Co-financed by the European Union
Connecting Europe Facility

Operational analysis and specifications.

Deliverable D1.1: Public

Keywords: High Performance Computing, Server, Data Centre, SSH, VMWare, Dask, Python, Datasets, Linux, Centos, SLURM, Help Desk, Back Up, Parallel development, Pilots.

CROSS Harmonization & HPC modelization of FOREST Datasets

Table of Contents

List of Figures

Contractual Date of Delivery to the EC:	May 2019
Actual Date of Delivery to the EC:	v1.0 31/05/2019 v2.0 31/08/2020
Editor(s):	Vicente Matellán (SCAYLE) Jesús Lorenzana (SCAYLE) Jennifer Abad (SCAYLE) Maria Jular (SCAYLE) Álvaro Fanego (SCAYLE)
Contributor(s):	Daniel Molina (TRAGSA) Spiros Michalakopoulos (CAMBrIc pilot) Jesús Candelas Segovia (TRAGSA) Asunción Roldán (TRAGSA)

DOCUMENT HISTORY

Version	Version date	Responsible	Description
0.1	29/04/2019	SCAYLE	Draft for SCAYLE comments
0.2	06/05/2019	SCAYLE	Draft for Cross-Forest consortium comments
0.3	28/05/2019	SCAYLE	Draft for Cross-Forest consortium comments
0.4	29/05/2019	SCAYLE	Draft for Cross-Forest consortium comments
1.0	30/05/2019	SCAYLE	Version to INEA
1.1	01/05/2020	SCAYLE	Draft for SCAYLE comments
1.2	09/05/2020	SCAYLE	Draft for Cross-Forest comments
1.3	17/05/2020	SCAYLE	Draft for Cross-Forest comments
1.4	25/05/2020	SCAYLE	Draft for Cross-Forest comments
1.5	25/05/2020	SCAYLE	Draft for Cross-Forest comments
1.6	15/06/2020	SCAYLE	Draft for Cross-Forest comments
2.0	30/06/2020	SCAYLE	Version ready to be sent to INEA

1 INTRODUCTION

This deliverable provides detailed information of the results of Activity 1 (HPC Facilities).

This deliverable is a voluntary update of D1.1 presented by the Cross-Forest consortium in 2019. The reason for this elaboration is because at that time the works to be developed were not yet fully defined and the part of the development external to the HPC infrastructure was still in the study phase. In the first version of this deliverable it was already stated that there would be an update of it in the future.

Large parts of this document contain texts from other deliverables already delivered by the consortium, namely D1.2, D3.1 and D4.1

In the development of this document, actions of a technical nature are included regarding the evolution of the work carried out by the Cross-Forest project consortium and its progress from a chronological point of view.

It starts with the description of SCAYLE's own and common tools for the management of users/clients and of the works that are sent to Caléndula Supercomputer. These tools are used by the pilots of the project and are also used by the members of the partnership to send their simulations and feed new data to the platform set up.

These are the Job Manager and the application manager that allow a strict control of the traceability of the tasks and jobs requested by the customers, both internal and external.

The description of the development of the jobs for the pilots follows.

This document begins with the tasks carried out with the CAMBrIc pilot, details the software installed and includes details of its workflow.

Then, the work related to FRAME is included, with the same structure as for the previous pilot.

Finally, and as it is a common facility, the technical characteristics of the hardware made available for both pilots are described in detail.

Afterwards, the infrastructure created for the Cross-Forest platform is explained, which houses the data sets transformed and produced by the project. Everything related to the software and hardware that has given rise to the infrastructure made available to all the Linked Open Data is described. It also details the layout of the virtual machines intended for their operation and the connections to be able to access them. It also incorporates a wide detail of the physical characteristics of the equipment that provides the mentioned virtual infrastructure.

Finally, everything related to the maintenance of the resources destined to the correct operation of the entire technological ecosystem of the project, to the storage destined for this purpose and to the backup system that protects it, is included and detailed.

The Activity 1 throughout this period encompasses the following tasks, included in the GA:

1. Task 1.1. Assessment of the working environment and workflow definition.
2. Task 1.2. Installation of the simulation software, adaptation to the working environment and identification of possible improvements.

3. Task 1.3. Setting-up the production environment.
4. Task 1.4. Design of the storage system for the simulations performed.
5. Task 1.5. Provision of access to the results obtained.

For sake of clarity, the deliverable has been structured following these tasks, except for tasks 1.2 and 1.3 which, because of their overlap, and because many of the works carried out correspond to both and it is very difficult to distinguish between them, are listed in this document together in the section 3, thus detailing all the activity performed for each task according to the Grant Agreement.

As provided for in the Grant Agreement, the Activity 1 defines the models for the use cases and the related algorithms. Simulations for fire propagation and their effects together with fire suppression technics (use case/demonstrator FRAME), and for forecasting wood quality in mixed forests on big surfaces (use case/demonstrator CAMBrIc) are also performed.

As the project is advancing, areas for possible improvement of the algorithms have been identified through the implementation of parallelism techniques. The performance of the models and the simulations has been evaluated to identify the best adaptation of the algorithms to the calculation environment. In addition, a set of procedures have been implemented to I) obtain the input data automatically, ii) perform the required calculations and iii) store the data produced by the simulations. Finally, HPC console presentation layer of the platform has been implemented to allow users to access the results of the calculations performed.

The outputs of this Activity are i) the assessment and adequacy of the working context, ii) the installation and setup of the simulation software and iii) the production environment. These results are explained in three deliverables: D1.1 Operational Analysis (on all the analysis and preparation activities); D1.2 HPC Systems: Inventory, Evaluation and Performance – Interim Report (this document); and D1.3 HPC Systems: Inventory, Evaluation and Performance - Final Report (this document will highlight the results of Activity 1 at the end of the Action).

2 Task 1.1. Assessment of the working environment and workflow definition

In this task, the process of knowing all the stages that formed the workflow to be implemented were carried out. This work was essential to acquire a thorough knowledge of all the data, programs and their interactions that determined the most appropriate supercomputer subsystem for project development.

The working environment of the platform is assessed, and the data processing workflow was defined. The workflow was based on the programs, the data that are used for the platform and the use cases and includes the interactions with the supercomputer subsystem. At the beginning, an intense task of prospecting and initial state of the members of the consortium was carried out.

Numerous meetings and interviews were held so that the technicians of SCAYLE could understand what the previous software of the partners was like and which parts were intended to be used.

There were also several parallel meetings with the developers of the respective pilots and with those responsible for putting the endpoints into operation.

Figure 1 – Training session in SCAYLE Headquarters

Several visits were made to SCAYLE by partners not familiar with the HPC environment to understand how Caléndula works.

Figure 2 – iuFOR visiting SCAYLE Headquarters

In the first technical meeting of the project held in León, a basic practical training session in supercomputing environment was developed so that the members of the partnership knew first-hand how to develop more efficient algorithms from the point of view of parallelism.

After these initial meetings, the work has been developed in an individualized way and adapted to the 4 scenarios identified by the members of the consortium. The following sections detail of how this work has been developed and the results achieved so far.

Figure 3 – Timeline HPC workflow

2.1 Initial analysis of pilots

It was necessary to perform a previous adaptation of both pilots to fit the characteristics of a supercomputing environment. Supercomputers have a large number of processors that can be used simultaneously by the programs to calculate complex tasks in much shorter times. To achieve that a program uses several processors in its execution it needs to be designed to use some type of simultaneous or parallelism execution. There are several technologies to fulfil this goal: MPI, OpenMP, MPI+OpenMP and simultaneous execution of multiple copies of the program through the use of Job Arrays managed by the HPC infrastructure job manager. From the beginning, the migration to open technology environments and standards has been considered.

Different technical meetings have taken place among the different technical teams involved to analyse the pilot needs. The main features identified so far are summarized below. As the work advances, new needs could be identified. In that case, this deliverable would be updated, and a new version would be generated.

The deliverable summarises the technical meetings held to obtain the analysis. Meeting agendas are included as annexes.

2.1.1 CAMBrIc

In the case of the pilot use of CAMBrIc, the software was initially designed for use in personal computers and workstations with Microsoft Windows environments, so it was essential to make a prior work of adaptation to the HPC environment. The original development was written in the C# language and was supported by .NET libraries, both proprietary Microsoft technologies. Although there are technologies, such as Mono [www.mono-project.com], that facilitate the use of .NET-based programs in Linux operating systems, their support is not complete and hidden incompatibilities may appear until the last phases of the adaptation project. Another requirement that was raised is the possibility that the same new development can be used in personal computers and supercomputers, selecting the appropriate option in each case.

Bearing in mind the above restrictions, the study phase of the code to be ported to the Python programming language was started, fully supported by the supercomputer operating system and using the Dask framework.

Dask is presented as the most suitable option for the needs of the migration as it allows to face the development of the parts of the code that can be parallelized, at the same time that it allows to generate the version to be executed in personal computers and workstations. As an important added advantage, Dask has libraries that allow integration with the task manager installed in the HPC infrastructure, SLURM [slurm.schedmd.com].

2.1.2 FRAME

For the HPC integration of the Frame software, two parallel approaches have been considered. On one hand, the adaptation of the calculation routines was considered, excepting the graphic visualization routines that do not make sense in a supercomputing environment, to be executed inside SLURM job arrays. This will allow the execution of a large number of simulations of terrain cells, which are sequential tasks, simultaneously. A job array allows to send a high number of executions of a program, in which the input parameters that describe the characteristics of the individual cell vary automatically.

In parallel, work is being done to study the migration of the code written in C# to C/C++, which will increase cross-platform compatibility and facilitate the development of parallel code using MPI libraries.

Later on, Mono was installed because it seemed to be the most appropriate at that time, although finally we chose to carry out the work in the .NET Core development environment. The reasons for this are explained in detail in section 3.3.2 of this document.

3 Task 1.2. Installation of the simulation software, adaptation to the working environment and identification of possible improvements & Task 1.3. Setting-up the production environment

As explained in the introduction and due to the fact that T1.2 and T1.3 have been performed simultaneously, we explain both together in this deliverable.

Under these tasks, the software executing the models has been installed and adapted to the calculation system. Installation tests of the different simulation software has been performed. Finally, various benchmarks and stability tests have been performed in order to achieve an optimal integration of the entire software stack, including the software supporting the simulations.

The interaction procedures between the job management module of the platform and the HPC resources have been designed to tailor different user requests in order to select the adequate level of HPC resources on the basis of the type of work to be performed.

3.1 SLURM Job Scheduler

All HPC jobs running in the system must be executed in the calculation nodes by sending a script to the SCAYLE job manager.

The job manager or queue manager is a system that sends the jobs to the calculation nodes to which each user has access. The manager used by SCAYLE is SLURM (Simple Linux Utility for Resources Management).

As a cluster workload manager, SLURM has three key functions. First, it assigns users exclusive and/or non-exclusive access to resources (computer nodes) for a certain time so that they can do their work. Second, it provides a framework for starting, running, and monitoring work (usually parallel work) on the assigned set of nodes. Finally, it arbitrates the containment of resources by managing a queue of pending jobs.

In SLURM there are two types of roles (corresponding to the different types of users in the system): "users" and "administrators". Both interact with SLURM through a set of simple commands, but their purposes are totally opposite:

- Users: They send jobs to be executed and wait for them to finish as fast as possible and in a correct way, without caring if the job has been done in one, two or three computation nodes.
- Administrators: Must find a way to execute parallel jobs in parallel nodes, and most importantly, must do as if the job will be executed in only one node. The tasks of an administrator are:
    - Assign resources from a computation node: Processors, memory, disk space...etc.

- Launch and manage jobs.

3.2 SCAYLE Ticketing (Help Desk) System

SCAYLE has implemented a ticket manager (Help Desk) to track incidents and requests from its users. The manager used is GLPI (Gestionnaire libre de parc informatique), a free software web tool that offers a comprehensive management of the computer inventory of a company and an incident management system (ticketing/helpdesk). The tool is developed for Apache-PHP-MySQL environments, so it can be installed on both Windows and Linux servers. Its easy installation and operation allows to manage all the support and maintenance of a company in a quick and easy way, so the deployment and implementation are quite reduced.

This double role of inventory manager (equipment, servers, peripherals, software licenses, network topology, reservation of shared resources, etc.) and helpdesk for monitoring interventions, allows administrators and support staff to link the interventions made to users and equipment, thus generating a complete traceability of the actions taken, groups that have worked on the resolution of the requests made by each user, all the steps taken from opening to closing, explanatory comments on each step taken, i.e. a history of each of the requests that have been requested.

It is an intuitive and simple interface for administrators, technicians and end users. All the requests have the possibility of notification via e-mail to users and to the support staff at the beginning, progress or closure of a request.

The Help Desk system increases productivity and increases the satisfaction of internal and external users, since the processes are developed in an orderly manner and much more agile.

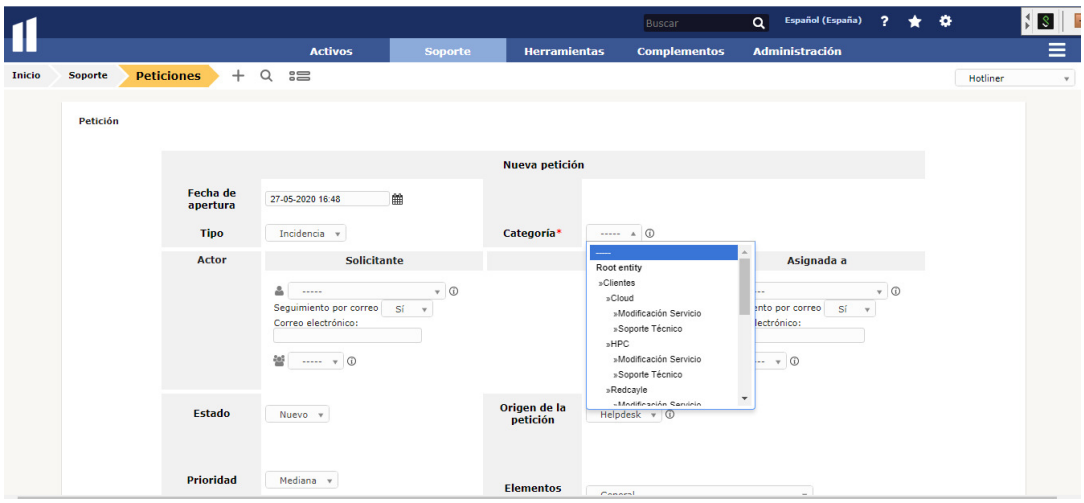

Figure 4 – SCAYLE Helpdesk screenshot

3.3 Cross-Forest Pilots

3.3.1 CAMBrIc

The necessary folder structure has been created within the calculation infrastructure and three calculation users have been created. The complete description of the whole process is described in Deliverable 3.2.

The SIMANFOR simulator is used to carry out two of the main tasks of the CAMBrIc pilot (calculating stocks and simulating their evolution). SIMANFOR is a web application that allows the simulation of sustainable forest management alternatives. It integrates different modules to manage forest inventories, simulate and project different stand conditions (through algorithms and formulas for prediction and projection), query systems, simulation outputs and security system. All the information about the CAMBrIc pilot and his characteristics are available in the Deliverable D3.2.

The use of SCAYLE's parallel computing infrastructure to run the SIMANFOR simulator will allow new calculations to be carried out more quickly and new studies to be carried out through the parallel execution of various simulations.

In order to migrate the current version of SIMANFOR to an operative version on a supercomputing infrastructure, Sngular (the private company carrying out this adaptation) decided to use Dask [https://dask.org]. The SCAYLE applications department has installed all the software required for the migration, to be used once the adaptation reaches the required stability.

3.3.1.1 Software

3.3.1.1.1 Dask

Dask is a flexible library for parallel computing from Python scripts.

It consists of two parts:

- Dynamic task planning optimized for computation tasks and interactive workloads.
- Collections of "Big Data" oriented functions like parallel arrays, dataframes and lists that extend common interfaces like NumPy, Pandas or Python iterators to larger than memory or distributed environments.

Dask uses Python as programming language. This is especially suitable as it integrates parallel calculation libraries to run the simulations on multiple servers simultaneously. In addition, and as an added value, it has tools that allow simulations to be executed in an integrated way with the SCAYLE job manager installed (section 3.1 present document). The job manager is the central piece of a supercomputer since it is in charge of receiving the job execution orders from the users and executing them according to different criteria (job priority, user priority, resource availability, ...).

Python Installation and Dask in annex F.

3.3.1.1.2 SIMANFOR Parallel Implementation

3.3.1.1.2.1 Overview

The original plans to use Apache Dask (ref: https://docs.dask.org/en/latest/) were dismissed because the simulator proved to be embarrassingly parallel (this is explained in the next section). However, some parts of the code which use Apache Dask should be left in the code for:

a) demonstrative purposes

b) to serve as a guide for implementing Dask in the simulator if the need arises: if it is required to process plots with thousands of trees, in which case the simulator will most likely cease to be embarrassingly parallel.

Subsequently we discuss Dask's use and potential use in SIMANFOR.

3.3.1.1.2.2 The simulator is embarrassingly parallel

An algorithm or section of code, or, as is the case in SIMANFOR, a whole program is considered embarrassingly parallel (ref: https://en.wikipedia.org/wiki/Embarrassingly_parallel) when the code can be executed on various independent cores which don't communicate with each other, without any loss of information. Typical examples of embarrassingly parallel problems are rendering of computer graphics (where each pixel can be rendered independently of all others), discrete Fourier transforms, or multiple independent text searches.

In SIMANFOR'S case, the input is a forest inventory, in the form of an Excel file with two sheets:

- the first one is a list of plots and their attributes, with one plot per row, and
- the second sheet is a list of trees and their attributes, with one tree per row, linked to a plot via a plot ID, which corresponds to a row in the first sheet.

The output is one Excel file per plot, which contains the results of executing the simulator on various scenarios. The key here is that there is exactly one output file per plot, with no interaction between plots. Thus, the program could conceivably execute on a separate core for each plot and would produce the desired output. In practice, as is discussed below, it is more efficient to run the sequential program on more than one plot per core. Somewhere between 16 and 32 seems to be very efficient, if not optimal.

Thus, neither Apache Dask nor any other parallel framework or library is needed. Instead, splitting the data input into smaller chunks (if and when needed), and submitting separate jobs for these chunks on the supercomputer, appears to be the ideal solution. This is the approach taken in this case. Next we describe how to install and run SIMANFOR on Caléndula and discuss performance issues and test runs.

3.3.1.1.2.3 The simulator on Caléndula

The simulator code is on github: https://github.com/simanfor-dask/simulator

All that is needed to install on Caléndula is to clone or download the code, make sure Python 3 is loaded, and that all the dependencies in **`.../simulator/requirements.txt`** are also installed.

