A Hadoop use case for engineering data

Benoit Lange, Toan Nguyen

A Hadoop use case for engineering data

Benoit Lange[1], Toan Nguyen[1]

[1] INRIA Rhône-Alpes,
38334, Saint-Ismier, France
{benoit.lange, toan.nguyen}@inria.fr

Abstract. This paper presents the VELaSSCo project (Visualization for Extremely LArge-Scale Scientific Computing). It aims to develop a platform to manipulate scientific data used by FEM (Finite Element Method) and DEM (Discrete Element Method) simulations. The project focuses on the development of a distributed, heterogeneous and high-performance platform, enabling the scientific communities to store, process and visualize huge amounts of data. The platform is compatible with current hardware capabilities, as well as future hardware.

Keywords: Hadoop; HPC; commodity nodes; visualization; Hive; HBase.

1 Introduction

For a long time, scientists have tried to understand natural phenomena. In the earlier days of science, researchers described natural phenomena with experimentations. Later, theories have been proposed by scientists to describe phenomena (Newton's laws, Maxwell's equations...). And, at the beginning of the 1980s, computational models have been developed to validate theories. These computational models are used with computer simulations. Computational models and IT hardware have evolved and bring now understanding to a higher level. With modern architectures, a computation takes only a couple of hours: but with efficient computations come huge amounts of data. In most cases, this quantity of information is managed easily, because some parts of the datasets are deleted. This is called filtering, which removes some elements which are none relevant, intermediary time-steps, etc. Unfortunately, this leads to important loses of information, and final analyses are not optimal anymore. To increase the accuracy of the analyses, it is necessary to store all the information produced, instead of deleting parts of it.

This paper is related to the European VELaSSCo project (Visualization for Extremely LArge-Scale Scientific Computing). The goal is to provide a storage platform for large datasets produced by engineering simulations. This paper introduces the VELaSSCo architecture and the components of the final platform.

Section II is an overview of related work on Big Data. Section III presents the VELaSSCo project and its requirements. Section IV details its architecture, layers and components. Section V concludes the paper.

2 Related Work

Big Data solutions have already been developed and widely discussed in many research papers. Nowadays, many application fields have adopted this paradigm, but the data produced by engineering applications has not yet been well evaluated regarding to the Big Data problematic.

Three dimensions are commonly used to represent Big Data (3Vs rule): Volume, Velocity and Variety [1][2]. Volume is related to size of datasets (bytes, number of records, etc.). Velocity concerns the acquisition of information (batch, real-time, etc.). And, Variety is linked to the data format (structured, unstructured, etc.).

Google is one of the major Big Data actors. They have developed a computational model specifically adapted to web search; the tools have been presented in different papers: [4] concerns the MapReduce programming model (MR), [5] deals with BigTable and [6] introduces its own virtual File System (Google FS). Other groups have developed some alternative solutions. One of these tools is Hadoop. It is an open-source and the most used Big Data framework[2], which supports all the requirements describe in the Big Data literature. The earlier implementation of this platform includes a specific file system named HDFS (Hadoop Distributed File System) [7] and the MapReduce [8] computational model. This framework is highly extensible, and many plugins have been developed: one example is HBase, presented in [11]. It is a Hadoop implementation of BigTable.

Alternative software have also been developed, e.g., Dryad[3]. It 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. Microsoft originally developed this software from scratch. But now, to fit with most of existing BigData infrastructures, Microsoft provides also a Hadoop implementation of Dryad. This implementation is based on YARN [10]. YARN is part of the second release of Hadoop. It is the task manager for Hadoop. It splits the *JobTracker* in two separate functions: *RessourceManager* and *ApplicationManager*. Hadoop computations have thus been improved, and MapReduce is not anymore the only computational model for Hadoop. The new Hadoop framework can run on up to 10.000 nodes.

3 The VELaSSCo project and goals

By 2020, complex simulations will produce more data than ever before and IT systems used by the scientific communities need to evolve accordingly. Nowadays, the scientists use mainly HPC facilities to run large simulations. Cloud systems are evolving and starting to offer some HPC cloud infrastructures. But they do not yet offer performance in par with current HPC facilities. However, in the near future, cloud providers will offer new kinds of services based on HPC nodes instead of commodity nodes. These future systems will achieve high performance figures.

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

[3] http://research.microsoft.com/en-us/projects/dryad/

In these new systems, the bottleneck will not be on computations but on I/O operations. With current engineering solvers, to avoid latency in I/O operations, the stored data is explicitly reduced bu the users. The research in this project aims to avoid this filtering by storing all the information produced.

