Visual Analysis for Extremely Large-Scale Scientific Computing

D2.4 – Design a petabyte sized engineering data solution in the HPC cloud infrastructure

Version 2.0

Deliverable Information

Grant Agreement no	619439
Web Site	http://www.velassco.eu/
Related WP & Task:	WP2, T2.4
Due date	March 31, 2015
Dissemination Level	
Nature	
Author/s	Benoit Lange, Toàn Nguyên
Contributors	Alvaro Janda, Andreas Dietrich, Miguel Tinte, Jochen Haenisch, Miguel Pasenau

Approvals

	Name	Institution	Date	OK
Author	Benoit Lange, Toàn Nguyên	**INRIA**	20/03/2015	
Task Leader	Jochen Haenisch	**JOTNE**		
WP Leader	Toàn Nguyên	**INRIA**		
Coordinator				

Change Log

Version	Description of Change
Version 0.1	TOC decomposition of the document
Version 1.0	First Release of the document
Version 2.0	Add of the partners contribution
Version 2.1	Review with minor corrections by Abel

Table of Contents

1. Introduction

After reviewing the *state of the art of Big Data* in deliverable *D2.1*, with their advantages and requirements, the architecture of the VELaSSCo platform has been specified and designed in deliverable *D2.2*, which integrates the data analytics needed by the scientific community to post-process, i.e. analyze and visualize, the results of the simulations that has been run on HPC clusters, with two big data frameworks.

The proposed solution is compatible with the requirements and restrictions of HPC clusters, which in summary include the no affectation to other user's processes, a global storage system, efficient network communication and a global resource manager. Only minor and acceptable changes has to be made on the HPC eco-system for an efficient operation of the VELaSSCo platform such as the incorporation and usage of local storage on some computation nodes and the submission of the VELaSSCo queries (jobs) to these nodes.

As reviewed in deliverable *D2.3 – HPC cloud infrastructure specification document suitable to the needs of e-Science*, the HPC infrastructure brings specific requirements to allow high efficient distributed memory calculations. A high initial investment is needed for a rigid computing capacity and expansions to support high load demands are also costly. Using HPC cloud infrastructure on the other side, allows for a smooth start in the high performance computing world for SME's and, as the calculation requirements increases over time, more computing nodes can be added to the HPC-cloud without much ado.

Taking into account the special characteristics of the HPC-cloud, such as CPU and storage flexibility and scalability [25], great care has to be taken in order to achieve a cloud-cluster configuration with a good network performance [26, 27, 28], as the efficiency of the designed platform depends on it.

Although the VELaSSCo architecture has been designed with the HPC ecosystem in mind, the flexibility of the adopted Big Data framework expands the possible uses of the designed platform by embracing cloud services.

This document shows that the platform can be easily adapted for its deployment in such environments. The following sections explain in detail how the different layers and modules of the designed architecture should be modified for a successful porting to HPC-cloud. Also the adjustments of the adopted tools are summed up for the ingestion and management of the petabyte-size simulation data into the platform under the cloud requirements. Finally special considerations are listed for the deployment of the platform on a specific HPC-cloud system.

2. Adaptation of VELaSSCo architecture to be used in Cloud

As stated in the VELaSSCo description of work (DOW) and in the previous deliverables, we target to provide a suitable architecture for HPC facilities. This choice was led by the existing computing facilities owned by the scientific community. But these IT systems are not particularly suitable for the requirements of Big Data.

Big Data is characterized by the 4V rules: Volume, Velocity, Variety and Veracity. In most HPC systems, the storage is performed by a networked storage system, specially designed to support high bandwidth transfers. These storage solutions are based on specific file systems like: Lustre, GPFS, PanFS, etc. But this strategy is not the most efficient when it is necessary to move computations next to the data. In a traditional HPC system, data is moved next to the computation units. Moreover, HPC facilities have some restrictions: for example a job scheduler is in charge of the management of the execution workflow. To fit with all the requirements of our project, it is necessary to move into a more flexible infrastructure, for example a cloud. The high flexibility of our architecture is defined by the Lambda architecture paradigm [18]. This paradigm has already been presented in D2.2. The current design of our platform is shown in Figure 1. This solution is very similar to the solution proposed by the NIST (National Institute of Standards and Technology) in the report [21].

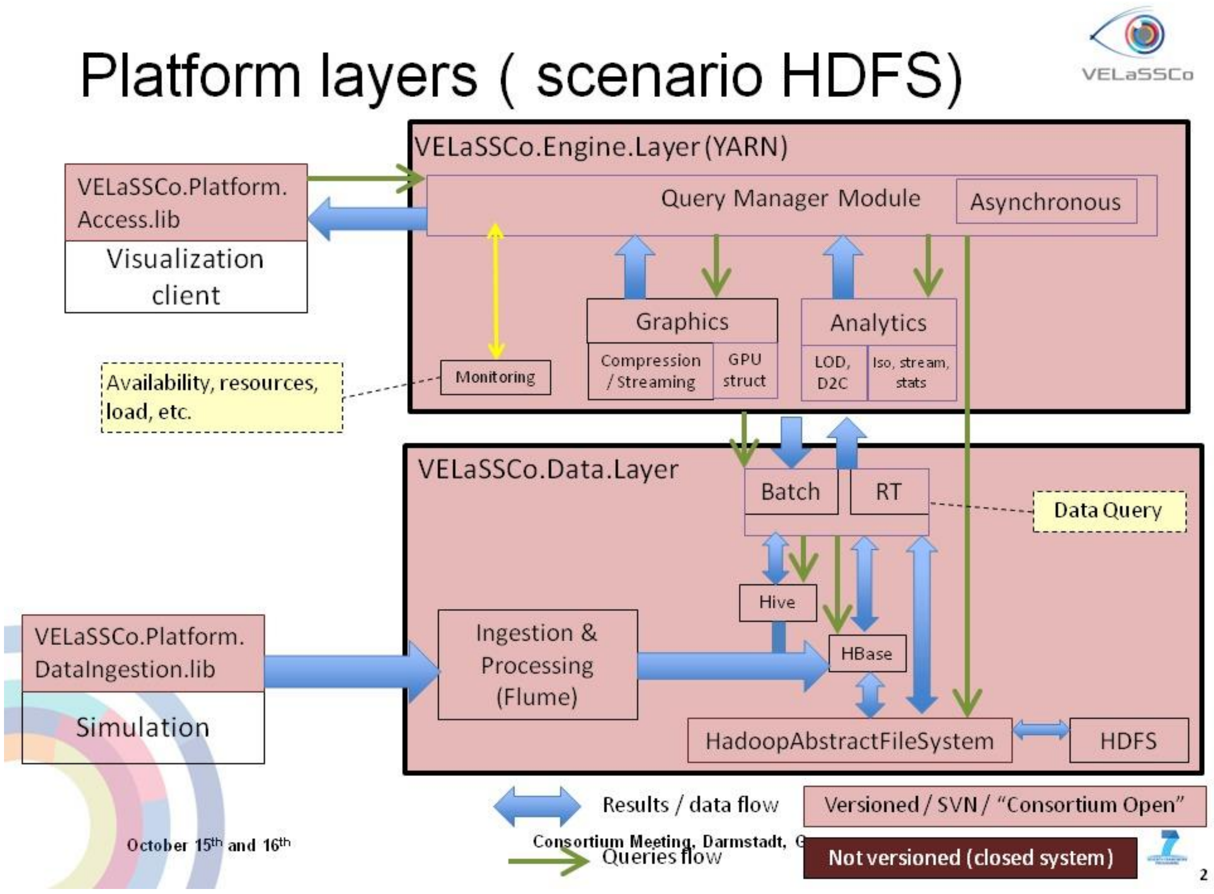

Platform layers (scenario HDFS)
VELaSSCo
VELaSSCo.Engine.Layer (YARN)
VELaSSCo.Platform.Access.lib
Visualization client
Query Manager Module
Asynchronous
Graphics
Compression / Streaming
GPU struct
Analytics
LOD, D2C
Iso, stream, stats
Monitoring
Availability, resources, load, etc.
VELaSSCo.Data.Layer
Batch
RT
Data Query
Hive
HBase
VELaSSCo.Platform.DataIngestion.lib
Simulation
Ingestion & Processing (Flume)
HadoopAbstractFileSystem
HDFS
Results / data flow
Queries flow
Versioned / SVN / "Consortium Open"
Not versioned (closed system)
October 15th and 16th
Consortium Meeting, Darmstadt,
2

Figure 1. The VELaSSCo platform.

This document aims to present our Big Data infrastructure. A representation of this architecture is presented in Figure 2. This architecture is designed around five blocks: System orchestrator, data provider, big data application provider, big data framework provider and data consumer. In the case of our own project, all of these components have been defined. The visualization module manages data consumer, external sources and HPC simulation nodes are our data provider.

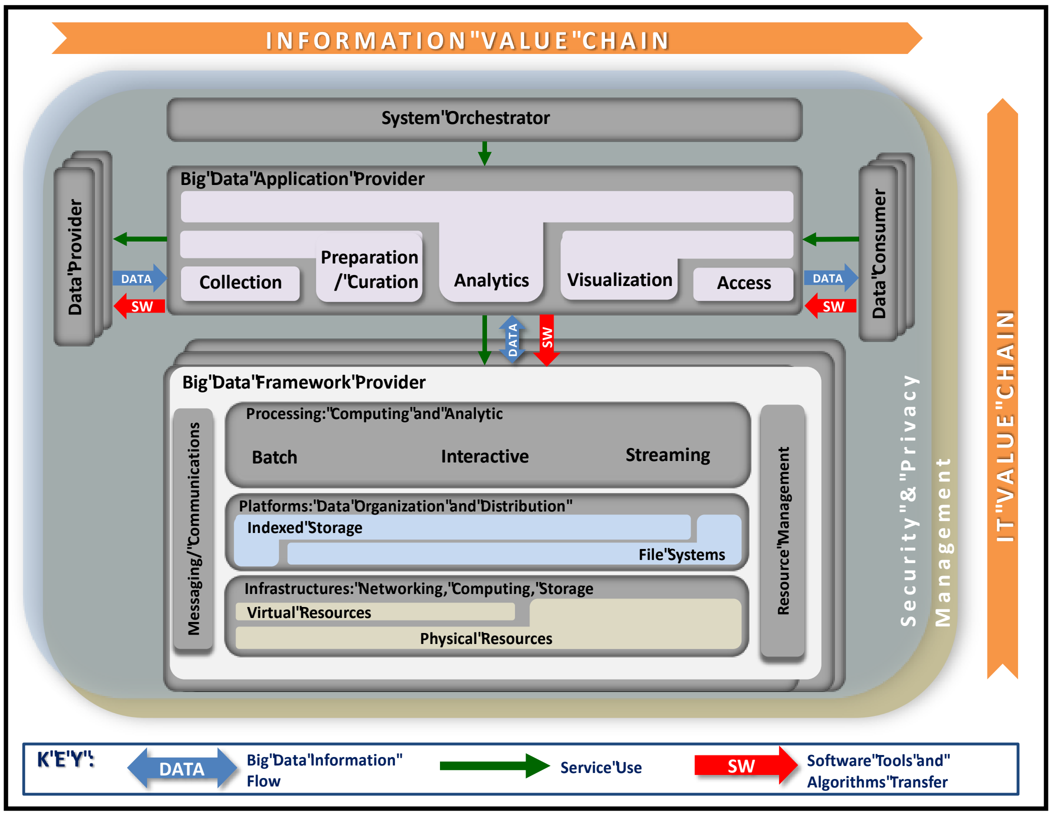


Figure 2. NIST Big Data Reference Architecture [21].

