093/nar/gkq429 doi:10.1093/nar/gkq429]

grain for large scale, distributed, and multi-service tasks orchestration (like Kepler[3]) and another one fine grain, which relates to the core data analytics part of the experiment and will coordinate the workflow execution at the level of big data analytics cluster instance (Ophidia[4][5]). Some additional infrastructure-level sub-points regarding the experiments run are reported in the following:

   a. different scenarios – directly related to the components deployment – will include: (i) static configurations, (ii) dynamic, and (iii) elastic ones; the first two options are key and mandatory for this use case;
   b. the deployment of the data analysis environment will be platform-agnostic;
   c. interoperable aspects related to the available security eco-system/infrastructure will be considered too;
   d. security will be taken into account at different levels (e.g. gateways, data and analysis services) and from different points of view (e.g. authN, authZ);
   e. search engines from the existing eco-system will enable the data discovery;
   f. data locality for the required analysis will be strongly taken into account;
   g. Workflow Management Systems (WfMSs) will be instantiated, according to a Workflow as a Service (WaaS) model.

4. Some of the tasks in the workflow can be related to running existing software like visualization tools well-known in the community (e.g. NCL, NCAR Command Language) or other ones for data manipulation and analysis, based on users needs and preferences. Moreover, specific interfaces are expected to support an easy definition of big data analytics experiments on large datasets through a declarative approach (e.g. massive, parallel statements in the workflow definition).
5. Considering that the experiment runs could take from minutes to hours (or even more depending on the set of inputs defined by the user) notification mechanisms (e.g. email) will inform users (when requested) about the status of the experiments.
6. The results of the experiments will be easily available to the end user for inspection, download, visualization through the scientific gateway interface. To this end, the user interface (e.g. scientific gateway) will provide specific/advanced support for data analytics and visualization.
7. The publication of the final results on HTTP/FTP or domain-specific services (e.g. OPeNDAP/THREDDS[6]) will be a possible option for the end users. That will enable a better sharing regarding the experiments results.

[3] Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J. and Zhao, Y. (2006), Scientific workflow management and the Kepler system. Concurrency Computat.: Pract. Exper., 18: 1039–1065. doi:10.1002/cpe.994

[4] S. Fiore, A. D'Anca, C. Palazzo, I. T. Foster, D. N. Williams, G. Aloisio: Ophidia: Toward Big Data Analytics for eScience. ICCS 2013: 2376-2385

[5] S. Fiore, C. Palazzo, A. D'Anca, I. T. Foster, D. N. Williams, G. Aloisio: A big data analytics framework for scientific data management. BigData Conference 2013: 1-8

[6] http://www.unidata.ucar.edu/software/thredds/current/tds/TDS.html

3.5 CTA Big Data Processing

The Cherenkov Telescope Array (CTA) is a large array of Cherenkov telescopes of different sizes and deployed on an unprecedented scale: a huge array of telescopes in two different sites (one in the southern hemisphere ~100 telescopes and one in the northern hemisphere ~20 telescopes).

The tens of telescopes within the Cherenkov telescope array will produce a huge amount of data (in the order of ~300PB), thus requiring a challenging software architecture for the whole observatory, since reliable data processing, data access, their dissemination and transmission are mandatory. The CTA data and their scientific products need to be preserved in a dedicated archive whose aim is to provide open access to a wide and diverse scientific community for several years after the end of the CTA operative life (~30 years).[7]

Strictly related to scientific/research challenges are the e-infrastructural challenges; it is possible to summarize the two principal challenges:

1. One of the e-Infrastructure challenges under consideration will consist in defining an adequate computing model for CTA, mainly distributed, and in exploiting the big data technology with an increased level of complexity added by other experiment sites.[8]
2. The second challenge will be to enable data dissemination to the scientific community through Science Gateway and Single Sign-On solutions for this huge amount of data.

The Computing Model should result from a series of technical solutions suitable for the data pipeline and data reduction constraints as well as for the major user requirements for CTA data management.

The high data rate of CTA, together with the large computing power requirements for Monte Carlo simulations, requires dedicated significant computer resources. Another serious functional constraint to the Computing Model is a required Archive System, i.e. a combination of the hardware and associated services for data I/O (in the usual astronomical meaning), for permanent storage of data products that must provide the scientific users with an efficient and organised access to data. A distributed approach could be more adequate for the scale of CTA. Grid solutions for CTA can be feasible in order to optimize tasks.

Another important CTA goal is to provide a common User Interface: a CTA Science Gateway with complex management of authentication and authorisation mechanisms. The CTA Science Gateway will provide access to resources and services from a distributed computing infrastructure. The resources

7 Though CTA data are Virtual Observatory compliant, the use of VO is recommended only for higher level science products from the observations. These VO products can be easily handled since the total size is not in the range of big-data horizon, like the whole data produced and handled by the CTA archive.

8 Different external experiments for Very High Energy gamma rays observatories like HESS, MAGIC and CTA prototypes like ASTRI will suggest several technological solution to be implemented in the CTA contest. The ASTRI/Mini-Array project in particular will lead the CTA preproduction phase so all identified solution will be first tested in the pre-production phase and then applied to the final production for CTA.

could be grid computing and storage resources, public and private cloud services, local personal computer resources, user-specific laboratory/institute storage and computing resources, together with CTA observatory storage and computing resources.

3.6 LBT Data Distribution and Archiving

Astronomical data are taken continuously at an operational Ground Based Telescope like Large Binocular Telescope (LBT). Several modes (service, observing, queue) are usually adopted to perform observations and data are archived and distributed accordingly to well-defined roles. The first step after data acquisition is to perform data preservation as a correct, fast and always reliable data storage. After this, accordingly with previously well-defined policies, data are distributed to end users both by replicating the data to different sites and allowing data retrieval through web interfaces. Information on data ownership is stored first in the data itself, and at a later time it is stored into customized databases with which software business logic could handle the proper data distribution to owners. The peak data rate of an optical telescope like LBT is about 100 GB/night, currently subdivided in packages (FITS files) every few seconds of about 100-500 MB each. In particular, LBT images are taken with different instruments depending on the scientific goal. Some instruments have very high resolution and large field of view. CCDs or detectors could create large size images with integration times that could span from less than one second to a few seconds of integration again depending on scientific goals. This could produce a peak of 100 GB/night of raw data. The procedure to ingest such data into an archive is not so critical as in other cases. Data propagation through geographically distinct sites is much more critical in terms of reliability and data integrity. In the LBT archive case, the Tucson general archive is mirrored to two distinct sites in Europe and provides data to other two locations. The current software developed for data distribution on transoceanic networks is based on VMs properly set up and configured to host several packages and a distributed control system. The default machine configuration requires 50 GB for the whole installation and two CPUs with 4 GB RAM. This kind of VM is equipped with a relational database (where the relevant metadata describing the data is stored), the control system for data ingestion and the software application for data and metadata distribution. It requires some open ports (4000, 5000). Both metadata and data distribution are currently handled via a customized business logic internal to software processes. The same machine could also host the web applications for data retrieval. The web application business logic, together with the database procedures, currently apply retrieval data policies without a federated Authentication and Authorization mechanism, simply relying on local business logic. Since the International Virtual Observatory Alliance directives foresee the possibility to deal with private data, a recent goal of the LBT archive web interface is to be able to manage Authentication and Authorization using SAML 2 and federated Identity and Service Providers. With such a system, data management via workflow management systems would also be facilitated. One example of this is the adoption of VOSpace for LBT data.

