A Hadoop distribution for engineering simulation

Benoit Lange, Toan Nguyen

A Hadoop distribution for engineering simulation

Benoit Lange
Project Opale
INRIA Rhône-Alpes
38334 Saint-Ismier, France
benoit.lange@inria.fr

Toan Nguyen
Project Opale
INRIA Rhône-Alpes
38334 Saint-Ismier, France
toan.nguyen@inria.fr

***Abstract*— In this paper, we discuss on the VELaSSCo project (Visualization for Extremely LArge-Scale Scientific Computing). This project aims to develop a specific platform to store scientific data for FEM (Finite Element Method) and DEM (Discrete Element Method) simulations. Both of these simulations are used by the engineering community to evaluate the behavior of a 3D object (for example fluid simulation in a silo). These simulations produce large files, which are composed of different time steps of a simulation. But the amount of produced data is too big to fit into a single node. Some strategies decompose data between nodes, but after several time-steps, some data has to be scratched to free memory.**

In this project, we aim to develop a platform, which enables the scientific community to store huge amounts of data on any kind of IT systems. We target to store data on any IT systems because most of scientists have access to modern computation nodes and not huge storage nodes. Our platform will try to fill the gap be-tween both worlds.

In this paper, we give an overview of the VELaSSCo project, and we detail our platform and deployment software. This platform can be deployed on any kind of IT system (dedicated storage nodes, HPC nodes, etc.). This platform is specially designed to store data from DEM and FEM simulations. In this paper, we present a performance analysis of our deployment tool compared to the well-defined myHadoop tool. With our tool we are able to increase computation capabilities with containers and virtualization.

Keywords*— *Hadoop ; HPC ; commodity nodes; visualization; Hive ; Hbase ; virtual machines; linux containers.

I. INTRODUCTION

Since a long while, scientists have tried to understand natural phenomena. In the earlier days of science, researchers described natural phenomena with experimentation. Later, theories have been proposed by scientists to describe phenomena (Newton's laws, Maxwell's equations...). And, at the beginning of the 1980, computational models have been developed to validate theories. These models are validated by computer simulations. Computational models and IT hardware have evolved and bring understanding to a higher level. With modern architectures, a computation takes only a couple of hours, but with efficient computations comes huge data. All the information produced by these high performance systems cannot be stored into traditional storage devices (storage systems have not followed the same path as computation units): scientists have to filter information. For the engineering field, filtering means scratching some elements: none relevant data, intermediary time-steps, etc. Unfortunately, this methodology leads to important loses of information, and analysis cannot be optimal. From this statement, it is necessary to adopt a new strategy to improve understanding on data.

This problem has already been introduced in [1]. In this book, the authors present their point of view concerning the evolution of computational research. Large multi-disciplinary data will dominate the future of science. These methods will unify theory, experimentation and simulation. This strategy is based on exploration of large data sets with efficient data mining tools. This new paradigm is mainly based on big data solutions, which provides all the necessary tools to extract content from large data sets.

This paper is related to the VELaSSCo (Visualization for Extremely LArge-Scale Scientific Computing – EC FP7) project. The goal of this project is to deal with large data sets produced by specific engineering simulations. For example the LSST produces every night 30 Terabytes of information [4]. Pure theoretical models can also compute large datasets; an example is presented in [2]. In that paper, a simulation method for astrophysical simulation is presented; it is a *N*-Body piece of software. It is based on the Dehnen algorithm [3]. For a plummer distribution, with 10 millions particles, this simulation engine produces 500 Megabytes of data for each time step, that is computed in 1.19s. In a near future, the research software will produce more and more data, thus it is necessary to anticipate this evolution and provide an efficient storage solution to deal with next generations of produced data. In our project, our interest is only focused on specific datasets produced by FEM and DEM simulations.

For the infrastructure of our platform different strategies are suitable: we choose to use an existing well-used framework. This choice was lead by different points: extensibility, scalability, reduction of development cost, etc. Our platform is composed of a custom Hadoop distribution, with some specific plug-ins, designed to store FEM and DEM data. This framework has been chosen due to its extensibility: this framework is used as a skeleton for an open-source and a closed-source distribution (composed by a commercial storage software). This skeleton is extended with some plugins at different levels (storage gathering, query). As stated before, this project has to be suitable for any kind of IT system (from HPC to dedicated cloud storage), and a particular focus on

visualization. The infrastructure availability point is crucial because most of IT facilities available for scientists are HPC systems. Unfortunately, the Hadoop framework is mainly developed for commodity storage nodes. Thus, we provide a deployment tool to fit with any kind of IT ecosystem. This tool uses a description (stored into a XML file) of the desired deployment.

In this paper, we present the VELaSSCO project, and our deployment tool. Then we provide a performance evaluation of performance achieves by a Hadoop distribution that uses our tool and a Hadoop distribution based on myHadoop.

Section II is an overview of related work on scientific data and Big Data. Section III presents the VELaSSCo project and how our tool can deploy a Hadoop distribution. Section IV is dedicated for experimental results. And Section V concludes this paper.

II. RELATED WORK

In this section, we present some work related to the BigData and the Hadoop framework.

BigData can be represented by 3 dimensions, named 3Vs rules: Volume, Velocity and Variety (these dimensions are sometime extended to 5V with Value and Veracity). Volume means size of dataset (bytes, number of records, etc.). Velocity is related to the gathering speed of information (batch, real-time, etc.). Variety concerns the data format (structured, unstructured, etc.). These dimensions are presented in [5][6]. The VELaSSCo project is linked to these different dimensions, because:

- For the future simulation engines, we expect to produce data in near real time (with a response time inferior to 100 ms [7]).
- Volume of produced data will be also very important (more than 1 Petabyte)
- Variety will be achieved by computation on different simulation engines.