But moving directly to a cloud infrastructure needs more study, if we want to preserve an efficient platform. Some adaptations will be necessary, and moreover, we assume that moving to a HPC-cloud infrastructure is the first step. In the rest of this section, we present how we plan to move to a cloud-HPC infrastructure, by providing comments on what is expected by moving this platform.

2.1. Access Library

A user accesses the VELaSSCo platform by operating a local visualization client (see Figure 3). The visualization client is separated from the database infrastructure, and communicates with the platform to send queries and to receive results. The visualization client basically consists of three main components: a visualization engine, the VELaSSCo access library, and one or more rendering nodes (Figure 3).

The visualization engine serves as rendering and post-processing framework, and is responsible for generating the final images that are presented to the user. Depending on the specific use case, this can either involve rendering of data that has been pre-computed by the VELaSSCo platform (e.g., a triangle mesh representing an iso-surface), or post-processing simulation data on the client side (e.g., generating an iso-surface from a volume dataset).
As visualization engines GiD (CIMNE) [23] and iFX (Fraunhofer) [24] are employed in the frame of the project. The rendering nodes are dedicated machines (ideally fitted with multiple CPUs and GPUs) that can be utilized by the visualization engine. This enables

distributed parallel visualization and post-processing on the client side, and allows for leveraging an available local HPC infrastructure.

The VELaSSCo access library acts as a communication layer, which connects the visualization engine to the VELaSSCo engine layer within the platform. To this end, the access library provides a specific application-programming interface (API) for sending queries and receiving results. Both GiD and iFX feature a plugin-mechanism to enable extensions. To attach the access library to the visualization engines, a plugin for each framework will be developed, where each plugin will be linked to the library. Keeping platform access in a separated library allows for targeting other frameworks besides GiD and iFX.

On the client side, a user interacts with a graphical user interface of one of the visualization engines. User actions are translated by the plugin component into a query message that is sent to the access library. In order to interact with the engine layer, the library will make use of Apache Thrift [22]. Thrift is a framework for remote procedure calls (RPC) and data serialization, and is used to interchange information between the access library and the query manager module (see section 2.2) in the VELaSSCo engine layer. Sending a query will trigger either retrieving simulation results (to be post-processed on the client), or the computation of geometric data (to be rendered on the client). The resulting data is sent back the same way to the visualization engine, which then makes use of its rendering nodes to display or process the data.

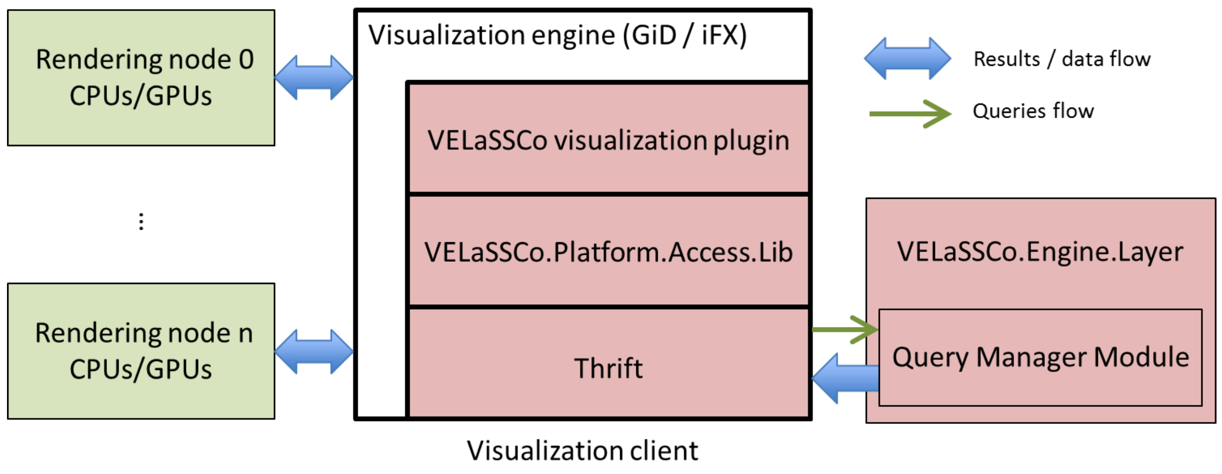


Figure 3 Integration of the VELaSSCo platform access library.

Particularly, the access library is responsible for:

- opening, maintaining and closing the connection to the VELaSSCo platform,
- user validation,
- model selection,
- forwarding queries issued from the visualization client,
- decompressing the data received and pass it to the client to visualize,
- providing information about the state of execution of the queries,
- providing interaction metaphors for specialized queries,
- and monitoring the health state of the VELaSSCo platform.

2.2. Query Manager Module

This module is in charge of the communication between the user and the platform. When a user is navigating into the data set, navigation displacement is mapped to queries, which are sent to the platform. These queries are sent to the VELaSSCo platform using an Access Library to the Query manager (QM). Two kinds of queries can be provided to the QM simple and complex ones. The received messages are analyzed by the QM and decomposed into subsets of queries. Then, QM uses these simplified queries to communicate with different modules of the platform: the analytics part, the rendering part, the I/O part and the monitoring modules. After the decomposition of the query, sub-queries are transmitted to the correct modules of the platform. This decomposition is performed with regard to the available data in the platform.

Simple queries interact directly with the storage platform, and the complex queries require some analysis performed on the data set first (produce intermediary information). For the case of direct query, the request is directly transmitted to the access storage repositories: HDFS, HBase, Hive or directly to any supported FS (Ext4, Lustre, …) or DBMS (Jotne EDM). Our QM provides a unique interface to communicate with one of these storage repositories. For the complex queries, the system will send a query to an intermediary layer (analytics and/or graphics). Another task of the QM is to trigger optimization regarding data and previous queries. This QM evaluates previous queries to determine if it is more suitable to store some computed data instead of analyzing them.

QM is deployed on the gateway node (which is reachable outside of the network). It does not yet require any specific high-end features. The only requirement of this module is to be reachable from outside, because it is in charge of the communication with visualization tools. The communication between both modules (visualization tool and QM) can be performed using Thrift (this tool uses a description file of the protocol and produces all the necessary data to communicate using a client/server paradigm) [22].

If we move this component to a cloud infrastructure, it seems that no change will be needed. This module is not designed to extract information or manage large data flows: it is a gateway between visualization and the extraction part of the platform.

2.3. Graphics

The VELaSSCo graphics module resides within the platform's engine layer (see Figure 1). Its primary purpose is to make use of server side HPC compute and memory resources to prepare query results in a suitable way, so that the information can be displayed by the visualization engines at high speed with minimal latencies. To this end, the graphics module converts the data into an internal format. Data structures resulting from this conversion are handed over to the query manager module, which sends them back to the visualization client.

The design of the data format is guided by the following requirements (a more detailed description of this internal format will be presented in D4.1):

- **Minimal data size.** In order to avoid spending time for parsing and conversion, the amount of control data, such as headers, separators and meta-data, should be minimized.
- **Binary data.** Since data has to be transferred over a network, and also to avoid conversion from an ASCII representation, the payload data should be in binary form.
- **Processing friendly layout.** Ideally, all data should be in a form, where is can directly be accessed by processing units without any conversion. It should be possible to load the payload data into buffers that are directly uploaded to the GPU. While the GPU is the typical target, this also applies to CPUs.

Additionally, the graphics module handles on-the-fly compression of the data, which reduces communication bandwidth. This can either be a lossy compression (e.g., by using a decreasing floating point accuracy), or it can be a lossless entropy-encoding (e.g., ZIP variants LZO or LZ4).

Moreover, the graphics module will handle streaming and progressive data transfer. Rather than sending the complete data set of a query in one step, information is sent on demand in small parts based on user input (e.g., depending on the current position of a moving camera in a visualization).

2.4. Analytics

This module is in charge of performing data analytics over stored simulation data in order to obtain new meaningful results of interest. The module will be also in charge of estimating the computational cost of the analytics queries in order to help the Query Manager to evaluate in which mode the query will be processed: Real time (low cost) or Batch (high cost). Furthermore, in the case of time-consuming analytics queries, the module will provide a fast feedback of the results to the user by means of issuing the same queries on a coarser or simplified version of the simulated data while the full-resolution results are processed. The HPC-cloud and virtualization overhead will impact the performance of these analytic queries and should be considered when the cost of these queries is evaluated.

The operations to be conducted by this module will be integrated into the Hadoop framework. Some of the analytics queries will use the YARN layer in order to decompose the queries into sub-queries so that the query can be computed in a distributed way. Once the sub-queries are finalized, a Map-Reduce like mechanism will be used to gather the results of sub-queries in order to obtain the final results of the query.

Since Analytics module will be integrated into the Hadoop framework, the specific features of its implementation in HPC systems are related to the deployment of Hadoop in this type of systems. As it will be explained in section 2.5, the implementation of Hadoop in HPC-clusters will be conducted by means of virtualization. In order to get a good performance, it will require a tuning of the configuration parameters of the virtualization processes.

It is not expected any major change in this module for its implementation on HPC-cloud systems. The only changes expected are related to the configuration file that will be necessary in order to fit the architecture of the HPC-cloud infrastructure.

2.5. Data Query

This module is in charge of the management of data stored in the platform. It is composed by different components: a query layer (which can interact in real-time or batch), an ingestion processor, and the data storage layer.

The batch access of storage is performed through a set of existing Hadoop plugins: Hive, HBase, and Hadoop I/O, and plug-ins for ISO 10303 STEP compliant storage (EDM) that will be developed as part of this project. In the current version of the prototype, we only deal with Batch processing, but to move to a real-time processing engine, we plan to use STORM.

These different access strategies provide efficient tools to extract information from the storage in the most efficient way. HBase, Hive and Hadoop data can be linked. These different solutions enable to take advantages of any access solution. Thus, by providing different access possibilities, we offer to users an enhanced version of a simple access library (based on a unique component). Hadoop I/O provides a low level access to data. These different components have some unique access strategies: Hive uses SQL, HBase has its own query language, and FS can be queried through Hadoop Abstract file system or with direct I/O operations. In [20], some explanations are provided concerning the Hadoop compatibility file system used with HDFS. Two methodologies are proposed to enable compatibility: with native FS or with the implementation of an abstract Hadoop class. Both methods have advantages and drawbacks. But the most reliable method is the implementation of the Hadoop class. Our solution does not only support Big data software stack, but also this solution has to be efficient with HPC components (for example Lustre, Ceph, or Ext3, ext4, etc.). We are also developing a storage system based on the EDM DB to enhance our platform with a new storage repository. EDM will be used into the closed sources scenario, and Hadoop interactions will be mapped to EDM queries using a specific plugin developed for this project.