3.7 GALAXY

ELIXIR groups Europe's leading life science organizations, aiming to create an infrastructure which integrates their research data and computational platforms, ensuring a seamless service provisioning that is easily accessible to all and providing tools to exploit this infrastructure.

The recent introduction of Next Generation Sequencing (NGS) and other data intensive technologies produce massive amounts of data that require powerful computational infrastructures, high quality bioinformatics software and skilled people to operate the tools used to analyse and exploit the data.

Galaxy is an open source web-based workflow manager platform for bioinformatics analysis. It is designed to allow data analysis using multiple tools and complex bioinformatics workflows. The results of each analysis can be viewed and published or shared with other users of the same platform. Galaxy capabilities can be expanded and tailored to the user needs by the platform administrator through the installation of tools from public repositories or by integrating any custom bioinformatics tool developed by users into the platform.

At present, there are more than 70 public Galaxy servers, with some oriented to specific types of analysis and others more general.

The use case described here consists in the development of a fully customizable Galaxy instance provider platform based on the technology developed within the INDIGO-DataCloud project. The use case aims to provide an easy setup of on-demand Galaxy workspaces, ready to be used both by life scientists and bioinformaticians lacking the necessary resources (either human or computational) to deploy their own custom instance.

Each single instance will allow the Galaxy administrator end user to add custom tools and data available or not in Galaxy's Tool Shed repositories and to each Galaxy user to upload their own data to the instance for analysis. New pipelines can be designed by the end users and deployed on the instance. Moreover, each Galaxy instance will be configured according to the Virtual Machine hardware configuration.

The requirements for this use case are listed in section 4.6 of this report.

3.8 Additional Use Cases

The previous use cases were proposed in INDIGO DoW. As Research Communities progress with their different activities and challenges, and at the same time start to explore the possibilities offered by INDIGO solutions, new use cases have been proposed. They are described in the following subsections.

3.8.1 POWERFIT

PowerFit is a software, developed by the Bonvin group, for automatic rigid body fitting of biomolecular structures in Cryo-Electron Microscopy (Cryo-EM) densities [R4]. PowerFit is a Python package and a simple command-line program, able to make use of single/multiple CPUs or GPUs to accelerate the calculations. The inputs (Protein Data Bank, PDB, Cryo-EM data) and outputs (PDB, log files etc.) for PowerFit are all text files. It is available at www.github.com/haddocking/powerfit.

PowerFit is included as an additional use case in this deliverable to provide an example employing Docker software with multiple CPUs, and especially GPU related cases. A use case for PowerFit can be described as a VM meeting the requirements listed in Section 4.8.1 of this report.

3.8.2 DISVIS

DisVis is another software developed by Bonvin group, for the visualization and quantification of the accessible interaction space of distance restrained binary biomolecular complexes [R5]. Like PowerFit, DisVis is a Python package and a simple command-line program, with the ability of harnessing multiple CPUs and GPU. The inputs and outputs for DisVis are all text files. It is available at www.github.com/haddocking/disvis.

The requirements of this software are similar to PowerFit, and are listed in Section 4.8.1 of this report.

3.8.3 Algae Bloom Modelling for Cdp Water Reservoir using Delft3D

The Case Study from the LifeWatch Research Community addressing the problem of Algae Bloom (see D2.1) is well suited for test and validation of INDIGO solutions. The reason is that it includes an HPC component, the Delft3D software suite, used in the modelling, a complex data management, due to the large and varied datasets required both as input to the model and as validation data, including instrumentation measurements, and a rich visualisation and post processing framework [R6].

Delft3D (see http://oss.deltares.nl/web/delft3d) is an open source software suite, that includes components to model a water reservoir first from a hydrological perspective, and second from the point of view of the water quality, including the evolution of algae under eutrophication conditions. Delft3D includes some components that have been adapted to HPC environments.

Delft3D results can be visualized using different internal components, and post-processed with tools like R-Studio or Jupyter to perform analysis like the fit of thermoclinae parameters.

Data management tools to integrate all the required input and obtained output files in the model are yet under development, and handled manually as Delft3D input/output.

Data management tools to support the integration of the instrumentation data, and also its remote and online visualisation, are based on open source and basic components, like mySQL DBMS and web portals. A common underlying distributed data storage system allowing the different actors (researchers, managers, developers) to access all these data and also preserve them, is also required.

3.8.4 TRUFA

TRUFA (TRanscriptome User-Friendly Analysis), is an open informatics platform offering a web-based interface that generates the outputs commonly used in de novo RNA-seq analysis and comparative transcriptomics [R7]. TRUFA provides a comprehensive service that allows performing dynamically raw read cleaning, transcript assembly, annotation, and expression quantification. Due to the computationally intensive nature of such analyses, TRUFA is highly parallelized and benefits from accessing high-performance computing resources. The complete TRUFA pipeline was validated using four previously published transcriptomic data sets. TRUFA's results for the example datasets showed globally similar results when comparing with the original studies, and performed particularly better when analyzing the green tea dataset. The platform permits analyzing RNA-seq data in a fast, robust, and user-friendly manner. Currently accounts on TRUFA are provided freely upon request at https://trufa.ifca.es. TRUFA has been developed by IFCA in collaboration with MNCN (Spanish Natural Science Museum, also in CSIC). Access to the web portal is available under subscription for the research community.

TRUFA is currently using HPC resources and is under migration to cloud environment. The aim of this use case is to use INDIGO solutions to deploy TRUFA on EGI FedCloud resources.

3.8.5 Medical Imaging Biobanks – Population Imaging

The use case of Medical Imaging Biobanks focuses on the support to the development of subprojects on population imaging. This use case constitutes one of the main targets of the BIM-CV (http://ceib.san.gva.es/bimcv) node of the EuroBioImaging (http://www.eurobioimaging.eu/), ESFRI. This node is providing the access to a huge medical imaging databank for population-level research, as it gathers historical data from the cases of a wide area in the Valencia Region.