The script **`simanfor.sh`** submits four jobs, on four different input files (inventories) that were created by manually splitting an example provided by UVA with 100 plots with 100 trees each. It simply calls the four scripts that individually submit the jobs:

```
CALENDULA[ sngular_aia_1_3@frontend2 scripts]$ cat simanfor.sh

#!/bin/bash
sbatch basic_engine_n4_1-25.sh
sbatch basic_engine_n4_26-50.sh
sbatch basic_engine_n4_51-75.sh
```



```
sbatch basic_engine_n4_76-100.sh

CALENDULA[ sngular_aia_1_3@frontend2 scripts]$
```

The Python code requires a scenario configuration file specifying the input as an inventory file described above, together with the output directory, the models being used, the forestry scenarios such as cuts and tree growth, and other configuration options.

Each script defines the SLURM directives, loads Python 3.7 and executes the python code, passing two arguments: the scenario configuration file and a logging configuration file path, e.g.:

```
CALENDULA[ sngular_aia_1_3@frontend2 scripts]$ cat basic_engine_n4_1-25.sh

#!/bin/bash
# numero de cores que serán reservados
#SBATCH -n 4
# particion en donde se ejecutara el trabajo
#SBATCH -p haswell
# limites que se aplicaran al trabajo
#SBATCH -q normal
# nombre
#SBATCH -J python_sngular
# tiempo maximo de ejecucion (p.e. 2 dias). Maximo permitido: 5 dias
#SBATCH --time=01:00:00
# archivos de salida y de error
#SBATCH -o ./salida/basic_n4-%j.o
#SBATCH -e ./salida/basic_n4-%j.e
# directorio de trabajo por defecto
#SBATCH -D .
# notificaciones por email relacionadas con la ejecucion del trabajo
#SBATCH --mail-user=spiros.michalakopoulos@sngular.com
##SBATCH --mail-type=ALL

ROOT=/home/sngular_aia_1/sngular_aia_1_3/dev/simulator

# carga de las variables necesarias para usar Python 3.7.7

module load python_3.7.7

python $ROOT/src/main.py -s $ROOT/files/scenario_claras_1-25.json -logging_config_file
$ROOT/config_files/logging.conf

CALENDULA[ sngular_aia_1_3@frontend2 scripts]$
```

3.3.1.1.2.4 Performance

Tests have been carried out with inventories of various sizes, and in various scenarios. Here we show the performance on the above-mentioned inventory of 100 plots by 100 trees, split into 4 smaller input files, each with 25 plots by 100 trees.

First the jobs are submitted via the **`simanfor.sh`** script:

```
CALENDULA[ sngular_aia_1_3@frontend2 scripts]$ ./simanfor.sh

Submitted batch job 350695
Submitted batch job 350696
Submitted batch job 350697
Submitted batch job 350698

CALENDULA[ sngular_aia_1_3@frontend2 scripts]$
```

While they are running, we check the queue:

```
CALENDULA[ sngular_aia_1_3@frontend2 scripts]$ squeue

             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```



```
350695  haswell python_s sngular_  R      1:13     1 cn3034
350696  haswell python_s sngular_  R      1:13     1 cn3034
350697  haswell python_s sngular_  R      1:13     1 cn3034
350698  haswell python_s sngular_  R      1:13     1 cn3034
```

```
CALENDULA[ sngular_aia_1_3@frontend2 scripts]$
```

Script output after execution:

```
CALENDULA[ sngular_aia_1_3@frontend2 salida]$ ls -lrt

total 24

drwxr-xr-x 2 sngular_aia_1_3 sngular_aia_1 4096 May 14 18:32 errors_old
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1    0 May 19 13:37 basic_n4-350698.e
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1    0 May 19 13:37 basic_n4-350697.e
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1    0 May 19 13:37 basic_n4-350696.e
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1    0 May 19 13:37 basic_n4-350695.e
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1 1727 May 19 13:39 basic_n4-350697.o
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1 1726 May 19 13:39 basic_n4-350698.o
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1 1727 May 19 13:39 basic_n4-350695.o
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1 1727 May 19 13:39 basic_n4-350696.o
drwxr-xr-x 2 sngular_aia_1_3 sngular_aia_1 4096 May 19 13:41 old

CALENDULA[ sngular_aia_1_3@frontend2 salida]$
```

Timings of each script:

```
CALENDULA[ sngular_aia_1_3@frontend2 salida]$ tail -2 basic_n4-350695.o

Models executions finished after 43.656333446502686 seconds.
Program finished after 67.93810319900513 seconds.

CALENDULA[ sngular_aia_1_3@frontend2 salida]$ tail -2 basic_n4-350696.o

Models executions finished after 47.366323709487915 seconds.
Program finished after 73.33272171020508 seconds.

CALENDULA[ sngular_aia_1_3@frontend2 salida]$ tail -2 basic_n4-350697.o

Models executions finished after 39.95999836921692 seconds.
Program finished after 60.887235164642334 seconds.

CALENDULA[ sngular_aia_1_3@frontend2 salida]$ tail -2 basic_n4-350698.o

Models executions finished after 42.91468381881714 seconds.
Program finished after 66.82035899162292 seconds.

CALENDULA[ sngular_aia_1_3@frontend2 salida]$
```

Note that the time between the end of the models executions and the program finishing corresponds to the time required to generate and print the output Excel files.

Those times don't include scheduler and queue tasks. Here are the full times of each job:

```
CALENDULA[ sngular_aia_1_3@frontend2 salida]$ sacct -j 350695,350696,350697,350698 --format='JobID,JobName,Elapsed,State'

       JobID    JobName    Elapsed     State

------------ ---------- ---------- ----------

350695       python_sn+   00:02:03  COMPLETED
350695.batch      batch   00:02:03  COMPLETED
350695.exte+     extern   00:02:04  COMPLETED
350696       python_sn+   00:02:08  COMPLETED
350696.batch      batch   00:02:08  COMPLETED
350696.exte+     extern   00:02:08  COMPLETED
350697       python_sn+   00:01:56  COMPLETED
350697.batch      batch   00:01:56  COMPLETED
350697.exte+     extern   00:01:56  COMPLETED
350698       python_sn+   00:02:02  COMPLETED
```



```
350698.batch     batch   00:02:02  COMPLETED
350698.exte+    extern   00:02:02  COMPLETED

CALENDULA[ sngular_aia_1_3@frontend2 salida]$
```

Finally, here is a part of the output directory after execution:

```
CALENDULA[ sngular_aia_1_3@frontend2 salida]$ ll /scratch/sngular_aia_1/sngular_aia_1_3/output_claras/

total 28572

-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1 328318 May 19 13:38 Output_Plot_1.xlsx
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1 329631 May 19 13:38 Output_Plot_10.xlsx
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1  98246 May 19 13:39 Output_Plot_100.xlsx
…
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1 374410 May 19 13:39 Output_Plot_97.xlsx
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1 377562 May 19 13:39 Output_Plot_98.xlsx
-rw-r--r-- 1 sngular_aia_1_3 sngular_aia_1  98145 May 19 13:39 Output_Plot_99.xlsx

CALENDULA[ sngular_aia_1_3@frontend2 salida]$
```

Note that the small output file size for some plots is not an error in the execution but due to the trees in the plots not fulfilling certain criteria which permits them to be included in the simulation. This results in empty sheets in the output file, sheets where the tree details are normally printed in the cases where the trees in the plot are taken into account for the simulation.

Thus we see that the above 100 plot by 100 trees simulation can be executed in just over 2 minutes on Caléndula, without any sophisticated parallelization beyond splitting the input file into smaller input files. It should also be noted that in our tests on the full 100 by 100 input file, the sequential code produces the correct results in just over 4 minutes.

3.3.1.1.2.5 Website to submit the jobs

In the above examples, the input file and scripts used were generated manually. It is suggested that if a web site is developed to allow users to perform simulations on their data, the scripts written will:

- check the inventory input file and if it contains above a certain numbers of plots, split the file accordingly. This can very easily be done using the Python library Pandas.
- If the inventory has been split into multiple input files, then the same number of jobs should be submitted. This is achieved by generating multiple scenario configuration files and submitting jobs using the same code, but with different scenario configuration files.

3.3.1.1.2.6 Apache Dask

As we mentioned in the beginning of this document section, Apache Dask was planned to be used, but was considered surplus to requirements. Nevertheless, if it is required to simulate tree growth on plots with a much larger number of trees in the future, Dask could be a good option for speeding up the execution of the simulators. For that reason and because some interesting work has already been done in the code using Dask, we present this work here.

Dask has various ways of executing parallel code (ref: https://github.com/dask/dask-tutorial), including Numpy-like Dask Arrays and Pandas-like Dask DataFrames. In this work we used

Dask Delayed (ref: https://docs.dask.org/en/latest/delayed.html).

To execute the simulator with the Dask parallel version, the "**-e 2**" flag must be used. This calls the **dask_engine.py** code instead of the sequential **basic_engine.py** code. In **dask_engine.py** you need to import delayed:

from dask import delayed

A side by side comparison between **basic_engine.py** and **dask_engine.py** illustrates the differences:

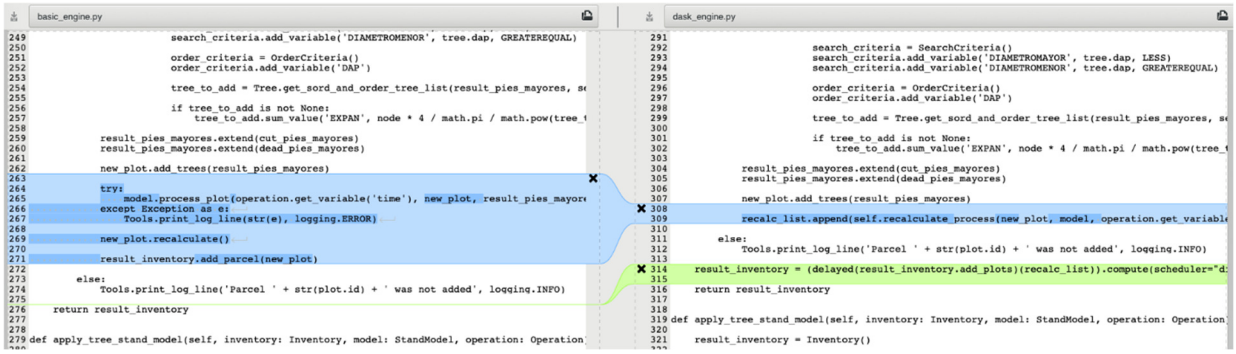

Figure 5 – Side by side comparison screenshot

This code is executed for each plot in the inventory. In **basic_engine.py**, on the left, the **model.process_plot()** function is called (passing in **new_plot** as a parameter), which performs calculations on some of **new_plot**'s variables, followed by **new_plot.recalculate()**, which performs further calculations on **new_plot**'s variables. Finally, the modified **new_plot** is added to the list of plots in **result_inventory**.

In the case of **dask_engine.py**, on the right, the function **recalculate_process()** is added to a list called **recalc_list**. This is what **recalculate_process()** looks like:

```
196
197     def recalculate_process(self, plot: Parcel, model: TreeModel, time, trees: list, result_inventory: Inventory):
198
199         plot = delayed(model.process_plot)(time, plot, trees)
200         plot = delayed(plot.recalculate)()
201
202         return plot
203
```

Figure 6 – Recalculate process Dask code

This code take the functions **model.process_plot()** and **recalculate()** and "delays" them. This tells the Dask kernel that you don't want to execute these functions right now, but just associate them with the specific plot. So, the code loops through all the plots and places their delayed functions **model.process_plot()** and **recalculate()** into the list called **recalc_list**.

Then, after all the plots are processed, **basic_engine.py** returns the **result_inventory** with all the plots and their modified values, whereas **dask_engine.py** does this:

```
313
314         result_inventory = (delayed(result_inventory.add_plots)(recalc_list)).compute(scheduler="distributed")
315
316         return result_inventory
317
```

Figure 7 – Result inventay Dask code

This code, tells the dask kernel to now go and **compute()** what is on the **recalc_list**, in parallel (using dask's "**distributed**" scheduler – there are other types of schedulers such as "**single-threaded**" which is very useful for debugging, see).

Dask comes with a useful diagnostics and visualization tool for the "**distributed**" scheduler, that

can be seen at **localhost:8787**

In the following screenshot, the code was run on an example inventory with 5 plots, on a machine with 4 cores:

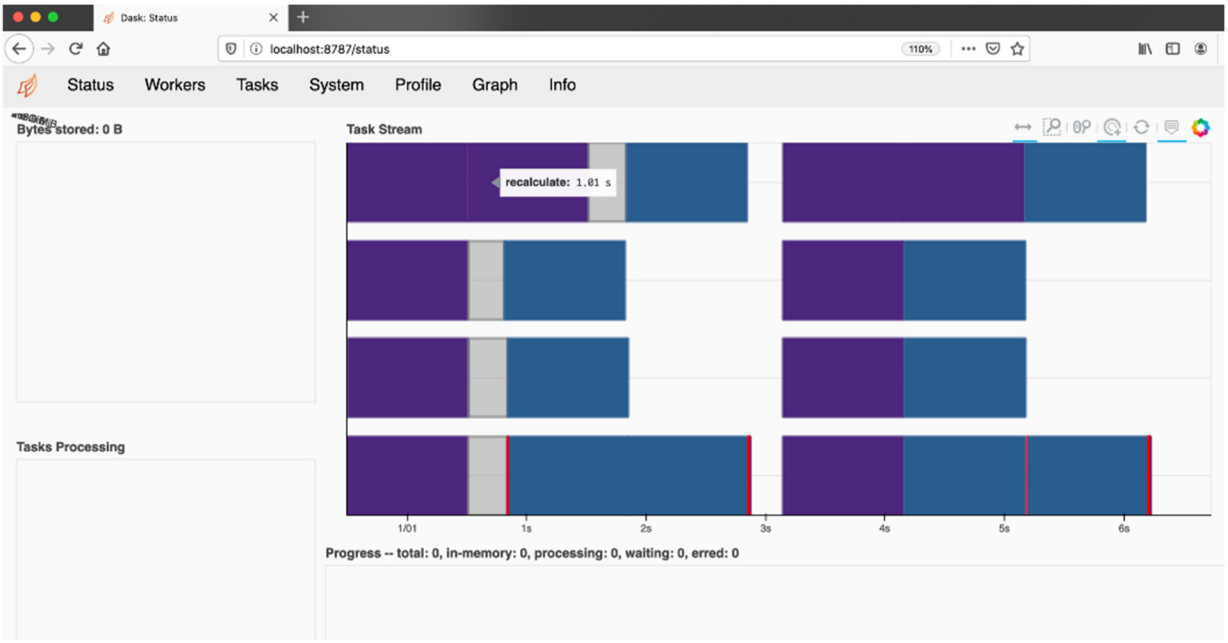

Figure 8 – Example inventory with 5 plots, on a machine with 4 cores screenshot

Note however that **sleep(1)** statements had been placed in the **recalculate()** and **process_plot()** functions, to be able to view the parallel execution better.

Here is another screenshot, this time without the sleep statements, on an 8 core machine, executing the 100 plot by 100 trees inventory:

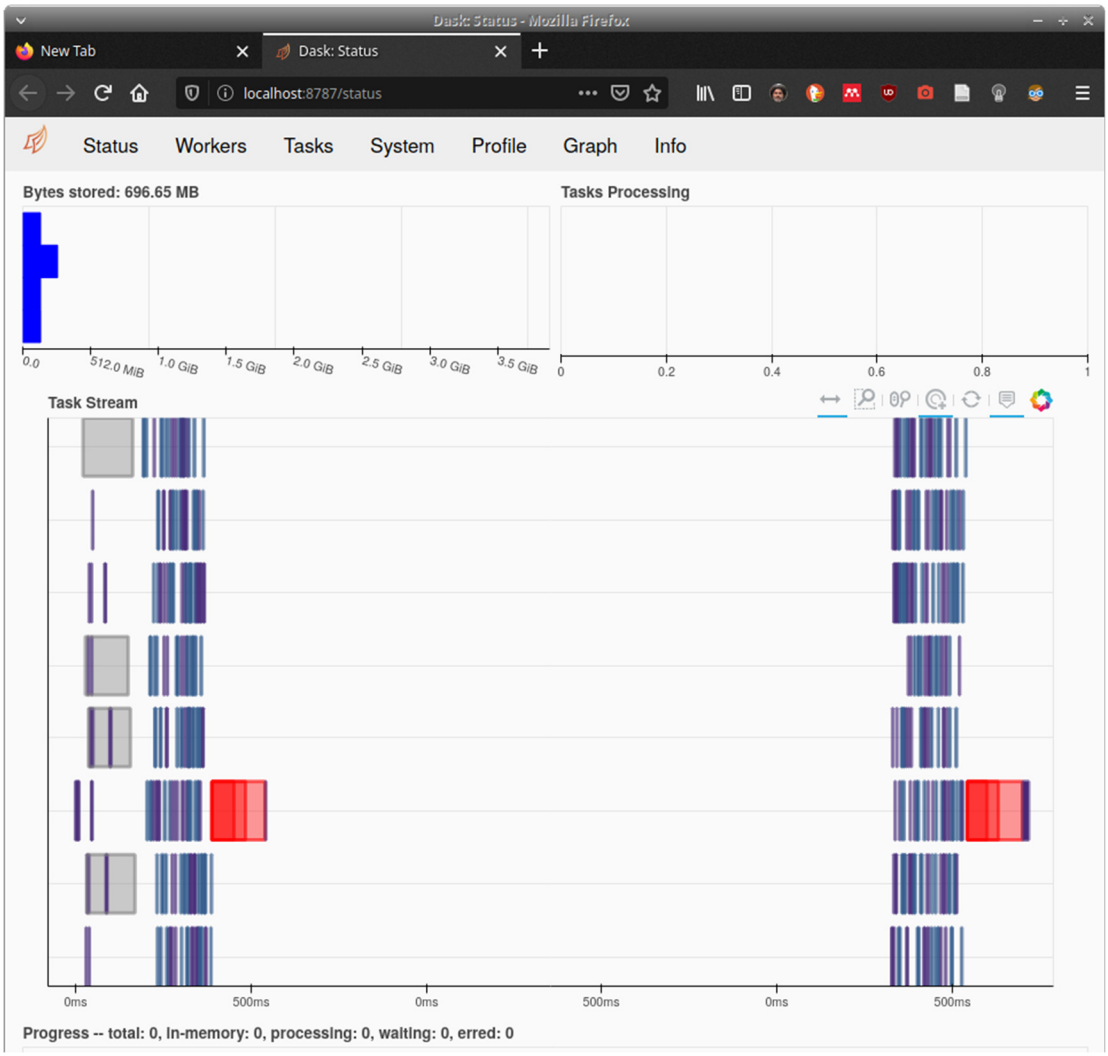

Figure 9 – Example with 100 plots by 100 trees, on a machine with 8 cores screenshot

Parallelizing those two functions however didn't show any increase in performance. On the contrary, in most cases, the execution times were slower, because of the overhead of Dask. It showed though, that the vast majority of time consumed was in generating the output file. So, the output file generation code was also parallelized with **dask.delayed**. On small inventories, this showed a very slight (seconds or microseconds) improvement on runtimes, but on larger inventories.

3.3.2 FRAME

3.3.2.1 Software

The source code, initially written in C#, has been ported to C/C++ to run in Linux-based operating environments. The complete description of the process is described in Deliverable 3.2. Below we include complementary information about the works performed for the adaptation to the HPC environment.

Initially, an attempt was made to carry out this task by installing MONO (an open source implementation of Microsoft's .NET Framework based on the ECMA standards for C#), but this proved to be unsatisfactory and was subsequently discarded and uninstalled.

Furthermore, as reported by TRAGSATEC, the effort made to rewrite the code to .NET Core produced highly satisfactory results and the stability required in the tests was achieved.

Detail of the installation of MONO and .NET in annexes G and H.

3.3.2.1.1 .NET Core

.NET Core is a general purpose open source development platform maintained by Microsoft and the .NET community at GitHub. It is multiplatform, supports Windows, MacOS and Linux and can be used to compile device, cloud and IoT applications, which are being positively used by TRAGSATEC to reach the objectives set.

Once rewritten, we performed validation and stability tests of the code in the cluster. Once these tests are finished, it was decided which execution modes allowed in cluster (multiple simultaneous executions, a parallel execution in multiple servers, array jobs, ...) were more adequate to achieve the optimal use of resources.

In this phase, SCAYLE set up all the needed folder infrastructure and permissions to allow access to the calculation system. As described previously, the same HPC hardware infrastructure is used for both pilots.

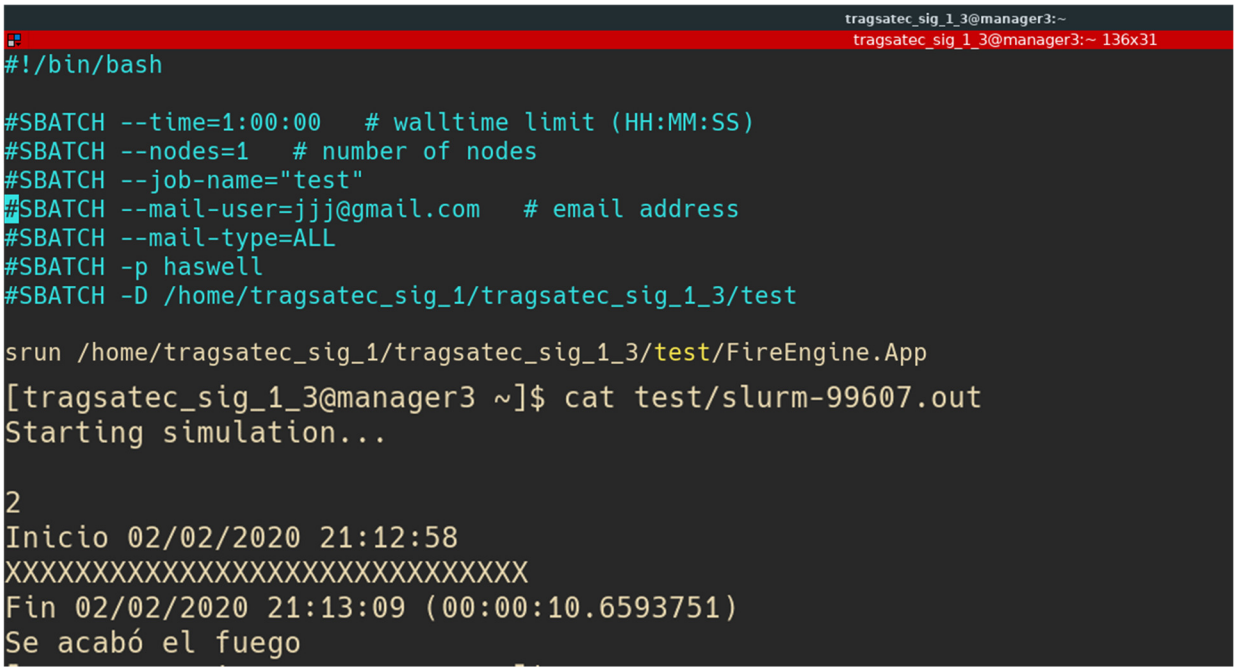

Figure 10 – FRAME code executing in Caléndula

3.3.2.1.2 Fire Simulator

3.3.2.1.2.1 Introduction

The goal of the Fire Simulator is to have a training and management tool, which will help in the decision making process for fire fighting. For this purpose, the system must be capable of executing, iteratively, different simulations for different combat scenarios, taking into account not only environmental variables, such as the orography of the terrain, fuel models or weather and atmospheric conditions, but also the different simulated actions that the deployed extinguishing means can execute on the ground.

Given that one of the characteristics to be taken into account in the implementation of the Fire Simulator should be independence from the operating system in which the calculation processes are carried out, it was considered appropriate to carry out the development using the .NET Core framework, since it is an open source framework that can be executed in the main operating systems such as Windows, Linux, macOS, ...

3.3.2.1.2.2 Input data

To carry out the execution, the fire simulator needs some input parameters that define the environment or scenario in which the simulation will take place. We could classify the input parameters in two groups:

- Information provided by the client application that starts or launches the execution of the fire simulator (extension of the terrain on which the simulation takes place, focus/s of the fire, start time, simulation time, weather conditions, ...)
- Information provided by additional services (weather conditions, fuel models, digital terrain model, roads, firewalls, ...)

Input data provided by the client application

The input data is indicated by a *json* object whose content, depending on the type of client application, can be a file located in the directory system, a stream in memory.

This *json* file contains information collected by the application invoking the service and must have a structure similar to the following example:


```
{
 "date": "2020-06-05 14:45",
 "extension": [ -5.0473326445, 40.1447562123, -5.0435990095, 40.1482908873 ],
 "focus": {
  "type": "FeatureCollection",
  "features": [
   {
    "type": "Feature",
    "geometry": {
     "type": "Point",
     "coordinates": [ -5.044973373413086, 40.14675935402329 ]
    },
    "properties": null
   },
   {
    "type": "Feature",
    "geometry": {
     "type": "LineString",
     "coordinates": [
      [ -5.0441408156693806, 40.14656252796877 ],
      [ -5.043818950260175, 40.146280409679406 ],
      [ -5.044102191794082, 40.14593924144995 ],
      [ -5.044342517656333, 40.14558330992102 ],
      [ -5.044679403370537, 40.14552426116728 ],
      [ -5.045241594314576, 40.145501297589306 ]
     ]
    },
    "properties": null
```



```
      }
     ]
    },
    "weather": [
     {
      "humidity": 15,
      "temperature": 25,
      "windSpeed": 5,
      "windAngle": 250,
      "date": "2020-06-05 14:45"
     },
     {
      "humidity": 20,
      "temperature": 20,
      "windSpeed": 0,
      "windAngle": 270,
      "date": "2020-06-05 13:45"
     },
     {
      "humidity": 10,
      "temperature": 30,
      "windSpeed": 10,
      "windAngle": 200,
      "date": "2020-06-05 16:30"
     }
    ]
   }
```


Figure 11 – Fire Simulator json sample code

Information provided by additional services

The additional services are a variable set of services that provide other data to the fire simulator. Additional services can be defined as services that connect to a database to obtain information, services that connect to information providers on the Internet, ... In general, these are services that connect to data providers, both local and remote.

Depending on the type of service and the global configuration set for the application, these services can provide the following types of information:

- Weather information. If a service containing meteorological information is defined, the simulator can invoke this service to obtain the values of variables such as wind speed, wind direction, humidity, ...
- Digital terrain model information.
- Maps of roads, paths, firebreaks, swamps, ...
- Forest map or fuel models, ...

3.3.2.1.2.3 Output data

After executing a simulation, an output file is generated in text format containing information on the progress of the fire.

In the current version, the output file contains information on the start and end time of combustion of a 1m. x 1m. cell, which allows for a progressive representation of the advance

of the fire by loading the file in the desktop application developed for that purpose.

In addition to the output file with the information of the combustion per cell, the execution records the time spent in carrying out the simulation, which will allow to carry out comparative analysis of performance in different servers or execution environments.

Example of an output file:

```
132358439230000000-132358439405000000 132358439145000000-132358439320000000
132358439135000000-132358439310000000 132358439035000000-132358439210000000
132358438905000000-132358439080000000 132358438895000000-132358439070000000
132358438820000000-132358438995000000 132358438745000000-132358438920000000
132358438645000000-132358438820000000 132358438535000000-132358438710000000
132358438430000000-132358438605000000 132358438355000000-132358438530000000
132358438285000000-132358438460000000 …
```

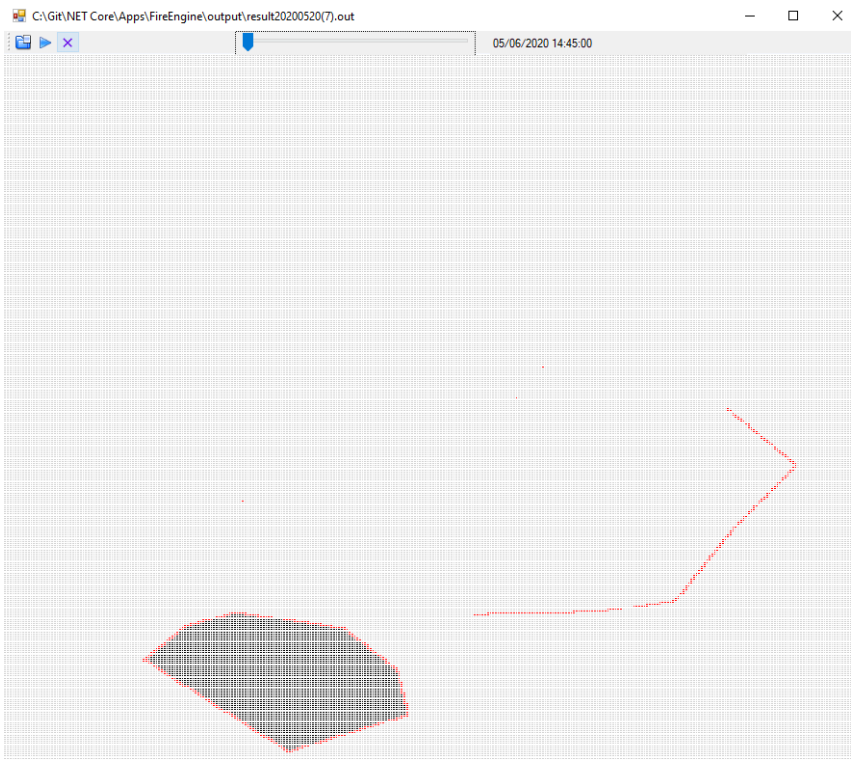

Figure 12 – Graphic representation of the simulation model output

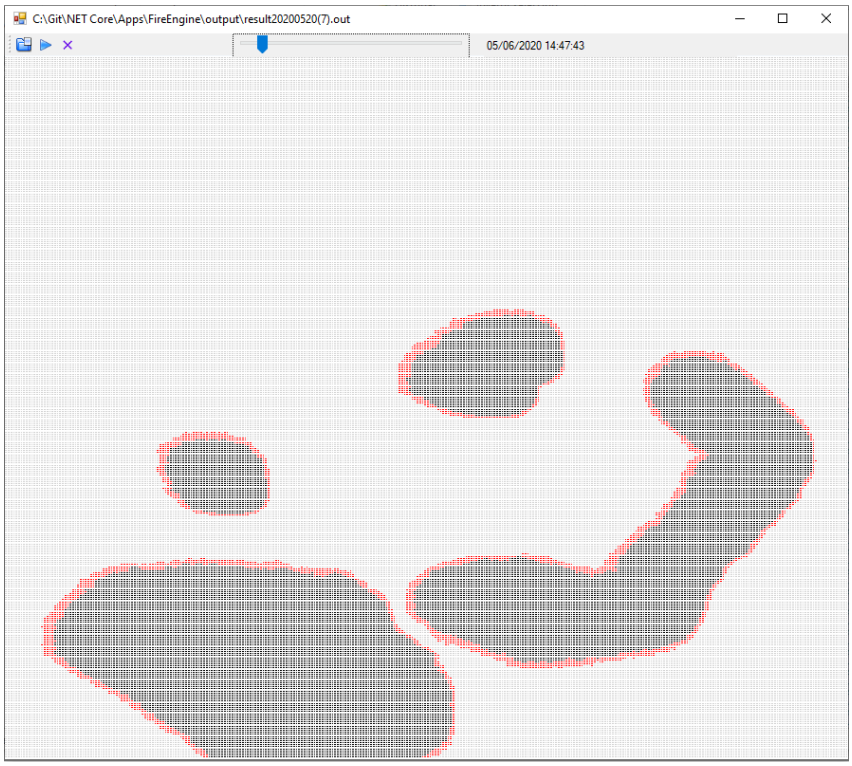

Figure 13 – Graphic representation of the simulation model output

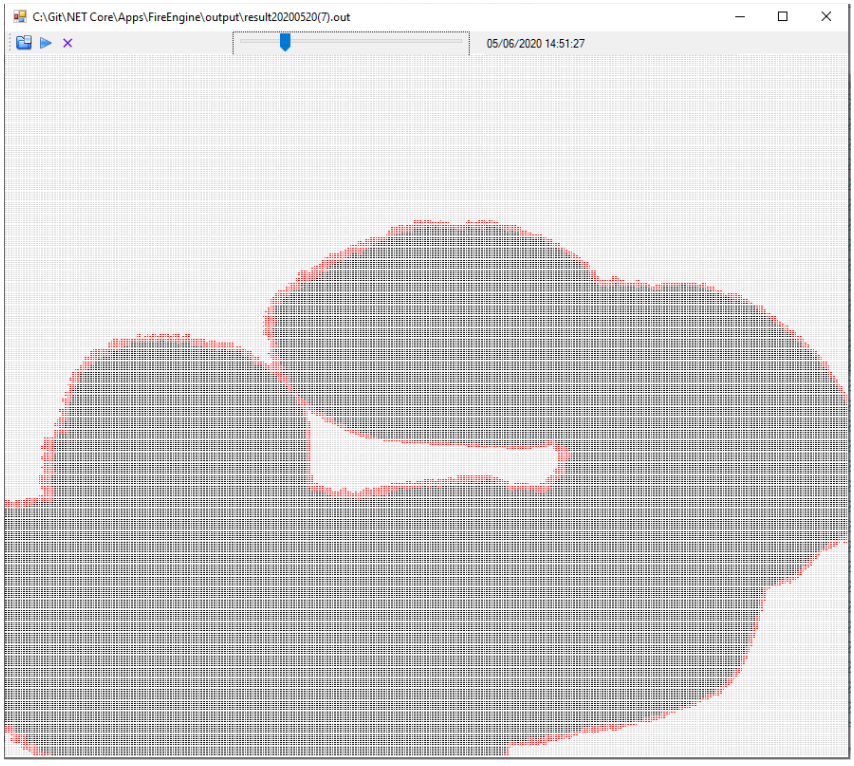

Figure 14 – Graphic representation of the simulation model output

3.3.2.1.2.4 The simulator on Caléndula

For execution in Caléndula it is sufficient to copy the directory containing the libraries into a Caléndula directory. Since the compilation contains all the necessary libraries to be executed in Linux, it is not necessary to carry out any previous installation of other packages or

additional modules; that is to say, there are no previous requirements as far as the OS is concerned.

Once the additional services have been configured, there are two possibilities to start the execution of one or several simulation processes:

- Launch by means of scripts several execution processes indicating, for each of them, its input file containing the data of the client application.
- Have a service (e.g. a web service) that receives the input data and launch the simulation process with this input data as client application data.

In this way, it is possible to simultaneously execute several simulations with different variables involved in the execution; that is, different meteorological conditions, different firewall locations, ... can be simulated simultaneously, thus allowing a predictable analysis of results to help in decision-making.

3.3.3 Common hardware for both pilots.

HPC hardware infrastructure assigned for both pilots, is the intensive calculation Haswell architecture, which has 114 servers with the following technical specifications:

- 2 Intel Xeon E5-2630 v3 processors (Haswell) 8 cores @ 2.40 GHz
- 32 GB RAM
- Infiniband interface FDR 56Gb/s

Figure 15 – Hashwell SCAYLE infrastructure

3.3.3.1 Haswell architecture.

Haswell is the codename for a processor microarchitecture developed by Intel as the "fourth-generation core" successor to the Ivy Bridge (which is a die shrink/tick of Sandy-Bridge-microarchitecture). Haswell CPUs are used in conjunction with the Intel 8 Series chipsets, Intel 9 Series chipsets, and Intel C220 series chipsets.

Design

The Haswell architecture is specifically designed to optimize the power savings and performance benefits from the move to FinFET (non-planar, "3D") transistors on the improved 22 nm process node.

Performance

Approximately 8% faster vector processing. Up to 5% higher single-threaded performance. 6% higher multi-threaded performance. Desktop variants of Haswell draw between 8% and 23% more power under load than Ivy Bridge. A 6% increase in sequential CPU performance (eight execution ports per core versus six). Up to 20% performance increase over the integrated HD4000 GPU (Haswell HD4600 vs Ivy Bridge's built-in Intel HD4000). Total performance improvement on average is about 3%. Around 15°C hotter than Ivy Bridge, while clock frequencies of over 4.6 GHz are achievable

Technology

22 nm manufacturing process 3D Tri-Gate FinFET transistors Micro-operation cache (Uop Cache) capable of storing 1.5 K micro-operations (approximately 6 KB in size) 14- to 19-stage instruction pipeline, depending on the micro-operation cache hit or miss (an approach used in the even earlier Sandy Bridge microarchitecture) Mainstream variants are up to quad-core. Native support for dual-channel DDR3/DDR3L memory, with up to 32 GB of RAM on LGA 1150 variants 64 KB (32 KB Instruction + 32 KB Data) L1 cache and 256 KB L2 cache per core A total of 16 PCI Express 3.0 lanes on LGA 1150 variants Variable Base clock (BClk) like LGA 2011. As well as lighter Ultrabooks, but the performance level is slightly lower than the 17 W version.

The following image shows and 3D representation of the SCAYLE Data Center. It shows the server cabinets and the 3 rooms that house Caléndula. In the room that houses all the computing, storage, cloud and telecommunications equipment, the cabinets that house the technology referred to in the different parts of this deliverable can be seen in green. Next figure and figures 17 and 36 of this deliverable show (in green) the equipment referred to in those parts of the document in the same room. In this specific case, it can be seen that the equipment used for the CAMBrIc and FRAME pilots is housed in cabinets Nº1, 9 and 10. They are the Haswell equipment described above.

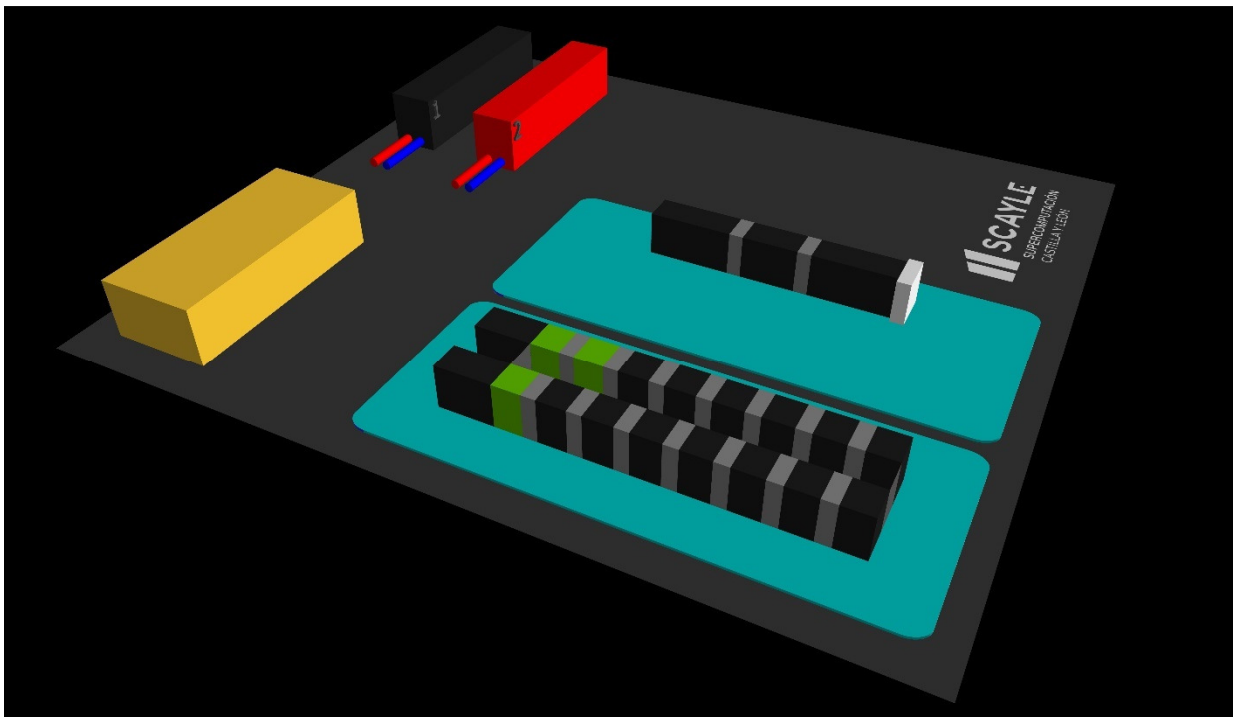

Figure 16 – Racks Nº 1, 9 and 10. Location Hashwell SCAYLE Data Centre

3.4 Cross-Forest Linked Open Data platform infrastructure

Within the framework of the project, two large Open Data warehouses are commissioned, one for UVA-GSIC/EMIC and another for TRAGSATEC.

Below, we make a brief technical description of what they are, the software and hardware required and their main functionalities.

As part of the Cross-Forest project, (Deliverables D2.1 and D2.2) a LOD version of the Spanish Forest Inventory and Forestry Map have been created. First, a suite of ontologies to model forest inventories (with concepts such as Plot, Tree, PlantSpecies, and TreeMeasure), forestry maps (Patch, Use, CanopyCover...), and geographical positions (SpatialEntity, Position, Polygon, CRS, Datum…) have been developed. Secondly, the Spanish sources have been transformed into LOD using the aforementioned ontology suite. The resulting dataset includes 92K plots, 1.4M trees, and 680K land cover patches. In addition to the original data, we provided WGS84 coordinates for all positions, a simplified low-resolution layer of land cover patches, and mappings to well-known resources. Finally, UVA-GSIC/EMIC has developed a web-based tool for exploring the resulting dataset.

UVA-GSIC/EMIC requested virtualization resources to host an end point which, apart from being a result of the project itself, has also served its forest viewer/scouting device (Forest Explorer). Initially the tests are carried out on the UVA's own equipment, but in January 2020 the migration and testing tasks to the assigned equipment in Caléndula begin.

To provide the Virtuoso facilities necessary for the development of the project, SCAYLE uses its private cloud infrastructure. The use of virtual machines to host multiple Virtuoso installations facilitates the backup of these machines and the dynamic modification of their characteristics according to the needs that may arise during the course of the project.

From February 15, 2020, the virtual resources requested by TRAGSATEC to host its own endpoint are into service. The functions and full description of this endpoint can be found in the Deliverable D2.2.

This second endpoint is hosted in a virtual machine that runs on the same physical machines described below: Lenovo SR630 server (section 3.4.2.1).

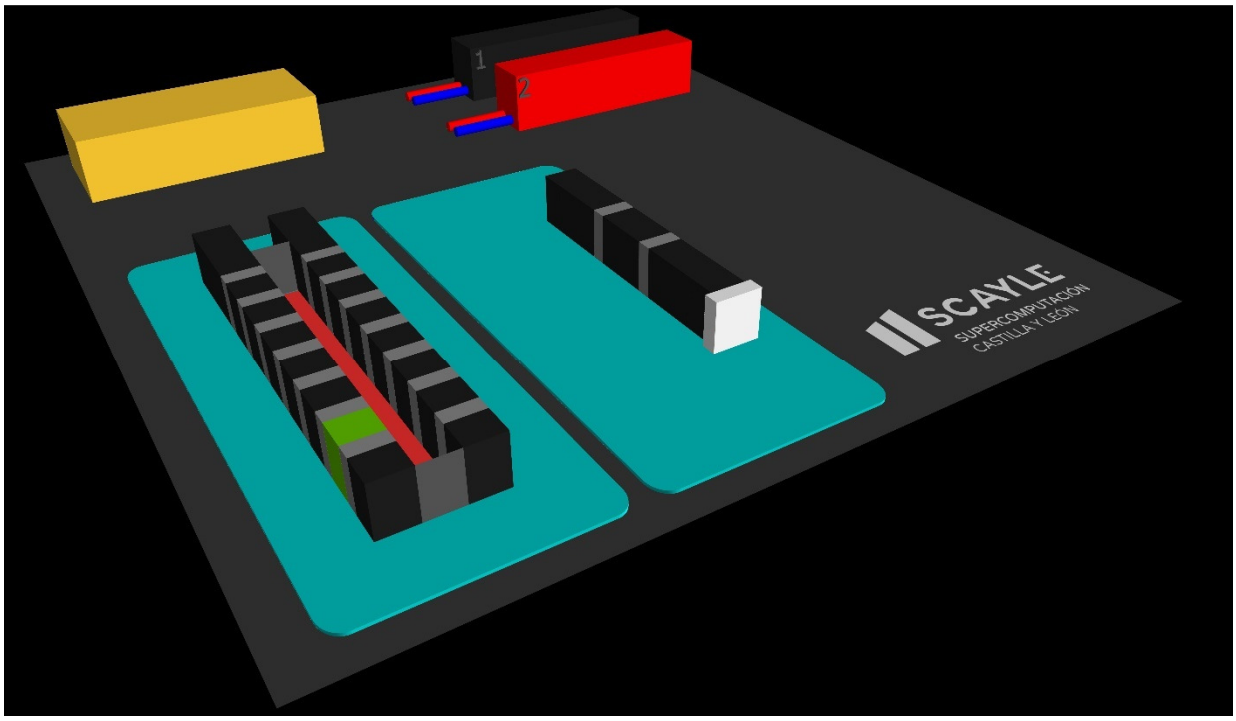

Figure 17 – Rack Nº 7. Location hardware Linked Open Data SCAYLE Data Centre

3.4.1 Software

SCAYLE's virtual infrastructure is based on VMware ESXi servers with a design focused on load balancing and high availability. These servers have access to a storage system, necessary for the storage of triples, with a total capacity of more than 350TB.

On this infrastructure, an installation of Virtuoso in its version 07.20.3230 has been made, compiled from its original sources stored in GitHub [https://github.com/openlink/virtuoso-opensource]. The installation from sources, thus avoiding the precompiled software, has allowed a better adaptation to the allocated resources and, hopefully, it will also produce an increase in performance. Detail of the installation of Virtuoso and preparation of the disks for it in annexes I and J.

External access to these facilities has been provided through an assignment of a public IP (and its associated rules in SCAYLE firewalls) that allows access to the machine's SSH server and the web interface provided by Virtuoso. With the local user created, it will be possible to access the machine, upload all its data and make the necessary queries to that data.

From the storage point of view, the virtual machines and the databases they run are backed up in the SCAYLE backup systems following the temporary policies (frequency of the backup, number of copies stored and their rotation) that the technicians consider necessary.

Machine disk storage modification process

Modifications were made to these virtual machines because of the need for more disk storage resources. These changes have been made using the LVM manager.

LVM Logical Volume Management, is a logical volume manager that allows us to extend, reduce and modify partitions on hard disks in real time, without the need to disassemble the file system.

In a simplified way, we could say that LVM is an abstraction layer between a storage device and a file system.

In cases where servers are using LVM, it is relatively easy to increase disk space or a particular partition. In the example below, the root partition of the CentOS 7.7 server has been expanded for the machine created for TRAGSA group. This machine is the one whose storage has been increased the most, up to 3 TB.

In the two increases of storage capacity made in TRAGSA machine, they have been done in a different way. In the first one, an existing disk is added to the machine and in the second one, a new disk is added to the machine. All these processes are described in Annex J.

After these two increases in disk storage, TRAGSA group machine has 3TB.

3.4.1.1 VMware ESXi

VMware ESXi is an enterprise-class, type-1 hypervisor developed by VMware for deploying and serving virtual computers. As a type-1 hypervisor, ESXi is not a software application that is installed on an operating system (OS); instead, it includes and integrates vital OS components, such as a kernel.

After version 4.1 (released in 2010), VMware renamed ESX to ESXi. ESXi replaces Service Console (a rudimentary operating system) with a more closely integrated OS. ESX/ESXi is the primary component in the VMware Infrastructure software suite. The name ESX originated as an abbreviation of Elastic Sky X.

Architecture

ESX runs on bare metal (without running an operating system) unlike other VMware products. It includes its own kernel. In the historic VMware ESX, a Linux kernel was started first, and is then used to load a variety of specialized virtualization components, including ESX, which is otherwise known as the vmkernel component. The Linux kernel was the primary virtual machine; it was invoked by the service console. At normal run-time, the vmkernel was running on the bare computer, and the Linux-based service console ran as the first virtual machine. VMware dropped development of ESX at version 4.1, and now uses ESXi, which does not include a Linux kernel at all.

The vmkernel is a microkernel with three interfaces: hardware, guest systems, and the service console (Console OS).

Interface to hardware

The vmkernel handles CPU and memory directly, using scan-before-execution (SBE) to handle special or privileged CPU instructions and the SRAT (system resource allocation table) to track allocated memory.

Access to other hardware (such as network or storage devices) takes place using modules. At least some of the modules derive from modules used in the Linux kernel. To access these modules, an additional module called vmklinux implements the Linux module interface. According to the README file, "This module contains the Linux emulation layer used by the vmkernel."

These drivers mostly equate to those described in VMware's hardware compatibility list. All these modules fall under the GPL. Programmers have adapted them to run with the vmkernel: VMware Inc. has changed the module-loading and some other minor things.

Service console

In ESX (and not ESXi), the Service Console is a vestigial general purpose operating system most significantly used as bootstrap for the VMware kernel, vmkernel, and secondarily used as a management interface. Both of these Console Operating System functions are being deprecated from version 5.0, as VMware migrates exclusively to the ESXi model. The Service Console, for all intents and purposes, is the operating system used to interact with VMware ESX and the virtual machines that run on the server.

In the event of a hardware error, the vmkernel can catch a Machine Check Exception. This results in an error message displayed on a purple diagnostic screen. This is colloquially known as a purple diagnostic screen, or purple screen of death (PSoD, cf. Blue Screen of Death (BSoD)).

Upon displaying a purple diagnostic screen, the vmkernel writes debug information to the core dump partition. This information, together with the error codes displayed on the purple diagnostic screen can be used by VMware support to determine the cause of the problem.

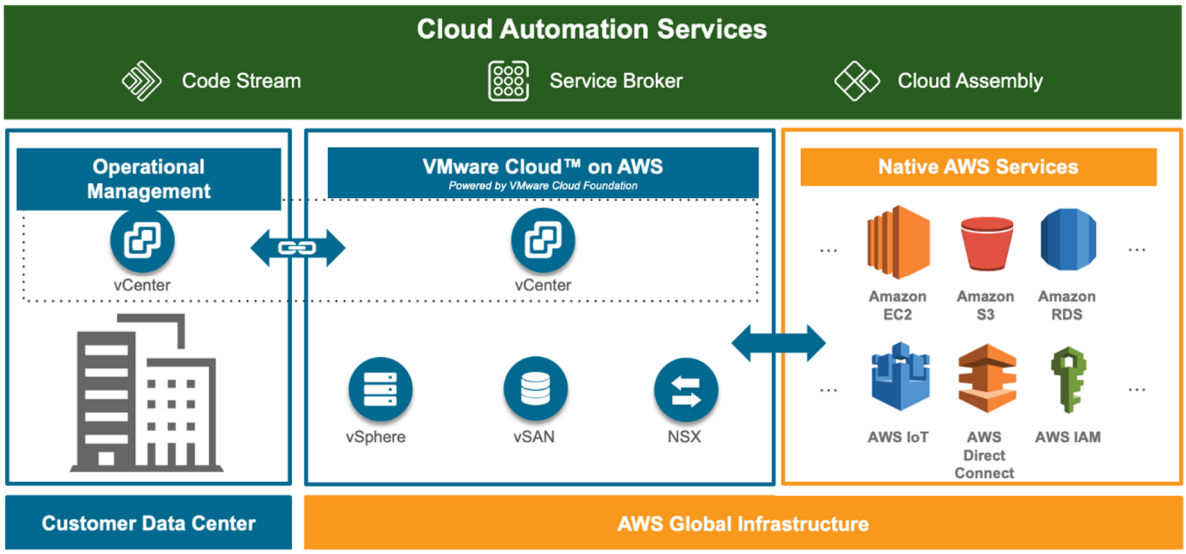

Figure 18 – Standard VMware infrastructure

3.4.1.2 SSH Server

Additionally, we have provided external access to this installation through a public IP (including associated rules in the SCAYLE firewalls) that allows access to the machine's SSH server and to the web interface provided by Virtuoso. A local user has also been created (18/11/2019) to access the machine, upload data and make the necessary queries to that data.

The SSH protocol works on the client/server-model. The SSH client always initiates the setup of the secure connection, and the SSH server listens for incoming connection requests (usually on TCP port 22 on the host system) and responds to them.

In the connection setup phase, the SSH server authenticates itself to the client by providing its public key. This allows the SSH client to verify that it is actually communicating with the correct SSH server (instead of an attacker that could be posing as the server).

After a successful authentication the server provides the client access to the host system. This access is governed with the user account permissions at the target host system.

The secure connection between the client and the server is used for remote system administration, remote command execution, file transfers, and securing the traffic of other applications. Automated SSH sessions are very often used as a part of many automated processes that perform tasks such as logfile collection, archiving, networked backups, and other critical system level tasks.

Availability of SSH Servers

Most server operating systems come with a native, pre-installed SSH server implementation. Those that are an exception to the rule are usually installed with an SSH server from a trusted security solution vendor, such as SSH Communications Security, Bitvise, or VanDyke Software. These companies sell SSH software and provide the technical support and maintenance services for it. The open source community maintains the OpenSSH project that provides a free to use, non-commercial SSH implementation.

Tectia SSH - Enterprise Grade SSH Clients and Servers - From the Inventors of the Protocol =button btn-success

Quality Equals Security

As security software, the SSH server has strict requirements for software quality. The SSH server process executes with wide system privileges, and acts as an access control "gatekeeper" to the host system. This makes the SSH server an attractive target for hackers and malware. The pivotal security role of the SSH server places stringent requirements for its code quality and reliability. Bugs and defects in the code can lead to serious security vulnerabilities.

Standardized Security

The SSH protocol has been standardized by the Internet Engineering Task Force (IETF). The standards are open and were authored as a joint effort by many security specialists and companies. As the original inventor of the protocol, SSH Communications Security was a key contributor in the standardization effort.

3.4.2 Hardware

From February 12th, after the acquisition and commissioning of the new resources by SCAYLE, one installation of Virtuoso (07.20.3230 version) will be available in a Caléndula virtual machine, at the request of the partner UVA-GSIC/EMIC.

Each 1U Lenovo SR630 server, has the following hardware configuration:

- 2x processors X86_64 with 24 cores e hyperthreading Intel 6252
- 1 TB of RAM.
- 4x interfaces 10Gbe SFP+ with GBics SFP+SR.
- 2x Hard Disks with mirroring SSD M.2 of 128GB.
- Compatibility with VMWare ESXi 6.5 U2 and 6.5 U3.

- Enterprise remote management module (Lenovo xClarityController Enterprise).

After evaluating the partner's requests, the minimum capacities needed for its implementation and after several meetings of the SCAYLE technical staff, it was decided to provide the Cross-Forest infrastructure and for this endpoint in particular, a virtual machine with the following characteristics:

128GB of RAM and 16 cores depending on the physical machines described above where they are running.

The access management is the same as it was in the previous machine used. it also has access to the SSH server and to the Virtuoso's web interface through a public IP. In this case, two local users have been created so that they can access the machine, upload their data and make any queries they need from that data.

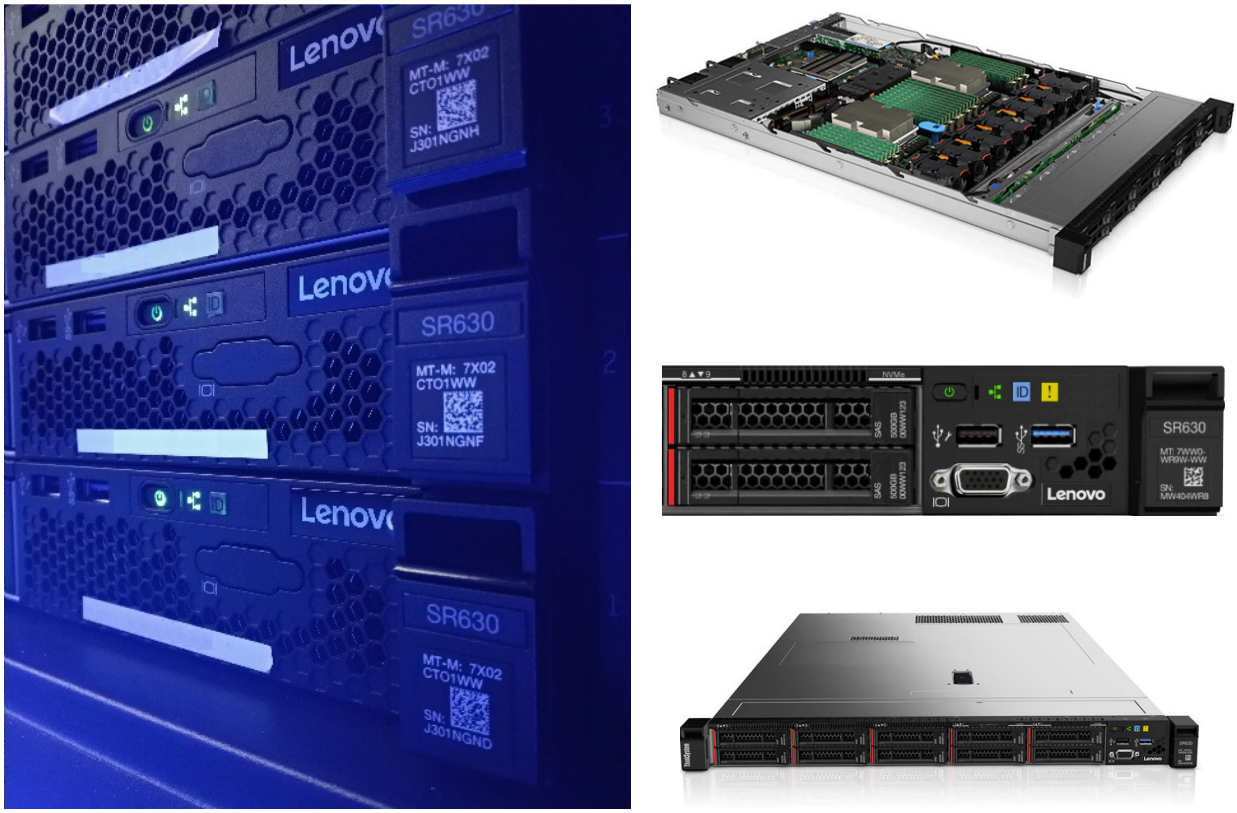

Figure 19 – SCAYLE Lenovo SR630 server

3.4.2.1 ThinkSystem SR630.

This serves offers support for data analytics, hybrid cloud, hyper-converged infrastructure, video surveillance, and high-performance computing.

Optimized support for workloads

Intel® Optane™ DC Persistent Memory provides a level of flexible memory designed specifically for data center workloads that offers an unprecedented combination of capacity, persistence.

Flexible storage

Lenovo AnyBay design offers different interface options in the same bay: SAS, SATA or U.2 NVMe PCIe drives. Freedom to configure some of the bays with PCIe SSD and use the remaining bays for high-capacity SAS, with the ability to expand to more PCIe SSDs in the future when you need it. This gives SCAYLE, and therefore the infrastructure designed for Cross-Forest, great versatility in the changes required.

Lenovo XClarity Controller

It's the integrated management engine in SCAYLE's ThinkSystem servers. It's designed to standardize, simplify and automate key server management tasks. Lenovo XClarity Administrator is a virtualized application that centrally manages ThinkSystem servers, storage and network, reducing provisioning times by up to 95 percent over manual operation. Running XClarity Integrator helps you streamline IT management, accelerate provisioning, and contain costs by seamlessly integrating XClarity into the SCAYLE IT environment.

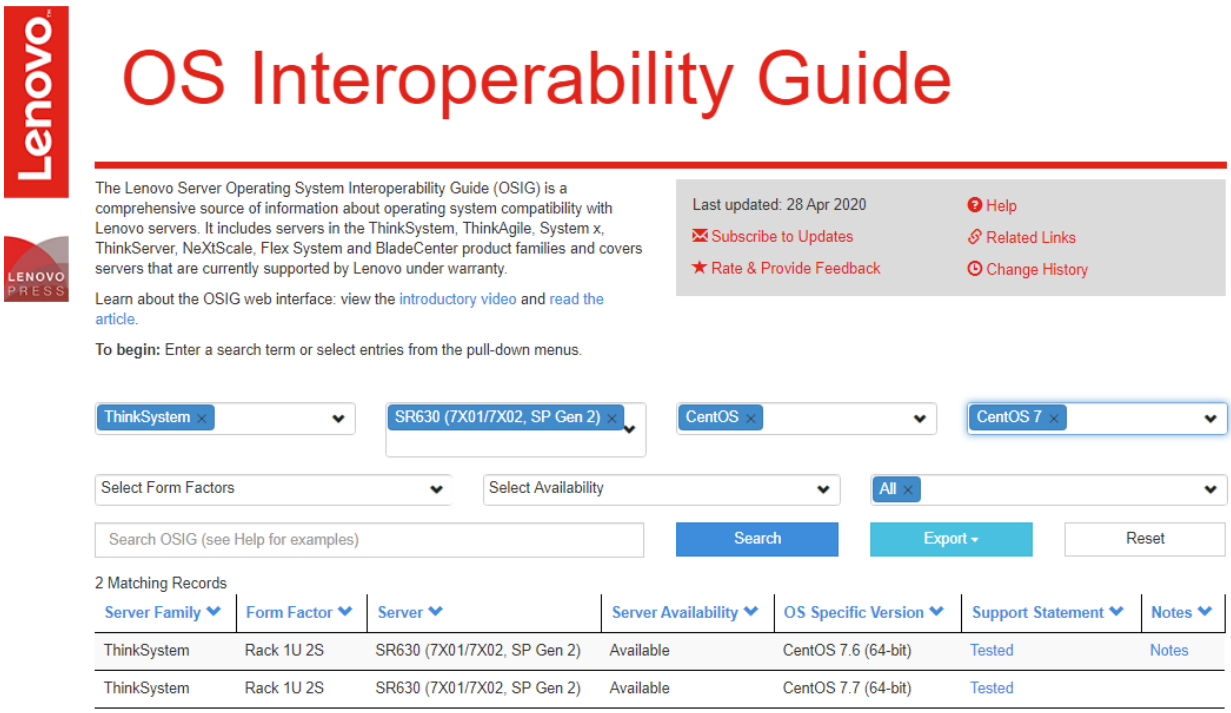

Figure 20 – Lenovo OS Interoperability Guide

3.4.2.2 Storage system

250GB of storage disk, housed in Rack Nº 16 of the SCAYLE Data Center, in Dell EqualLogic booths with the following specifications:

1. Model PS6110 with 24 2TB SAS disks.
2. Model PS6210 with 24 SAS 2TB disks.
3. Model PS6110 with 24 SAS 600GB disks.

Depending on the needs of software availability, the hard disk supplies may vary among the components of the cabin.

These servers were installed and added to the set of servers that serves the project after the stability tests.

Dell Equalogic.

The EqualLogic PS6110 Series doesn't force to make a choice between scaling storage performance or expanding capacity. A virtualized, peer-scale architecture delivers high-performance to demanding enterprise applications, even as they grow. PS Series arrays make it possible to purchase only the storage you need when they are need it. His management features speed SAN configuration while built-in intelligence senses network connections, automatically builds RAID sets and conducts system health checks. Even multi-generations of PS Series arrays can work together to automatically manage data, load balance across all resources or seamlessly expand the storage pool when you add a new array. The PS6110 Series arrays reduce single points of failure with fully redundant and hot-swappable controllers, fan trays, power supplies, disk drives with hot spares, and vertical port sharing.

PS Series arrays include enterprise-class storage software features like sub-volume automated tiering, thin provisioning and pointer-based snapshots so you can efficiently deliver the right data to the right place at the right time. In addition, 10GbE arrays feature Data Center Bridging designed to greatly reduce or eliminate dropped packets by standardizing performance over your converged SAN and LAN network.

EqualLogic Host Software extends the functionality of the array-based software, optimizing server and storage cooperation. Host Integration Tools are available for major OS and hypervisor providers, including Microsoft, VMware and Linux.

EqualLogic SAN Headquarters (SAN HQ) gives in-depth reporting and analysis, consolidated performance and robust event monitoring across multiple EqualLogic groups so to have a well-tuned EqualLogic SAN to meet and surpass the business requirements. His specs are included as annex K.

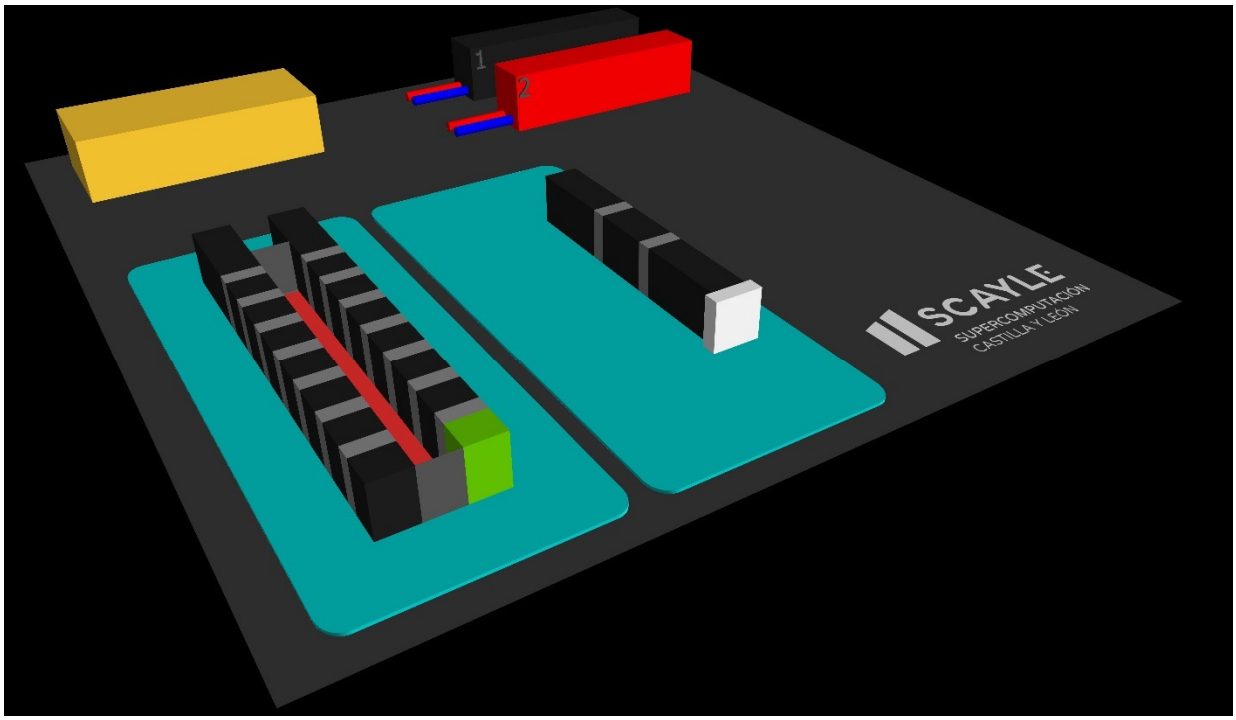

Figure 21 – Rack Nº16. Location Dell Equalogic SCAYLE Data Centre

4 Task 1.4. Design of the storage system for the simulations performed.

Although this phase of the work is part of task 1.2 and does not yet correspond to the dates on which this report is being produced, SCAYLE already has a policy related to the security of its installations, the data housed in them and the guarantee of the services it offers.

An easily scalable storage infrastructure, including backup and simulation history restoration procedures, has been designed and implemented in order to store the simulations carried out for future study and review. Specific work on Cross-Forest will be carried out as simulations and tests are generated by the partners.

In the next sections, the general characteristics that SCAYLE has currently implemented in this domain can be found. The services offered by SCAYLE are extremely flexible and versatile, with a selection of appropriate services and a very demanding quality control.

SCAYLE has an additional and complementary backup system for the data of the virtual instances in IaaS (Infrastructure as a Service) regime, thus guaranteeing the redundancy and high availability of the storage of the virtualization productive environment.

This system consists of an asynchronous replication of complete volumes at the level of storage booths that allows for an off-site copy of the data.

The restoration of these copies will not be available to the user except in the event of an event or failure of SCAYLE's own storage infrastructure and in no case for recovery of particular data loss. This replication may not guarantee the integrity of open files, especially databases.

A second Dell Storage PS6610 is therefore available for this purpose, located in a building outside of SCAYLE's premises to ensure remote backup of data in the event of failure scenarios.

As for the replication policy, work is scheduled to be performed once a week, with the last two replications being saved. These tasks are usually executed in the afternoon or evening to have as little impact as possible on the performance of the booths, but may vary depending on the number of volumes, the amount of data to be replicated from each volume, the load on the booths themselves and other factors that may influence this task.

4.1 Availability.

SCAYLE has its own comprehensive monitoring system and 24x7 surveillance and action on the platform. The technical team acts proactively and automatically in the event of anomalies.

SCAYLE guarantees a monthly availability of the platform (virtual server) of 99.90%. The calendar month will be taken into account for the calculation of availability.

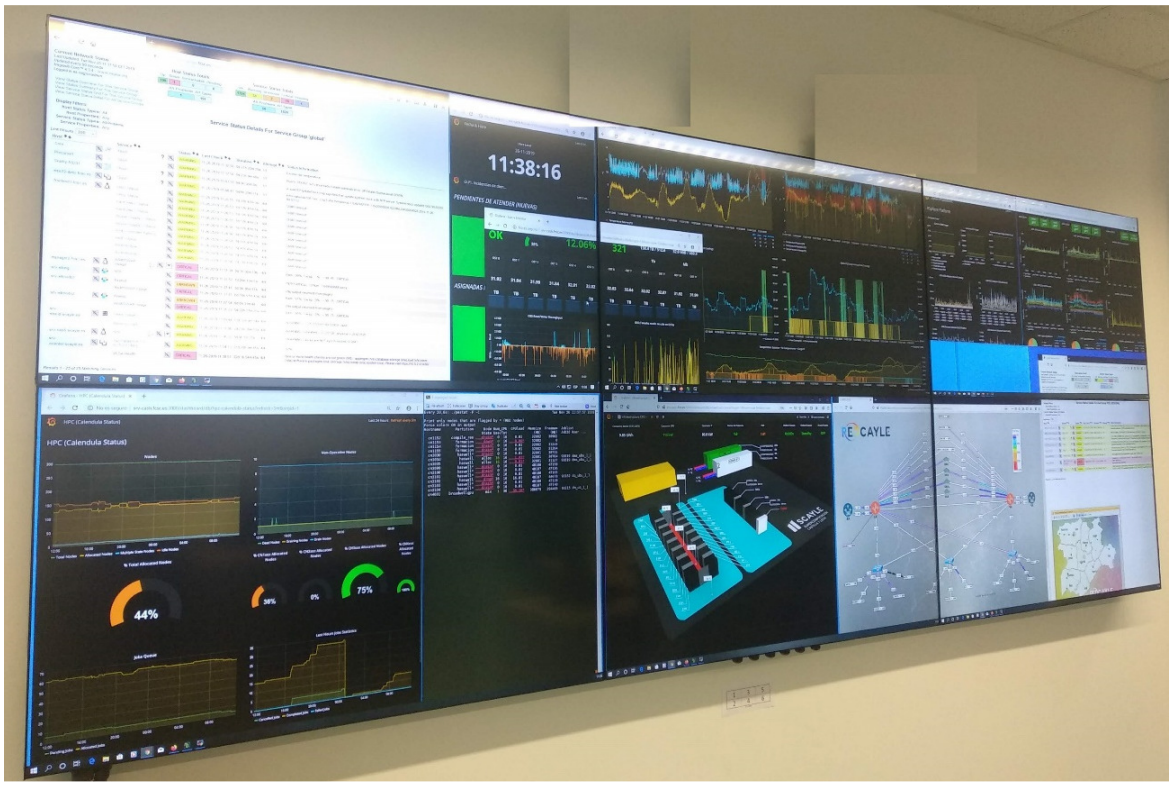

Figure 22 – Caléndula monitoring system

Figure 23 – Caléndula Supercomputer

4.2 Backup.

The client data, whether it is information contained in virtual instances in IaaS (Infrastructure as a Service) mode or files in the High Performance Computing system in time sharing mode are the property of the client and it is the responsibility of the client to safeguard them. The client may contract the corresponding backup services, for which the following alternatives are available:

1. Purchase of NAS space: SCAYLE will provide a directory on a NAS storage system, which will be billed at the current rate. It will be the customer's responsibility to configure the copy system.
2. Hiring of external cloning: The customer may hire the cloning of its machines against an external data processing center. This service will be billed according to the rate in force at any given time.

Service 1 is valid for both IaaS virtual machine instances and HPC services. Service 2 is exclusive for virtual machine instances.

However, the above is backed up by SCAYLE in order to guarantee the integrity of the platform in accordance with the following scheme:

- IaaS VM instances shall be cloned weekly using automatic backup software. The copy software creates a snapshot of the virtual machine and an incremental copy of it, keeping at least two. The copy will be made on a different storage system than the one on which the machine is in production. Due to the volume of information to be copied and the fact that the copy is done in a batch process, SCAYLE cannot guarantee the time window in which the copy will be done. Since it is a virtual machine clone, the copy will be made only on its definition and virtual disks, excluding those components that are not part of it: NAS system disks, shared lun's, etc.
- The files of the storage system of the high performance computing system will be copied weekly and two (2) copies will be kept.

5 Task 1.5. Provision of access to the results obtained.

In this task, a virtual machine based environment has been implemented. The environment constitutes the infrastructure that serves as the basis for the mobile and web interfaces, as well as the content management features, which are part of the presentation layer. This infrastructure is also connected to the storage system implemented in the previous tasks.

This phase of the work is already reaching a certain maturity and the part of the infrastructure prepared for this is already available and it is understood that it will require very slight modifications in the future. The tasks related to the pilots running on HPC have continued in a sustained manner as can be seen in the relevant sections of this document.

6 Results and conclusions

The operational needs of the pilots were assessed and appropriate HPC capabilities were sized. In the case of the pilots, both share the same structure and characteristics with respect to the hardware for each of the resources that were designated for each one. In the case of the software, it has been necessary to provide them with different characteristics since the two pilots came from different development environments and, in addition, neither of them had, in their initial design, a codification foreseen for a future parallelization of their code nor a simultaneity of simulations.

In the case of the Cross-Forest platform the opposite happened, the same software has been used for the 2 partners who demanded the data repository but both had different needs in terms of hardware resources and therefore it was necessary to perform the corresponding sizing for each of the requests.

The HPC infrastructures required for the execution of the project have been installed and have been in operation for several months. The partners involved in the different tasks have access accounts to the SCAYLE systems, and the tests and applications are working properly. From now on, and until the end of the project, SCAYLE will keep supporting partners involved in the different tasks.

7 References

https://lenovopress.com/lp0643-thinksystem-sr630-server-xeon-sp-gen1
https://www.vmware.com/
https://en.wikipedia.org/wiki/VMware_ESXi
https://www.ssh.com/ssh/
https://slurm.schedmd.com/
https://www.python.org/
https://dask.org/
https://en.wikipedia.org/wiki/.NET_Core
https://www.mono-project.com/

Annex A: List of abbreviations

EC	European Commission
INEA	Innovation and Networks Executive Agency
DGT	Direção-Geral do Território
TRAGSA	Empresa de Transformación Agraria, S.A.
TRAGSATEC	TRAGSA subsidiary company
UVA-iuFOR	Universidad de Valladolid - Instituto Universitario de Gestión Forestal Sostenible
GSIC-EMIC	Grupo de Sistemas Inteligentes y Cooperativos / Educación, Medios, Informática SCAYLE Fundación Centro de Supercomputación Castilla y León
GA	Grant Agreement
HPC	High Performance Computing
CAMBrIc	CAlidad de la Madera en Bosques mIxtos
FRAME	Forest fiRes Advanced ModElization
SIMANFOR	Sistema para la simulación de alternativas de manejo forestal sostenible
SQL	Structured Query Languages
SSH	Secure SHell
VSP	Virtuoso´s Web Language
OS	Operating System
LOD	Linked Open Data
MPI	Message Passing Interface
OpenMP	Open Multi-Processing
SLURM	Simple Linux Utility for Resource Management
GLPI	Gestionnaire Libre de Parc Informatique
GCC	GNU C Compiler

LVM	Logical Volume Manager
TCP	Transmission Control Protocol
ESX/ESXi	Elastic Sky X
TB	Tera Byte
Ghz	Giga Hertz
RAM	Random Access Memory
FDR	Fourteen Data Rate
IaaS	Infrastructure as a Service
VM	Virtual Machine
NAS	Network Attached Storage
EPEL	Extra Packages for Enterprise Linux
GPU	Graphics Processing Unit
CPU	Central Processing Unit
SAN	Storage Area Network
NAS	Network Attached Storage
IETF	Internet Engineering Task Force
RAID	Redundant Array of Independent Disks
LAN	Local Area Network
GBic	Gigabit Interface Converter

Annex B: Activity 1 Meeting

CROSS-FOREST. 01-Activity1 Meeting

Summary of meeting

Date: 11/02/2019

Location: SCAYLE

Address: Edificio CRAI-TIC, Campus de Vegazana s/n. Universidad de León

Contact Person: Álvaro Fanego Lobo, SCAYLE Supercomputación Castilla y León

AGENDA/OBJECTIVES

Analyze the viability of being able to take the SIMANFOR simulator code to a favorable environment for Caléndula/HPC.

ATTENDEES

Felipe Bravo Oviedo, Universidad de Valladolid
Cristóbal Ordóñez Alonso, Universidad de Valladolid
Vicente Matellán Olivera, SCAYLE Supercomputación Castilla y León
Jesús Lorenzana Campillo, SCAYLE Supercomputación Castilla y León
María Jular Castañeda, SCAYLE Supercomputación Castilla y León
Álvaro Fanego Lobo, SCAYLE Supercomputación Castilla y León

CONCLUSIONS AND NEXT STEPS

SIMANFOR is developed in Microsoft's .NET technology. Its fundamental use is in the academic environment and for this reason they have never considered a change.
From the University of Valladolid it is reported that the intention is to contact the developer company to assess whether they are interested in carrying out a new approach that will lead to a new version of the simulator compatible with the HPC environment and with the Caléndula supercomputer.
They will subsequently report on the conversations with the company and on the progress related to this activity.
Without further business, the session is adjourned.

Annex C: Activity 1 Meeting

CROSS-FOREST. 02-Activity1 Meeting

Summary of meeting

Date: 10/04/2019

Location: Telephone conference

Address: Telephone conference

Contact Person: Tragsa, SCAYLE

AGENDA/OBJECTIVES

Analyze the viability of being able to take the FRAME simulator code to a favorable environment for Caléndula/HPC.

ATTENDEES

Víctor Gonzalvo Morales, TRAGSA
Eduardo Hombrados Carrillo, TAGSATEC
Jesús Estrada Villegas, TAGSATEC
Jesús Lorenzana Campillo, SCAYLE Supercomputación Castilla y León
María Jular Castañeda, SCAYLE Supercomputación Castilla y León
Álvaro Fanego Lobo, SCAYLE Supercomputación Castilla y León

CONCLUSIONS AND NEXT STEPS

Two possible scenarios are analyzed:
Given that the FRAME simulator was initially developed in C# and that this is a tool not very compatible with the usual HPC environments, the possibility of using *MONO* software to take all or part of the FRAME code to that environment and from there, carry out the relevant tests on the Caléndula supercomputer is assessed.
On the other hand, TRAGSA informs that it is being valued (given that it is believed that it would be the best option) to factor all the software towards programming language C, which would mean a complete compatibility towards HPC environments. This decision has not yet been taken, it will be taken in one direction or another in the near future.
It is agreed, while the decision is being made by TRAGSA, to move forward in parallel. SCAYLE will install and configure *MONO* in its equipment once this has taken place and it's a stable environment. TRAGA will provide all or part of the FRAME code and the relevant tests will be carried out.
Without further business, the session is adjourned.

Annex D: Activity 1 Meeting

CROSS-FOREST. 03-Activity1 Meeting

Summary of meeting

Date: 22/05/2019

Location: Meet Google

Address: Meet Google

Contact Person: Sngular, SCAYLE

AGENDA/OBJECTIVES

Beginning of the actions to implement the new SIMANFOR simulator code in the Calendar/SCAYLE environment.

ATTENDEES

Moisés Martínez, Sngular
Juan Tomás García, Sngular
Cristóbal Ordóñez Alonso, UVA
Jesús Lorenzana Campillo, SCAYLE Supercomputación Castilla y León
Álvaro Fanego Lobo, SCAYLE Supercomputación Castilla y León

CONCLUSIONS AND NEXT STEPS

The company Sngular is being responsible for the development of a more current version of simanfor based on Dask. Dask is a flexible library for parellel computing in Python. It is composed of two parts: Dynamic task scheduling optimized for computation. This is similar to Airflow, Luigi, Celery, or Make, but optimized for interactive computational workloads. “Big Data” collections like parallel arrays, dataframes, and lists that extend common interfaces like NumPy, Pandas, or Python iterators to larger-than-memory or distributed environments. These parallel collections run on top of dynamic task schedulers.

Caléndula has operating system Centos 7.5 and that there are already installed applications that work in Python environment does not seem that there can be problems.

The planning that is planned from Sngular is based on agile methodologies and consists of 4 Sprints of 15 days of execution each. In one month the first installable version will be available.

Next days, SCAYLE technicians will install Dask on the servers dedicated to the Cross-Forest project and access accounts will be created for Sngular users.

We will continue with on-line meetings coinciding with the sprints planned to advance further in the integration.

Without further business, the session is adjourned.

Annex E: First technical Meeting

CROSS-FOREST. REUNIÓN TÉCNICA
Agenda 7/Marzo/2019

Date: 7/marzo/2019
Location: LEÓN
Conference room: Aula 108
Address: SCAYLE. Centro de SuperComputación Castilla y León
Edificio CRAI-TIC, Campus de Vegazana s/n. Universidad de León
Contact Person: Álvaro Fanego (alvaro.fanego@scayle.es, 987293173, 987293160)

	Título ponencia	Ponente
10:00	**CROSS-NATURE: resultados reutilizables para CROSS-FOREST** • Datos, ontologías y casos de uso	Ramón Baiget (TRAGSATEC) Mat Max Montalvo (UC3)
11:00	**Ontología propuesta desde UVA**	Guillermo Vega (UVA)
11:30	**PAUSA CAFÉ Y VISITA A CALÉNDULA**	
12:00	**HPC**	
	• Introducción a MPI y OpenMP.	Lidia González (ULE) Ángel Manuel Guerrero (ULE)
	• Ejercicios prácticos en Caléndula. SCAYLE	
14:15	**COMIDA**	
	PILOTOS > estado actual y próximos pasos	
15:00	• CAMBrIc	Cristóbal Ordóñez (UVA)
15:30	• FRAME	Víctor Gonzalvo (TRAGSA) Álvaro Carrillo (TRAGSA)
16:00	**Arquitectura CROSS-FOREST**	
	• Propuesta	Telmo Jurado (TRAGSA)
	• Decisiones a tomar	Todos
16:30	Conclusions and next steps	
16:45	**End of meeting**	

Annex F: Python and Dask installations

Python Installation

Since Dask has been developed to run in Python, the first task in the installation process has been to install a Python interpreter. Python version 3.7.7 was chosen as the last one available at the time of installation.

The installation is done from one of the servers that has temporarily active direct access to the Internet. For security reasons, the calculation servers (nodes) have limited access to the Internet. Access is only allowed upon request and justification of the need for such access for downloading or sending data to perform calculations.

To achieve an optimal installation, we use the version 8.2.0 of the GCC compilers since they generate code more optimized for the Haswell processors used than the compilers installed by default with the CentOS 7.7 operating system, in its version 4.8.5.

