Quber2 COOPERATION

VELaSSCo

Visual Analysis for **E**xtremely **La**rge-**S**cale **S**cientific **Co**mputing

D3.3 – Pre-computed, or on-demand computed, transformations stored in HPC: Engine to create multi-resolution models & co. from simulation data, first version ready for first prototype (A Revision)

Version 1.1

Deliverable Information

Grant Agreement no	619439
Web Site	http://www.velassco.eu/
Related WP & Task:	WP3, T3.2
Due date	February , 2016
Dissemination Level	PU
Nature	P
Author/s	Miguel Pasenau
Contributors	Giuseppe Filippone, Alvaro Janda, Heidi Dahl, Ivan Martinez Rodriguez, Jochen Haenisch, Abel Coll

Approvals

	Name	Institution	Date	OK
Author	Miguel Pasenau	CIMNE	16/2/2016	
Task Leader	Miguel Pasenau	CIMNE	16/2/2016	
WP Leader	Miguel Pasenau	CIMNE	17/2/2016	
Coordinator	Abel Coll	CIMNE	17/2/2016	

Change Log

Version	Description of Change
Version 0.0	Content definition
Version 1.0	Draft
Version 1.1	Reviewed

Table of Contents

Table of figures

1. Introduction

This deliverable was meant to present the revision of the complex VELaSSCo Queries (VQueries) according to the study and evaluation of the first prototype of VELaSSCo Platform by the consortium and on the evaluation of the same prototype by the user panel members on an evaluation event. The very short time slice between the release of the prototype, the evaluation event and this report, makes it impossible to include the revisions and modifications of the VELaSSCo platform that are corollary of the users' evaluation of the first prototype.

Thus, this report will only lean on some of the impressions won by the study and internal evaluation of the platform done by the consortium and some first-handed considerations done by the users in the evaluation event.

Due to some deviations, like the architecture redefinition, some of the work outlined in D3.3 will be included in the final version of the VELaSSCo platform: like the CalculateSimplifiedMesh, and 'pre-computed' path of the VQueries, i.e. storing the results of the analytics operation for later retrieval when the query is issued, thus avoiding repeated calculations.

First the revisions of the in D3.2 presented VQueries are presented after an internal evaluation, and then a preliminary revision after the Evaluation Event are presented.

2. Revision after an internal evaluation

During the implementation process, the architectural design has been changed several times.

Architecture (updated 21.10.2015)

- The final Close Source architecture:

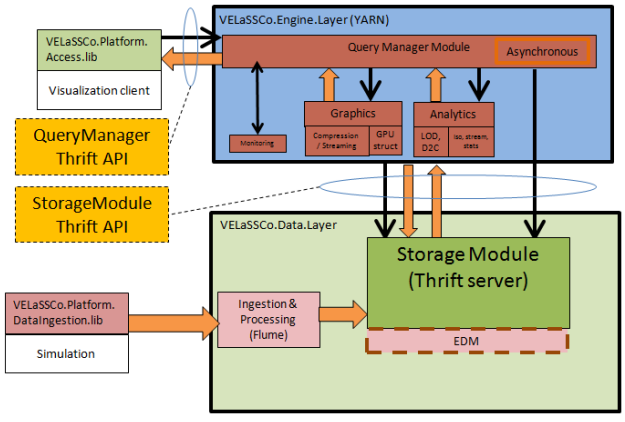
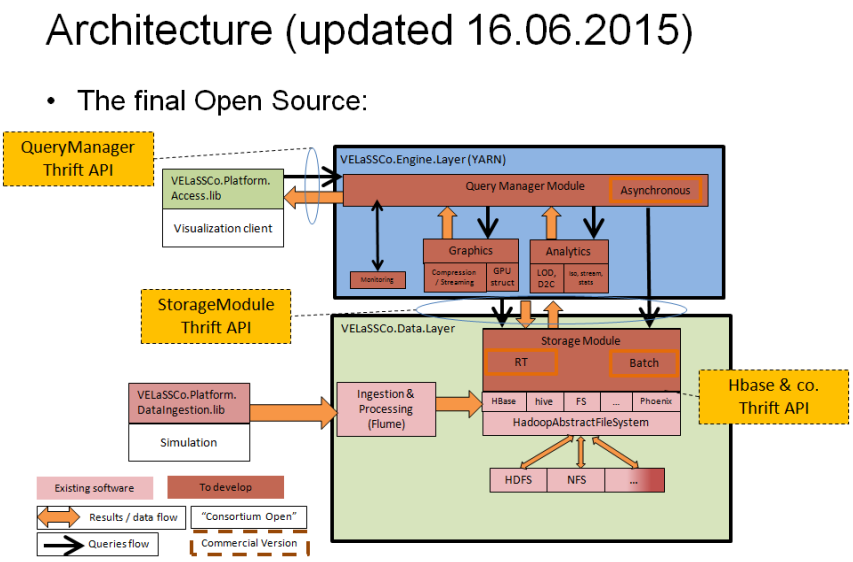