Some Big Data solutions have already been developed and widely discussed in different research fields, but engineering data has not yet been well evaluated regarding to the BigData needs.

Google is one of the major BigData leaders. They have reintroduced an existing computational model specially adapted for their web search; tools developed have been presented into different papers: [8] concerns the MapReduce programming model (MR), [9] deals with BigTable and [10] introduces its own virtual File System (Google FS). GFS is a pure virtual file system, where storage is distributed among different nodes. A file is splitted into small files, distributed and replicated among the storage nodes. The hierarchical structure of the file is stored on another node. This FS is largely linked to the computational model named Map Reduce. This model is able to extract information from a large set of files using a distributive decomposition for the computation: a Map and a Reduce function. The Map function is used to extract content from small files and the Reduce phase extract information from intermediate files produced by the Map phase. Google has also developed another tool to store more structured data, it is named BigTable. This tool is a table oriented storage system based on the GFS. Unfortunately, all mentioned tools are not freely available, thus it is not possible to deploy a private Google BigData ecosystem.

Other groups have developed some alternative solutions, some of these solutions are closed source and other are fully open source. One of these tools is Hadoop. It is the most used big data framework [11]. It is an open-source framework, which support all the big data requirements. The earlier implementation of this platform includes a specific file system named HDFS (HaDoop File System) [12] and the MapReduce [13] computational model. This framework is highly extensible using plugins. An example of these plugins is Hbase, an open source implementation of the Bigtable piece of software [14][15]. Other tools have been proposed as Hadoop alternatives, an example is Dryad [16]. But most of them are not anymore maintained. Dryad provides a more complex model than the traditional MapReduce model. With Dryad, it is possible to add intermediary layers between the Map and the Reduce phases. This methodology enables more complex computations. But now, to fit with most of existing BigData IT systems, Microsoft provides a Hadoop implementation of Dryad. This implementation is based on the new Hadoop computational model named YARN [17]. YARN is included in Hadoop 2, and splits the *JobTracker* of Hadoop into two separate functions: *RessourceManager* and *ApplicationManager*. With this methodology performance of Hadoop have been increased, and MapReduce is not anymore the only suitable computational model for Hadoop. Now, it is possible to deploy any computational model over a Hadoop ecosystem. The new Hadoop framework can run over a huge amount of commodity nodes: deployment with more than 10.000 nodes, see [18].

Major IT companies now provide their own Hadoop distribution, see Intel [19], Microsoft [32] for example. These distributions are specially developed to fit with some requirements. For example, the Intel distribution is tweaked to be suitable with Intel Hardware. All parameters from the motherboard to Hard disk are described on the documentation of the Intel Hadoop ecosystem. These parameters are presented to provide the best solution to reduce energy consumed by their Hadoop distribution.

Other providers have all the necessary tools to deploy a complete Hadoop ecosystem. Two of the major's contributors of Hadoop provide their own distribution and sandbox with specific pre-installed plug-ins; see Hortonworks [20] and Cloudera [21]. Both sandboxes are based on virtual machines. Virtualization has been massively used in big data infrastructures to increase the computation capabilities without the needs of modification for the software part. The Standard version of Hadoop does not support efficiently new features of modern CPU like: SSE or multithreading. Thus to increase computation capabilities, the usage of virtual machines can be an efficient method. Most of sophisticated deployment tools use virtual machines see Ambari [22]. Ambari is one of the most advanced tools to deploy a Hadoop ecosystem. All nodes can be configured through a web UI.

Software companies, which are involved in virtualization also provide their own specific Hadoop distribution based on their virtualization tool, see VMware [23].

Hadoop has been mainly developed to be deployed on commodity servers. Some studies have been focused on support of modern hardware capabilities, for example accelerators [24] or using specific CPU features [25][26]. Commodity facilities are not the most deployed infrastructure, and most scientists have only access to HPC centers. HPC systems have some specifications: local files are erased at regular interval; a scheduler shares the available resources, etc. Thus, to be suitable with their IT system, some Hadoop tools have been proposed. Most of these strategies redevelop a big data framework with a dedicated MapReduce computation; see [27][28][29][30]. But these solutions are in most cases not anymore supported or they are not available to download. Moreover, these solutions are specifically designed to run on HPC file systems like Lustre. An alternative solution based on Hadoop has been presented in [31], and named myHadoop. They developed a deployment tool, that builds an on demand Hadoop (permanent or temporary) ecosystem on the top of a HPC facility. This distribution supports two strategies: a temporary approach with an important overhead due to file transfers, the second method with permanent data. In both cases, this Hadoop distribution is not able to provide the best performance, because it does not support natively usage of multicore capacities. For our project, we decide to extend this method by providing a more configurable Hadoop distribution.

In the Section 3.A, we will present the VELaSSCo project, and in the Section 3.B, the architecture of the platform. We then describe how our development platform works, and how to configure it.

III. THE VELASSCO PLATFORM

This section detailed the VELaSSCo project and the architecture of our distribution.

A. The VELaSSCo project

The European Union through the Seventh Framework Programme (FP7) has funded this project. Different companies (ATOS and JOTNE) and laboratories/universities (UEDIN, Franhaufer, SINTEF, CIMNE and INRIA) are involved in this project. This project regroups people from different fields to reach a common goal: provide an innovative and efficient platform to store engineering data produced by simulations.

For this project, three domains are targeted:

- Big Data infrastructure,
- Data analytics,
- Visualization.