This module is tightly linked to the storage module. We have selected Hadoop as a basic software stack for our platform, and we also selected a couple of plugins to enable different access strategies for the datasets. This module is massively parallel at different levels: compute and storage. The components used are designed to run on massive parallel architectures, composed by commodity nodes.

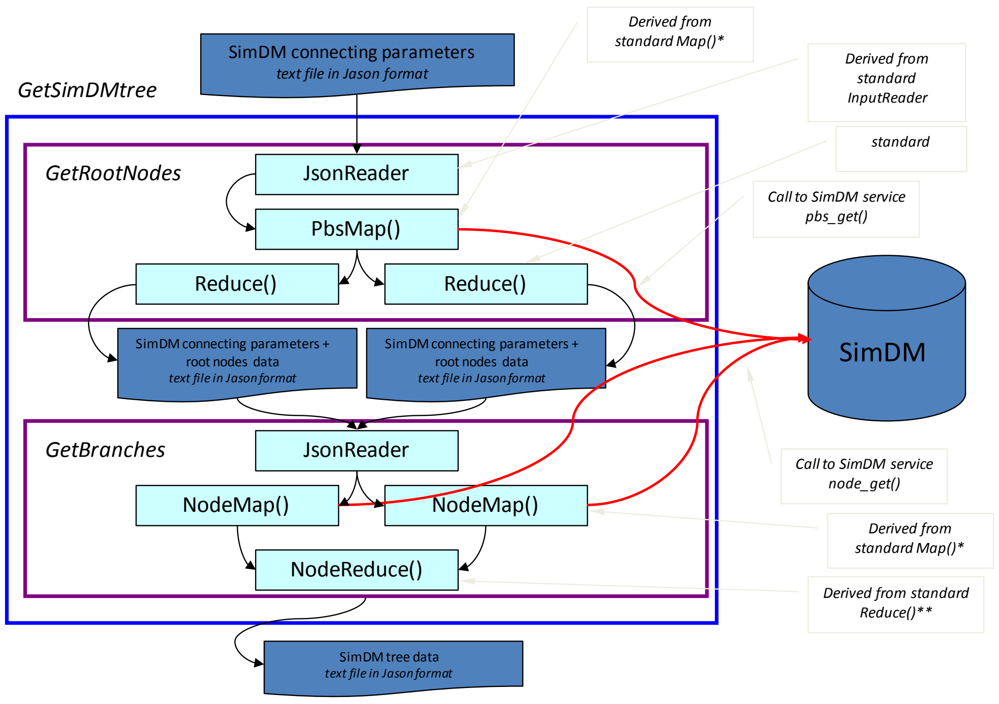
But to fit with HPC capabilities (very high compute performance), we use intensively virtual machines. This approach enables our solution to fit efficiently with multi-core capabilities of hardware. But virtualization has a cost and to reduce impact of this method, it is necessary to evaluate the most suitable set of parameters regarding virtualization. This strategy is performed using an automatic benchmark tool, which starts and stops virtual machines when activity of the cluster is idle.

We expect that when we will move to HPC-cloud infrastructures, no major changes will be necessary. Only configuration files will have to be adapted to fit with the new platforms. But this task is always necessary when we move to a new architecture. Moreover, our strategy based on auto-configuration will enable to adapt parameters of the storage platform. The virtualization parameters can be impacted by these changes: maybe less virtual machine might be needed.

2.6. EDM plug-in

The VELaSSCo platform will be validated against two types of storage: the open source alternative HBase/Hive with a table oriented database and the commercial solution EXPRESS Data Manager (EDM) with its object data models that are compliant to ISO 10303, STEP.

Whereas the Hadoop framework for distributed data management is already integrated with Hbase/Hive, no such integration exists with EDM and STEP. VELaSSCo will develop this integration so that the VELaSSCo platform can be used with either Hbase/Hive or EDM.



SimDM connecting parameters
text file in Jason format
GetSimDMtree
GetRootNodes
JsonReader
PbsMap()
Reduce()
Reduce()
SimDM connecting parameters + root nodes data
text file in Jason format
SimDM connecting parameters + root nodes data
text file in Jason format
GetBranches
JsonReader
NodeMap()
NodeMap()
NodeReduce()
SimDM tree data
text file in Jason format
SimDM
Derived from standard Map()*
Derived from standard InputReader
standard
Call to SimDM service pbs_get()
Call to SimDM service node_get()
Derived from standard Map()*
Derived from standard Reduce()**

Figure 4: Example of the dataflow of a YARN-query to EDMopenSimDM

Initial prototyping has revealed the need for two EDM plug-ins. EDM needs to receive the data queries (section 2.5) in an EDM understandable format and shall deliver results in VeLaSSCo format. The queries are triggered in the YARN-layer (sections 2.4 and 3) in a VELaSSCo unified way and, thus, independent of the underlying storage solution. This first EDM plug-in is, therefore, located in the YARN-layer. The VELaSSCo data domain is engineering analysis or simulation, which is the domain of the EDM application *EDMopenSimDM*. YARN-queries will be translated into queries to *EDMopenSimDM*. See Figure 4 for an example of the dataflow that will be implemented by this plug-in.

On the other side, EDM shall be able to benefit from big data management capabilities of Hadoop and its file system. This requires the development of the second plug-in. For importing big data, EDM needs to read data not from the Windows or UNIX file system, as it usually does, but from the HDFS. And, after having processed data into the EDM DBMS format, data will still be big data and will require the HDFS support for being managed. The second EDM plug-in for Hadoop will be the porting of the EDM data access API to HDFS. Especially this second plug-in will provide EDM with the crucially needed big data capabilities.

Big data, however, not only requires access to data that are distributed because they cannot be hold in a single location. Big data also requires special means for processing queries, validation and manipulation of extraordinary amounts of data in reasonable times. HPCs provide the raw power that EDM will want to use. EDM has a concept called Application Server (*EDMappServer*™). The *EDMappServer* allows multiple clients to access an *EDMserver*™ simultaneously and encloses all work intensive operations to improve performance. The *EDMserver* configuration is flexible, and an *EDMappServer* may run in any system in the network. Thus, we will use this concept to leverage the power of multi-node HPCs. The concept may be coupled with an algorithm for parallelization of queries and the map/reduce paradigm of Hadoop.

Running EDM in a Cloud-based HPC adds both benefits and issues. An obvious benefit is the availability of virtual machines in the Cloud. *EDMserver*, which is the foundation of the server-side software of *EDMopenSimDM*, runs only on MS-Windows. This operating system is rather rare among HPC installations. On the other hand, Cloud environments demand stricter security and authentication measures, which need to be harmonized with the EDM intrinsic measures. This additional effort needs to be spent to exploit the benefits that automated communication among applications can give on Clouds.

2.7. Data Ingestion

Data Ingestion module is in charge of feeding persistence layer represented by specific databases. This module formats the received dataset, which have been produced after the processing of a simulation. To do so, Data Ingestion module aims to implement a Data Injector component, which take as input simulation information and insert this output into the persistence layer. This process will be achieved in two iterative approaches:

- The first *batch* methodology is focused on the extraction of information from the simulation process to the storage layer (in our case NoSQL databases and distributed file systems). This process is synchronized sequentially. In the first stage of this process, the simulation engine should has already generated data files into a specific path (located in the HPC cluster) and then Data Injector component will take these files as input to process their information.
- Finally, a *near real-time* solution should be ideally implemented, allowing the streaming synchronization between simulation data files generation and data ingestion processes.

In both cases, computing resources needed could be very high. This heavy computation workload comes from the size and number of simulation files generated. Besides this, batch and streaming will required a lot of memory; this large amount comes from the need of reading multiple files at the same time.

Thus, it is necessary to take into account some features in order to move Data Ingestion module to a cloud architecture:

- HPC provides specific computing capacities, which should be kept for a proper performance of the data ingestion process. Therefore, HPC-Cloud resources (which are suitable to support virtualization) can provide enough resources for each module according to their requirements.

- Besides performance problems, virtualization of distributed resources could be an important issue. HPC typically provides a set of nodes, which allows distributing resources along the topology of the system, providing flexibility and increasing the computing capacity. HPC-Cloud environment could affect the maintenance of integrity among the system by introducing a virtualization layer among nodes, services, etc.

Typically, a complex cloud infrastructure will be divided into different layers, at least three:

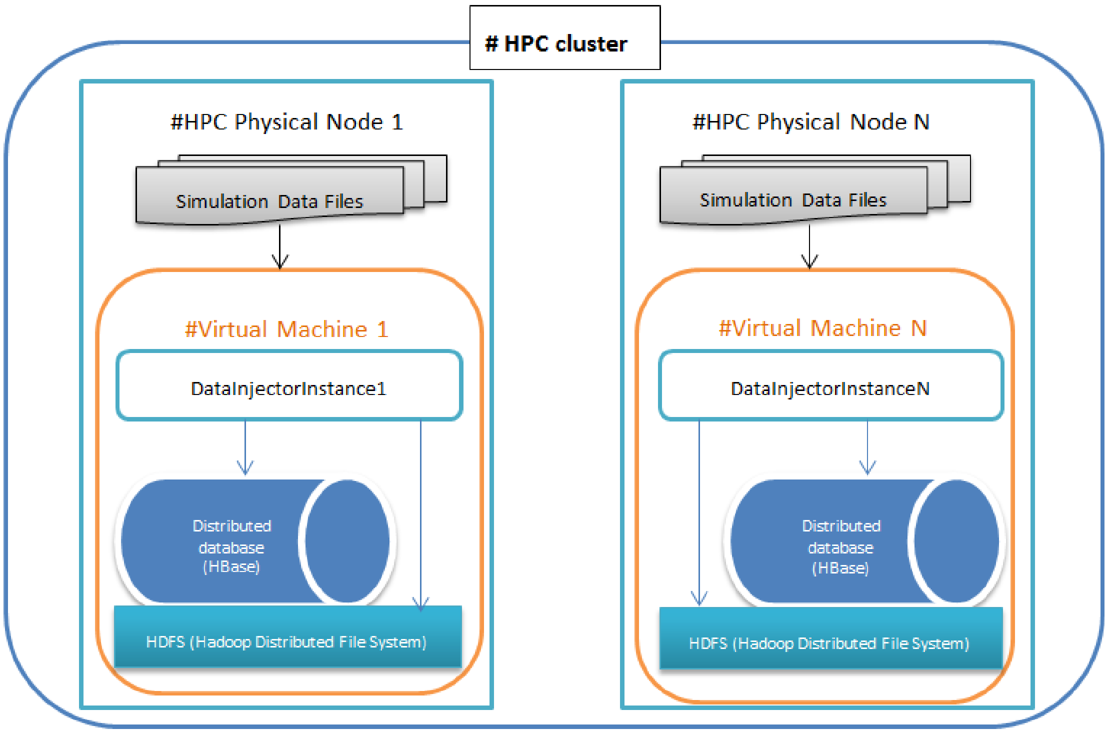


Figure 5.HPC-Cloud architecture of Data Ingestion

SaaS[1], PaaS[2] and IaaS[3], as exposed on Figure 11. This structure aims to gather services and resources into different levels according to their functionality.