Figure 1 shows the latest versions of both architectures, the open source one and the closed source one. Still, in the closed source architecture, the connection of the analytics queries between the QueryManager, StorageModule and EDM has not been solved as the different proposals were not satisfactory for all partners. Some iterations in the design of this part are still been considered.

Also during the implementation process, in the open source architecture the HBase table structure has been changed several times after the release of D3.1, and different tables were used in addition to the official ones. The latest HBase table structure definition used in the prototype is described in D2.7.



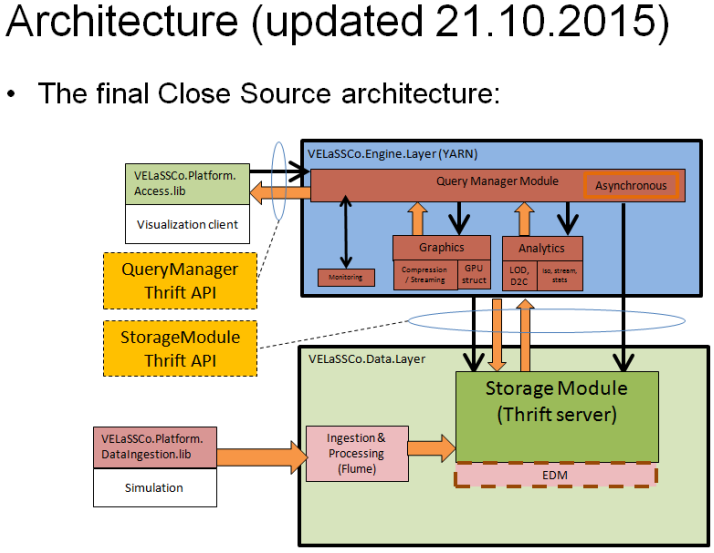


Figure 1: Latest version of the VELaSSCo platform's architecture showing the layers, modules, the tools and interfaces involved in the different components

Several mechanisms have been implemented in the VELaSSCo platform to support the evolution of these architectures like:

- support for several HBase tables and structures (see Figure 2);
- different SessionID and modelID management strategies for the open and closed architectures;
- binary temporary files are written as hexadecimal strings;
- standard java and Hadoop's YARN calling methods have been added;

- several implementations of GetBoundaryOfAMesh have been implemented to overcome some of the yarn configuration settings;
- and support for the deployment on two clusters: CIMNE's Acuario (pez001) and UEDIN's Eddie (velassco-cluster01).