```
CALENDULA[ sngular_aia_1_3@cn3007 ~]$ module load haswell/gcc_8.2.0
CALENDULA[ sngular_aia_1_3@cn3007 ~]$ cd ../COMUNES/
CALENDULA[ sngular_aia_1_3@cn3007 ~]$ mkdir sources
CALENDULA[ sngular_aia_1_3@cn3007 ~]$ cd sources/
CALENDULA[ sngular_aia_1_3@cn3007 sources]$ wget https://www.python.org/ftp/python/3.7.7/Python-3.7.7.tgz
CALENDULA[ sngular_aia_1_3@cn3007 sources]$ tar xvf Python-3.7.7.tgz
CALENDULA[ sngular_aia_1_3@cn3007 sources]$ cd Python-3.7.7
CALENDULA[ sngular_aia_1_3@cn3007 Python-3.7.7]$ ./configure --prefix=/home/sngular_aia_1/COMUNES/python/3.7.7 --
enable-shared --enable-optimizations
CALENDULA[ sngular_aia_1_3@cn3007 Python-3.7.7]$ make altinstall
CALENDULA[ sngular_aia_1_3@cn3007 Python-3.7.7]$ cd /home/sngular_aia_1/COMUNES/python/3.7.7/bin/
CALENDULA[ sngular_aia_1_3@cn3007 bin]$ ln -sf python3.7 python
```

Dask Installation 2.10.1

We used the Python 3.7.7 installation done earlier. It is also necessary to have available a library to run parallel software MPI Message Passing Interface, technology that allows to distribute the calculations between several different servers in a coordinated way. In this case we use the OpenMPI 3.1.2 library compiled with the same compilers used with Python (GCC 8.2.0).

The installation paths of both software tools are defined by executing the following *module load* commands:

```
CALENDULA[ sngular_aia_1_3@cn3007 ~]$ module load python_3.7.7
CALENDULA[ sngular_aia_1_3@cn3007 ~]$ module load haswell/openmpi_3.1.2_gcc8.2.0
```

For the software developed for CAMBrIc to work properly, it is necessary to install additional Python packages that add new functions to the basic version of the Python language interpreter.

```
CALENDULA[ sngular_aia_1_3@cn3007 dev]$ pip3 install -r requirements.txt
CALENDULA[ sngular_aia_1_3@cn3007 dev]$ pip3 install mpi4py
```

The installed packages and versions contained in the archive requirements.txt are:

```
bokeh==1.4.0
Click==7.0
cloudpickle==1.3.0
dask==2.10.1
distributed==2.10.0
```



```
et-xmlfile==1.0.1
fsspec==0.6.2
HeapDict==1.0.1
jdcal==1.4.1
Jinja2==2.11.1
locket==0.2.0
MarkupSafe==1.1.1
msgpack==0.6.2
numpy==1.18.1
openpyxl==3.0.3
packaging==20.1
pandas==1.0.1
partd==1.1.0
Pillow==7.0.0
psutil==5.6.7
pyparsing==2.4.6
python-dateutil==2.8.1
python-i18n==0.3.7
pytz==2019.3
PyYAML==5.3
scipy==1.4.1
six==1.14.0
sortedcontainers==2.1.0
tblib==1.6.0
toolz==0.10.0
tornado==6.0.3
xlrd==1.2.0
zict==1.0.0
```

Finally, the dask-jobqueue software is added, allowing the integration of the software developed in Dask with the SLURM job management software available in Caléndula.

```
CALENDULA[ sngular_aia_1_3@cn3007 dev]$ pip3 install dask-jobqueue –upgrade
```

To check the correct installation of all the software, a test is carried out using a script to send a job to SLURM:

```
#!/bin/bash
# numero de cores que serán reservados
#SBATCH -n 16
# particion en donde se ejecutara el trabajo
#SBATCH -p haswell
# limites que se aplicaran al trabajo
#SBATCH -q normal
# nombre
#SBATCH -J python_sngular
# tiempo maximo de ejecucion (p.e. 2 dias). Maximo permitido: 5 dias
#SBATCH --time=2-00:00:00
# archivos de salida y de error
#SBATCH -o python_sngular_test-%j.o
#SBATCH -e python_sngular_test-%j.e
# directorio de trabajo por defecto
#SBATCH -D .
# notificaciones por email relacionadas con la ejecucion del trabajo
#SBATCH --mail-user=email@usuario
#SBATCH --mail-type=ALL
# carga de las variables necesarias para usar Python 3.7.7
module load python_3.7.7
python src/main.py -s /home/sngular_aia_1/sngular_aia_1_3/dev/simulator/files/scenario_claras.json
```


Annex G: Mono installation

Installation is done directly from the Mono developer repositories by adding the repository configuration:

```
# [root@cn3007 ~]# vim /etc/yum.repos.d/centos7-stable.repo
…
[mono-centos7-stable]
name=mono-centos7-stable
baseurl=https://download.mono-project.com/repo/centos7-stable/
enabled=1
gpgcheck=1
gpgkey=https://download.mono-project.com/repo/xamarin.gpg
…
```

To install, we run the CentOS package installer, yum:

```
# [root@cn3007 ~]# yum install mono-complete
...
Dependencias resueltas
================================================================================
Package          Arquitectura          Versión          Repositorio       Tamaño
================================================================================
Instalando:
mono-complete          x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   3.4 k
Instalando para las dependencias:
giflib                 x86_64        4.1.6-9.el7           base            40 k
ibm-data-db2           x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    39 k
libexif                x86_64        0.6.21-6.el7          base           347 k
libgdiplus-devel       x86_64  6.0.4-0.xamarin.1.epel7     mono-centos7-stable   180 k
libgdiplus0            x86_64  6.0.4-0.xamarin.1.epel7     mono-centos7-stable   527 k
libmono-2_0-1          x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   3.3 k
libmono-2_0-devel      x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    51 k
libmono-llvm0          x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   8.9 M
libmonoboehm-2_0-1     x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    6.3M
                                                                libmonoboehm-2_0-devel    x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-
stable   3.1 k
libmonosgen-2_0-1      x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    6.8M
libmonosgen-2_0-devel  x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    6.3k
mono-core              x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    33 M
mono-data              x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    4.4M
mono-data-oracle       x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    80 k
mono-data-sqlite       x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    67 k
mono-devel             x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    25 M
mono-extras            x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   398 k
mono-llvm-tools  x86_64  6.0+mono20190708165219-0.xamarin.1.epel7  mono-centos7-stable 17M
mono-locale-extras     x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   291 k
                                                    mono-mvc              x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   433 k
mono-reactive          x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   332 k
mono-wcf               x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   973 k
mono-web               x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   2.2 M
mono-winforms          x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   1.5 M
mono-winfxcore         x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable   263 k
monodoc-core           x86_64  6.8.0.96-0.xamarin.3.epel7  mono-centos7-stable    19 M
msbuild                noarch  1:16.5+xamarinxplat.2020.01.10.05.36-0.xamarin.2.epel7                        mono-centos7-stable    10 M
msbuild-libhostfxr     x86_64  3.0.0.2019.04.16.02.13-0.xamarin.4.epel7        mono-centos7-stable    167 k
msbuild-sdkresolver    noarch  1:16.5+xamarinxplat.2020.01.10.05.36-0.xamarin.2.epel7                                 mono-centos7-stable
51 k

Resumen de la transacción
================================================================================
Instalar  1 Paquete (+30 Paquetes dependientes)
...
[root@cn3007 ~]# yum install xsp
...
Dependencias resueltas
```

```
================================================================================
Package     Arquitectura      Versión           Repositorio     Tamaño
================================================================================
Instalando:
xsp         x86_64     4.5-0.xamarin.2.epel7   mono-centos7-stable  315 k

Resumen de la transacción
================================================================================
Instalar  1 Paquete
...
```

As mentioned above, the installation of Mono served as a proof of concept to evaluate one of the available means to quickly adapt the Windows .NET code to run on a Linux system.

Once it was confirmed that it was not the optimal way to perform the adaptation, the software was removed from the calculation nodes where it was installed to eliminate unused software and improve performance.

To uninstall:

```
[root@cn3007 ~]# yum history
Complementos cargados:fastestmirror, langpacks
ID    | Linea de comandos       | Día y hora      | Acción(es)    | Modific
-------------------------------------------------------------------------------
  57 | install xsp             | 2020-01-30 11:56 | Install      |    1
  56 | install mono-complete   | 2020-01-30 10:45 | Install      |   31 EE
  55 | erase ruby ruby-libs    | 2020-01-28 17:08 | Erase        |    9
  54 | erase epel-release      | 2020-01-28 17:00 | Erase        |    1
  53 | erase cpan perl-Tie-IxHa | 2020-01-28 14:07 | Erase       |   42
  52 | install cpan perl-Tie-Ix | 2020-01-28 08:11 | Install     |   40
  51 | install ruby            | 2020-01-28 08:10 | Install      |    9
  50 | install epel-release    | 2020-01-28 08:06 | Install      |    1
  49 | install dotnet-sdk-3.1.1 | 2020-01-16 09:25 | Install     |    5 E<  48 | erase dotnet-host.x86_64 | 2020-01-16 08:49 | Erase        |    4>
  47 | erase aspnetcore-runtime | 2020-01-16 08:46 | Erase       |    1 <  46 | install libserf-devel    | 2020-01-07 17:51 | Install      |    1>
  45 | install libserf         | 2020-01-07 17:46 | Install      |    1
  44 | install utf8proc.x86_64  | 2020-01-07 17:27 | Install     |    2
  43 | install apr-util-devel   | 2020-01-07 17:18 | Install     |    4
  42 | install apr-devel       | 2020-01-07 17:16 | Install      |    1
  41 | install kmod-lustre-clie | 2019-12-11 11:22 | Install     |    3
  40 | erase openmpi-devel ucx  | 2019-12-11 11:21 | Erase       |    3
  39 | erase kmod-lustre-client | 2019-12-11 11:19 | Erase       |    3 EE
  38 | install openmpi-devel.x8 | 2019-12-11 11:14 | Install     |    3
history list
[root@cn3007 ~]# yum history undo 56
[root@cn3007 ~]# yum history undo 57
```

x64.rpm dotnet-targeting-pack-3.1.0-x64.rpm netstandard-targeting-pack-2.1.0-x64.rpm"

Annex H: .NET Core installation

References:

https://dotnet.microsoft.com/download

https://docs.microsoft.com/es-es/dotnet/core/install/linux-package-manager-centos7#install-the-aspnet-core-runtime

In a similar way to the Mono software, the nodes are configured to be able to download and install the necessary programs from a pre-compiled software repository for Linux from Microsoft, developer of the .NET technology.

We add the repo configuration in one of the nodes:

```
[root@cn3007 ~]# vim /etc/yum.repos.d/microsoft-prod.repo
packages-microsoft-com-prod]
name=packages-microsoft-com-prod
baseurl=https://packages.microsoft.com/centos/7/prod
enabled=1
gpgcheck=1
gpgkey=https://packages.microsoft.com/keys/microsoft.asc
sslverify=1
```

We run the following command to get the list of packages needed to install the .NET Core and ASP.NET Core running environment:

```
 [root@cn3007 ~]# yum install aspnetcore-runtime-3.1
...
Dependencias resueltas
================================================================================
Package            Arquitectura   Versión      Repositorio         Tamaño
================================================================================
Instalando:
 aspnetcore-runtime-3.1  x86_64      3.1.1-1 packages-microsoft-com-prod 7.5 M
Instalando para las dependencias:
 dotnet-host           x86_64      3.1.1-1 packages-microsoft-com-prod  39 K
 dotnet-hostfxr-3.1      x86_64      3.1.1-1 packages-microsoft-com-prod 148 k
 dotnet-runtime-3.1      x86_64      3.1.1-1 packages-microsoft-com-prod  29 M
 dotnet-runtime-deps-3.1 x86_64      3.1.1-1 packages-microsoft-com-prod  2.8k
Resumen de la transacción
================================================================================
Instalar  1 Paquete (+4 Paquetes dependientes)
...
```

A problem appeared at the time of installation as there seems to be a problem with the files stored on Microsoft's servers. To install, it is necessary to do so with rpm since CentOS' package manager, yum, does not correctly resolve the dependencies of the downloaded packages. Report that you are waiting for another version.

As an alternative solution we downloaded the packages from the route https://packages.microsoft.com/centos/7/prod and store them in a local route accessible from all the servers of the supercomputer (/home/software/net_core).