The European Union is funding the VELaSSCo project, in the Seventh Framework Programme (FP7). It groups researchers and engineers from different fields to reach a common goal: provide an innovative and efficient platform to manipulate engineering data produced by simulations. Different industry partners (ATOS and JOTNE) and laboratories/universities (UEDIN, Fraunhofer IGD, SINTEF, CIMNE and INRIA) are involved in the project. The goal is to provide an efficient storage, analytics and visualization platform, specially designed for engineering data.

The platform is supported by Big Data software, in order to handle large datasets produced by simulation engines. The goal is to deal with FEM and DEM simulations. These applications produce huge amounts of data, with complex data structures, e.g., spherical particles, none spherical particles, etc). The second requirement concerns analyses. In order to provide specific information to the users, some analyses need to be executed on the platform, for example extract splines, iso-surface, level of details, etc. These analyses must be supported by our architecture and fit with computational models of HPC, but also cloud infrastructures. Finally, the platform needs to be specifically designed using a user centric paradigm (focused on real-time visualization). When the users perform rendering queries, the response times need to be short. Other requirements and the preliminary design have been described in [11].

From all these specificities and regarding the extensibility of Hadoop, we selected this framework as the foundation of our software stack. In the next section, we describe how we plan to use it. We combine Hadoop with existing plugins and also provide new pieces of software to respond to the needs of the project.

4 The VELaSSCo solution

As stated in the previous section, the foundation of the platform is Hadoop. This software stack can be deployed on any kind of IT architecture with some advantages for commodity nodes. But currently, most scientific communities have their own HPC facilities, based on high-end computers. Further, Hadoop is not well suited for this kind of hardware. The project goal is therefore to provide a flexible Hadoop distribution to support efficiently a large variety of IT infrastructures. In this section, we present a global overview of the VELaSSCo architecture (Figure 1). The architecture is decomposed into layers. Four layers are presented: simulation, storage, engine and client.

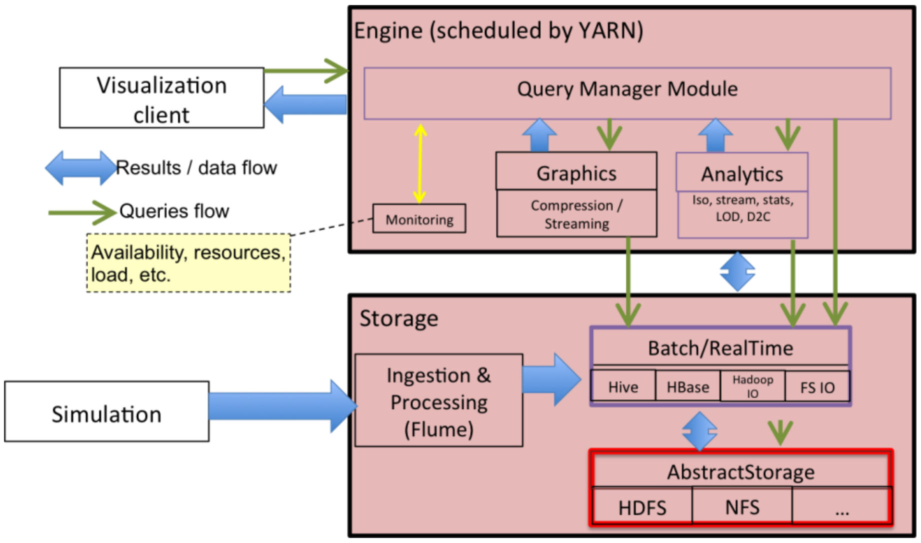

Figure 1.VELaSSCo Architecture

The first layer named *simulation* already exists. This layer is in charge of the production of data. Simulations are executed on specific nodes and the produced data is stored locally (on the HPC FS). These files contain tabular data with different time steps information from the simulations. The simulation software already exist and they produce very large amounts of information. All the time-steps are not stored because this would require too much storage space. Another issue comes from I/O operations, which are slow. With a DEM simulation, a scientist can produce 1 Petabyte of data with 10 millions particles and 1 Billion of time-steps. After the production of data, it is necessary to store the results in the storage layer. A Flume agent is in charge of this task.