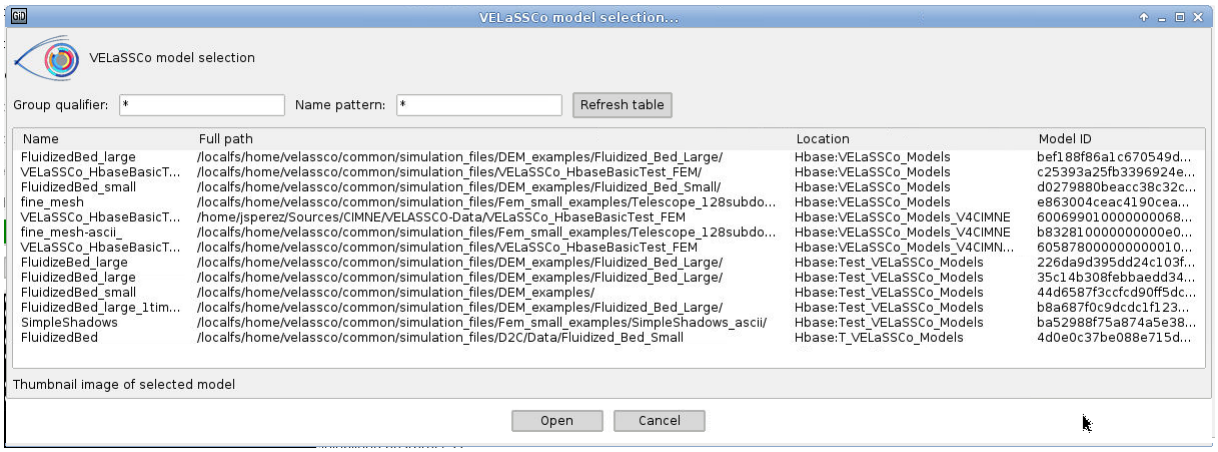


Figure 2: Model selection window showing the different models used to develop the platform and the 3rd column list the different tables used with different structures.

2.1 CalculateBoundaryOfAMesh

Quick overview

This analytics has been already explained in detail in D3.2, but in a nutshell, given a volume mesh, a set of tetrahedrons, this VQuery splits each tetrahedron in four triangles and removes the repeated ones, i.e. the triangles shared between two attached tetrahedrons, returning a list of the unique triangles which form the skin of the boundary mesh.

Following figure shows the usage of the analytics with the telescope example using YARN is shown in the log included in listing 1 in the Annex 1 of this document.

Problem detected: not running in parallel

In the first implementation the analytics operation GetBoundaryOfAMesh run flawlessly on a standard java machine, but crashed using Hadoop's YARN, a tool to launch and manage jobs on a distributed computing and data cluster. After modifying the configuration settings for the job containers, yarn can run the GetBoundaryOfAMesh.

The telescope example used in the use-cases is split in 128 partitions, as shown in *Figure 3*. Each partition was ingested in the HBase tables in separate rows and we've configured HBase to store the rows in separated *region-servers*, i.e. data nodes, as described in previous deliverables, in order to distribute the processing effort of the analytics operations across different data nodes.