To illustrate the problematic of virtualizing resources, diagram in Figure 5 represents a HPC-Cloud implementation example of Data Ingestion module.

Architecture above displays a simplified implementation example where it is shown how HPC nodes resources can be virtualized with one or more Virtual Machines. These virtual machines may contain the Data Ingestion logic and access directly to distributed persistence layer. The simulation data files could be placed in physical nodes or virtual machines, depending on the synchronization solution adopted. Similarly, the possibility to virtualize resources can affect to the decision of moving persistence layer within Virtual Machines or not, depending of scalability, performances, number of accesses, etc. All these variables should be taken into account at the time of defining our HPC-Cloud architecture.

[1] Software as a Service
[2] Platform as a Service
[3] Infrastructure as a Service

2.1. Simulation

FEM simulations

A simulation of a physical process is performed by solving the equations describing this process using a discretization of the domain of the problem, depending on the method used to solve these equations. In a Computer Fluid Dynamics (CFD) simulation, the domain, for instance, is the air surrounding a telescope as shown in Figure 7. This domain is subdivided into a set (mesh) of very simple geometric elements, usually tetrahedra, sharing their vertices, called nodes.

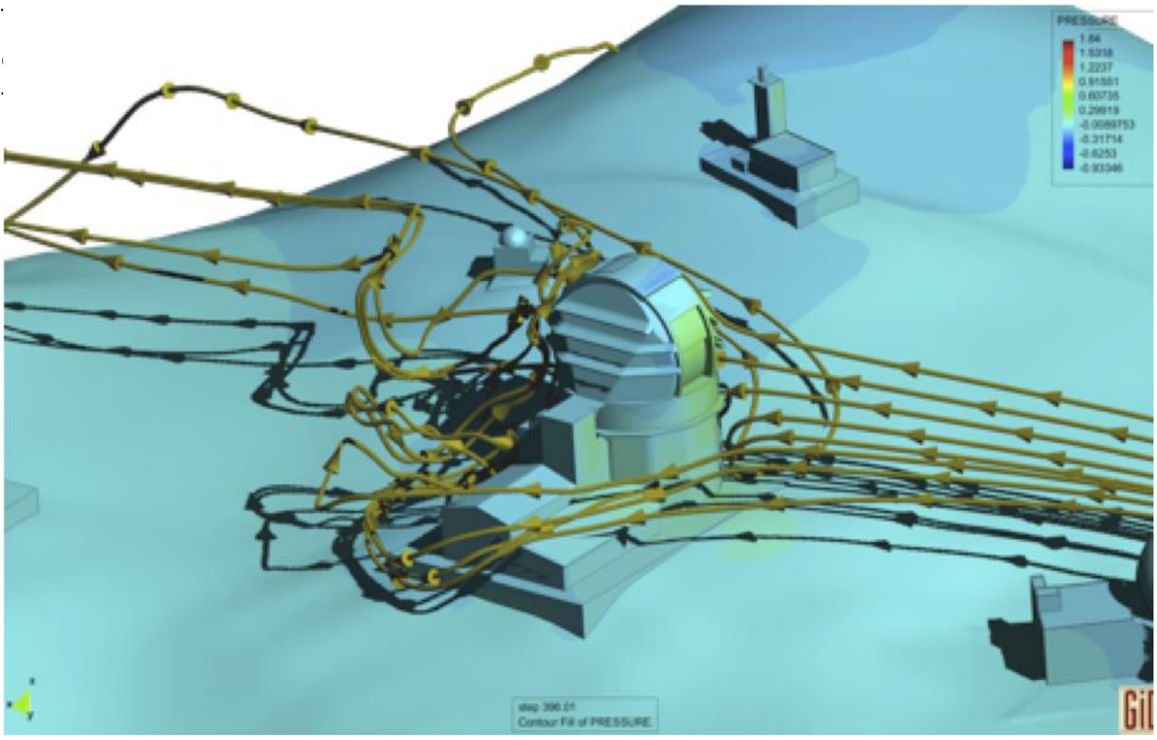


Figure 6. Air flow surrounding a telescope. These stream lines have been computed using the FEM method.

domain is partitioned into several sub-domains, usually one for each node of the cluster. That means that the initial mesh of billion elements is sub-divided into smaller sets and distributed across the cluster as shown in Figure 8. The simulation program, which runs on each node of the cluster, reads their partition and communicates with its neighbours to solve the problem. Then, at given time intervals, the program writes its mesh portion and results into its own file. At the end of the simulation the simulation results are stored in as many files as partitions are.

In the first use case of the VELaSSCo platform, these files are ingested into the platform for a later analysis and visualization.
In the second use case, the simulation program communicates and ingests the data directly into the VELaSSCo platform by using the VELaSSCo's DataIngestion library, and so avoiding the usage and storage of local (to the HPC facility) huge files. This communication is accomplished by a network of flume agents and pipelines.

Porting simulations from the HPC environment to HPC-clouds is straightforward as shown in the Fortissimo project [29], only the already mentioned specific aspects should be considered in the deployment, like the lower per-cpu performance and eventually high network latency.

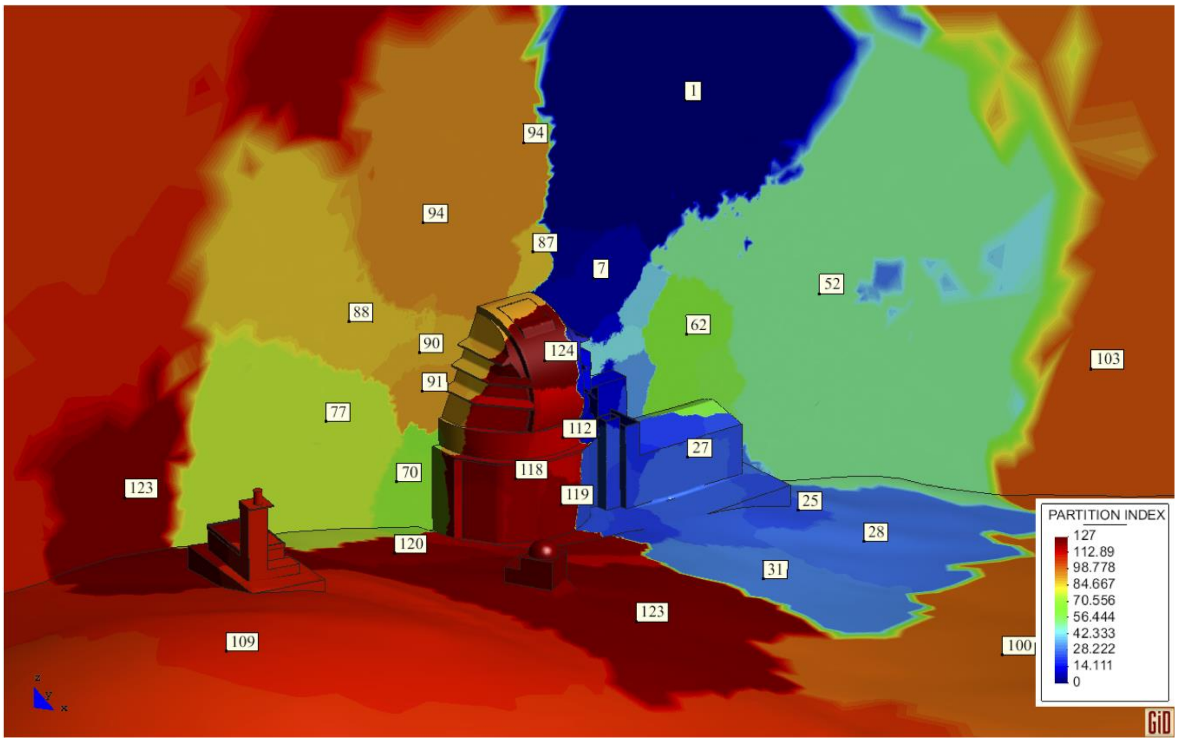


Figure 7. Space decomposition of the air surrounding the telescope (provided by CIMNE)

Discrete based simulations

Discrete based simulations are mainly conducted by mean of Discrete Element Method (DEM) models. In this kind of simulations, the material is represented by means of discrete particles that interact between themselves through the contacts forces. The results obtained from the DEM simulations are discrete and related to the individual particles and the individual contacts. An example of a silo discharge simulation is shown in Figure 9.

The most majority of DEM simulation codes are capable of running in parallel through a large number of nodes of HPC systems. The parallelism of the code is typically conducted by means of Message Passing techniques (MPI) and spatial decomposition of the simulation domain. The simulation results are typically written into a single file containing all the simulation output data. Nevertheless, in the case of very large simulations and highly distributed computation, it can be more convenient that each node writes a file containing the results of its own partition data.

We expect that moving to HPC-cloud infrastructures, most of DEM codes will be fully compatible. Indeed, it can be found that different DEM codes have been already run on HPC-cloud infrastructure such as Amazon EC2.

The only difference between HPC-clusters and HPC-clouds can be the computational time needed to run the simulations. HPC-clusters typically show a better performance than HPC-Clouds but this is strongly dependent on the hardware specifications of the infrastructure.

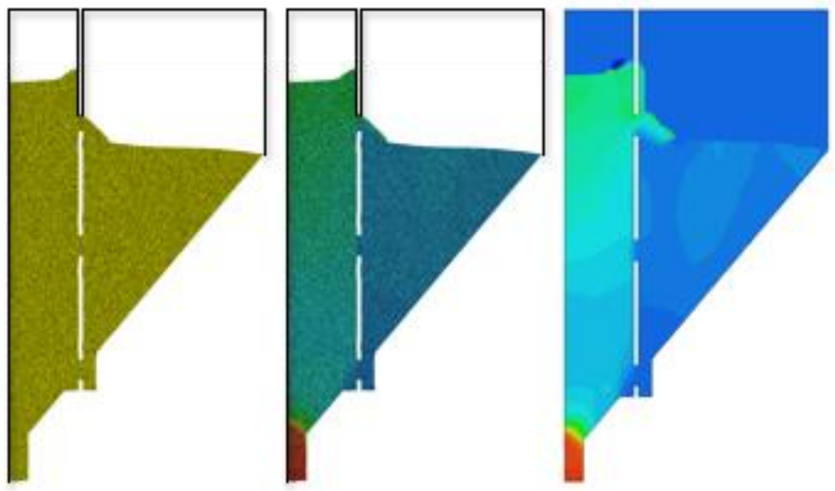


Figure 8. Example of DEM simulation provided by the UEDIN.

3. Specific treatment of petabyte data storage

This section provides the specifications of the petabyte tools of the VELaSSCo platform. This section discusses the storage and the ingestion part.

Two initial datasets have been identified; their data come from simulation or simply exist already as computed data. Data from external sources are already produced and available on dedicated storage repositories. This second type of data sets is produced by FEM and DEM simulations.

One example is a Computational Fluid Dynamics (CFD) simulation of the air surrounding a telescope. In the case of DEM and FEM, the amounts of the produced data can be huge, and it is necessary to use multiple computational nodes of a cluster to compute it. An example of a DEM simulation is presented in Figure 9. In Figure 7 and Figure 8, we present an example of a FEM simulation on a telescope is presented.