Three modules compose the *storage* layer: the ingestion (extract information), query (communication part) and storage modules. The ingestion module is in charge of gathering data from the simulation layer. This module is also used to format data and store these new elements into the correct storage repositories. This strategy is based on a Flume agent, specially designed for our datasets (where simulations are run). Then, the agent sends to the Batch/real-time module (purple box), which is composed by different "I/O query software". This module can be decomposed into different Hadoop extension: Hive, Hbase or the traditional HDFS I/O (Hadoop Distributed File System) query engines. We also have included the standard I/O operations from an operating system, to ensure read and write operations on most existing file systems. These different accesses enable to support automatic benchmarking. This part will be discussed at the end of paper. To select the desired file system, it is necessary to specify in the Hadoop configuration file this parameter. This configuration impacts directly the *AbstractStorage* module.

The third layer is related to the *client* part. In this project we target two visualization tools: IFX[4] (developed at IGD Fraunhofer) and GID[5] (from CIMNE). We

[4] http://www.i-fx.net

extend these tools to enable connections and gathering of information with the future versions of the VELaSSCo platform. This part will be managed using Thrift.

The last layer is named *engine*. Four modules compose this layer: a query manager, a monitoring module, a graphical module and an analytic module. This layer is in charge of communications with the user, and all these communications go through the query manager. The monitoring module, is in charge of checking the health of the platform, and sends this information to the user. The second (graphic) module is in charge of translating RAW data extracted from the platform to a suitable GPU friendly format. This module can communicate with the client with two different strategies: one in batch (chunks of data are send to the client), one in real time (data is streamed to the client). The third module concerns the analysis part of the platform. Some queries performed by the users imply new computations on the datasets. These analyses are performed by the VELaSSCo platform (by the analytics module). The user does not only execute these analyses, and the platform can also trigger some of them to increase the access on specific datasets. An example of such a kind of computations is: extract different resolutions of a model in order to enable streaming visualization. Another example is: extract an iso-surface of a selected part of a model. All these new produced data are directly stored into the platform. The final module concerns the query manager (QM). This module is under development, and the main goal of this module is to receive a query from a user, decompose it into sub-queries, which are able to interact with all available queries (from the storage layer), analytics or graphics modules. This module is also equipped with a set of evaluation tools, which are executed automatically (when no computation is running on the platform) to evaluate the best storage and access (using Hive, HBase or HDFS) strategy for a data set. This evaluation is configured by previous analyses of users. This QM is extendable to support more query engines or analytics operations.

5 Preliminary results

In order to validate the VELaSSCo approach, we performed some preliminary tests [11]. Two scenarii have been used for this evaluation. The visualization tool asks for all data from a data set, and second, 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 the particle id, location, acceleration, etc. For the complete extraction, the 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 the evaluation, we use a single node with an Intel® Core™ i7-2620M Processor, (with 2 physical cores) and 8 GB of memory, running Fedora Core 20. For our experiments, we compare three alternatives: 1) a myHadoop installation (which supports Hadoop 2.x) on a bare metal node, 2) our simplified architecture deployed on three VirtualBox (version 4.3.14) machines, and 3) a simplified distribution with three Dockers (version 1.1.2) containers.

[5] http://www.gidhome.com

Here, we only deal with simple queries (extract data directly from the storage module), and we have made a comparison on existing extensions of Hadoop. The main idea is to find the most effective way to extract content from one of the three data access systems: NFS Gateway (linked to HDFS), Hbase and Hive.

Using the NFS Gateway to read all data and read a specific part of a dataset, we show that NFS gateways have the same performance on both solutions. For the second benchmark, we compare Hbase read with the same hardware parameters. For read operations (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 adequate and the containers is 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. These evaluations are presented in Figure 2 and 3.

For the third benchmark, our experiments are performed with the Hive plugin. Hive is a data warehouse solution built on top of the Hadoop platform. It facilitates queries by providing a high level language called HiveQL. For read operations, the best solution is the architecture with containers, tied to the bare to the metal solution (Figure 4). Usage of pure virtual machines is not efficient. For the read-all operations, 3 containers provide the best architecture (Figure 5). The bare metal solution is not really efficient in this case, because Hadoop does not use efficiently multi-threading capabilities. Thus, for this purpose, it is more efficient to use a solution based on containers.

Another advantage is that Hive and Hbase can be combined, and Hive can query a Hbase data set easily. This 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 concern is how to increase the computation capabilities: with containers, the solution can reach the highest performance, when compared to bare to the metal or using VirtualBox. The NFS Gateway is not the most efficient solution, but it provides a simple way to deal with Hadoop data. The most useful feature with this gateway is the ease of data accesses: with Hbase and Hive, it is necessary to use a specific protocol based on Thrift. However, the Thrift compiler produces all necessary classes to communicate with these plugins.