For the Big Data infrastructure, we have planned to provide an efficient platform to store our data. This platform has to be suitable with all kind of IT architectures (HPC to commodity servers). This statement is important for scientists; most of scientific IT architectures are composed of HPC facilities, which are not dedicated to support big storage. High

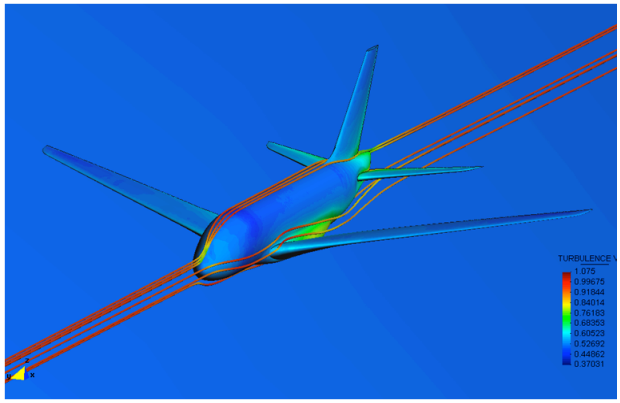

Figure 1. Streamline of an airplane.

performance nodes mainly compose HPC and storage is performed on external systems (for example GPFS [40]). HPC nodes have local storage, but with a limited space. These nodes are interconnected with specific high-speed networks. And computation on these nodes is managed by as schedulers (Slurm, Torque, …).

At the opposite, IT systems for big data infrastructures are composed of large sets of commodity nodes with large storage attached locally. These systems use standard network hardware. They are highly distributed: each local storage and local compute unit are viewed as a single machine. To aggregate information from different nodes, a distributive file system is implemented on the top of the existing storage system; it is a pure virtual file system. With this strategy, each node stores a part of the data, and to achieve higher performances, the computation is spawned on slave nodes (specific computation nodes). This strategy has been implemented for several frameworks, but the most common one is Hadoop. It provides a complete framework to deploy an efficient distributive environment.

In the next section, we will present our Hadoop distribution that enables the support of most of IT systems.

By 2020, complex simulations will produce more data than ever before. Global evolution of IT ecosystems will lead these complex computations to run on cloud IT systems (more storage capabilities than traditional IT system). Moreover, production of data is not anymore a bottleneck (CPU can achieve high GFLOPs). Now, the scientific community has to study storage bottleneck. It is necessary to find new solutions to store more and more data (reach higher number of I/O operations). Today, scientists have to filter part of the produced data to reduce the amount of stored information. Discarding part of the information can lead analyze to a global mistake, and also accuracy is reduced. With our platform, we aim to not discard any information and keep all time-steps of a simulation to bring a better understanding of computed data.

Storage is not the only target of our platform; we also aim to include some analytic features. Future Big Data platforms have to deal with advanced analyses on data. These platforms will also be used to produce information from the stored data. In our project, this data extraction can be classified into two categories: complex analysis and simple analysis. Simple

analyzes consist in extracting information directly from the raw data, without any computation. For complex queries, it will be necessary to apply extra computations on raw data. Regarding to the needs of the visualization part, this extra computation is time consuming. Some examples of this computation is: filtering data, extraction of splines, extraction of streamlines, etc. Figure 1 presents an example of a complex query: extraction of streamlines.

The last part of this project concerns data visualization. Our platform has to be suitable with some specific use cases:

- Providing some data chunks,
- Providing a data stream.

For the first statement, data is extracted in batch. A query is performed among the platform, and all results produced by this query are transmitted into one data element. With this strategy, all data needed by the viewer is transmitted once. This methodology can be considered as ”offline” visualization, this method is suitable with huge amount of data.

For the second methodology, data is extracted in real-time. The visualization software performs online queries, and the platform transmits data in real-time. With this approach, data needs to be stored using a specific multi-resolution strategy (different levels of detail of a data set). Thus, data can be streamed to the visualization tool, which gives a smooth rendering to the user. Real-time interaction of our platform will be discussed in a future paper.

B. The architecture of our platform

Our platform has a lot of requirements, and we will describe a subset of these. The VELaSSCo platform needs to:

1. Be deployed on any kind of IT architecture (from HPC to big data solution),
2. Support storage of files produced by simulation engines (large or small files),
3. Support extensibility. For example, we want to use EDM DB (JOTNE software) as a storage solution for our Big data architecture,
4. Support computation, IT systems used for our platform will not only be used for storage,
5. Support offline and real-time queries,

The Hadoop framework covers most of the requirements of this project. We provide a specific extended Hadoop distribution.

More precisely, we provide two different versions: an open-source and a close-source version. In the next section, we present some implementation details of our preliminary work. This version will cover point 1, 2, 3 (with already existing plug-ins) and 5 (batch part).

For the first point, we present how an user can deploy a Hadoop ecosystem with our deployment tool, on any platform. This deployment can be done using bare metal computation nodes, or virtualization.

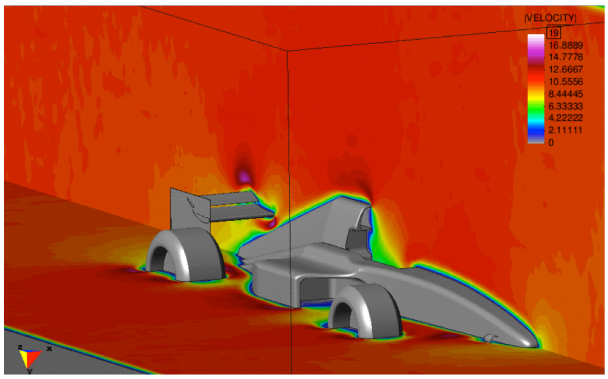

Figure 2. F1 simulation using FEM computation.