BIM-CV medical imaging databank is the main resource exploited in this node. BIM-CV is reorganizing its structure to deal with the expected higher demand that the participation as a node of EuroBioImaging will produce. In this sense, BIM-CV will need a mechanism to deploy virtual infrastructures on demand including selected data, processing tools and virtual hardware. The main use case is the following:

1. External users submit a proposal of a subproject. A subproject will involve a subset of the data in the imaging databank, a set of processing tools and a request for resources. If successful, the subproject will have access granted. For the sake of better understanding, we will use one example: The development of an automatic biomarker for the evaluation of hip joint fracture in patients with osteoporosis.
2. Subprojects need to be prepared in three steps:
    a. Data selection and ingestion. A wide off-line query is performed to the databank, extracting the subset of data. In our example, CT abdominal images of patients of age above 65 that suffered a hip joint fracture in the following 5 years, as well as an equivalent set of controls (patients who did not suffer from a fracture). Images are anonymised, keeping sex, age, ICD diagnosis and final evolution.
    b. Definition of a common software processing environment for such case, including an XNAT server, a processing pipeline, MRIcron, dcm4chee and other basic tools, including the configuration steps required. This is complemented with the project-specific requirements, prepared in conjunction with the subproject IT manager. Add the virtual infrastructure requirements. In the case of the example, caffe, octave, python and Jasper will be needed. The results may be a combination of a TOSCA specification plus a previously configured VMI.
    c. Bundle everything and provide a service to deploy and delete such a virtual infrastructure.
3. Once the virtual infrastructure is prepared, the subproject IT manager is in charge of booting it, monitoring it, creating new users for the XNAT portal, and dealing with other management tasks.
4. Users accessing the platform will see the reference data (the extracted files from the Imaging Databank), and run processing pipelines through the XNAT portal tuning up the parameters, updating the models in Caffe, or updating the octave code. This should work as it would be running on a local infrastructure available to the end user. Interactions will normally be performed on the portal, but many projects may require direct access to the infrastructure.
5. Results obtained from the segmentation, quantification or quality enhancement are new images and reports. Therefore, it is necessary to browse results, re-use them, remove the irrelevant files and download those that the user will like to. Although not necessary in this use case, it may be possible that multiple users will have separate storage areas. This may be performed at the application level.
6. Neither the end-user, nor the IT manager should need to increase/decrease the number of processing resources available. This must be done automatically, except for the front-end.
7. Data should be preserved for further executions, system maintenance shutdown, or updating. Not all the data should be kept for the long term, but all the data should be available during the execution of a subproject.
8. Subproject will be bounded temporarily. They will have a deadline that could be extended by decision of the BIM-CV node. Access for the deployment of the virtual infrastructure should be revocable by the BIM-CV.

9. The final outcome of the subproject will include provenance information that could enable repeating the experiment. It is envisaged that end users will provide access to the platform to external reviewers of projects and papers. The virtual infrastructure can be used for the concept of “executable paper”.

4 REQUIREMENTS ON RESOURCES AND FUNCTIONALITIES TO IMPLEMENT TEST USE CASES

As explained in the previous sections, once the different use cases of interest for test and validation activities in collaboration with WP3 and WP6 have been identified, and after the functional requirements for INDIGO architecture have been compiled (see D2.1 and its update in D2.4), it is time to start considering the practical implementation in the different e-infrastructure facilities.

The purpose of this section is to integrate a first compilation of the basic requirements on functionalities **and resources, related to the corresponding research communities,** to plan test and validation facilities to be operative as soon as INDIGO solutions start to be available, and even anticipate the discussion on potential issues.

Given the diverse status of maturity of the applications and of the technology required to support the corresponding functional requirements, the information provided has a varying level of details.

4.1 Local Version of HADDOCK

The requirements for a local, self-contained version of the HADDOCK cluster in a VM are the following:

Computational requirements:

- Launching a single VM on a set of nodes, where the nodes are multicore
- Each node should ideally have 1 GB of memory per core
- Master node should have at least 4 cores

Storage requirements:

- Master node should have enough storage space (/home should have >250-500 GB)
- Size of /tmp on the nodes should be at least 0.5 GB per core to allow enough scratch /tmp space during job execution

Network requirements:

- Connectivity between nodes and master should be of at least 1 GB switch

Software requirements:

- Batch system installed and preconfigured with the number of cores per node equal to number of queue slots per node (TORQUE/Maui are the preferred schedulers)
- NFS mount of the home partition on all nodes
- All components running Scientific Linux, with Python version 2.7 or higher (within the 2.X range), together with csh/tcsh, gcc, fortran compilers, awk (or gawk), sed.

Once a VM with the properties stated above is launched, the testing should be carried out on the test-cases already present in the HADDOCK software. However, the HADDOCK portal operator should perform testing of the HADDOCK VM, since specific expertise is required to run the software. In this use case, the HADDOCK portal operator can deploy and configure the cluster with the specifications listed above, using EC3: Elastic Cloud Computing Cluster and test the performance.

4.2 Multi-Threading and MPI-based Molecular Dynamics

The requirements of a VM to run Molecular Dynamics simulations using MPI or multi-threading, depending on the specific application, are the following:

Computational requirements:

- Launching a single VM on a set of nodes. For the MPI application the nodes should be multicore, with at least four cores per processor
- Each node should have at least 5 GB of memory

Storage requirements:

- Master node should have at least 200 Gb of storage available. MD applications generate multiple large files, which together define the "trajectory" (i.e. coordinates as a function of simulated time)

Network requirements:

- Connectivity between nodes and master should be of at least 1 GB switch

Software requirements:

- Batch system installed and preconfigured with the number of cores per node equal to number of queue slots per node (TORQUE/Maui are the preferred schedulers)
- All components running Scientific Linux, with Python version 2.7 or higher (within the 2.X range), together with csh/tcsh, gcc, fortran compilers, awk (or gawk), sed.
- OpenMPI libraries

An implementation of multi-threading Molecular Dynamics has been performed also for GPGPUs. This requires CUDA.

4.3 The Maximum Occurrence Approach for the Characterization of Internal Dynamics in Multi-Domain Proteins

To deploy the MaxOcc software on a VM, the following requirements should be met:

Computational requirements:

- Launching a single VM on a set of nodes. The total number of cores available should in the order of 100
- Each node should have at least 1 GB of memory

Storage requirements:

- Master node should have at least 100 Gb of storage available. The output of MaxOcc consists of a few thousands of individual protein coordinate files, each up to a MB in size

Network requirements:

- Connectivity between nodes and master should be of at least 1 GB switch

Software requirements:

- Batch system installed and preconfigured with the number of cores per node equal to number of queue slots per node (TORQUE/Maui are the preferred schedulers)
- All components running Scientific Linux, with Python version 2.7 or higher (within the 2.X range), together with csh/tcsh, gcc, fortran compilers, awk (or gawk), sed.
-

4.4 Climate Model Intercomparison Analysis

The three classes of data analysis mentioned in Section 3.4 (anomalies analysis, trend analysis, climate change signal analysis) are strongly related to climate model inter-comparison and will involve both

accessing large datasets (e.g. multi-Terabyte order) and running complex analytics workflows with tens/hundreds of data analytics operators[9].