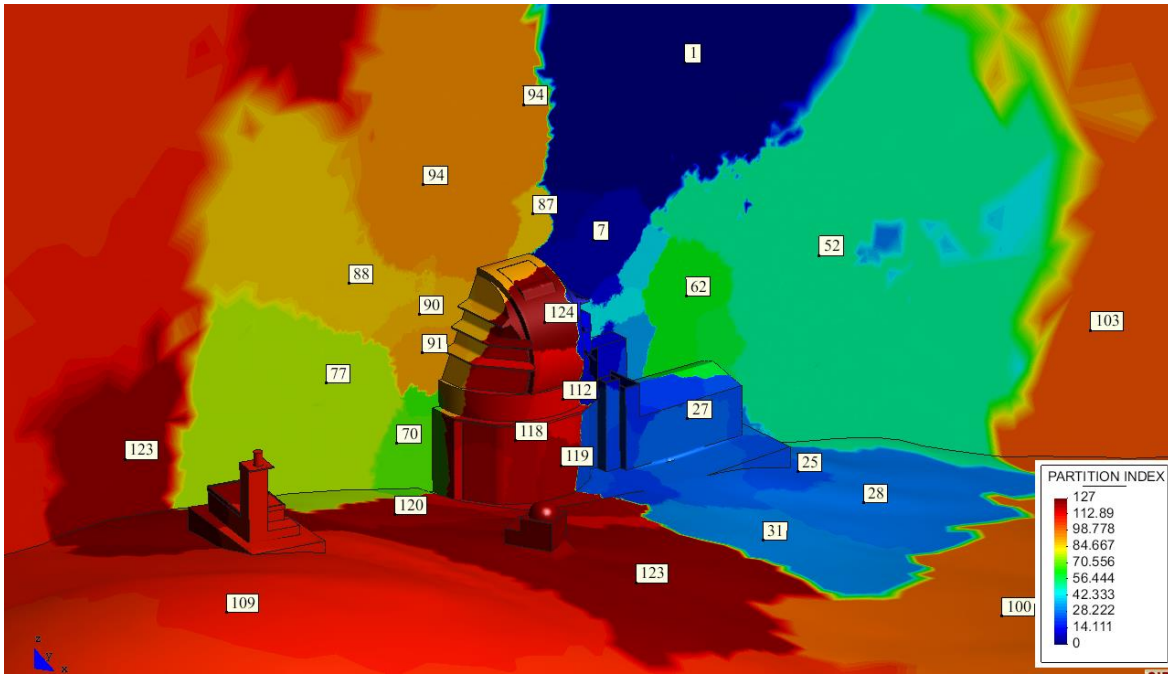


Figure 3: Telescope FEM model used in some use-cases of the evaluation event. The different colours are the sub-domains, partitions, in which the model is subdivided to run the simulations and which are ingested in separated rows in the hbase tables.

But, although the 128 rows were detected by the yarn tool (in listing of Annex 1: `Map input records=128`), the work could not be split (in listing 1: `INFO mapreduce.JobSubmitter: number of splits:1`) and all the work was done in serial, as shown in listing 1:

```
Job Counters
        Launched map tasks=1
        Launched reduce tasks=1
        Rack-local map tasks=1
```

This serialization with YARN results in longer execution times compared to the java version:

Job management	GetBoundaryOfAMesh	GetListOfVertices	Total
YARN	381 s.	163 s.	620 s.
Java	159 s.	131 s.	311 s.

Table 1: execution times for the jobs that compose the VQuery: GetBoundaryOfAMesh and GetListOfVertices, and the total time for the VQuery GetBoundaryOfAMesh. This times were taken on the Acuario cluster (Intel(R) Xeon(R) CPU E5410 @ 2.33GHz).

Table 1 shows the overhead yarn brings with itself, when the jobs are executed in serial. This may reflect some issues with the yarn configuration or the data ingested in the HBase tables.

In order to distribute the data and the computing effort of the analytics, two techniques are available [1, 2, 3]:

1. Pre-splitting: as we known the characteristics of our simulation data we can configure hbase to split the ingested data and send the rows to different *region-servers*, based on the row-key.
2. Salting: add some prefix or suffix to the row-key to break the order and sequence of the row-keys so that they appear more random and each record will be sent to a different *region-server*.

We have decided to use the first technique but it seems that it's not working well.

At the present, we are scrutinizing and analysing the yarn and HBase logs in order to detect and solve this problem.

Problem detected: too long transit times

During the preparation of the evaluation event use-cases, one of the partners (ATOS) reported that using the local visualization client with the AccessLibrary plug-in against the VELaSSCo platform deployed in UEDIN's Eddie cluster, the VQuery GetBoundaryOfAMesh took more than 20 minutes until the telescope skin mesh was shown in the visualization client.

A closer look at the different steps of the VQuery execution showed that despite the fact that the VQuery takes less than 2 minutes to be executed in the Eddie cluster, the transit time of the triangles and vertex coordinates to a visualization client at CIMNE, was 9 minutes. The data size to be sent from Eddie to the visualization client at CIMNE is 33 Mbytes, resulting in a bandwidth of 64 Kbytes in average. It is worth mentioning that the connection to Eddie cluster is done through a VPN connection and a SSH tunnel.

To reduce the size of the returned data to the visualization client, a compression mechanism will be implemented between the QueryManager and the AccessLibrary.

Some preliminary tests using different compression techniques has been performed which reduced the size of the returned result of 1/3 of the original size, and up to 1/10 in some cases, in less than a second.