These multiple clusters stored small sets of data from every simulation time step. These multiple time steps need to be aggregated together to form a unique file. Flume agents perform this task: they extract and store data from the HPC to the storage repository. These agents can be configured to gather information from external sources. This space decomposition is presented in Figure 8. This information is merged by agents into one repository. These agents write data into the selected storage system (HBase), but these agents can also write data on raw storage systems (HDFS, Lustre, etc). Flume is very efficient with lots of agents; this system needs to be highly distributed to be efficient. More agents imply a more powerful system. This extracting strategy can also be used with our external providers. This method can be plugged into a data transfer strategy based on a non-HPC solution. Flume is also suitable with the Lambda architecture by providing requirements in term of scalability and fault tolerance. This tool can be tuned to be efficient by adding gathering agents.

The second part of this petabyte architecture system is composed by the storage and compute platform. For this part, we have selected Hadoop. This Big Data software stack supports many other applications. In [12], authors give an overview of this software stack and examples of available extensions are presented in Figure 10. This framework enables high distributed computing which enables high availability of the system. Hadoop is built on top of HDFS, the HDFS storage file system. But in the earlier age of Hadoop, the only computation model was Map Reduce. Now several extensions are available to enable more complex computations. Hadoop is a distributed storage system composed by two different nodes: the resource manager and the slaves (with node managers). All data are stored into a VELaSSCo repository. This repository is composed by part of the Hadoop Big Data software stack.

The new version 2.x of Hadoop is managed by YARN, a specific Hadoop scheduler and resource manager. The resource manager is in charge of distributing computations and storage. It keeps a trace of where data are stored, and which are the running jobs.

For this project, using Hadoop seems to be a good choice as foundation of the platform. This framework enables to have an efficient and already tested framework to support petabyte storage. But this system was not designed to support HPCs. Thus, our work will develop efficient Hadoop frameworks for HPC. Enabling virtualization at different levels on the platform provides efficiency for HPC environments when using Hadoop. These improvements are presented below.

In the rest of this section, we present the deployment of Flume agents and distributed storage on a HPC system, then in an HPC-cloud system and finally on a pure cloud system.

3.1. HPC systems

In this section, we discuss the integration of our petabyte solution on HPC infrastructures. It is focused on different aspects: cost, performance, flexibility, architecture, virtualization, management of resources and services, fault tolerance and analysis mechanisms.

First of all it is necessary to define HPC systems. High Performance Computing is composed by a high-end computer facility. HPC is able to provide fast computation capabilities. These systems are composed by two kinds of nodes: compute nodes and gateways. Compute nodes are elements that are used to perform the computation. They are composed by high-end CPUs which provide modern hardware features for the computation: multi-threading, SIMD, etc. To avoid job concurrency, these nodes are not directly reachable by the users. A scheduler manages them. The second element of a HPC is the gateway. This node is necessary to manage user computations. It is in charge of the scheduling and starting the computations on nodes. This node manages workload priorities, and the health of the system. This element executes the task scheduler (Torque, Slurm, etc). In the gateway, no computation is running

The cost of HPC systems is high and requires recurrent hardware updates. But the cost of such kind of systems is not only composed by hardware. As stated in [3], the hardware only represents 7% of the global bill. These systems are very expensive in terms of maintenance and power consumption, only a few of SMEs can afford this technology. The same issue is

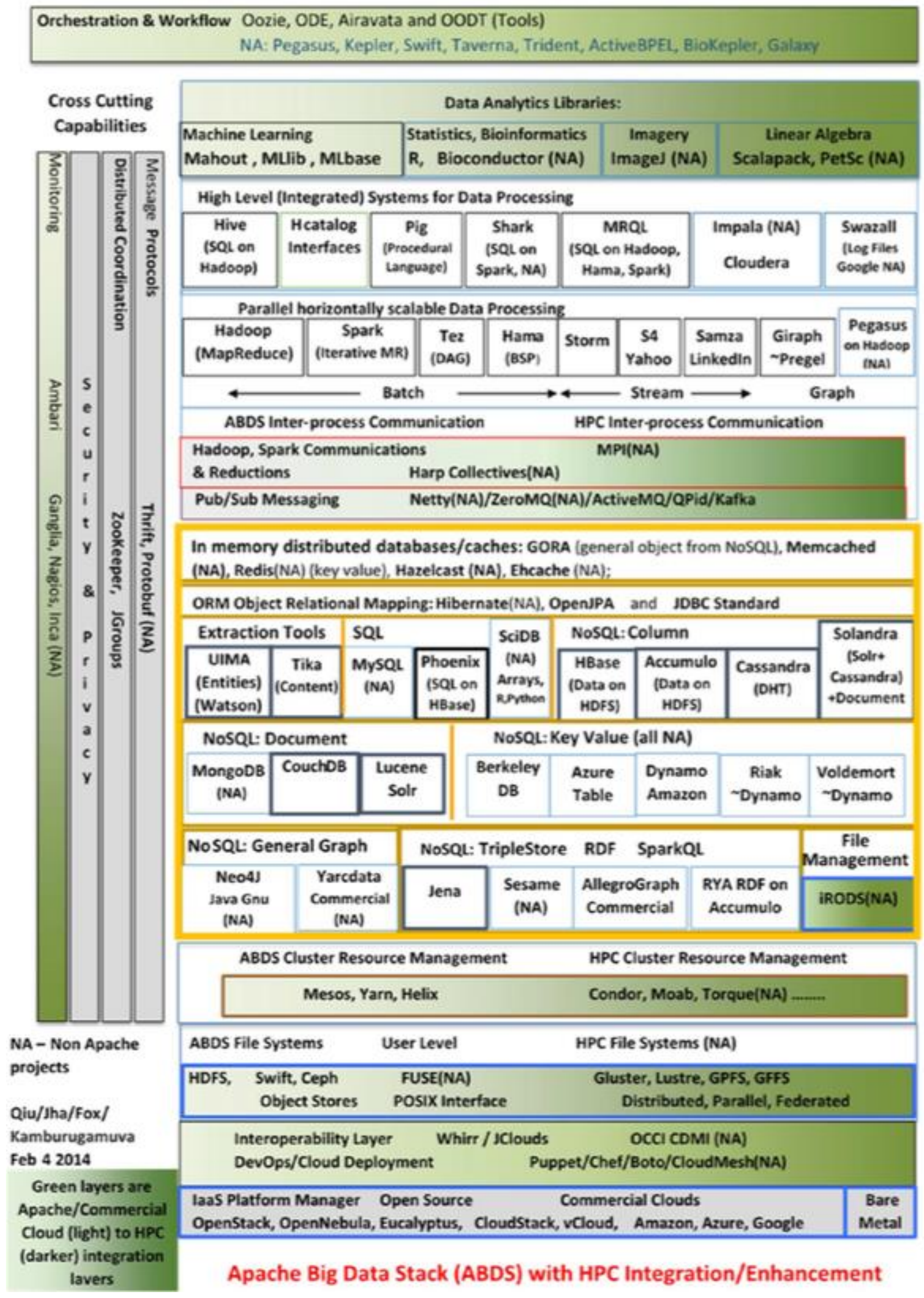

Orchestration & Workflow Oozie, ODE, Airavata and OODT (Tools)
NA: Pegasus, Kepler, Swift, Taverna, Trident, ActiveBPEL, BioKepler, Galaxy
Cross Cutting Capabilities
Monitoring Ambari Ganglia, Nagios, Inca (NA)
Security & Privacy
Distributed Coordination ZooKeeper, JGroups
Message Protocols Thrift, Protobuf (NA)
Data Analytics Libraries:
Machine Learning Mahout , MLlib , MLbase
Statistics, Bioinformatics R, Bioconductor (NA)
Imagery ImageJ (NA)
Linear Algebra Scalapack, PetSc (NA)
High Level (Integrated) Systems for Data Processing
Hive (SQL on Hadoop)
Hcatalog Interfaces
Pig (Procedural Language)
Shark (SQL on Spark, NA)
MRQL (SQL on Hadoop, Hama, Spark)
Impala (NA) Cloudera
Swazall (Log Files Google NA)
Parallel horizontally scalable Data Processing
Hadoop (MapReduce)
Spark (Iterative MR)
Tez (DAG)
Hama (BSP)
Storm
S4 Yahoo
Samza LinkedIn
Giraph ~Pregel
Pegasus on Hadoop (NA)
Batch
Stream
Graph
ABDS Inter-process Communication
HPC Inter-process Communication
Hadoop, Spark Communications & Reductions
MPI(NA)
Harp Collectives(NA)
Pub/Sub Messaging
Netty(NA)/ZeroMQ(NA)/ActiveMQ/QPid/Kafka
In memory distributed databases/caches: GORA (general object from NoSQL), Memcached (NA), Redis(NA) (key value), Hazelcast (NA), Ehcache (NA);
ORM Object Relational Mapping: Hibernate(NA), OpenJPA and JDBC Standard
Extraction Tools
UIMA (Entities) (Watson)
Tika (Content)
SQL
MySQL (NA)
Phoenix (SQL on HBase)
SciDB (NA) Arrays, R,Python
NoSQL: Column
HBase (Data on HDFS)
Accumulo (Data on HDFS)
Cassandra (DHT)
Solandra (Solr+ Cassandra) +Document
NoSQL: Document
MongoDB (NA)
CouchDB
Lucene Solr
NoSQL: Key Value (all NA)
Berkeley DB
Azure Table
Dynamo Amazon
Riak ~Dynamo
Voldemort ~Dynamo
NoSQL: General Graph
Neo4J Java Gnu (NA)
Yarcdata Commercial (NA)
NoSQL: TripleStore RDF SparkQL
Jena
Sesame (NA)
AllegroGraph Commercial
RYA RDF on Accumulo
File Management
iRODS(NA)
ABDS Cluster Resource Management
HPC Cluster Resource Management
Mesos, Yarn, Helix
Condor, Moab, Torque(NA) ....
NA – Non Apache projects
ABDS File Systems
User Level
HPC File Systems (NA)
HDFS,
Swift, Ceph Object Stores
FUSE(NA) POSIX Interface
Gluster, Lustre, GPFS, GFFS Distributed, Parallel, Federated
Qiu/Jha/Fox/ Kamburugamuva Feb 4 2014
Interoperability Layer
Whirr / JClouds
OCCI CDMI (NA)
DevOps/Cloud Deployment
Puppet/Chef/Boto/CloudMesh(NA)
Green layers are Apache/Commercial Cloud (light) to HPC (darker) integration layers
IaaS Platform Manager
Open Source
Commercial Clouds
OpenStack, OpenNebula, Eucalyptus, CloudStack, vCloud, Amazon, Azure, Google
Bare Metal
Apache Big Data Stack (ABDS) with HPC Integration/Enhancement

Figure 9. HPC integration opportunities for the Hadoop Big Data stack [12]

present in the scientific community the cost and extensive update of HPC nodes is a problem. Both communities prefer to use cheaper solutions like Cloud-based ones.