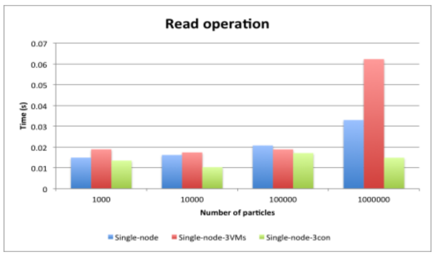

Figure 2. Read operation for Hbase.

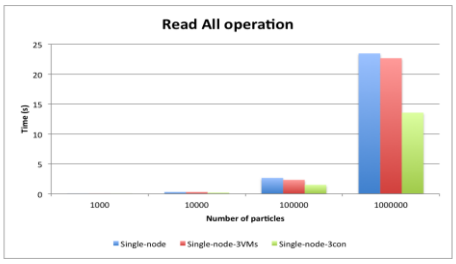

Figure 3. Read all operation for Hbase.

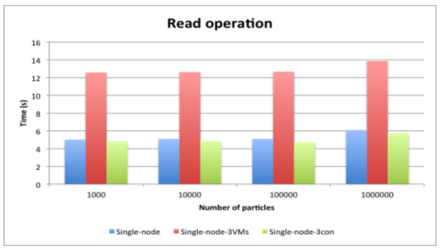

Figure 4. Read operation for Hive.

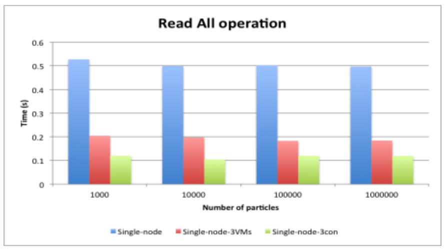

Figure 5. Read all operation for Hive.

From these results, it appears necessary to adopt a container virtualization approach to increase the performance of the Hadoop platform and to avoid important modifications on the whole framework. Containers reduce the overhead for such a kind of virtualization. The system is drastically reduced compared to a virtual machine engine. Moreover, the time needed to deploy our container platform is also less important than the time needed for virtual boxes. For the bare to the metal, the deployment time is 1.7 seconds, while for the container distribution 8.8 seconds are necessary, and for VirtualBox it is a big 188.2 seconds. Unfortunately, all IT systems are not designed for virtualization and even less for containers. Finally, we tried to evaluate the performance using more containers for read and read all operations. In this evaluation we see the overhead of using Hive and Hbase at the same time. And we can see that using three containers is the optimal solution. Results are produced faster than with the other methods. For this test, Hive is used to create the Hbase table.

6 Conclusion and perspectives

This paper provides an 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, manipulate and visualize scientific data. Because modern simulation software used by the scientific communities produce enormous information, it becomes increasingly difficult to deal the corresponding large volumes of data.

A specific architecture is defined which supports the requirements of the project regarding computation, and also the available hardware. The platform is composed by existing and new software. The architecture is organized using a layered approach: simulation, storage, client and engine. The simulation part is related to the production of data. The client layer is the access interface of the platform. The storage layer is in charge of different storage methodologies. It is extensible by new storage and access libraries. The last part concerns the engine part. It is in charge of the communications with the client and data layers. It decomposes complex queries provided by the users into specific parts. These sub-queries gather information from the storage layer and apply computations on the data sets, and finally provide a GPU friendly format of the extracted data sets to support efficient visualization.

The platform is currently under development and different parts need to be evaluated. We are developing an automatic evaluation tool, which measures the best storage and access strategies for a specific query regarding a data set. We also plan to extract information from this platform in real time.

Acknowledgments

This work is supported in part by the EC for the FP7 project VELaSSCo, project number 619439, Call FP7-ICT-2013-11. We thank all the members of the consortium: CIMNE (SP, Coordinator), University of Edinburgh (UK), SINTEF (No.), Fraunhofer IGD (D), JOTNE (No.) and ATOS (SP) for their invaluable contributions and for many fruitful discussions.

References

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

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

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

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

[5] 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)

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

[7] D. Borthakur, The Hadoop Distributed File System: Architecture and design. Hadoop Project Website, 11, 21 (2007).

[8] C. Lam, Hadoop in action. Manning Publications Co. (2010).

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

[10] 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. O Malley, S. Radia, B. Reed, and E. Baldeschwieler. Apache Hadoop Yarn: Yet another resource negotiator. In: 4th Annual Symposium on Cloud Computing (SOCC'13), ACM, New York, USA (2013).

[11] B. Lange and T. Nguyen, Bigdata architecture for large-scale scientific computing, In: 2014 International Conference on Advances in Big Data Analytics (ABDA), pp. 181-184, Las Vegas, USA (2014).