At the present this mechanism is being implemented in the VELaSSCo platform for all the implemented VQueries, the layered approach of the architecture makes possible to apply this change without modifying any VQuery, only the QueryManager and the AccessLibrary.

Another way to reduce the size of the returned data for this specific VQuery, as the returned results are graphic data (triangles and coordinates), is to use some specialized graphic compression techniques and formats, like the Real Time Format presented in D4.1.

3. Revision after the Evaluation Event

As the writing of this document is done in parallel with the deliverables of WP5 in charge of reporting the user feed-back on the evaluation event, some conclusions can

be drawn from the evaluation event but not actions could be performed nor modifications implemented yet.

As commented in the introduction, as soon as we get the reports including the feed-back from the users, the new improvements have been implemented; a review of this deliverable will be released.

One of the first-hand impressions from the users at the Evaluation Event was the slow performance of some of the VQueries, including the GetBoundaryOfAMesh transmission problem explained in the previous section. A screenshot of the result from this VQuery using GiD as visualization client is depicted in Figure 4.

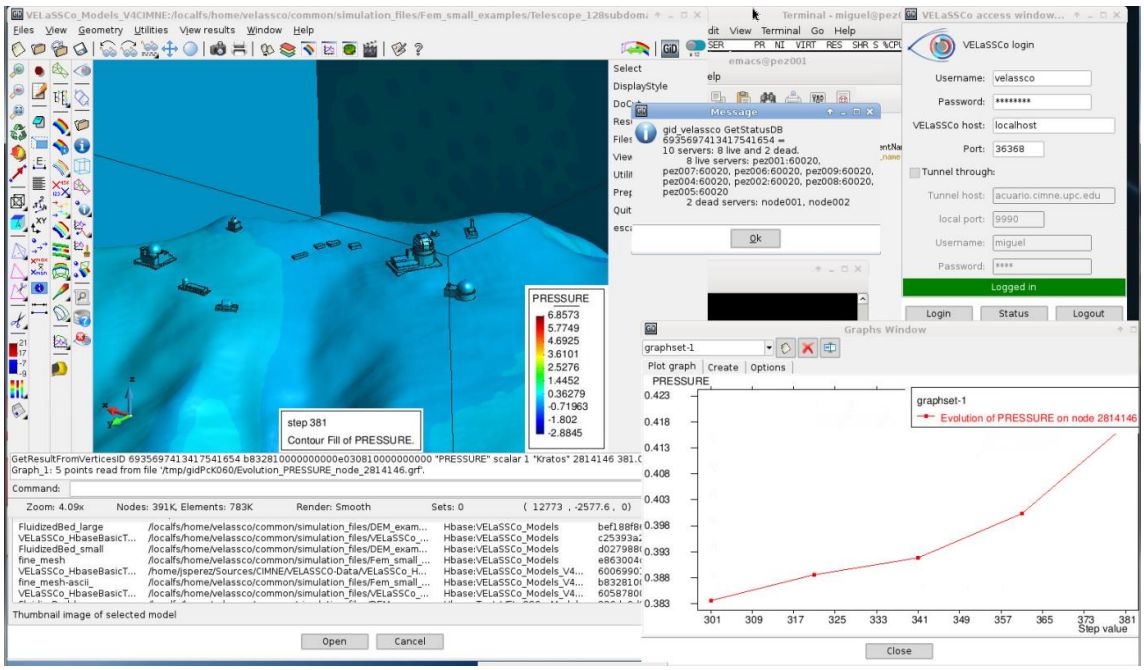


Figure 4: The Telescope FEM model used in some use-cases of the evaluation event. Showing on the left the Pressure colour map drawn over the boundary mesh of the model. On the right the evolution of the pressure on one of the vertices.

About the other complex VQuery included in an use-case of the evaluation event, the Discrete to continuum (D2C) transformation, the users complained about the little feed-back was provided during the D2C execution, as in other VQueries, and that they had to do several steps in order to view the result of this analytics: close the current DEM model, open the D2C_model, calculate the boundary of the new mesh, select a result to visualize. A screenshot of the result from this VQuery using GiD as visualization client is depicted in Figure 5.