HPC systems are not flexible and cannot be managed dynamically. Resources used are specified at the beginning of the execution. The architecture of these systems is monolithic, one node is used as a gateway, and multiple nodes are used for the computations. The storage of HPC is performed on external nodes (mostly through a specific storage cluster). On HPC clusters, virtualization is never used to not affect the execution time; it is necessary to provide application modification to support multi-threading, and SIMD programming. Only seldom fault tolerance is managed: when a node crashes, it is necessary to restart the whole computation.

Selecting the adapted HPC infrastructure to perform big data analytics is complex. As shown in [2], it is necessary to make an evaluation of the infrastructure. The hardware and software stack used in HPC environments depends on the targeted applications. In the VELaSSCo project, simulations are not run on the developed infrastructure. These heavy computations are performed on an external HPC system. The computation on our infrastructure will be mainly focused on specific computation based on extracting features from datasets. The need in term of computations has guided our choice for the software stack. And to fit with the available hardware, we use virtualization on each node.

To reduce the impact of virtualization, containers can also be used. This idea is also validated by the recent change on Hadoop framework, now, the Resource Manager (RM) trigger containers to improve computation on the Hadoop clusters, see [19].

The efficiency of HPC architecture can be improved at different levels: computing and throughput. But it is also possible to deploy a HPC infrastructure specially designed for I/O operations. Read and write operations are not performed locally on a HPC cluster; data is stored on external repositories. The architecture depends on the targeted optimizations.

For the VELaSSCo project, we have selected a software stack, which is not designed to fit with HPC: it is a Big Data software stack. To bring Hadoop to HPC, it is necessary to provide some evaluations regarding the available hardware. The performance reached by the software stack heavily depends on possibilities of modifying hardware and available software. An example of this tweaks are: installing container software or a supplementary hard disk on the node (this strategy can be used to provide local storage on each node).

As stated before, in a HPC environment, a specific component call scheduler performs the management of jobs. In a Hadoop environment another scheduler is provided: it is the resource manager. An HPC scheduler is in charge of managing available resources of the system. And this availability can be an issue if one node is too busy. Hadoop is a big data framework, which was not designed to run on HPC systems. But, this framework shares some common features with them, see [5]. This software stack has not yet been widely spread into the HPC community. Hadoop is not yet fully suitable with HPC [4]; it is considered as an invader: the HPC community has not designed it. Hadoop is developed in JAVA. It reinvents a HPC programing model, and also uses an old communication methodology. But now, with the new task scheduler of Hadoop (Yarn), it seems to be a more realistic software stack for HPC. YARN provides most of HPC features; one case study is presented in [6].

To conclude, Hadoop was not specially design to run on HPC but we hope to be able to provide an efficient evaluation tool to tweak parameters of the Hadoop software stack. This evaluation process will be performed at the top level of the VELaSSCo platform in the Query manager. The knowledge extracted from this module allows enhancing the parameters of the whole platform. In the next section, we present what is expected if we move our solution to a cloud-HPC infrastructure.

3.2. For HPC-cloud systems

At the end of the VELaSSCo project, we expect to move to a new kind of platform. For this purpose, it is necessary to study the impact in terms of cost and benefits of this perspective.

This study needs to be performed for each component of the final architecture. From related works, we can see that HPC cloud infrastructures are cheaper than traditional HPC. In [3] the authors provide a cost study which compares HPC and HPC-cloud infrastructures. Pure HPC-clouds are better with a factor of 3.7 compared to traditional HPC, but a hybrid solution, which mixes private HPC and clouds HPC, can achieve higher performance, up to 4.2.

On a Cloud-HPC infrastructure, flexibility is well suited with the proposed architecture. In fact the system is especially designed to be flexible. Nodes can be added dynamically, this extensibility is managed by credit cards.

Cloud-HPC infrastructures mainly depend on what is proposed by a provider. For example for Amazon, the HPC cloud is located on a specific cluster at Amazon. These nodes are interconnected with specific network hardware.

As stated in the previous deliverable, virtualization on HPC-clouds really depends on the provider, and most of provider offers a pure bare-metal computation facility. On Amazon, things are less clear and it is complex to obtain information on their software stack.

But there exist some scientific alternatives to industrial HPC-cloud systems. The Dutch scientific community proposes one of these clusters, detailed in [10]. It presents two clusters: one dedicated to traditional HPC computations, while the second one is composed by a cloud infrastructure. The goal is to bring scientific computations to a new era: the future. Finally, the UK community also provides a Cloud infrastructure dedicated to HPC. In [15], the authors present a new compute facility located in UK. This cluster is specially designed to run specific algorithms from physics, astronomy engineering and healthcare.

Cloud-HPC infrastructures are not yet massively used. These infrastructures are not spread. Thus moving to such kind of architectures can be a challenging task. In the case of VELaSSCo, we think that move will be an efficient strategy. CPU inside HPC-cloud can have lower CPU performance but more cores are available. Thus, we think that deploying a VELaSSCo infrastructure into a cloud-HPC system will provide similar or even better performance. For the Flume agent, performance will be similar than the Hadoop deployment. If we are able to deploy a larger cloud-HPC infrastructure (with more nodes), the gathering will be limited by the network throughput.

3.3. For cloud systems

A cloud infrastructure has some important differences compared to traditional HPC or even HPC-cloud infrastructure. These systems are mainly composed by virtual machines. The level of virtualization depends on the provider. On the Amazon cloud system, virtualization impact is reduced compared to the selected hypervisor. The data sets used in cloud environment have evolved, and now big data produced are common from different domains: climate modeling, astrophysics, health, etc. NIST presents in [21] a mapping of their proposed architecture to cloud architecture. This solution can be used in our context.

For the final user of a cloud, the cost implied depends on the selected environment. Regarding the cloud provider, a user can buy more CPU, more bandwidth, or higher IO performance, etc. The actual cost of a cloud instance is now very cheap and it is possible to have access to lot of cores with a credit card.

In a public cloud environment, the evaluation of performance is not a trivial task. The computation of these systems is mainly composed by virtual machines and user can only access to these virtual nodes and not directly to the bare-metal cores. Thus, virtualization brings an important overhead compared to a real machine. But this virtualization impact is not validated on the amazon cluster. As stated in [14], Amazon is able to provide an efficient cloud infrastructure without impact on CPU performance. This difference comes from the hypervisor. At Amazon, they have developed their own supervisor, which reduced the overhead; other providers use a traditional hypervisor, which is not so efficient.

Most cloud systems are highly flexible. It is possible to remove nodes or add nodes dynamically. This strategy is mainly performed by virtualization hypervisor; it is provide a rapid and efficient tool to deploy new nodes. Specific SDK developed by providers gives this flexibility.

The architecture of these systems is widely distributed. It is possible to have access on nodes located in the same cluster, or even in external clusters. In most architecture, interconnection of nodes is performed by standard network links.

In cloud infrastructures, virtualization is the main feature. The software stack used in cloud environment is mainly focused on these features. These ecosystems are based on hypervisors, which enable to deploy any kind of hardware. An evaluation highlighted that the cloud-HPC infrastructure provided by Amazon has equal performance than a traditional HPC cluster, see [9].

The fault tolerance is not mandatory on clouds. A user of a cloud environment may have to deal with such kind of things to enhance the platform.

Several researchers have provided study regarding to cloud capabilities. This kind of architecture is more and more used to provide cheaper computation solutions. To ensure an efficient platform to store and compute data, it is necessary to use the knowledge of the application scientists to improve the platform, see [9].

A

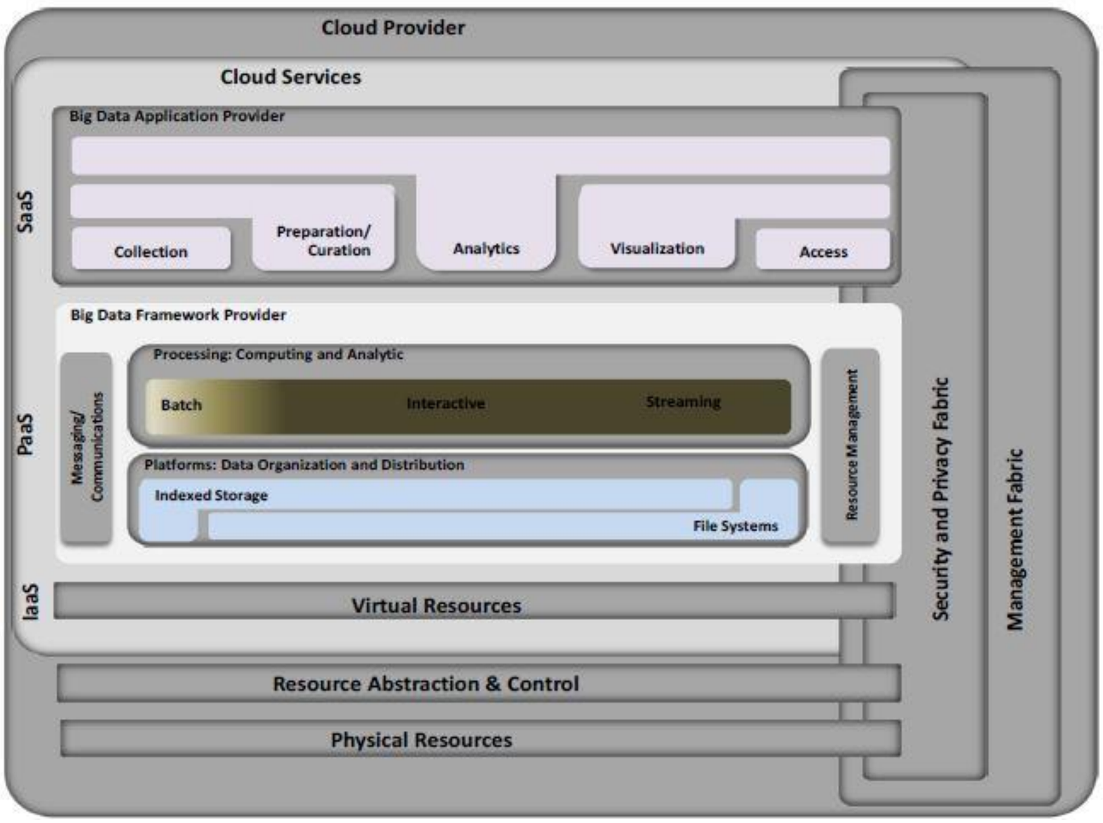


Figure 10 NIST big data architecture: cloud deployment opportunities [21].

strategy used by the scientific community is to develop their own cloud solutions. An example of this methodology is presented in [7]. The authors of this paper have developed their own cloud solution, which deals with satellite images. The development of this platform comes from the need of managing and applying computation on larger datasets. The existing solutions were not suitable for their needs. They have developed a highly flexible and extensible solution to fill the gap of managing large image datasets. It is based on an Azure cloud infrastructure. This cloud infrastructure gives to scientists a flexible and extensible solution to fit with their requirements. With this cloud strategy, their visualization and managing tools are able to provide an efficient platform, more efficient than high-end machines. The execution time is faster by a 90 factor.