```
[root@cn3007 ~]# cd /home/software/net_core/
[root@cn3007 net_core]# ll
total 37236
-rw-r--r-- 1 root root  7852785 ene 14 20:40 aspnetcore-runtime-3.1.1-x64.rpm
-rw-r--r-- 1 root root    40065 ene 14 20:40 dotnet-host-3.1.1-x64.rpm
-rw-r--r-- 1 root root   151817 ene 14 20:40 dotnet-hostfxr-3.1.1-x64.rpm
-rw-r--r-- 1 root root 30066054 ene 14 20:40 dotnet-runtime-3.1.1-x64.rpm
-rw-r--r-- 1 root root     2857 ene 15 00:19 dotnet-runtime-deps-3.1.1-centos.7-x64.rpm
-rw-r--r-- 1 root root      206 ene 15 13:35 microsoft-prod.repo
```



```
[root@cn3007 net_core]# rpm -Uvh aspnetcore-runtime-3.1.1-x64.rpm dotnet-host-3.1.1-x64.rpm dotnet-hostfxr-3.1.1-x64.rpm dotnet-runtime-3.1.1-x64.rpm dotnet-runtime-deps-3.1.1-centos.7-x64.rpm
advertencia:aspnetcore-runtime-3.1.1-x64.rpm: EncabezadoV4 RSA/SHA256 Signature, ID de clave be1229cf: NOKEY
Preparando...                          ################################# [100%]
Actualizando / instalando...
   1:dotnet-runtime-deps-3.1-3.1.1-1  ################################# [ 20%]
   2:dotnet-host-3.1.1-1              ################################# [ 40%]
   3:dotnet-hostfxr-3.1-3.1.1-1       ################################# [ 60%]
   4:dotnet-runtime-3.1-3.1.1-1       ################################# [ 80%]
   5:aspnetcore-runtime-3.1-3.1.1-1   ################################# [100%]
```

To leave the installation ready for future needs in the development and improvement of the pilot, the .NET SDK Software Development Kit was also installed. The SDK provides the libraries and development environment needed to generate new code or add new functions.

```
[root@cn3007 ~]# yum install dotnet-sdk-3.1
...
Dependencias resueltas

================================================================================
 Package                Arquitectura   Versión      Repositorio         Tamaño
================================================================================
Instalando:
 dotnet-sdk-3.1         x86_64   3.1.101-1  packages-microsoft-com-prod  63 M
Instalando para las dependencias:
 aspnetcore-targeting-pack-3.1    x86_64   3.1.0-1  packages-microsoft-com-prod   1.4 M
 dotnet-apphost-pack-3.1          x86_64   3.1.1-1  packages-microsoft-com-prod    67 k
 dotnet-targeting-pack-3.1        x86_64   3.1.0-1  packages-microsoft-com-prod   3.4 M
 netstandard-targeting-pack-2.1   x86_64   2.1.0-1  packages-microsoft-com-prod   2.1 M

Resumen de la transacción
================================================================================
Instalar  1 Paquete (+4 Paquetes dependientes)
...
```

We downloaded the necessary packages in the same location /home/software/net_core.

As an optimization and security measure, we removed the installed repo from the node:

```
[root@cn3007 ~]# rm /etc/yum.repos.d/microsoft-prod.repo
rm: ¿borrar el fichero regular «/etc/yum.repos.d/microsoft-prod.repo»? (s/n) s
```

The installation was done with the command:

```
[root@cn3007 net_core]# yum install dotnet-sdk-3.1.101-x64.rpm aspnetcore-targeting-pack-3.1.0.rpm dotnet-apphost-pack-3.1.1-x64.rpm dotnet-targeting-pack-3.1.0-x64.rpm netstandard-targeting-pack-2.1.0-x64.rpm
...
Instalado:
  aspnetcore-targeting-pack-3.1.x86_64 0:3.1.0-1          dotnet-apphost-pack-3.1.x86_64 0:3.1.1-1
  dotnet-sdk-3.1.x86_64 0:3.1.101-1                       dotnet-targeting-pack-3.1.x86_64 0:3.1.0-1
  netstandard-targeting-pack-2.1.x86_64 0:2.1.0-1
¡Listo!
```

The execution of the programs on the supercomputer on the different servers is decided on the basis of the system load state and it is not possible to know in advance on which servers the jobs will be executed. Therefore, it has been necessary to replicate the installation of all the software used on all the servers of the supercomputer.

To do this, it is usual in HPC environments to use a tool called *pdsh* that allows a command to be executed remotely on multiple servers simultaneously. In the following lines, the installation process is executed in the different architectures or groups of servers that make up the supercomputer.

```
[root@manager1 ~]# pdsh -w cn3[008-114] "cd /home/software/net_core; yum install -y aspnetcore-runtime-3.1.1-x64.rpm aspnetcore-targeting-pack-3.1.0.rpm  dotnet-apphost-pack-3.1.1-x64.rpm  dotnet-host-3.1.1-x64.rpm  dotnet-hostfxr-3.1.1-
```

x64.rpm dotnet-runtime-3.1.1-x64.rpm dotnet-runtime-deps-3.1.1-centos.7-x64.rpm dotnet-sdk-3.1.101-x64.rpm dotnet-targeting-pack-3.1.0-x64.rpm netstandard-targeting-pack-2.1.0-x64.rpm"

[root@manager1 ~]# pdsh -w cn1[001-186] "cd /home/software/net_core; yum install -y aspnetcore-runtime-3.1.1-x64.rpm aspnetcore-targeting-pack-3.1.0.rpm dotnet-apphost-pack-3.1.1-x64.rpm dotnet-host-3.1.1-x64.rpm dotnet-hostfxr-3.1.1-x64.rpm dotnet-runtime-3.1.1-x64.rpm dotnet-runtime-deps-3.1.1-centos.7-x64.rpm dotnet-sdk-3.1.101-x64.rpm dotnet-targeting-pack-3.1.0-x64.rpm netstandard-targeting-pack-2.1.0-x64.rpm"

[root@manager1 ~]# pdsh -w cn2[001-006] "cd /home/software/net_core; yum install -y aspnetcore-runtime-3.1.1-x64.rpm aspnetcore-targeting-pack-3.1.0.rpm dotnet-apphost-pack-3.1.1-x64.rpm dotnet-host-3.1.1-x64.rpm dotnet-hostfxr-3.1.1-x64.rpm dotnet-runtime-3.1.1-x64.rpm dotnet-runtime-deps-3.1.1-centos.7-x64.rpm dotnet-sdk-3.1.101-x64.rpm dotnet-targeting-pack-3.1.0-x64.rpm netstandard-targeting-pack-2.1.0-x64.rpm"

[root@manager1 ~]# pdsh -w cn4001 "cd /home/software/net_core; yum install -y aspnetcore-runtime-3.1.1-x64.rpm aspnetcore-targeting-pack-3.1.0.rpm dotnet-apphost-pack-3.1.1-x64.rpm dotnet-host-3.1.1-x64.rpm dotnet-hostfxr-3.1.1-x64.rpm dotnet-runtime-3.1.1-x64.rpm dotnet-runtime-deps-3.1.1-centos.7-x64.rpm dotnet-sdk-3.1.101- Hardware

Annex I: Virtuoso installation

Virtuoso installation process on the virtual machine and Operating system installation.

The first thing that had been done in these virtual machines has been the installation of the basic system of the operating system, CentOS 7.7.

Type of installation:

Gnome (Gnome Applications and Development Tools)

Security Policy: Disable "Apply security policy"

Kdump: *disable*

Hostname: *prj-cf-virtuoso*

Partitions: we partition the disk into three partitions: boot, root and swap. (These parameters may vary according to the needs of each machine).

Machine Configuration

Once the operating system has been installed, the machine is configured so that the installation of the virtuoso can then be carried out.

From inside the machine a *yum update* is done to update.

```
# [root@prj-cf-virtuoso ~]# yum update
```

The required EPEL (Extra Packages for Enterprise Linux) repository is installed. This repository provides an installation point from which to install multiple software packages that are not available from the original CentOS repositories.

```
[root@prj-cf-virtuoso ~]# rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

Some necessary packages are installed.

```
[root@prj-cf-virtuoso ~]# yum install gcc kernel-devel kernel-headers dkms make bzip2 perl
```

SELINUX and iptables are disabled in order to access the machine.

```
# [root@prj-cf-virtuoso ~]# vim /etc/selinux/config
SELINUX=disabled
# [root@prj-cf-virtuoso ~]# iptables -L
# [root@prj-cf-virtuoso ~]# systemctl stop firewalld
# [root@prj-cf-virtuoso ~]# systemctl disable firewalld
# [root@prj-cf-virtuoso ~]# systemctl status firewalld
```

The machine is restarted to check that all changes have been made correctly.

```
# [root@prj-cf-virtuoso ~]# reboot
# [root@prj-cf-virtuoso ~]# sestatus
SELinux status:                 disabled
```

Virtuoso installation

To generate the configuration script and all other necessary build files, you need to have some packages installed.

Virtuoso dependencies (autoconf, automake, libtool, flex, bison, gperf, gawk, m4, make, OpenSSL) and other necessary ones are installed.


```
# [root@prj-cf-virtuoso ~]# yum install autoconf automake libtool flex bison gperf gawk m4 make openssl-devel readline-devel wget git gcc gmake
```

The Virtuoso repository at https://github.com/openlink/virtuoso-opensource is cloned and you can see the files that have been created.

```
# [root@prj-cf-virtuoso~]#git clone https://github.com/openlink/virtuoso-opensource.git
# [root@prj-cf-virtuoso ~]# cd virtuoso-opensource/
# [root@prj-cf-virtuoso virtuoso-opensource]# ll
total 532
drwxr-xr-x. 15 root root   4096 nov 22 12:26 appsrc
-rw-r--r--.  1 root root    201 nov 22 12:26 AUTHORS
-rwxr-xr-x.  1 root root   3614 nov 22 12:26 autogen.sh
drwxr-xr-x.  3 root root     76 nov 22 12:26 bin
drwxr-xr-x. 42 root root   4096 nov 22 12:26 binsrc
-rw-r--r--.  1 root root 212256 nov 22 12:26 ChangeLog
-rw-r--r--.  1 root root  90693 nov 22 12:26 configure.ac
-rw-r--r--.  1 root root  18092 nov 22 12:26 COPYING
-rw-r--r--.  1 root root  17881 nov 22 12:26 COPYING.md
lrwxrwxrwx.  1 root root     10 nov 22 12:26 CREDITS -> CREDITS.md
-rw-r--r--.  1 root root   6613 nov 22 12:26 CREDITS.md
drwxr-xr-x.  3 root root   4096 nov 22 12:26 debian
drwxr-xr-x. 11 root root    224 nov 22 12:26 docsrc
lrwxrwxrwx.  1 root root     10 nov 22 12:26 INSTALL -> INSTALL.md
-rw-r--r--.  1 root root   9478 nov 22 12:26 INSTALL.md
drwxr-xr-x. 13 root root    207 nov 22 12:26 libsrc
lrwxrwxrwx.  1 root root     10 nov 22 12:26 LICENSE -> LICENSE.md
-rw-r--r--.  1 root root   1484 nov 22 12:26 LICENSE.md
-rw-r--r--.  1 root root   5333 nov 22 12:26 Makefile.am
lrwxrwxrwx.  1 root root      7 nov 22 12:26 NEWS -> NEWS.md
-rw-r--r--.  1 root root  31433 nov 22 12:26 NEWS.md
-rw-r--r--.  1 root root  14149 nov 22 12:26 README
-rw-r--r--.  1 root root  11328 nov 22 12:26 README.GeoSPARQL.md
-rw-r--r--.  1 root root   5576 nov 22 12:26 README.GIT.md
-rw-r--r--.  1 root root    511 nov 22 12:26 README.hibernate.md
-rw-r--r--.  1 root root    871 nov 22 12:26 README.jena.md
-rw-r--r--.  1 root root    504 nov 22 12:26 README.jsse.md
-rw-r--r--.  1 root root   6345 nov 22 12:26 README.MACOSX.md
-rw-r--r--.  1 root root   4427 nov 22 12:26 README.OpenSSL.md
-rw-r--r--.  1 root root   7373 nov 22 12:26 README.php5.md
-rw-r--r--.  1 root root    602 nov 22 12:26 README.sesame2.md
-rw-r--r--.  1 root root    607 nov 22 12:26 README.sesame3.md
-rw-r--r--.  1 root root   9969 nov 22 12:26 README.UPGRADE.md
-rw-r--r--.  1 root root  10720 nov 22 12:26 README.WINDOWS.md
```

It is run autogen.sh and passed to the configure as an "opt" installation directory.

```
# [root@prj-cf-virtuoso virtuoso-opensource]# ./autogen.sh
# [root@prj-cf-virtuoso virtuoso-opensource]# ./configure --with-layout=opt -with-readline
```

A check of the program is compiled and executed before installation to verify that the compilation process has been performed without errors.

```
# [root@prj-cf-virtuoso virtuoso-opensource]# make -j4
# [root@prj-cf-virtuoso virtuoso-opensource]# make check
```

Virtuoso is installed.

```
# [root@prj-cf-virtuoso virtuoso-opensource]# make install
# [root@prj-cf-virtuoso ~]# cd /opt/virtuoso-opensource/
# [root@prj-cf-virtuoso virtuoso-opensource]# ll
total 12
drwxr-xr-x.  2 root root   81 nov 22 12:56 bin
drwxr-xr-x.  2 root root   26 nov 22 12:56 database
drwxr-xr-x.  2 root root 4096 nov 22 12:56 doc
```



```
drwxr-xr-x.  2 root root  239 nov 22 12:57 hosting
drwxr-xr-x. 11 root root 4096 nov 22 12:56 lib
drwxr-xr-x.  2 root root   54 nov 22 12:56 vad
drwxr-xr-x.  5 root root 4096 nov 22 12:56 vsp
# [root@prj-cf-virtuoso virtuoso-opensource]# cd database/
# [root@prj-cf-virtuoso database]# /opt/virtuoso-opensource/bin/virtuoso-t
```

Create the user or users who will use the machine, also the group they will belong to and add the user or users to that group.

```
# [root@prj-cf-virtuoso ~]# adduser xxxx
# [root@prj-cf-virtuoso ~]# passwd xxxx
# [root@prj-cf-virtuoso ~]# groupadd yyyy
# [root@prj-cf-virtuoso ~]# gpasswd -a xxxx yyyy
Añadiendo al usuario xxxx al grupo yyyy
```

The owner of the Virtuoso is changed to be the user who has already created its owner.

```
# [root@prj-cf-virtuoso ~]# cd /opt
# [root@prj-cf-virtuoso opt]# ll
total 0
drwxr-xr-x. 2 root root  6 oct 30  2018 rh
drwxr-xr-x. 9 root root 92 nov 22 12:56 virtuoso-opensource
# [root@prj-cf-virtuoso opt]# chown -R xxxx:yyyy virtuoso-opensource/
# [root@prj-cf-virtuoso opt]# ll
total 0
drwxr-xr-x. 2 root         root         6 oct 30  2018 rh
drwxr-xr-x. 9 xxxx yyyy 92 nov 22 12:56 virtuoso-opensource
```

A Systemd service is created so that Virtuoso runs automatically every time the machine is started and we don't need to run it again.

```
# [root@prj-cf-virtuoso ~]# cd /usr/local/bin
# [root@prj-cf-virtuoso bin]# vim virtuoso
 1 #!/bin/bash
 2
 3 cd /opt/virtuoso-opensource/database/
 4 /opt/virtuoso-opensource/bin/virtuoso-t
# [root@prj-cf-virtuoso bin]# chmod 755 virtuoso
# [root@prj-cf-virtuoso bin]# ll
total 4
-rwxr-xr-x 1 root root 92 nov 25 10:12 virtuoso
# [root@prj-cf-virtuoso bin]# cd /etc/systemd/system/
# [root@prj-cf-virtuoso system]# vim virtuoso.service
 1 [Unit]
 2 Description=Virtuoso
 3 After=networking.target
 4
 5 [Service]
 6 Type=oneshot
 7 ExecStart=/usr/local/bin/virtuoso
 8 RemainAfterExit=yes
 9
10 [Install]
11 WantedBy=multi-user.target
# [root@prj-cf-virtuoso system]# chmod 755 virtuoso.service
# [root@prj-cf-virtuoso system]# ll
...
-rwxr-xr-x  1 root root  179 nov 25 10:15 virtuoso.service
...
# [root@prj-cf-virtuoso system]# systemctl daemon-reload
# [root@prj-cf-virtuoso system]# systemctl start virtuoso.service
# [root@prj-cf-virtuoso system]# systemctl enable virtuoso.service
```



```
Created symlink from /etc/systemd/system/multi-user.target.wants/virtuoso.service to
/etc/systemd/system/virtuoso.service.
# [root@prj-cf-virtuoso system]# systemctl status virtuoso.service
● virtuoso.service - Virtuoso
   Loaded: loaded (/etc/systemd/system/virtuoso.service; enabled; vendor preset: disabled)
   Active: active (exited) since lun 2019-11-25 12:55:52 CET; 53s ago
  Process: 1006 ExecStart=/usr/local/bin/virtuoso (code=exited, status=0/SUCCESS)
 Main PID: 1006 (code=exited, status=0/SUCCESS)
    Tasks: 18
   CGroup: /system.slice/virtuoso.service
           └─1056 /opt/virtuoso-opensource/bin/virtuoso-t
nov 25 12:55:50 prj-cf-virtuoso systemd[1]: Starting Virtuoso...
nov 25 12:55:52 prj-cf-virtuoso systemd[1]: Started Virtuoso.
```


Annex J: Disks preparation

You do *fdisk -l* to see the disk partitioning.