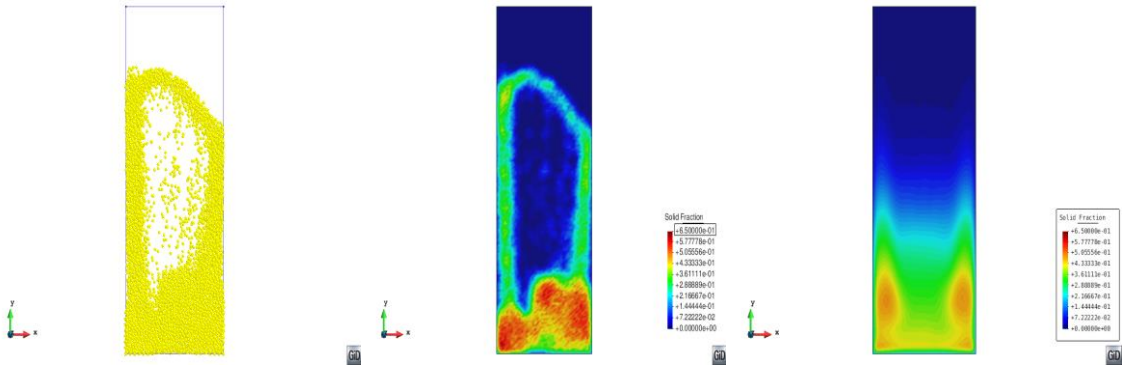


Figure 5. Example of the results obtained by means of the Discrete to Continuum transformation: Left, discrete data provided by DEM simulation solver. Center: D2C results for a single time-step. Right: D2C results including temporal averaging.

4. References

[1] From "Hints for optimizing LOAD" https://www-01.ibm.com/support/knowledgecenter/SSPT3X_4.1.0/com.ibm.swg.im.infosphere.biginsights.analyze.doc/doc/bigsql_loadhints.html , last visited on February 2016.

[2] From "Best Practices for Managing HBase in a High Write Environment" by Woodstock, March 2013, http://www.appfirst.com/blog/best-practices-for-managing-hbase-in-a-high-write-environment/ , last visited on February 2016.

[3] From "HBaseWD: Avoid RegionServer Hotspotting Despite Sequential Keys" by Alex Baranau, April 2012 , http://blog.sematext.com/2012/04/09/hbasewd-avoid-regionserver-hotspotting-despite-writing-records-with-sequential-keys/ , last visited on February 2016.

Annex 1 – Execution log

Execution log of the GetBoundaryOfAMesh operation, including the execution of the GetBoundaryOfAMesh and GetListOfVertices Yarn jobs is shown in Listing 1.

The following convention has been followed:

- In **black**: the launch of the yarn job with the parameters which, in order of appearance corresponds to: HBase_file_configuratino SessionID ModelID Simulation_Data_TableName MeshID ElementType MeshType
- In **orange**: the alternative paths to overcome the different hbase table versions
- In **red**: the current problem: the analytics is not running in parallel

```
[VELaSSCo-EL] 2016.02.16 16:32:43
[VELaSSCo-EL] 2016.02.16 16:32:43 ----- Query() -----
[VELaSSCo-EL] 2016.02.16 16:32:43
[VELaSSCo-EL] 2016.02.16 16:32:43 Input:
[VELaSSCo-EL] 2016.02.16 16:32:43   sessionID : 2616036699551920637
[VELaSSCo-EL] 2016.02.16 16:32:43   query     :
{
  "name"       : "GetBoundaryOfAMesh",
  "modelID"    : "b832810000000000e030810000000000",
  "meshID"     : "Kratos Tetrahedra3D4 Mesh",
  "analysisID" : "",
  "stepValue"  : "0"
}

S   2616036699551920637
dlS 2616036699551920637
M  -b832810000000000e030810000000000-
Msh-Kratos Tetrahedra3D4 Mesh-
An --
Sv -0-
looking for the Mesh Kratos Tetrahedra3D4 Mesh in order to get it's id
Mesh name Kratos Tetrahedra3D4 Mesh has mesh number = 1 and elementType = Tetrahedra
in AnalyticsModule::getInstance
[VELaSSCo-EL] 2016.02.16 16:32:43 yarn jar ../Analytics/GetBoundaryOfAMesh_pez001.jar
/localfs/home/velassco/common/hbase/conf/hbase-site.xml 2616036699551920637
b832810000000000e030810000000000 Simulations_Data_V4CIMNE 1 Tetrahedra static
Heap size: current = 481.5 MB, max = 889.0 MB, free = 461.2366638183594 MB.

>>> Doing GetBoundary of mesh 1 of model b832810000000000e030810000000000 from table
Simulations_Data_V4CIMNE
         it is a static mesh with Tetrahedra
        Using Hasmap in Mapper ? Yes
        Using HBase configuration: /localfs/home/velassco/common/hbase/conf/hbase-site.xml
16/02/16 16:32:45 INFO zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0x7840df80
connecting to ZooKeeper ensemble=pez001:2181
(…)
pez001/10.0.0.1:2181, initiating session
16/02/16 16:32:46 INFO zookeeper.ClientCnxn: Session establishment complete on server
pez001/10.0.0.1:2181, sessionid = 0x152c67bb4660211, negotiated timeout = 90000
rowkey( 56) = b832810000000000e03081000000000000000000000000000000000000 not found!
        ( using the column qualifier prefix = m000001cn_)
16/02/16 16:32:48 INFO client.HConnectionManager$HConnectionImplementation: Closing zookeeper
sessionid=0x152c67bb4660211
16/02/16 16:32:48 INFO zookeeper.ZooKeeper: Session: 0x152c67bb4660211 closed
16/02/16 16:32:48 INFO zookeeper.ClientCnxn: EventThread shut down
Warning: standard qualifier ( "m%06dcn_" ) not found, using alternative ( "m%06d_" ).
16/02/16 16:32:48 INFO zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0x3a1e5f00
connecting to ZooKeeper ensemble=pez001:2181
(…)
```

```
16/02/16 16:32:48 INFO zookeeper.ClientCnxn: EventThread shut down
Accessing HBase: Found the specified mesh.
16/02/16 16:32:49 INFO client.RMProxy: Connecting to ResourceManager at pez001/10.0.0.1:18050
16/02/16 16:34:31 INFO zookeeper.ClientCnxn: Session establishment complete on server
pez001/10.0.0.1:2181, sessionid = 0x152c67bb4660213, negotiated timeout = 90000
16/02/16 16:34:31 INFO util.RegionSizeCalculator: Calculating region sizes for table
"Simulations_Data_V4CIMNE".
16/02/16 16:34:31 INFO mapreduce.JobSubmitter: number of splits:1
16/02/16 16:34:31 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1455028933081_0020
16/02/16 16:34:31 INFO impl.YarnClientImpl: Submitted application application_1455028933081_0020
16/02/16 16:34:32 INFO mapreduce.Job: The url to track the job:
http://pez001:18088/proxy/application_1455028933081_0020/
16/02/16 16:34:32 INFO mapreduce.Job: Running job: job_1455028933081_0020
16/02/16 16:34:54 INFO mapreduce.Job: Job job_1455028933081_0020 running in uber mode : false
16/02/16 16:34:54 INFO mapreduce.Job:  map 0% reduce 0%
16/02/16 16:38:33 INFO mapreduce.Job:  map 67% reduce 0%
16/02/16 16:38:57 INFO mapreduce.Job:  map 78% reduce 0%
16/02/16 16:38:59 INFO mapreduce.Job:  map 100% reduce 0%
16/02/16 16:39:09 INFO mapreduce.Job:  map 100% reduce 100%
16/02/16 16:39:09 INFO mapreduce.Job: Job job_1455028933081_0020 completed successfully