Managing a scientific application in the cloud is a challenging task. It requires a specific infrastructure to provide an efficient solution for providing an effective execution workflow. In [13], they evaluate the management workflow, needed for the related data. This paper proposes an overview of existing solutions on clouds and shows that cloud infrastructures are able to provide a better control on resources, compared to a grid solution or a HPC. Their solution enables to provide efficiently the plan for the cloud system.

Efficient analysis is not only limited to hardware, it is also necessary to provide a software stack. In [8], the authors define the problematic of e-science. In this paper, they present their own solution deployed on a cloud infrastructure and which can be accessed by a web service. To validate their deployment, they perform an evaluation with three datasets: spectral, medical and chemical data. E-science is also discussed in [14]. This document discusses the feasibility of using cloud computing for the e-science domain. It highlights that it is the first time in the computing history that renting 1000 nodes for one hour has the same price of rending 1 node for 1000 hours. The authors also proposed an interesting picture (see Figure 12), which depicts the existing cloud solutions in the context of services levels (SaaS, PaaS, IaaS).

To avoid the providers' limitations, scientific communities develop their own cloud clusters. Such a kind of cluster is presented in [16]. The project was named CARMEN, and was an e-science platform designed to share specific datasets. This platform is composed by a specific set of tools, which provide improved data extraction methods. They evaluate their platform with a specific use case from neuroscience.

An alternative solution is to use a hybrid strategy based on private and public clouds. But the management of both clouds is not easy and it is necessary to use tools to perform this task. Resiliin is presented in [17]. This software enables to combine private and public clouds into a unique compute and storage instance. This tool provides all the necessary software to execute jobs using both clouds. To validate their solution, the authors do an evaluation based on existing Hadoop benchmarks. With their tool, they are able to interact with most of open-source or commercial cloud providers. Usage of the virtual cloud is performed through a web service.

We assume that our proposed architecture will be able to be deployed efficiently in a pure cloud infrastructure. The foundation software stack that we have selected will enable us to deploy our whole system on most common facilities: HPC, cloud, or HPC-cloud. Automation of the performance analyses allows deploying of an efficient platform on all IT system. Moving to another architecture will imply a small cost and resources investment.

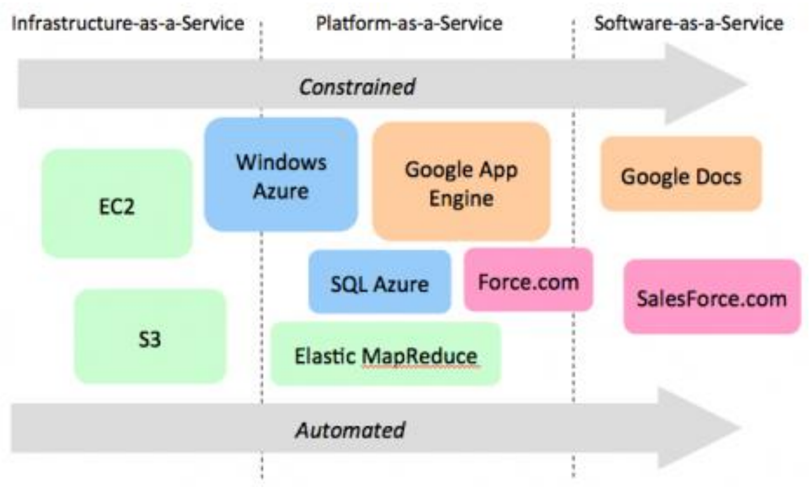


Figure 11. Example of cloud providers [21].

4. Physical setup-up of the platform

Our Big data infrastructure will be initially deployed in a HPC cluster. The next step will deploy this infrastructure into a HPC-cloud, and finally we hope to deploy it into a pure cloud infrastructure. In this section, we discuss on the deployment of some components of our solution in different architectures.

4.1. Simulation

This module is not directly linked to the VELaSSCo platform. The simulation engines are already deployed on HPC clusters and we assume that these applications are already enough optimized to fit with capacities of HPC cluster. These applications are specially designed to support multithreading and SIMD instructions. But these systems are not designed to be compatible with massive storage. And running several time steps can produce very important amount of information. Each node in a HPC cluster owns a small local storage system and if a user desired to keep a subset of produced data, he/she would have to transfer them to the network storage system.

Moving this module to a HPC-cloud will just provide new computation capabilities to the system. The efficiency of such a kind of application will depend on how application has been designed to support HPC features (MPI for massive parallelism, multi-threading, etc). Moving to an HPC-cloud has an advantage: the number of available nodes. On a HPC-cloud, more nodes are available compared to a traditional HPC environment. But the most important drawback of moving to HPC-cloud is access latency (not local nodes) and the availability of nodes (all nodes are not always available for computation; time is required to have access to enough resources).

Finally, if we move this solution to cloud, most of the performance reached by this software will drop. In fact, compute resources on pure cloud infrastructure do not offer similar performances than HPC resources. And to achieve similar performance, it will be necessary to use a larger set of nodes. This strategy only works when an application is specially designed to run efficiently on massive parallel architectures.

4.2. Storage and IO module

As stated in a previous deliverable, we have selected Hadoop as basic infrastructure for our platform. This framework will be in charge of multiple modules of the platform: storage, IO operations and job scheduling. Our infrastructure will be deployed on HPC environment. But the selected framework is designed to run on complex distributed infrastructure (with many commodities nodes). This framework has been written in Java without support of HPC feature. On a HPC system and without any tweak, this solution is not viable; in this specific case Hadoop needs some improvement to provide the most efficient platform. Hadoop parameters have to be tweak to be efficient, these modifications are operated at different level: number of computation and storage nodes, IO operator, etc. One example of a possible improvement is presented below. Our strategy studies virtualization on a single node to achieve the highest performance. An evaluation to find the most suitable configuration is presented here.

In this evaluation, we use two different methodologies. Our evaluation tool asks for all data from a data set, and asks for a subset (a filtering based on particle ID) of the dataset. The dataset is composed of information produced by simulations stored into a column-oriented format. Each file contains particle id, particle location, particle acceleration, etc. For the complete extraction our tool asks the platform for all data from a specific dataset. For the read query, only a subset of data (identify by particles id) is extracted.

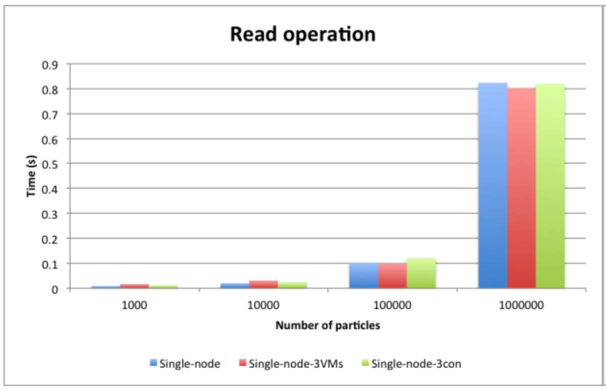


Figure 12. Read operation for NFS gateway

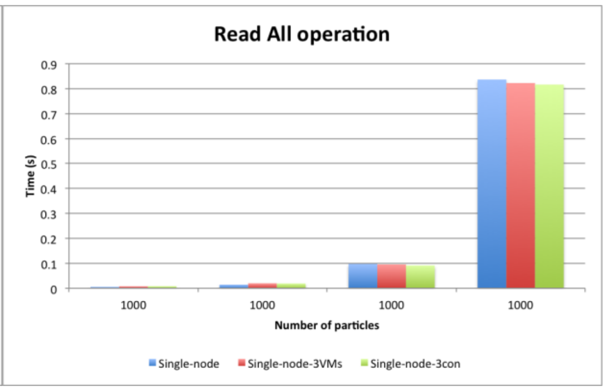


Figure 13. Read all operation for NFS gateway.

For our evaluation, we use a single node with an Intel® Core™ i7-2620M Processor (with 2 physical cores), 8 GB of memory and this machine runs a Fedora 20. For our experimentation, we have chosen three parameters: a myHadoop (which supports Hadoop 2) installation on a bare metal node, our distribution with 3 Virtualbox (version 4.3.14) machines, and our distribution with 3 Dockers (version 1.1.2) containers.

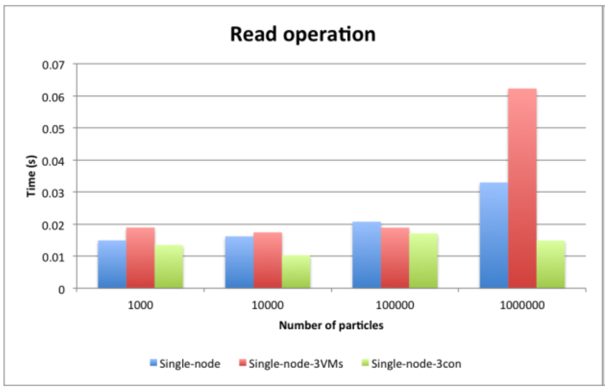


Figure 15. Read operation for Hbase

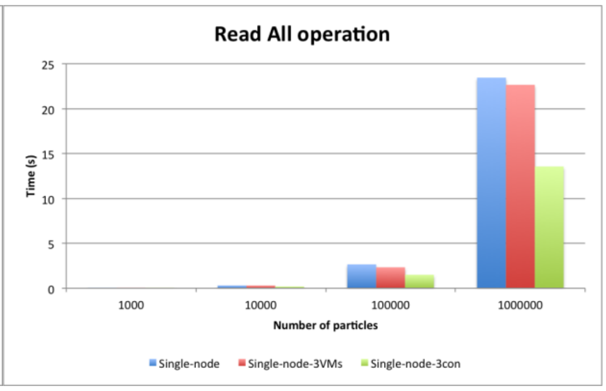


Figure 14. Read all operation for Hbase.

In this document, we only deal with simple queries, thus we compare existing extensions of Hadoop to extract content from three data access systems: NFS Gateway (linked to HDFS), HBase and Hive. The visualization tool in this project is developed in C++ and the connection to the platform is done using thrift for HBASE and Hive, and with a mounted point for NFS. For the VELaSSCo project, we have planned to use Flume as a gathering agent, but for this experiment, data is written using a different way (using Hive, HBase or file writes).

The time needed to read data using the NFS Gateway is shown in Figure 14 and Figure 13. In both cases (read all and read operations), we have to read the whole file (because data is not locally sorted). If we compare the amount of time required gathering data from VMs cluster or containers cluster, it is not so different. Thus, to gather information from a NFS gateway, no recommendation can be done for one or another strategy.

For the second benchmark, we compare HBase read with the same hardware parameters, see Figure 16 and Figure 15. For read operation (sub-selection on the data set), on a small distribution (from 1.000 to 100.000 particles), all three methods need the same amount of time to gather a subset of records. But, when the number of particle is higher (1.000.000), Virtualbox is not anymore suitable and containers become the fastest solution. For the second test (gather all data from a dataset), a similar pattern appears. The best solution is the use of 3 containers.