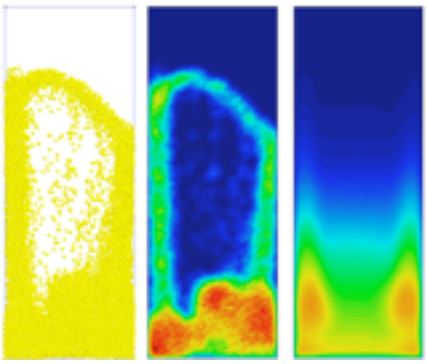

Figure 3. Fluid simulation (DEM)

For the second statement, we have performed an evaluation of storage strategy to find the most relevant method to store data for FEM and DEM simulations. An example of produced file from FEM simulation is presented in Figure 2 (the rendering has bee produced by CIMNE). This simulation deals with the decomposition of space using a mesh structure. For the DEM, an example of rendering is presented in Figure 3 (the figure has been produced by the University of Edinburg). Both of these solutions produce large files, for example: with 10 millions particles and 1 billion of time-steps, DEM needs 1 Petabytes of storage. For 1 billion elements with 25000 time-steps, FEM needs 50 TB of free space. Currently, all the data produced by these simulations are simply not stored, and several time-steps are deleted from the storage devices. FEM and DEM simulations store data in a similar way. Output data is composed of raw data, composed of different rows: ID, velocity, acceleration etc. An example of produced file (for a DEM simulation) is presented below:

```
TIMESTEP PARTICLES

0 868019

ID GROUP RADIUS MASS PX PY PZ VX VY VZ Angular_Velocity_X Angular_Velocity_Y Angular_Velocity_Z

1 1 5.3705e-05 0.198709 0.0233948 -0.105028 0.0233836 -3.56961e-05 -5.10717e-05 9.75788e-06 0.00106043 0.000390757 -0.000536836

2 1 5.74099e-05 0.212416 0.0865132 -0.104191 0.0238821 -0.000101408 -5.40777e-05 - 0.000558674 - 0.0015792 -0.00176641 0.00125497

…
```

This output format is quite simple, and used by many simulation engines. A file storage system seems to be the perfect solution to store directly raw data.

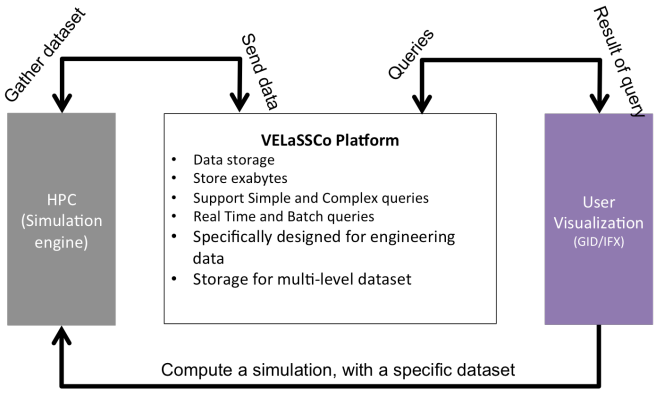

Figure 5. Overview of the components of the VELaSSCo platform.

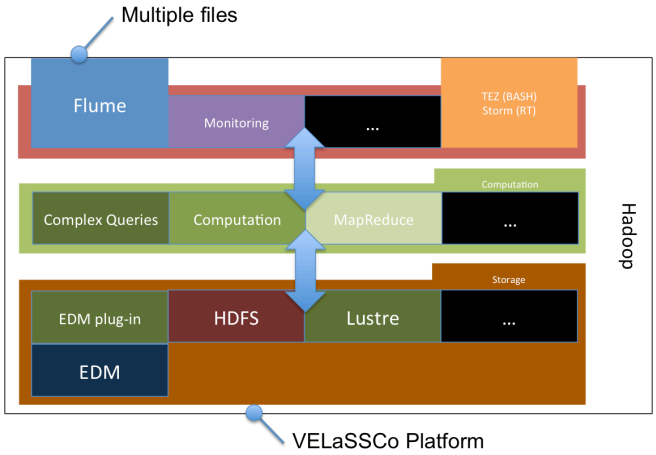

Figure 4. Overview of the VELaSSCo platform.

For the third point, we have evaluated the possibility to use existing extensions (like Hbase or Hive) to improve read and write operation of a simple query.

In the next section, we will present our benchmark for Hive, Hbase and the NFS gateway to perform simple queries on a myHadoop distribution, and on our own distribution.

For the last point, we only have studied the possibility to execute batch queries, so far the real-time part needs a deeper study (for example with Spark) or maybe to develop a specific plugin for visualization queries.

IV. IMPLEMENTAION DETAILS

As myHadoop, we provide a Hadoop deployment tool. Our tool is an extended version of the myHadoop one; it provides all the necessary code to instantiate a VELaSSCo platform on most IT systems. Communication between computational nodes, users and the storage platform is presented in Figure 4. An user starts a simulation on the compute nodes (gray box). Then compute nodes send some data to the VELaSSCo platform (this data transfer is assumed by FLUME agent). The platform stores this information into its file system (virtual or not). After this step, the users are able to gather parts of the data through specific applications.

In Figure 5, we present a detailed view of the VELaSSCo platform. Different layers compose it: a communication layer with external elements (users or computational nodes), a computational layer and a storage layer. Two distinct parts compose the communication layer: the Flume agent (that gather information from HPC) and the user communication part. For the user communication pipeline, numerous extensions are available to allow communication with Hadoop (in the final stage of our platform, only one query layer pipe will be open to the user, and a decomposition of a query will be performed by this layer). In the next section, we have benchmarked three of them: the NFS Gateway, Hbase and Hive. Interfacing with Hbase or Hive is done through Thrift [33]. Actual queries for the distribution are simple, complex queries are not yet implemented. Thus, with such simple queries, the traditional Hadoop computational model is enough. Finally, for the storage part, Hadoop is already supporting different file systems: HDFS (a pure virtual FS), Ceph [34], NFS, Lustre, etc. Due to some requirements from our partners, we plan to propose a Hadoop extension that supports EDM Data Base also.