[root@prj-cf-virtuoso~]# fdisk –l

```
[root@prj-cf-virtuoso ~]# fdisk -l

Disk /dev/sdb: 1099.5 GB, 1099511627776 bytes, 2147483648 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/sda: 2040.1 GB, 2040109465600 bytes, 3984588800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Identificador del disco: 0x0009152d

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200   209715199   103808000   8e  Linux LVM

Disk /dev/mapper/VG00-LV--root: 71.9 GB, 71936507904 bytes, 140500992 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/VG00-LV--swap: 34.4 GB, 34359738368 bytes, 67108864 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Figure 24 – Add an extension to an existing disk storage code

The following package needs to be installed in order to make the extension.

[root@prj-cf-virtuoso~]# yum install cloud-utils-growpart

Extend the partition that has been increased.

[root@prj-cf-virtuoso~]# growpart /dev/sda2

```
[root@prj-cf-virtuoso ~]# growpart /dev/sda 2
CHANGED: partition=2 start=2099200 old: size=207616000 end=209715200 new: size=3982489567 end=3984588767
```

Figure 25 – Extend the partition in a disk code

See that the size has changed.

[root@prj-cf-virtuoso~]# pvresize /dev/sda2

```
[root@prj-cf-virtuoso ~]# pvresize /dev/sda2
  Physical volume "/dev/sda2" changed
  1 physical volume(s) resized or updated / 0 physical volume(s) not resized
```

Figure 26 – A disk size change code

Extend the size of the logical volume.

[root@prj-cf-virtuoso~]# lvextend -L+(extends) /dev/mapper/VG00-LV—root


```
[root@prj-cf-virtuoso ~]# lvextend -L+2040.1G /dev/mapper/VG00-LV--root
  Rounding size to boundary between physical extents: 1,99 TiB.
  Insufficient free space: 522266 extents needed, but only 460800 available
[root@prj-cf-virtuoso ~]# lvextend -l+460800 /dev/mapper/VG00-LV--root
  Size of logical volume VG00/LV-root changed from <67,00 GiB (17151 extents) to 1,82 TiB (477951 extents).
  Logical volume VG00/LV-root successfully resized.
```

Figure 27 – Extend the size of the logical volume code

Resize the file system by mounting the logical volume. (must be ext2, ext3 or ext4)

[root@prj-cf-virtuoso~]# resize2fs /dev/mapper/VG00-LV—root

```
[root@prj-cf-virtuoso ~]# resize2fs /dev/mapper/VG00-LV--root
resize2fs 1.42.9 (28-Dec-2013)
resize2fs: Bad magic number in super-block mientras se intentaba abrir /dev/mapper/VG00-LV--root
No se pudo encontrar un superbloque válido para el sistema de ficheros.
```

Figure 28 – Resize the file system code

LVM is using xfs as its file system. So, instead of using the command resize2fs, use the command xfs_growfs.

[root@prj-cf-virtuoso~]# xfs_growfs /dev/mapper/VG00-LV—root
[root@prj-cf-virtuoso~]# fdisk –l

```
[root@prj-cf-virtuoso ~]# xfs_growfs /dev/mapper/VG00-LV--root
meta-data=/dev/mapper/VG00-LV--root isize=512    agcount=4, agsize=4390656 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0 spinodes=0
data     =                       bsize=4096   blocks=17562624, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal               bsize=4096   blocks=8575, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 17562624 to 489421824
```

Figure 29 – LVM using xfs sample code

```
[root@prj-cf-virtuoso ~]# fdisk -l

Disk /dev/sdb: 1099.5 GB, 1099511627776 bytes, 2147483648 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/sda: 2040.1 GB, 2040109465600 bytes, 3984588800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Identificador del disco: 0x0009152d

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200  3984588766  1991244783+  8e  Linux LVM

Disk /dev/mapper/VG00-LV--root: 2004.7 GB, 2004671791104 bytes, 3915374592 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/VG00-LV--swap: 34.4 GB, 34359738368 bytes, 67108864 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Figure 30 – LVM using xfs sample code

Add a new disk to LVM, a fdisk is made to view the disk partitioning.

[root@prj-cf-virtuoso~]# fdisk –l

```
[root@prj-cf-virtuoso ~]# fdisk -l

Disk /dev/sdb: 1099.5 GB, 1099511627776 bytes, 2147483648 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/sda: 2040.1 GB, 2040109465600 bytes, 3984588800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Identificador del disco: 0x0009152d

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200  3984588766  1991244783+  8e  Linux LVM

Disk /dev/mapper/VG00-LV--root: 2004.7 GB, 2004671791104 bytes, 3915374592 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/VG00-LV--swap: 34.4 GB, 34359738368 bytes, 67108864 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Figure 31 – Add a new disk to LVM code

The new partition is created with the following indications.

[root@prj-cf-virtuoso~]# fdisk /dev/sdb

Commands are entered in the given order to create a new primary partition that uses 100% of the new hard drive and is ready for LVM.

n = create new partitions

p = create primary partition

1 = partition the disk

Press Enter twice to accept the default value of first sector and last sector.

To prepare the partition to be used by LVM

t = change of partition time

8e = changes in LVM partition time

Verify and write the information to the hard drive.

p = configuration partition view so we can review before writing changes to the disk

w= Write the changes to the disk.

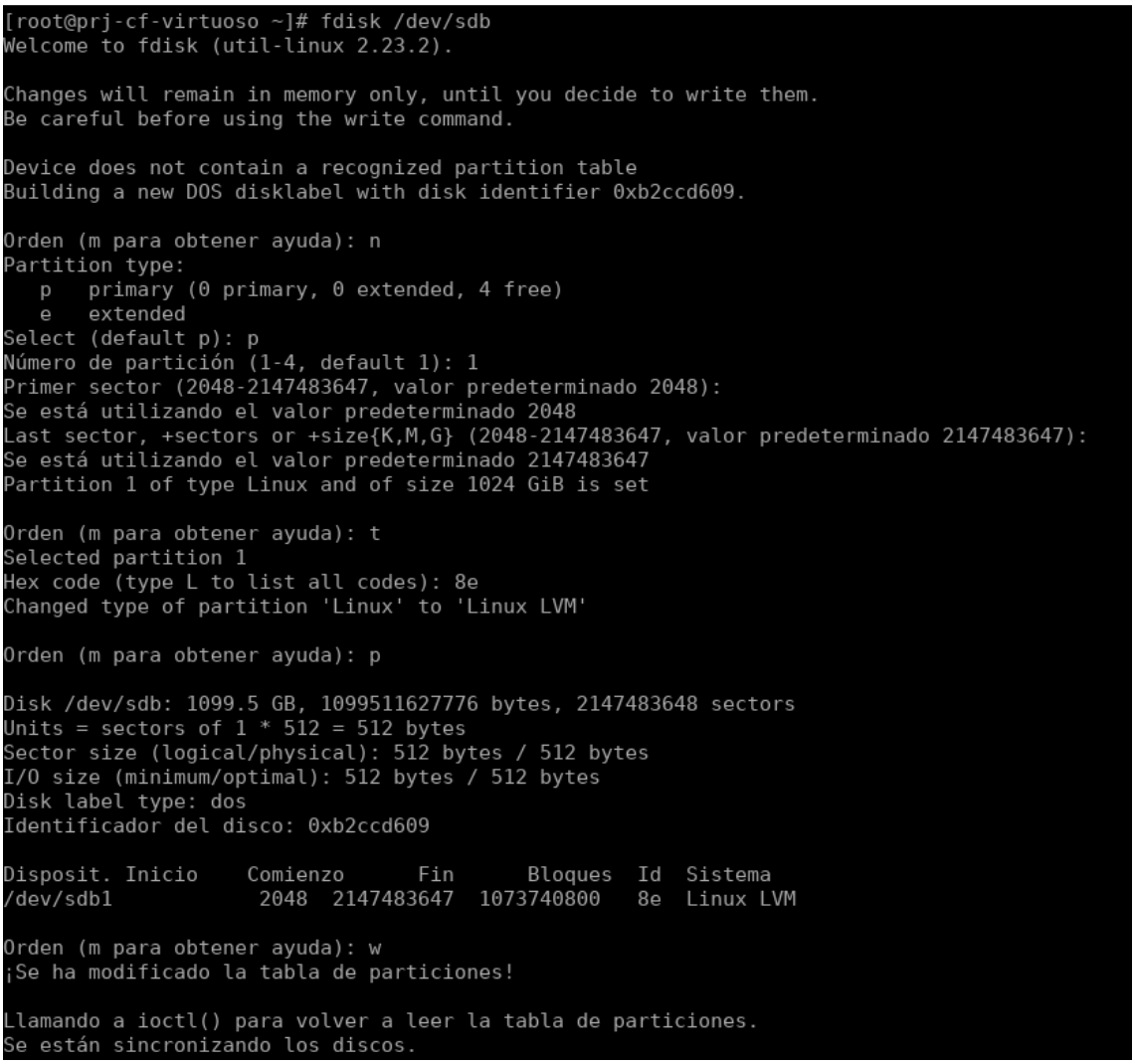

Figure 32 – Verify and write the information to the hard drive code

The new /dev/sdb1 partition has been created.

[root@prj-cf-virtuoso~]# fdisk –l

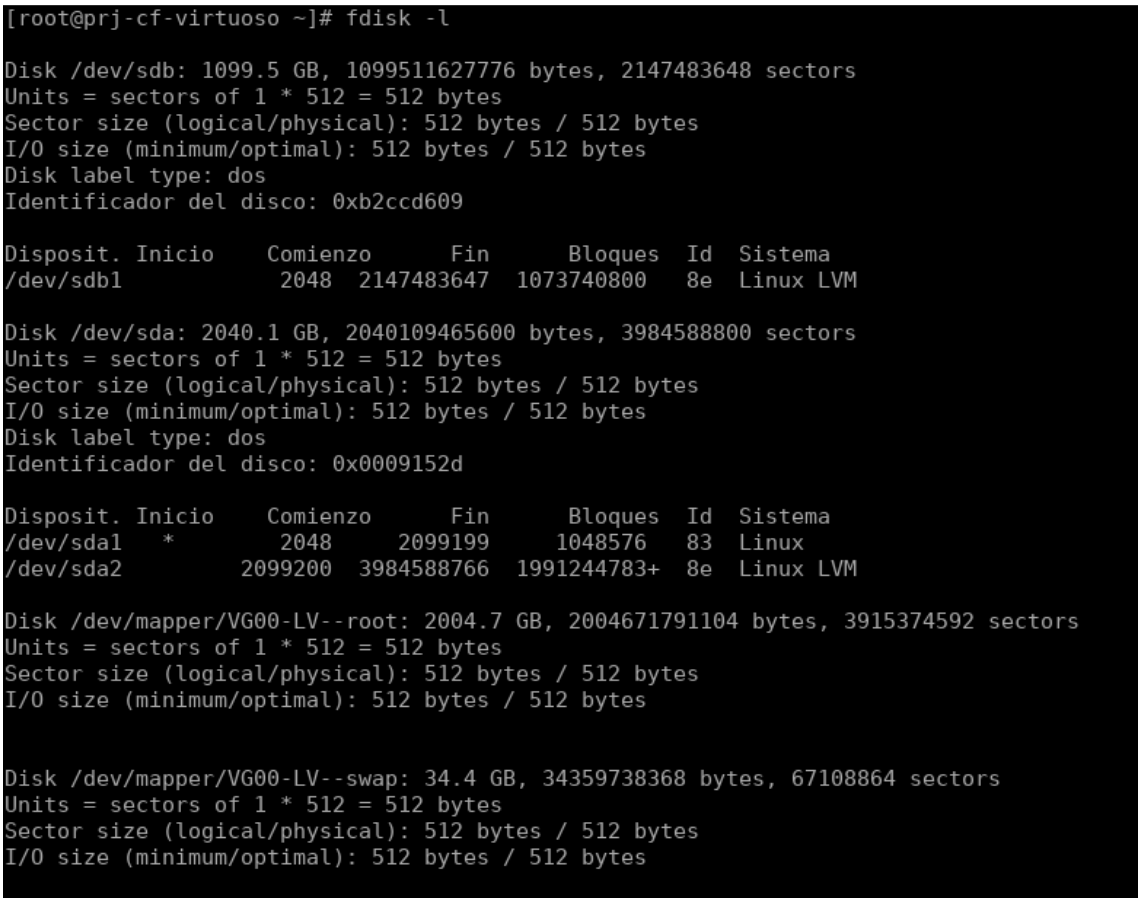

Figure 33 – Creating new disk partition has been created

Create a physical LVM volume on the newly created partition.

[root@prj-cf-virtuoso~]# pvcreate /dev/sdb1

Figure 34 – Creating a physical LVM volume

/dev/sdb1 is a new physical volume.

[root@prj-cf-virtuoso~]# pvdisplay

```
[root@prj-cf-virtuoso ~]# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sda2
  VG Name               VG00
  PV Size               1,85 TiB / not usable 1,98 MiB
  Allocatable           yes (but full)
  PE Size               4,00 MiB
  Total PE              486143
  Free PE               0
  Allocated PE          486143
  PV UUID               Moog8Q-JgPY-xarf-9Xhl-914Z-T8Do-O5OfXd

  "/dev/sdb1" is a new physical volume of "<1024,00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb1
  VG Name
  PV Size               <1024,00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               p0kmEh-ml32-x8za-2ijh-T9Es-7nNN-wabwkW
```

Figure 35 – Showing new physical volume

Vgextend, adds another physical volume to the volume group.

[root@prj-cf-virtuoso~]# Vgextend VG00 /dev/sdb1

```
[root@prj-cf-virtuoso ~]# vgextend VG00 /dev/sdb1
  Volume group "VG00" successfully extended
[root@prj-cf-virtuoso ~]# vgdisplay
  --- Volume group ---
  VG Name               VG00
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  6
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                2
  Open LV               2
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               2,85 TiB
  PE Size               4,00 MiB
  Total PE              748286
  Alloc PE / Size       486143 / 1,85 TiB
  Free  PE / Size       262143 / <1024,00 GiB
  VG UUID               LPXuNa-6qrd-vHld-Nprc-SXmH-14Tz-tfS6cd
```

Figure 36 – Adding another physical volume

All that remains is to expand the logical volume or lv. LVM is using xfs as its file system. Xfs comes with its own set of commands. So, instead of using the command resize2fs, use the command *xfs_growfs.*

[root@prj-cf-virtuoso~]# lvextend –L+1099.5G /dev/mapper/VG00-LV--root

```
[root@prj-cf-virtuoso ~]# lvextend -L+1099.5G /dev/mapper/VG00-LV--root
  Insufficient free space: 281472 extents needed, but only 262143 available
[root@prj-cf-virtuoso ~]# lvextend -l+262143 /dev/mapper/VG00-LV--root
  Size of logical volume VG00/LV-root changed from 1,82 TiB (477951 extents) to 2,82 TiB (740094 extents).
  Logical volume VG00/LV-root successfully resized.
[root@prj-cf-virtuoso ~]# xfs_growfs /dev/mapper/VG00-LV--root
meta-data=/dev/mapper/VG00-LV--root isize=512    agcount=112, agsize=4390656 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0 spinodes=0
data     =                       bsize=4096   blocks=489421824, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal               bsize=4096   blocks=8575, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 489421824 to 757856256
```

Figure 37 – Command xfs_growfs code

It is proven how the increase in storage has been.

[root@prj-cf-virtuoso~]# fdisk –l

```
[root@prj-cf-virtuoso ~]# fdisk -l

Disk /dev/sdb: 1099.5 GB, 1099511627776 bytes, 2147483648 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Identificador del disco: 0xb2ccd609

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/sdb1            2048  2147483647  1073740800   8e  Linux LVM

Disk /dev/sda: 2040.1 GB, 2040109465600 bytes, 3984588800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Identificador del disco: 0x0009152d

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200  3984588766  1991244783+  8e  Linux LVM

Disk /dev/mapper/VG00-LV--root: 3104.2 GB, 3104179224576 bytes, 6062850048 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/VG00-LV--swap: 34.4 GB, 34359738368 bytes, 67108864 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Figure 38 – Disk storage modification process code

Annex K: Dell Equalogic Specs

Features	PS6110E/PS6110X	PS6110S/PS6110XS	PS6110XV/PS6110XV 3.5”
Product configurations	Capacity for your data-hungry applications	High performance for your demanding enterprise applications	High-performance storage optimized for your most critical applications
Storage controllers		Dual controllers with 4GB1 non-volatile memory per controller	
Network interfaces		Management network: One 100BASE-TX per controller Interface ports: One 10GBASE-T with RJ45 and one 10GbE SFP+ for fibre or twin-ax copper cabling	
Hard disk drives	PS6110E: Twenty-four (24) hot-pluggable NL-SAS drives PS6110X: Twenty-four (24) hot-pluggable SAS drives	PS6110S: Twenty-four (24) hot-pluggable SSDs PS6110XS: Seven (7) hot-pluggable SSDs and seventeen (17) hot-pluggable SAS drives	PS6110XV/PS6110XV 3.5”: Twenty-four (24) 15K SAS hot-pluggable drives
Drive capacities	PS6110E: 3.5” 7.2K NL-SAS drives available in 1TB†, 2TB 3TB, or 4TB (3TB and 4TB available in NL-SAS SED) PS6110X: 2.5” 10K SAS drives available in 600GB, 900GB, and 1.2TB (900GB also available in SAS SED)	PS6110S: 2.5” SSDs available in 400GB and 800GB PS6110XS: Combines 2.5” 400GB and 800GB SSDs and 2.5” 600GB and 1.2TB 10K SAS drives	PS6110XV: 2.5” 15K SAS drives available in 300GB (300GB also available in SAS SED) PS6110XV 3.5”: 3.5” 15K SAS disk drives available in 600GB (600GB also available in SAS SED)
System capacities	PS6110E: Up to 96TB PS6110X: Up to 28.8TB	PS6110S: Up to 19.2TB PS6110XS: Up to 26TB	PS6110XV: Up to 7.2TB PS6110XV 3.5”: Up to 14.4TB
Self-Encrypting Drives (SED)	PS6110E: 96TB using twenty-four (24) 4TB NL-SAS drives PS6110X: 21.6TB using twenty-four (24) 900GB 10K SAS drives	N/A	PS6110XV: 7.2TB using twenty-four (24) 300GB 15K SAS drives PS6110XV 3.5”: 14.4TB using twenty-four (24) 600GB 15K 3.5” SAS drives