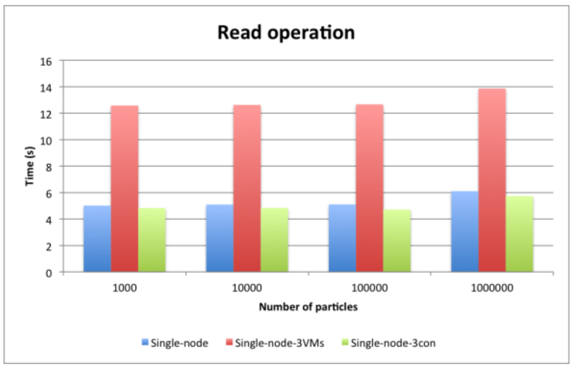


Figure 17. Read operation for Hive.

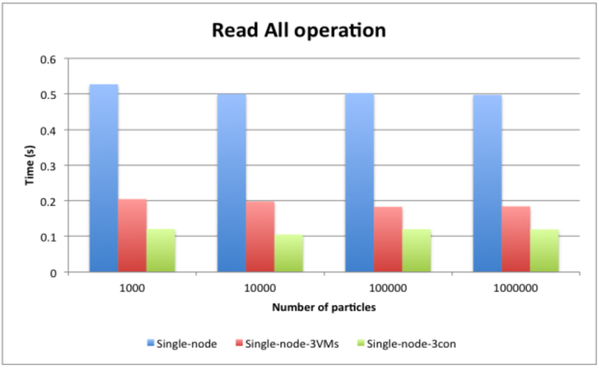


Figure 16. Read all operation for Hive.

For the third benchmark, our experimentation has been performed with the Hive plugin. Hive is a data warehouse solution built on top of the Hadoop platform. This software facilitates queries by providing a high level language called HiveQL. Result of this experimentation is presented in Figure 17 and Figure 18. For read operation, the best solution is the architecture with containers which is tied to the bare to the metal solution. Usage of pure virtual machines is not efficient. For the read-all operation, 3 containers provide the best architecture. The bare metal solution is no really efficient in this case, because Hadoop is not able to use efficiently multi-threading capabilities. Thus for this purpose, it is more efficient to use a solution based on containers.

Another advantage, which is not mentioned above, is that Hive and HBase can be combined, and Hive can query a HBase data set easily. This statement is important, because from these previous tests, a mixed solution will be necessary to extract information as fast as possible. Indeed, HBase has to be used for select operations, while Hive is more efficient to gather a complete dataset. Another statement concerns how to increase computation capabilities: with container, the solution is able to reach the highest performance (compare to bare to the metal or using VirtualBox). The NFS Gateway is not the most efficient solution; it only provides a simple way to deal with Hadoop data. The most useful feature with this gateway is the ease on data access: with HBase and Hive, it is necessary to use a specific protocol based on Thrift (but Thrift compiler produces all necessary classes to communicate with these plugins).

From these results, it is necessary to adopt a container virtualization approach to increase performance of the Hadoop platform and to avoid important modifications of the whole framework. Overhead of such a kind of virtualization system is drastically reduced compared to a real virtual machine engine. Moreover, the time needed to deploy our container platform is also drastically less important than time needed for virtual boxes. For the bare to the metal, deployment time is 1.7 seconds, while for the container distribution 8.8 seconds are necessary and for virtual box 188.2 seconds. Unfortunately, all IT systems are not suitable with virtualization and even less with containers. To be able to use the most of resources, containers have to become major features of HPC facilities. But this habit is not yet well used in HPC centers.

Finally, we have tried to evaluate performance on using more containers for read and read all operations. This statement is present in Figure 19. In this figure we see the overhead of using Hive and HBase at the same time (compared to previous strategy). And we can see that using 3 containers is the optimal solution. Results are produced faster than with other methods. For this test, Hive is used to create the HBase table.

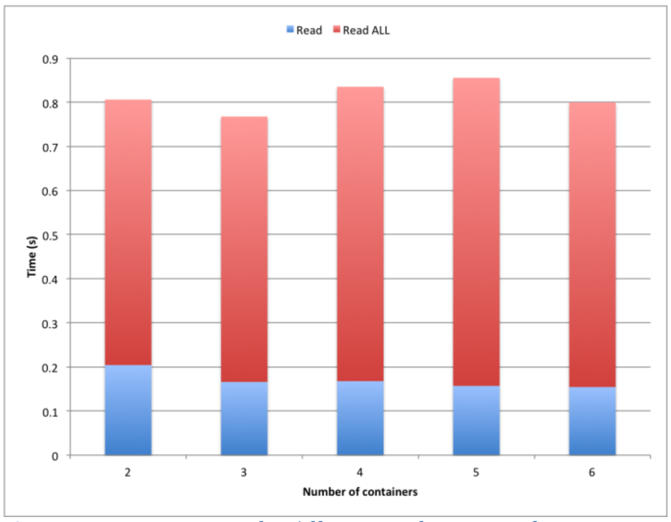


Figure 18 Time execution for different configuration of containers

On a HPC-cloud infrastructure, we plan to use this solution on a similar way than HPC. With this new deployment, we will be able to provide a larger instance of VELaSSCo. But some evaluation will be necessary in order to provide the most efficient set of parameters. To avoid complex management of these parameters, automation tools will be necessary to perform an efficient evaluation of these parameters.

When this architecture will move to the cloud, we hope to be able to reach the highest efficiency. In fact all features described in this section will be used in the most suitable condition. In this case performances will be released, and we will be able to achieve a very high paralleled computation platform for engineering data.

4.3. Networking

In a distributed environment, network is one of the most important features. This part of hardware is important to enable an ensure data transfer among nodes. In a HPC configuration, interconnection between nodes is performed using high efficient network system. But such kind of hardware has an important cost.

In the case of VELaSSCo, this high-speed networking can be an advantage to transfer large amount of data between nodes. But in our architecture, the software stack used is not developed to deal with this network transfer. Only small sets of information transit in the network. Thus in our configuration this high bandwidth capability will not be optimally used.

In an HPC-cloud environment, network speed is quite similar to traditional HPC. Thus, we will have the same statement if the correct cloud configuration is used; see [26][27][28].

Finally moving to a Cloud infrastructure will change the networking paradigm. And our solution will be more efficient due to the large set of nodes. In this configuration, traditional network devices interconnect nodes. This strategy enables to have more computes nodes and is thus more suitable for massive parallel computation.

5. Conclusions

Thanks to the flexibility of the chosen tools, the deployment of the VELaSSCo platform on an HPC-cloud environment that supports petabyte sized engineering data is feasible with small modifications that are related to the specifics of cloud services. The small modifications only refers to configuration files of the chosen tools, adjustments on the cost evaluation of the analytics queries to ensure the feedback to the user, and the corresponding modifications to the storage layer when expanding or reducing the number of used nodes in the same fashion as the traditional big data framework.

Public HPC-cloud services are charged mainly for two concepts: number of cloud-computing nodes, with their CPU and storage capacity; and the traffic and bandwidth in-and-out of the cloud cluster. Network traffic between nodes inside the same cloud cluster is usually discarded compared to the two first concepts.

For simulations running in the cloud, the deployment of platform in the same cloud as the simulation will reduce the cost of the analysis and visualization of the results, as the data generated by the calculation program is analyzed in the same cloud and only the visualization information is transferred between the HPC-cloud and the visualization client.

6. References

[1] The Apache HBase Reference Guide: http://hbase.apache.org/book/book.html
[2] http://insidehpc.com/hpc101/hpc-architecture-for-beginners/
[3] https://www.theubercloud.com/cost/
[4] http://glennklockwood.blogspot.fr/2014/05/hadoops-uncomfortable-fit-in-hpc.html
[5] http://www.admin-magazine.com/HPC/Articles/Is-Hadoop-the-New-HPC
[6] http://randomlydistributed.blogspot.fr/2014/01/saga-hadoop-20.html
[7] Li, J., Humphrey, M., Agarwal, D., Jackson, K., van Ingen, C., & Ryu, Y. (2010, April). E-Science in the cloud: A modis satellite data reprojection and reduction pipeline in the windows Azure platform. In Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on (pp. 1-10). IEEE.
[8] Hiden, H., Woodman, S., Watson, P., & Cala, J. (2013). Developing cloud applications using the e-science central platform. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 371(1983), 20120085.
[9] http://escience.washington.edu/get-help-now/comparing-performance-cloud-vs-local-clusters-green's-function-simulations
[10] https://surfsara.nl/news/sara-and-big-grid-make-hpc-cloud-infrastructure-available
[11] http://www.commit-nl.nl/projects/e-infrastructure-virtualization-for-e-science-applications
[12] Fox, G., Qiu, J., & Jha, S. (2014). High Performance High Functionality Big Data Software Stack. Big Data and Extreme-scale Computing (BDEC).
[13] Deelman, E., Juve, G., Malawski, M., & Nabrzyski, J. (2013). Hosted science: Managing computational workflows in the cloud. Parallel Processing Letters, 23(02).
[14] http://escience.washington.edu/get-help-now/understanding-cloud-computing-research-and-teaching
[15] Hey, T., & Trefethen, A. E. (2002). The UK e-science core programme and the grid. Future Generation Computer Systems, 18(8), 1017-1031.
[16] Watson, P., Lord, P., Gibson, F., Periorellis, P., & Pitsilis, G. (2008, May). Cloud Computing for e-Science with CARMEN. In 2nd Iberian Grid Infrastructure Conference Proceedings (pp. 3-14).
[17] Iordache, A., Morin, C., Parlavantzas, N., Feller, E., & Riteau, P. (2013, May). Resilin: Elastic MapReduce over multiple clouds. In Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on (pp. 261-268). IEEE.
[18] http://www.drdobbs.com/database/applying-the-big-data-lambda-architectur/240162604
[19] https://issues.apache.org/jira/browse/YARN-1964

[20] https://wiki.apache.org/hadoop/HCFS

[21] NIST Big Data Interoperability Framework: Volume 5, "Architectures White Paper Survey". Draft Version. National Institute of Standards and Technology (USA). January 2015.

[22] Apache Thrift: https://thrift.apache.org

[23] GiD: http://www.gidhome.com

[24] iFX: http:// www.i-fx.net

[25] Bias, Randy. Grid, Cloud, HPC … What's the Diff?, http://cloudscaling.com/blog/cloud-computing/grid-cloud-hpc-whats-the-diff/, retrieved 2015.03.30

[26] Iyer , Rajkumar and Prateek, Anshu. "Boosting Amazon EC2 Network for High Throughput" August 18, 2014. http://www.aerospike.com/blog/boosting-amazon-ec2-network-for-high-throughput/ , retrieved 2015.03.09

[27] Mytton , David (Server Density). "Need for speed: Testing the networking performance of the top 4 cloud providers", April 12, 2014. https://gigaom.com/2014/04/12/need-for-speed-testing-the-networking-performance-of-the-top-4-cloud-providers/ , retrieved 2015.03.09

[28] Topchiy , Serhiy. "Testing Amazon EC2 network speed", March 4th, 2013. http://epamcloud.blogspot.com.es/2013/03/testing-amazon-ec2-network-speed.html , retrieved 2015.03.09

[29] Fortissimo EU project, http://www.fortissimo-project.eu/project/the-project.html