Our deployment tool is an enhanced version of myHadoop. This software has inspired our deployment tool, because it is quite efficient for deploying a Hadoop ecosystem over a HPC facility. Our tool enables to deploy a Hadoop cluster on any kind of IT system. For this purpose, users describe their desired IT architecture in a single file. Through this XML file, users can select nodes where the plugins will be deployed, where and which kind of storage solution will be used, if they want to use virtualization, if a task scheduler controls nodes, etc. The launching script currently assumes that all machines used in a previous Hadoop session are available (integrity of the dataset was to yet controlled). In Figures 6 and 7, we present two examples of XML files.

In the first example (see Figure 6), we deploy a VELaSSCo architecture on two physical nodes. To increase computation capabilities (use multi-threading), we deploy on each nodes virtual machines (networking between virtual nodes is done through a Virtual Network). In the first node, Dockers manages virtual machines, while in the second node, Virtualbox is in charge of the management of virtual machines. As shown in this example, it is possible to mix virtual nodes and real nodes; *Shark* is in charge of all services (Master node, Hive, Hbase, NFS gateway and Pig manger). And, all other nodes (the virtual nodes and *Shok*) are considered as storage / compute units.

In the second example (see Figure 7), each node is used without any virtualization engine. To reduce execution overhead, all services are dispatched among the different nodes.

```
<?xml version="1.0" encoding="UTF-8"?>
<VELaSSCo>
    <!-- Loc ressrouces Hadoop, Hive, ... -->
    <path>
        <install>/local_home/lange/distrib</install>
        <tools>/local_home/lange/tmp/tools</tools>
    </path>
    <real>
        <node>

            <host>shark</host>
            <hadoop>
                <master></master>
                <hbase></hbase>
                <hive></hive>
                <pig></pig>
                <NFS></NFS>
            </hadoop>
            <virtual>
                <dockers>
                    <host>velassco01</host>
                    <hadoop>
                        <FS>
                            <kind>HDFS</kind>
                            <path>/tmp</path>
                        </FS>
                    </hadoop>
                </dockers>

                <dockers>
                    <host>velassco02</host>
                    <hadoop>
                        <FS>
                        </FS>
                    </hadoop>
                </dockers>
                <dockers>
                    <host>velassco03</host>
                    <hadoop>
                        <FS>
                        </FS>
                    </hadoop>
                </dockers>
            </virtual>
        </node>
        <node>
            <host>shok</host>
            <hadoop>
                <FS>
                </FS>
            </hadoop>
            <virtual>
                <vbox>
                    <host>velassco11</host>
                    <ip>192.168.64.102</ip>
                    <hadoop>
                        <FS>
                        </FS>
                    </hadoop>
                </vbox>
                <vbox>
                    <host>velassco12</host>
                    <ip>192.168.64.103</ip>
                    <hadoop>
                        <FS>
                        </FS>
                    </hadoop>
                </vbox>
            </virtual>

        </node>
    </real>
</VELaSSCo>
```

Figure 6. Example of a mixed distribution.

```
<?xml version="1.0" encoding="UTF-8"?>
<VELaSSCo>
    <!-- Loc ressrouces Hadoop, Hive, ... -->
    <path>
        <install>/local_home/lange/distrib</install>
        <tools>/local_home/lange/tmp/tools</tools>
    </path>
    <real>
        <node>

            <host>shark</host>
            <hadoop>
                <master></master>
                <pig></pig>
                <NFS></NFS>
                <FS>
                    <kind>local</kind>
                    <path>/mnt/tmp</path>
                </FS>
            </hadoop>
        </node>
        <node>
            <host>shok</host>
            <hadoop>
                <FS></FS>
                <hbase></hbase>
            </hadoop>
        </node>

    <node>
        <host>sheep</host>
        <hadoop>
            <FS></FS>
            <hive></hive>
        </hadoop>
    </node>
</real>

</VELaSSCo>
```

Figure 7. Example of bare metal distribution on multiple hosts.

With the XML file of our deployment tool, we can describe two kinds of "virtualization":

- Nodes can contain "Virtual" nodes: our distribution can deploy virtual machines based on the Virtualbox engine[35].
- Nodes can contain "Container" nodes: our distribution can provide container machines based on the *Dockers* tool [36].

The goal of this virtualization step is to increase the computational capability of a node without modifying the Hadoop software stack. The standard version of Hadoop is not well suited to run in a multi-threading environment. With "virtualization", computation capabilities can be increased. In future versions of our platform, we hope to enhance our tool and use other virtual engines or container engines.

Our deployment tool is composed of a script written in Perl that provides all necessary tools to create a full Hadoop distribution. This script produces all configuration files and the description of the architecture. It provides also a starting script for the Hadoop ecosystem instead of the traditional *start-all.sh* script.