In such a context, the testing activity associated to the proposed use case will mainly support two scenarios:

- "single model": the test suite will be related to a data analysis experiment involving the big data analytics framework (e.g. Ophidia) only. This will not include testing any coarse-grain workflow system, as the focus will mainly be on the single-model workflow part.
- "multi-model": the test suite will be related to a data analysis experiment involving a (i) WfMS (e.g. Kepler), (ii) multiple big data analytics framework instances, (iii) a FG Engine instance and (iv) a Scientific Gateway for climate data analysis. This will include testing both the coarse-grain workflow systems and multiple big data analytics frameworks (at least two), as well as WP6 components like the FG Engine and WP5 ones from the INDIGO PaaS (e.g. to dynamically instantiate a big data cluster in the INDIGO infrastructure, or to address/support security) as the focus will be related to a complete ensemble analysis on several models.

The requirements on functionalities and resources for a local test about the "**large scale climate model intercomparison data analysis**" use case are reported below.

Single-model scenario:

- launching an Ophidia cluster either with a single-node hosting the entire stack (supernode) or a multiple-nodes deploy (cluster mode).

  Two possible deployment scenarios are:

  - single-node: small scale deployment; in this case the Ophidia server, the compute and I/O nodes will be on the same VM. A pre-configured VM[10] is under preparation and will be made available for testing and validation purposes by the end of January.
  - multiple-node: large scale deployment; in this case the Ophidia server, the compute and I/O nodes will be hosted on separate VMs.
    - each compute/IO node should ideally have 4 GB of memory per core;
    - the server front-end should be at least 2-cores and 4 GB of memory;
    - a VM hosting the Ophidia CLI is not needed for test and validation purposes. The CLI could be deployed on the Ophidia server machine;
    - "orchestrated" deployment of the Ophidia stack will be needed, based on the INDIGO PaaS. VMIs for each separate component will be available.

  The single-node scenario is mandatory for test and validation purposes. It can be also used for training and dissemination purposes. The second type of deployment is much more related to production-level scenarios, where an entire cluster is automatically deployed and configured.
- batch system (SLURM) installed and properly configured;
- connectivity between nodes and master of at least 1 GB switch;

[9] A data analytics operator represents a core data transformation/analysis task. Some examples are: data sub-setting (slicing and dicing), data aggregation, and data reduction. Analytics operators include both data- and metadata-oriented functionalities.

[10] Dependencies: ncurses 5.7.3 ; GNU readline 6.0 ; graphviz 2.26; GTK+2 2.24.23; libssh2 1.4.2; gSOAP 2.8.15; libcurl 7.19.7 ; VOMS 2.0.12 ; globus-common-devel 15.29, globus-gsi-credential-devel 7.8, globus-gsi-proxy-core-devel 7.7, globus-gssapi-gsi-devel 11.18, globus-gram-client-devel 13.12; libxml2 2.7.6; OpenSSL 1.0.1; Jansson 2.6; MPICH2 1.2.1 ; NetCDF 4.1.1; HDF5 1.8.5; GNU Scientific Library (GSL) 1.13; MySQL C library 5.6.25 ; GNU libmatheval 1.1.10; zlib 1.2.3; MySQL server 5.6 ; SLURM scheduler 2.6.

- storage disk attached to the Ophidia cluster. NFS mount of the home partition on all nodes to enable import/export of the data. 1TB disk could be considered ok for testing and validation purposes. Data challenge will need tens of TBs disk space to run ensemble analysis.
- all nodes running CentOS 6.x (it is planned to port several components on CentOS7.x in the second half of the project (e.g. Ophidia framework)).

The requirements addressed by the single-model scenario are: ENES#1, ENES#2, ENES#3, ENES#5, ENES#6, ENES#7, ENES#9, ENES#10, ENES#11.

Multi-model scenario

In this case it will be also needed to consider:

- launching a Kepler VMI properly configured and including the Ophidia CLI, and two Ophidia cluster instances. A 2-core, 4GB machine, 50GB disk, could be considered for testing and validation purposes reagarding Kepler. The Ophidia instance configuration requirements can be considered the same of the previous Single-model scenario.

It is worth mentioning here that the Multi-model scenario can be considered as an integration test where multiple components of INDIGO are tested together and able to validate the proposed the entire use case.

The requirements addressed by the back-end components (e.g. Kepler, Ophidia, FG Engine) of the multi-model scenario are: ENES#4, ENES#5, ENES#6, ENES#7, ENES#8. On the other hand, the requirements ENES#12, ENES#13, ENES#14, ENES#15 are mainly addressed by front-end parts of the test case (e.g. Scientific Gateway or mobile apps).

Running the test

In the single-model scenario the Ophidia CLI will be invoked in batch mode (-e option of the oph_term), whereas in the multi-model scenario, the tests will run (1) through the Scientific Gateway (from a web browser) in the integration test, but also, (2.1) through Kepler (in batch mode), (2.2) through a FG-Engine REST API Client and (2.3) through the applications in the FG-Engine back-end (e.g. adaptors part) to perform unit tests and validation on a restricted set of components.

4.5 CTA Big Data Processing

The first step to test a CTA prototype archive is to identify a test-bed as a virtual infrastructure capable to fulfil CTA requirements. A virtual machine should be able to handle in parallel a lot of data (mainly in FITS format), archive, process and reduce them using appropriate pipelines (in development phase).[11]

We can identify two virtual components to be developed in order to accomplish the first case study on the identified infrastructure: 1) an archive module and 2) a pipeline module.

To develop the archive I/O module:

11 The amount of data for instrument still to develop can change, even with produced cameras the data-rate can change a lot since it depends on different factors like trigger-rate, compression algorithms, effective pixels involved and so on. A good approximation can consider ~0.8TB/night for a single telescope prototype, 3TB/night for 9telescopes of the miniarray preproduction and increases to ~24TB/night for the whole CTA array of telescopes.

- C/C++ with multi-threads (~50 archiving jobs simultaneous), CFITSIO
- mysql (and mongo) databases
- distributed repository ~100TB "redundant" (located at least in three sites?)[12]

To develop the pipeline module:

- gcc C/C++ compiler
- Python + NumPy + SciPy + Ctools
- fftw3 and pyFFTW for running on multiple CPUs
- CUDA (last version) for GP-GPUs acceleration
- git, pip and Cython for installation

The basic test for archiving performance can be accomplished using a subset of simulated Monte Carlo data for about 16 simultaneous archiving jobs running on different multi-core CPUs (having at least 2GB RAM per core).

The typical file size is about 1GB but a set of smaller data chunks have to be considered during a common observation.

There are different levels of reduction: 1) following the data acquisition rate so each incoming file should be processed close to where storage resources are; 2) stored files should be accessed by external user pipelines and the intermediate reduction could be refined to obtain final science ready products; 3) external users should be able to access both intermediate files and final products.

For external users access to the archive any technology can be used, since there is no requirement except the one concerning Authentication & Authorization and generic portal/gateway usage.

We foresee and recommend that the massive I/O data reduction in production phase performed by the CTA pipelines are run in a dedicated high-throughput infrastructure such as the usual a Grid infrastructure (i.e. similar to the one used by LHC) using a dedicated Virtual Organization (CTA VO) to reserve CTA resources.

The pipeline module can be tested starting from the archiving tail using a linear queue and performing multi-processor / multi-core approach as well as GPU acceleration, since the developed code should be able to run on different accelerators (e.g. NVIDIA TESLA).