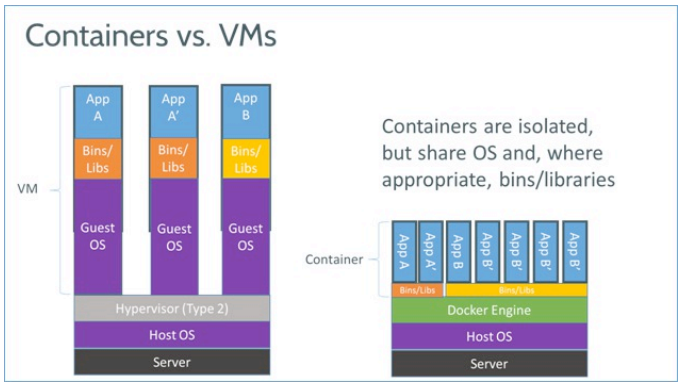

Figure 6. Containers vs VMS, from [41]

As a reminder, containers are a specific operating system-level virtualization, which provides methods to run multiple isolated Linux OS on a single host. Compared to virtual machines, different layers are removed from the computation stack. It is not anymore necessary to run a guest OS, and applications directly interact with the OS through a container engine. Computation overhead is thus drastically reduced. An overview of VMs and containers is presented in Figure 6.

In the next section, we present an evaluation of our platform compared to the myHadoop distribution [2].

V. PERFORMANCE AND EVALUATION

For this evaluation, our measurement protocol is based on read requirements of simple queries in the VELaSSCo project. Some pieces of software have been used to benchmark applications like: HiBench [37] or Hadoop Blast [38]. But some of these tools are too old and require too much work to be suitable for our Hadoop distribution. Thus, our evaluation has only been focused on our requirements, and two queries are studied: extract a whole dataset from a recent simulation and extract a subset of the same dataset.

As stated before, VELaSSCo will deal with two specific kinds of queries: complex and simple ones. Complex queries will be developed later. An example of such a kind of computation is spline extraction.

For simple queries, the platform will gather information directly from the data set, without any transformation.

Two methodologies have been used for this evaluation. Our visualization 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.

For our evaluation, we use a single node with an Intel® Core™ i7-2620M Processor, (with 2 physical cores) and 8 GB

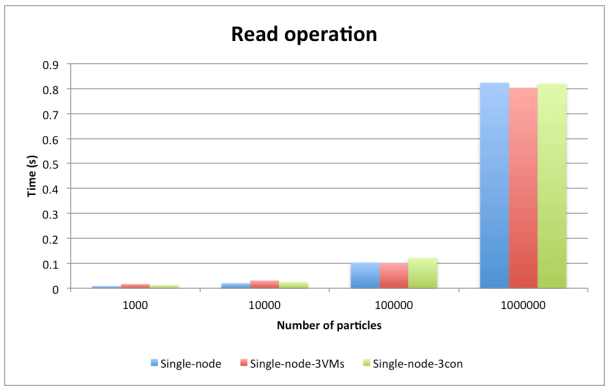

Figure 9. Read operation for NFS gateway.

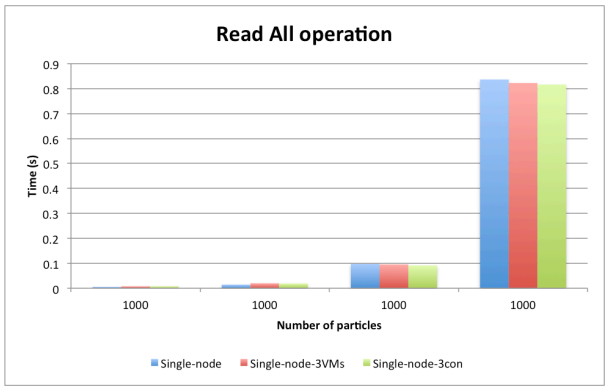

Figure 10. Read all operation for NFS gateway.

of memory, 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.

For this paper, 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 [33] 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).

Figure 9 and 10 show the time needed to read data using the NFS Gateway. 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 11 and 12. For read operation (sub-selection on the data set), on a small distribution

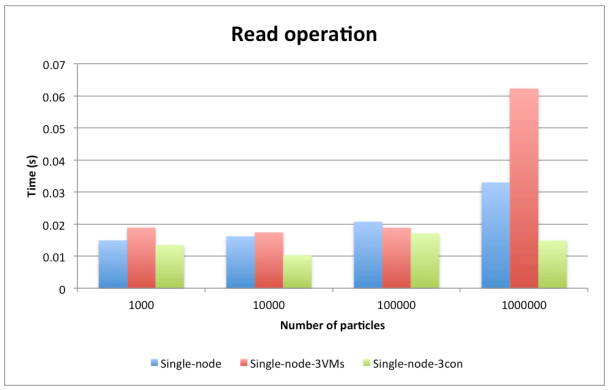

Figure 11. Read operation for Hbase.

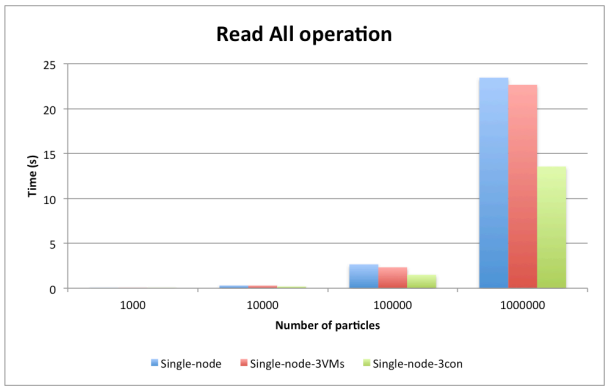

Figure 12. Read all operation for Hbase.

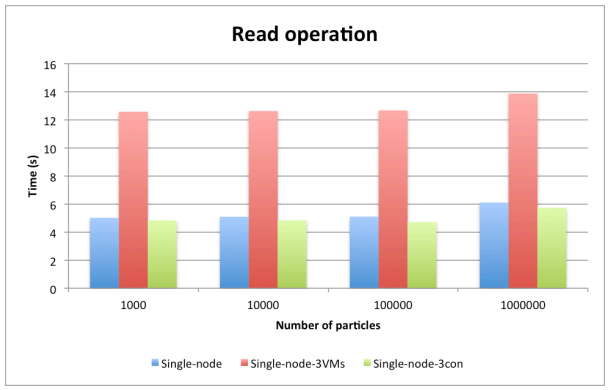

Figure 13. Read operation for Hive.