All output results must be easily handled by the archive module at the end of the pipeline processing steps. All outputs should be easily browsed by a dedicated user in the archive at all levels of reduction. The user interface can be provided as a simple user interface/user portal of the archive distributed resources.

4.6 LBT Data Distribution and Archiving

Minimum requirements for LBT Archiving System virtual machine:

- Dual core CPU
- 4GB ram
- 50 GB VM disk space occupancy
- open ports for data and meta data distribution services (port 4000 and port 5000)

[12] To test a suitable archive, we can use a subset of data capable to handle at least the pre-production phase for a small period of time (thus a small requirement for the storage of only 100TB, redundant) to test connections, I/O, A&A for a generic external user (INDIGO or CTA user).

- available disk space of about 10 TB. An HDD (SDDs are not required for long term preservation) type of storage could be SAN mounted via NFS or others distributed file systems. Data are images formatted in atomic FITS files. An architectural suggestion on path hierarchy is the ability to write them in a /mnt_point/YYYY/MM/DD/instrument_name/file_version/file.fits. This could be modified with some work from our side (not desirable).
- NFS / SAMBA / SCP / TFTP servers on the VM for the incoming directory
- Operating system requirements 1:
  - **Distribution:** Centos 6.4 64 bit
  - **MySQL:** 5.6.14-1
  - **Java JDK:** openJDK 1.7.0_51
  - **GCC:** Red Hat 4.4.7-4
  - **Python:** 2.6.6
- Operating system requirements 2:
  - **Distribution:** Debian GNU/Linux 7.1 64 bit (wheezy)
  - **MySQL:** 5.5
  - **Java JDK:** openJDK 7u25-2.3.10-1~deb7u1
  - **GCC:** 4.7.2 (Debian 4.7.2-5)
- Software requirements:
  - **Boost library:** 1.55.0
  - **CCfits:** 2.4
  - **Cfitsio** 3.3.10
  - **OmniORB:** 4.1.7
  - **Protocol Buffer:** 2.5.0
  - **SOCI:** 3.2.1
  - **Tango:** 8.1.2
  - **ZeroMQ:** 3.2.3

An already configured VM is available for deployment over KVM, XEN Citrix and it is portable on VMWare virtualization systems.

Currently the data retrieval web interface is hosted on a different virtual machine in order to separate the archiving (LAN) infrastructure by the web access one (WAN) into distinct security zones.

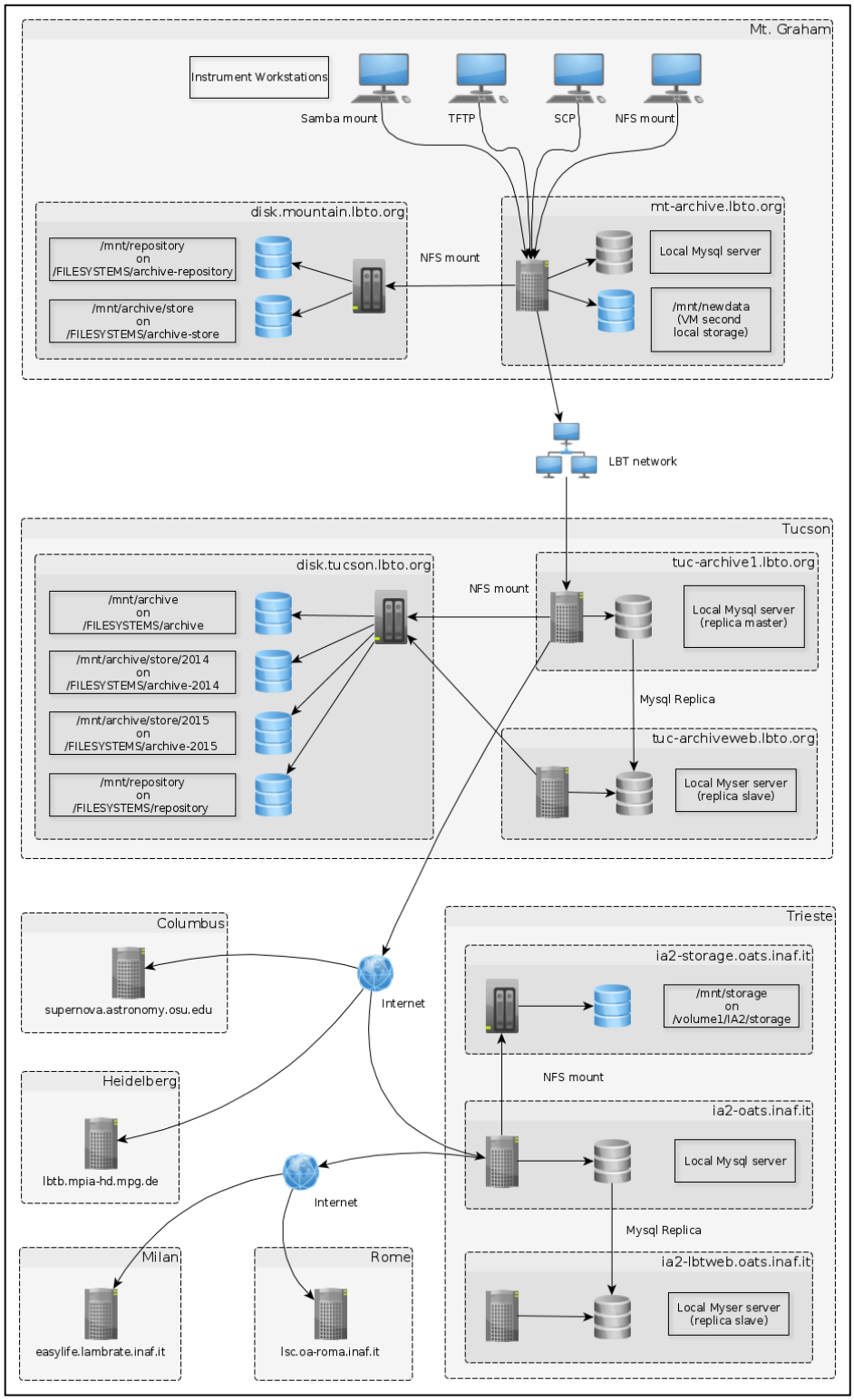

Figure 1: Proposed workflow for INAF/LBT use case

4.7 GALAXY

The requirements for a virtual Galaxy instance (standard state-of-the-art production instance) as intended for the use case are the following:

Computational requirements

- A virtual machine or Docker container with the Galaxy workflow manager;
- Fulfil extra computing requirements with a special VM with higher individual resources or with a large number of them;
- Processing may involve a set of operations with high throughput stages.

The requirements associated are EL#1, EL#7, EL#12.

Testing procedure: the galaxy instance will be tested running a set of predefined tools and workflow (a set of validation tests will be defined).

Storage requirements

- Each Galaxy deployment needs an external volume (at least 500 GB) for reference data and one (at least 1 TB) for users' data.
- The reference data volume could be shared among different Galaxy deployment, if allowed by the administrators.
- The volume filesystem has to manage multiple small files or big ones.
- Each volume should be separated, to increase scalability and privacy.