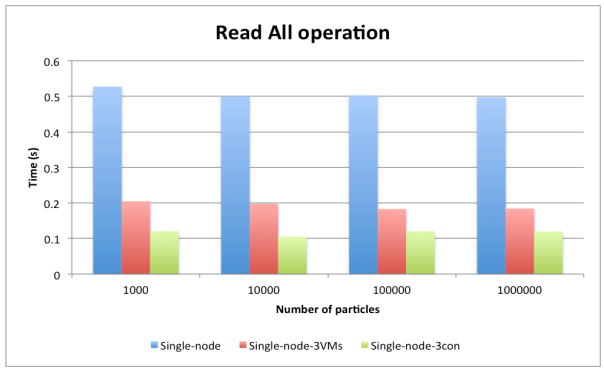

Figure 14. Read all operation for Hive.

(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.

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 13 and 14. For read operation, the best solution is the architecture with containers, 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 performances of the Hadoop platform and to avoid important modifications of the whole framework. This strategy is validated by [39]: 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 more 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.

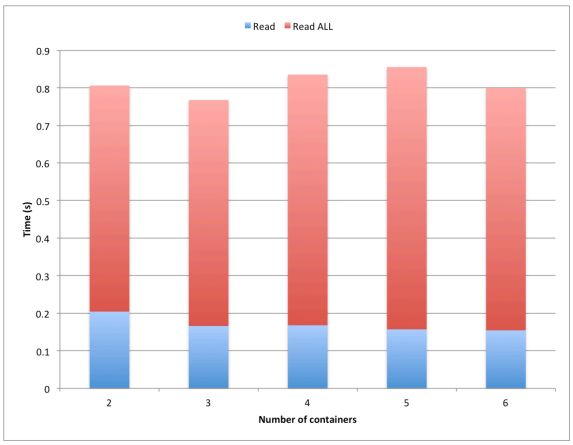

Figure 15. Time execution for different configuration of containers.

Finally, we have tried to evaluate performance on using more containers for read and read all operations. This statement is present in Figure 15. 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.

VI. CONCLUSIONS AND PERSPECTIVES

In this paper, we provide a brief overview of the VELaSSCo project and a new Hadoop distribution, which has been designed to answer the VELaSSCo requirements.

This European project aims to develop a new kind of storage platform specifically designed to store data from scientific computations. Scientific solvers produce huge amounts of data.

In this project, two specific kinds of tools are used to produce data. The first one is dedicated to FEM simulations and the second one deals with DEM simulations. In both cases, these simulation engines produce large files, which can contain different information and time-steps. These simulations are mostly run over HPC systems. This kind of system is designed to be efficient for extreme computing, but storage is often limited by overhead. In the literature, solutions to store big data are based on usage of a specific storage cluster [10]. Commodity nodes compose this cluster. Unfortunately, most of scientists do not have access to this specific kind of IT infrastructure, they only own HPC facilities. Thus, for this project, we have designed a deployment tool for a Hadoop system, that can be configured easily to be deployed on any kind of IT infrastructure.

For this purpose, our deployment tool is based on a description of the IT architecture. This description of architecture concerns all critical points of a Hadoop ecosystem: which plugins (NFS gateway, Hive, Hbase), where data is stored, which kind of storage system (HDFS, GPFS, ext4, …).

To increase the computation capability of the ecosystem, we also provide to the user the possibility to deploy virtual machines on the top of the IT system. In the current version, virtualization can be performed using different virtualization engines: a container (with *Dockers*) and a virtual engine (with Virtualbox).

The XML file allows deploying all necessary tools over the desired machines. With our distribution tool, deployment of different plugins is facilitated and thus we have been able to benchmark different components on a specific architecture composed of one node. The most efficient solution is composed of two plugins: Hive and Hbase. Each plugin is used for a specific use case: Hive read all operation on the dataset, while Hbase can be used to select a subset of the data set. Moreover, to have the best response time, the best architecture needs to use containers as virtualization engine. As stated in [39], virtualization software is not so efficient, and utilization of containers seems to be more fruitful.

Some future works are also necessary. Current development branch does not yet support connection of Virtualbox or *Dockers* containers with external nodes. This problem is not discussed in this paper but in a future release of our tool, virtual interconnection between nodes will be managed. In the current version, user has to set all necessary information regarding to Hbase and Hive, we have not yet chosen the storage structure (this step will be discuss in few months). We have not yet accessed to remote a computation cluster (located in Edinburg) for VELaSSCo, thus tests have been performed on desktop nodes.

More important statements have to be discussed in this section. In most HPC centers, users cannot deploy virtual machines (using containers or virtual engine). IT administrators have to provide these features to the scientific community to be able to use the maximum computation capacity available on their nodes.

Moreover such a system has to evolve, HPC nodes are not attached with a large file system, and storage is performed through specific storage nodes (connected with a virtual network). To increase performance of Hadoop ecosystems on a HPC cluster, evolutions have to be performed.

Finally, we have to perform tests on different operating systems, and improve the user interface of our deployment tool. Moreover data sent by this platform is not yet in real-time and some extensions will be necessary with STORM to fit with all the requirements of the visualization tools.

ACKNOWLEDGMENT

This work was supported in part by the EU FP7 project VELaSSCo, project number: 619439, FP7-ICT-2013-11.

We also want to thanks the members of the consortium, CIMNE, UEDIN, SINTEF, Fraunhofer, JOTNE and ATOS.

REFERENCES

[1] S. Tansley & K. M. Tolle, The fourth paradigm: data-intensive scientific discovery, 2009.

[2] B. Lange & P. Fortin, "Parallel dual tree traversal on multi-core and many-core architectures for astrophysical n-body simulations," Euro-Par 2014, 2014.

[3] W. Dehnen, "A hierarchical o (*n*) force calculation algorithm,"Journal of Computational Physics, vol. 179, no. 1, pp. 27–42, 2002.

[4] C. F. Claver, D. W. Sweeney, J. A. Tyson, B. Althouse, T. S. Axelrod, K. H. Cook, L. G. Daggert, J. C. Kantor, S. M. Kahn & V. L. Krabbendam, "Project status of the 8.4-m lsst," in *Astronomical Telescopes and Instrumentation*. International Society for Optics and Photonics, 2004, pp. 705–716.

[5] W. Fan and A. Bifet. Mining big data: current status, and forecast to the future. ACM SIGKDD Explorations Newsletter, 14(2):1 to 5, 2013.

[6] D. Laney. 3d data management: Controlling data volume, velocity and variety. META Group Research Note, 6, 2001.

[7] J. Dean and L. A. Barroso. The tail at scale. Communications of the ACM, 56(2):74 to 80, 2013.

[8] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. Commun. ACM, 51(1):107 to 113, Jan. 2008.

[9] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008

[10] Ghemawat, S., Gobioff, H., & Leung, S. T. (2003, October). The Google file system. In ACM SIGOPS Operating Systems Review (Vol. 37, No. 5, pp. 29-43). ACM.

[11] K. Saroj. Idc report to hadoop leads the big data analytics tool for enterprises - http://cloudtimes.org/2013/11/06/idc-report-hadoop-leads-the-big-data-analytics-tool-for-enterprises/.

[12] D. Borthakur, The hadoop distributed file system: Architecture and design. Hadoop Project Website, 11, 21, 2007.

[13] C. Lam, Hadoop in action. Manning Publications Co., 2010.

[14] M. N. Vora, Hadoop-HBase for large-scale data. In Computer Science and Network Technology (ICCSNT), 2011 International Conference on (Vol. 1, pp. 601-605). IEEE, 2011.

[15] R. C. Taylor, An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. BMC bioinformatics, 11(Suppl 12), S1, 2010.

[16] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. ACM SIGOPS Operating Systems Review, 41(3):59 to 72, 2007.

[17] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. Oâ€™Malley, S. Radia, B. Reed, and E. Baldeschwieler. Apache hadoop yarn: Yet another resource negotiator. In Proceedings of the 4th Annual Symposium on Cloud Computing, SOCC'13, New York, NY, USA, 2013. ACM.

[18] G. Rommel, Hadoop 1.x vs Hadoop 2.

[19] Intel Distribution for Apache Hadoop* Software: Optimization and Tuning Guide

[20] http://hortonworks.com

[21] http://www.cloudera.com

[22] http://ambari.apache.org

[23] Virtualizing Apache Hadoop, VMware, 2012

[24] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang. Mars: a mapreduce framework on graphics processors. In Proceedings of the 17th international conference on Parallel architectures and compilation techniques, pages 260 to 269. ACM, 2008.

[25] Z. Xiao, H. Chen, and B. Zang. A hierarchical approach to maximizing mapreduce efficiency. In PACT, pages 167–168, 2011.

[26] R. M. Yoo, A. Romano, and C. Kozyrakis. Phoenix rebirth: Scalable mapreduce on a large-scale shared- memory system. In Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on, pages 198 to 207. IEEE, 2009.

[27] Z. Fadika, E. Dede, M. Govindaraju, and L. Ramakrishnan. Mariane: Mapreduce implementation adapted for hpc environments. In Grid Computing (GRID), 2011 12th IEEE/ACM International Con- ference on, pages 82 to 89. IEEE, 2011.

[28] Z. Fadika, E. Dede, J. Hartog, and M. Govindaraju. Marla: Mapreduce for heterogeneous clusters. In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Comput- ing (ccgrid 2012), pages 49 to 56. IEEE Computer Society, 2012.

[29] E. Dede, Z. Fadika, J. Hartog, M. Govindaraju, L. Ramakrishnan, D. Gunter, and R. Canon. Marissa: Mapreduce implementation for streaming science applications. In E-Science (e-Science), 2012 IEEE 8th International Conference on, pages 1 to 8, Oct 2012.

[30] https://github.com/erikfrey/bashreduce

[31] S. Krishnan, M. Tatineni, and C. Baru. myhadoop-hadoop-on-demand on traditional hpc resources. San Diego Supercomputer Center Technical Report TR-2011-2, University of California, San Diego, 2011.

[32] http://azure.microsoft.com/en-us/services/hdinsight/

[33] http://thrift.apache.org

[34] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn. Ceph: A scalable, high-performance distributed file system. In Proceedings of the 7th symposium on Operating systems design and implemen- tation, pages 307 to 320. USENIX Association, 2006.

[35] https://www.virtualbox.org

[36] https://www.docker.com

[37] https://github.com/intel-hadoop/Hibench

[38] https://portal.futuregrid.org/manual/hadoop-blast

[39] W. Felter, A. Ferreira, R. Rajamony, J. Rubio, An Updated Performance Comparison of Virtual Machines and Linux Containers, IBM Research Division, 2014

[40] http://www-03.ibm.com/systems/platformcomputing/products/gpfs/

[41] http://koin.github.io/what-is-docker/