The associated requirements are: EL#4, EL#5, EL#10, EL#14.

Testing procedure: the external volumes for reference and user data are created and mounted. The reference data will be downloaded in the corresponding volume and tested using galaxy tools (e.g. Lift-Over, https://genome.ucsc.edu/util.html).

Network requirements

- A ssh command line access to the virtual server should be granted to the Galaxy administrator in order to perform Galaxy administrative tasks only available through the command line.
- Data can be transferred from a local facility to the Galaxy instance through the http and ftp protocols.
- For each Galaxy instance, a public IP address should be granted for http access.

The associated requirements are: EL#11, EL#13.

Testing procedure: ssh terminal access to a running Galaxy instance and data transfer through ftp/http protocols.

Software requirements

- Postgresql as database, NGINX as web server, UWSGI as interface between the web server and the web service, PROFTPD as FTP server;
- All components running on CentOS linux, with Python version 2.6 or 2.7, GCC compiler, GNU make and GIT, R >= 3.2;
- Galaxy instance configured according to the virtual hardware configuration (e.g. CPU number, ram, storage, public/private IP address);
- A set of self installation scripts to setup different "flavors" of Galaxy depending on the user needs: e.g. "metagenomics Galaxy", "transcriptomics Galaxy", etc…

The associated requirements are: EL#6, EL#9.

Testing procedure: a Galaxy instance will be deployed through web page in terms of CPU, RAM, storage and desired tools. A set of validation tests will be defined and performed.

Other requirements

- A way to isolate user data from any unauthorized access, including cloud platform administrator(s).
- Each deployment needs its own administrator.

The associated requirements are EL#2, EL#3, EL#8.

Testing procedure: access to the instance and to the data volume (both user and reference data) with the Galaxy administrator credentials should be granted. The same using a different user's credentials is expected to fail.

4.8 Additional Use Cases

4.8.1 POWERFIT and DISVIS

The requirements for a local PowerFit VM and a local DisVis VM are the same. Therefore, these are gathered under one section and they are the following:

Computational requirements:

A system with multiple CPUs and/or GPGPUs

Software requirements:

- Python version 2.7 with NumPy 1.8+, SciPy
- gcc or another C-compiler
- fftw3 and pyFFTW for running on multiple CPUs
- openCL1.1+, pyopencl, clFFT, gpyfft for GPU acceleration
- git, pip and Cython for installation

Performance of PowerFit and DisVis on local sources:

Testing for the performance of PowerFit running on local sources shows that harnessing GPU power instead of using single CPU speeds up the process by a factor of 20 (CPU: AMD Opteron 6344 using FFTW3; GPU: NVIDIA GeForce GTX 680 using clFFT) for two different testing cases, namely GroEL-GroES and RsgA-Ribosome. The following commands are used to run the test:

PowerFit test case 1: GroEL-GroES (EMD-1046 directory in test-cases of PowerFit repository on github)

For single CPU:

```
powerfit GroES_1gru.pdb 1046.map 23 -a 4.1 -l -d results
```

For GPU (-g flag):

```
powerfit GroES_1gru.pdb 1046.map 23 -a 4.1 -l -g -d results
```

PowerFit test case 2: RsgA-Ribosome (EMD-2017 directory in test-cases of PowerFit repository on github)

For single CPU:

```
powerfit 4adv_V.pdb 2017.map 13.3 -a 4.71 -l -d results
```

For GPU (-g flag):

powerfit 4adv_V.pdb 2017.map 13.3 -a 4.71 -l -g -d results

It is also possible to run the same commands with a flag of –p *N* instead of –g, where *N* is the number of processors.

For the DisVis test case PRE5-PUP2 complex (located in test-cases directory of the DisVis repository on github), the performance of DisVis on local resources (AMD Opteron 6344 node) for an increasing number of cores is shown in Figure 1. Running on more than 16 cores in this particular example does not lead to further speed up of the computations. The same figure reports the GPGPU performance (on a GTX680 card). Following commands are used for this test-case:

For single CPU:

disvis O14250.pdb Q9UT97.pdb restraints.dat -a 9.72 -p 1 -vs 2

For multiple CPUs (number of CPUs is determined by –p flag, here set to 16)

disvis O14250.pdb Q9UT97.pdb restraints.dat -a 9.72 -p 16 -vs 2

For GPU (-g flag)

disvis O14250.pdb Q9UT97.pdb restraints.dat -a 9.72 -g -vs 2

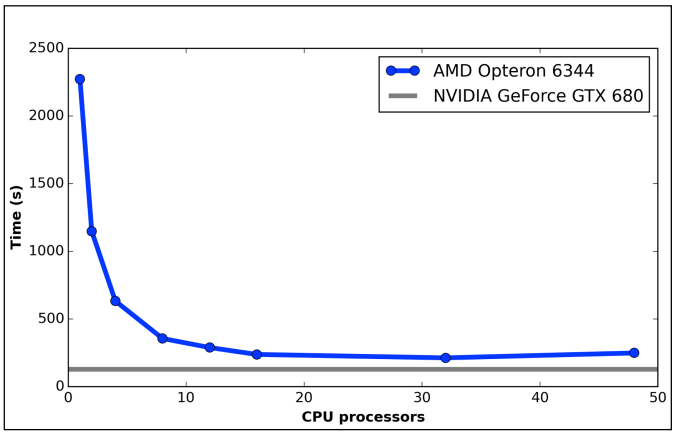

Figure 2: Performance trend for DisVis on single/multiple CPUs (AMD Opteron 6344, 48 cores) and GPGPU (NVIDIA GeForce GTX 680)

Various scenarios are possible for this use case:

1) A preconfigured multi-core VM with Docker is launched and the DisVis/PowerFit containers are run.
2) A preconfigured multi-core VM with all the required software dependencies is launched in which DisVis/PowerFit can be installed (eventually automatically) and run.
3) A preconfigured GPGPU enabled VM with all the required software dependencies is launched in which DisVis/PowerFit can be installed (eventually automatically) and run making use of the GPGPU resources.

For all scenarios, the INDIGO solutions should provide a similar performance / speedup as a function of the number of cores and GPGPU for DisVis and PowerFit cases. The testing procedure can be carried out using the examples stated above.

4.8.2 Algae Bloom Modelling for Cdp Water Reservoir using Delft3D

The requirements to implement this test use case are the following:

Computational requirements:

- Virtual Machine or Docker container. At least 4 CPU cores to run the Delft3D modules. At least 16GB of memory. User portal, for checking results and model outputs: at least 4CPUs, 8GB, 300 GB local disk, access to distributed storage.
  Testing:
  - Run of medium and high resolution pre-defined testing models. Checking output.

Software requirements:

- Delft3D Software (Delft3D dependencies of external packages are detailed in its documentation; it can be installed in Unix or Windows).

Storage requirements:

- Users need a storage system where they can upload files and get outputs from simulations. The modelling suite needs a few input files, not very big and produces a few big files, up to 1TB of data. After simulation, output data is required to check the results, so the file system needs to be accessible by the users and by the user portal as well.
  Testing:
  - Check interaction between computing part and storage system: use of uploaded input files and creation of output.
  - Downloading of output files in an acceptable time.

Other requirements:

- An AAI solution is required.

4.8.3 TRUFA

The requirements to implement TRUFA as test are the following:

Computational requirements:

- Docker Containers to process the pipeline implementing the TRUFA workflow: at least 48 cores available to run under MPI (local or in a cluster), with 2GB RAM /core
  Testing:
  - Deploy of pipeline software in containers.
  - Run pre-defined tests using parallelizable steps of the pipeline. These test are submitted from the web portal using specific input files.

Software requirements:

- For the Web portal: python, python-based web server, web-based file manager. Also a batch system (currently SLURM, but it can be replaced thanks to the TRUFA modular architecture). Currently, the pipeline installed in our supercomputer needs the Altamira management software, that can be switched depending on the deployed solution and the orchestrator selected. TRUFA also needs the following open source packages, some of them parallelized: B2G BLASTP BLAT BOWTIE CEGMA CUFFLINKS CUTADAPT FASTQC HMMER INTERPROSCAN MPIBLAST PRINSEQ RSEM SAMTOOLS TRINITY_RNA_SEQ. MPI is also needed for parallelization.

  Testing:
  - Deploy the different modules of TRUFA within INDIGO environment: web-side, orchestrator, computing layer.
  - Submission through web portal of pre-defined TRUFA workflows, taking in account all the available steps.

Storage requirements:

- Users need their own storage space with at least 20GB available. All the system interacts with it to get input files and store outputs. The file system need to be accessible within the web portal.

  Testing:
  - Upload of big input files (~4GB) through the web-based file system.
  - Download big output files (~4GB) from the web-based file system.
  - Compute uploaded input files within the pipeline.

Other requirements:

- An AAI solution is required.

4.8.4 Medical Imaging Biobanks – Population Imaging

This use case will host temporary subprojects that will be provided with a virtual infrastructure, a subset of the data in the long-term repository, and processing services.

The requirements of the use case related to EuroBioImaging are the following:

Storage requirements:

- User working data should be stored on persistent disk volumes, where project / user specific data is made available. These volumes may persist during the lifetime of an experiment, although they are not intended to be the long-term persistent storage. When a new subproject is started, data from the main repository should be copied in the volume (this will be performed by the application) and such volume attached to a deployment. The virtual infrastructure may be shutdown and restarted, without losing the data in the volume. Volumes will be released at the end of a subproject.

  A platform administrator would need to create such volumes (with associated metadata, as user permission, maximum size and searchable tags), mount them locally on a remote machine, transfer the data (if mounted as POSIX, direct “cp” will work) and unmount them.

  The associated requirements are:

- Persistent (but medium-term) data storage volumes with standard POSIX file access (EB#1).
- ACL in the access to data (EB#2)
- Online access to data (EB#8)
- Management of users and groups (EB#9)
- Long-term availability of results (EB#10)

The testing procedure will include four tests:

- Request as special privileged user the creation of a volume for a specific user, populate it with a set of files structured as a DICOMDIR (it works with POSIX), which will include approximately a few GB and tens of thousands of files and unmount it
- Mount the disk on a remote machine with the credentials of the user for whom the volume was created.
- The same as above but using a different user's credentials, expecting it to fail.
- Request the deployment of a VM on the system with such volume attached.

Deployment of a computing pipeline:

- The main objective of making data accessible is to process them. Processing may involve a set of operations with high throughput stages. A workflow engine may be used. The support of a batch queue back-end may be sufficient.

  The requirements associated to this case are the following:

  - Execution of data-driven and computing-intensive workflows (EB#3)
  - Resources adaptation to workload (EB#6)
  - Provenance and repeatability of experiments (EB#11)

  The testing procedure will include 3 tests.

  - Deploy a batch queue system including the front / end without requiring the user to create the images or to install the software. Define the metadata that will guide the elasticity feature.
  - Submit a predefined workload implying a peak of a few tens of jobs to trigger the start-up of new VMs and their further shutdown. Jobs are expected to run for several minutes up to hours, so a booting delay on the order of few minutes may be acceptable.
  - Deploy a batch queue system with a shared directory where a predefined volume can be mounted. This process should not require user intervention at the administration level.

Customizing processing software:

- The execution pipelines will require pieces of software that are not common or standard. This will require users to provide the means to install, configure and/or mount such software. We envisage three alternatives: (1) A user may be able to specify the dependencies and installation instructions which will be automatically applied. This may be the most flexible and desirable approach from the general system point of view, although it may require more effort on the user's side. (2) A user defines a VMI or Docker image where the software is included and registers it in the system. This will be the basic image to be used in the instantiation of the previous virtual hardware. (3) A user requires direct access to the running image to adjust or tune up some parameters. This is the worst scenario, since changes will not be persistent, but it will be unavoidable, especially during development.

The requirements associated are:

- Availability of customised software (EB#4).
- Deployment of own software (EB#5).
- Terminal access to the resources (EB#7).

The testing procedure will include 4 tests:

- Define a basic specification including external dependencies available at public repositories, such as dcm4chee, which do not require further configuration and deploy a VM with such dependency, including a validation test (e.g. see section 9 in https://dcm4che.atlassian.net/wiki/display/ee2/Installation).
- Define a more complex specification that includes several VMs (XNAT + LONI) and deploy a cluster with a shared disk and two roles.
- Embed a user's defined software on a Docker image, register it, and use it for the deployment of a batch queue whose workers will use such Docker image. Use ITK and run the image registration tests on a mounted test volume.
- Terminal access to a running instance (XNAT) and manually configure users and permissions.

As a summary, the technical requirements of this use case are the following:

- Computational requirements: Between 1 and 25 instances of 4-8GB RAM each for the test. The number of active instances will depend on the execution stage.
- Storage requirements: Around 100GB in a read-only volume, and about three times more for a medium-term persistent volume. Both volumes should be mounted on the computing nodes.
- Network requirements: External connection is required, and limited by end-user connection. No need for internal high-speed network.
- Software requirements: dcm2nii, Caffe, XNAT, CONDOR, octave, matlab (interesting to test the floating license model), dcm4chee, own software.
- Other requirements: Availability of GPGPUs will be relevant. Elastic management of the computing instances is a must. Automatic reconfiguration of the processing nodes.

5 PROGRESS IN THE IMPLEMENTATION OF USE CASES FOR TEST AND VALIDATION

As anticipated in section 4, quite a different level of progress is expected for the different use cases considered. Also the available platforms to perform these initial tests are yet not many, and mainly associated to local resources from the different Research Communities. This section will be updated thanks to the ongoing work in T2.3 that will be reported in the future deliverable D2.8, describing both the final plans and the experience with the test and validation framework.

5.1 Local Version of HADDOCK

A local version of HADDOCK, with all the required associated third-party software (with the requirements specified in Section 4.1 of this report) has been successfully tested on a VM provided by EC3: Elastic Cloud Computing Cluster, deployed on the EGI Federated Cloud using FedCloud proxy key. The corresponding archive is available from the U. Utrecht partner for testing purposes, but because of licensing issues cannot be made freely available nor redistributed. All examples provided with the HADDOCK distribution have been successfully tested.

5.2 Multi-Threading and MPI-based Molecular Dynamics

We are in the progress of setting up Docker containers with the SANDER program from the AmberTools toolkit (available under GPL). This container is currently being tested.

In parallel, we are exploring the preparation of a full VM for the PMEMD software to run on GPGPUs. PMEMD is the GPU-optimised version of SANDER; however, its usage does require a license.

5.3 The Maximum Occurrence Approach for the Characterization of Internal Dynamics in Multi-Domain Proteins

The in-house software implementing the MaxOcc method has been rewritten. We are currently updating the WeNMR portal to use this new version of the software. The implementation of the software using VMs will be based on an architecture.similar to the HADDOCK use case.

5.4 Climate Model Intercomparison Analysis

A local version of the Ophidia big data software stack has been successfully tested on a VM both on OpenNebula and OpenStack private clouds at the CMCC and INFN-Catania in the context of providing a first prototype of the use case (single-model scenario). The VMI is under testing now and will be released by the end of January on the EGI-AppDB. That will enable automated testing/validation activities jointly with WP3.

A preliminary workflow on precipitation trend analysis has been successfully tested using the current prototype of the Scientific Gateway for Climate Data Analysis running at INFN-Catania and an Ophidia instance running at CMCC. In this regard, a demo has been presented at the EGI Community Forum 2015, held in Bari in November 2015, during the “Virtual Research Environments” session.

5.5 CTA Big Data Processing

The principal software components involved in the archive module of the CTA archive use case have been developed in a centralized scenario. This has to be converted to a distributed platform in order to

easily scale out. The metadata keywords will be stored in a dedicated document family database (i.e. Mongo DB).

The pipeline components to process the scientific data is under development at a 50% of progress. For the archive testing process it is only needed a queue implementation, eventually with a second stage of GPGPUs implementation to boost up the performance.

For the complete case study, a customized cloud service orchestration can be envisaged in order to provide a SaaS for the whole archive test-bed with a desired user level approach to stored data.

5.6 LBT Data Distribution and Archiving

Currently the status of the LBTDA is in operation. The software for data archival, distribution and retrieval is already debugged for FITS files. This software is written in C++ / C and uses other interesting technologies for data serialization (Google Protocol Buffer). Some other formats were developed for other projects (Italian Radio Archives). A VM is available for both the mentioned OS with the full set up already in place. For the other purposes (proposal submission, data pipeline orchestration with workflows, data access Virtual Observatory compatible, data visualization and download) some cases are under development and others, for the moment, only desiderata.

5.7 GALAXY

A local virtual machine with all the required accessory software has been deployed on the INDIGO-DataCloud infrastructure in Bari.

All third party software specified in section 4.10 of this report have been integrated and are fully functional. Data can be uploaded and imported in Galaxy through ftp. Moreover, an external volume (500 GB) has been deployed and mounted.

A first set of relevant tools for data analysis have been installed through Tool Shed repository [R8]. Reference data have been uploaded in this volume and used in the testing phase.
A first bunch of tests have been performed and gave the expected results.
The VM tests are still ongoing. The yaml template to deploy the Galaxy instance is still under development. A way to insulate the Galaxy instance is also under study.

5.8 Additional Use Cases

5.8.1 POWERFIT and DISVIS

Docker containers have already been built for those applications, for both the multithreading and GPGPU versions. The latter ones do however have dependencies on the hardware (GPGPU cards).
Dockerfiles for building Docker containers of Powerfit and DisVis, running on multiple CPUs (without GPU flavor) are available at:

https://github.com/haddocking/docker.git

The GPGPU version for DisVis is available at:

https://github.com/indigo-dc/docker-disvis

Testing using examples specified in Section 4.8.1 still continues for both multithreading and GPGPU versions.

5.8.2 Algae Bloom Modelling for Cdp Water Reservoir using Delft3D and TRUFA

As indicated in previous sections, the two additional use cases explored in the LifeWatch community have previous implementations using a mixture of Cloud and local HPC resources. The latest implementations developments have been exploring solutions as OpenShift and also an Open Data Preservation portal to provide an integrated framework for data management, and are currently being deployed in FedCloud resources at the new LifeWatch facilities in Seville. It is not possible to offer yet details on the performance of these tools in this framework.

5.8.3 Medical Imaging Biobanks – Population Imaging

Three main activities have been performed in this use case. First, software requirements for the installation of the image processing part have been identified and are being coded as recipes for the on-the-fly configuration and instantiation of VMs / containers. The INDIGO platform will use a TOSCA specification for the deployment of multi-VM applications, and different approaches on software configuration (basic installation, pre-installation of complex deployments, full installation) and the balance container/VM are being tested. Second, a software architecture has been defined and reported. Finally, as data access is a main requirement, features and capabilities of one-data are being analysed in pre-prototypes to validate the approach.

6 REFERENCES

R 1	S.J. de Vries, M. van Dijk and A.M.J.J. Bonvin, "*The HADDOCK web server for data-driven biomolecular docking*". Nature Protocols, 5, 883-897 (2010).
R 2	Bertini I, Case DA, Ferella L, Giachetti A, Rosato A., "*A Grid-enabled web portal for NMR structure refinement with AMBER"*. Bioinformatics. 2011 Sep 1;27(17):2384-90. doi: 10.1093/bioinformatics/btr415
R 3	Bertini I, Ferella L, Luchinat C, Parigi G, Petoukhov MV, Ravera E, Rosato A, Svergun DI. "*MaxOcc: a web portal for maximum occurrence analysis"*. J Biomol NMR. 2012 Aug;53(4):271-80. doi: 10.1007/s10858-012-9638-1
R 4	G.C.P. van Zundert and A.M.J.J. Bonvin. "*Fast and sensitive rigid-body fitting into Cryo-EM density maps with PowerFit*".AIMS Biophysics, 2, 73-87 (2015).
R 5	G.C.P. van Zundert and A.M.J.J. Bonvin. "*DisVis: Quantifying and visualizing accessible interaction space of distance-restrained biomolecular complexes*". Bioinformatics 31, 3222-3224 (2015)
R6	Coterillo, I. et al., "*Integrating a Multisensor Mobile System in the Grid Infrastructure*", in Remote Instrumentation for eScience and Related Aspects, p59-73, Springer, 2010, doi: 10.1007/978-1-4614-0508-5
R7	E.Kornobis et al, "*TRUFA: A User-Friendly Web Server for de novo RNA-seq Analysis Using Cluster Computing*", Evol Bioinform. 2015; 11: 97–104.doi: 10.4137/EBO.S23873
R8	Galaxy Wiki, https://wiki.galaxyproject.org/ToolShed