16/02/16 16:39:10 INFO mapreduce.Job: Counters: 59
        File System Counters
                FILE: Number of bytes read=48837498
                FILE: Number of bytes written=73509353
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=199
                HDFS: Number of bytes written=44687088
                HDFS: Number of read operations=5
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Rack-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=730296
                Total time spent by all reduces in occupied slots (ms)=22041
                Total time spent by all map tasks (ms)=243432
                Total time spent by all reduce tasks (ms)=7347
                Total vcore-seconds taken by all map tasks=243432
                Total vcore-seconds taken by all reduce tasks=7347
                Total megabyte-seconds taken by all map tasks=747823104
                Total megabyte-seconds taken by all reduce tasks=22569984
        Map-Reduce Framework
                Map input records=128
                Map output records=1933408
                Map output bytes=54135424
                Map output materialized bytes=24418746
                Input split bytes=199
                Combine input records=1933408
                Combine output records=813958
                Reduce input groups=798971
                Reduce shuffle bytes=24418746
                Reduce input records=813958
                Reduce output records=783984
                Spilled Records=2441874
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=99945
                CPU time spent (ms)=864570
                Physical memory (bytes) snapshot=3402072064
                Virtual memory (bytes) snapshot=9420881920
                Total committed heap usage (bytes)=2953314304
        HBase Counters
                BYTES_IN_REMOTE_RESULTS=3556711056
                BYTES_IN_RESULTS=3556711056
                MILLIS_BETWEEN_NEXTS=212920
                NOT_SERVING_REGION_EXCEPTION=0
```

```
                NUM_SCANNER_RESTARTS=0
                REGIONS_SCANNED=1
                REMOTE_RPC_CALLS=131
                REMOTE_RPC_RETRIES=0
                RPC_CALLS=131
                RPC_RETRIES=0
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=44687088
Job time: 256.766 s.
CPU time: 380.961 s.
Output: Number of triangles = 783984
(…)
16/02/16 16:39:12 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
doing MapReduce::getListOfVerticesFromMesh
[VELaSSCo-EL] 2016.02.16 16:39:17 yarn jar ../Analytics/GetListOfVerticesFromMesh_pez001.jar
/localfs/home/velassco/common/hbase/conf/hbase-site.xml 2616036699551920637
b832810000000000e030810000000000 Simulations_Data_V4CIMNE 1 static

>>> Doing GetListOfVertices of mesh 1 of model b832810000000000e030810000000000 from table
Simulations_Data_V4CIMNE
         it is a static mesh
        Using HBase configuration: /localfs/home/velassco/common/hbase/conf/hbase-site.xml
(…)
Accessing HBase: Found the specified mesh.
(…)
16/02/16 16:39:22 INFO util.RegionSizeCalculator: Calculating region sizes for table
"Simulations_Data_V4CIMNE".
16/02/16 16:39:23 INFO mapreduce.JobSubmitter: number of splits:1
16/02/16 16:39:23 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1455028933081_0021
16/02/16 16:39:23 INFO impl.YarnClientImpl: Submitted application application_1455028933081_0021
16/02/16 16:39:23 INFO mapreduce.Job: The url to track the job:
http://pez001:18088/proxy/application_1455028933081_0021/
16/02/16 16:39:23 INFO mapreduce.Job: Running job: job_1455028933081_0021
16/02/16 16:39:29 INFO mapreduce.Job: Job job_1455028933081_0021 running in uber mode : false
16/02/16 16:39:29 INFO mapreduce.Job:  map 0% reduce 0%
16/02/16 16:40:52 INFO mapreduce.Job:  map 67% reduce 0%
16/02/16 16:41:43 INFO mapreduce.Job:  map 100% reduce 87%
16/02/16 16:41:45 INFO mapreduce.Job:  map 100% reduce 93%
16/02/16 16:41:48 INFO mapreduce.Job:  map 100% reduce 99%
16/02/16 16:41:49 INFO mapreduce.Job:  map 100% reduce 100%
16/02/16 16:41:49 INFO mapreduce.Job: Job job_1455028933081_0021 completed successfully
16/02/16 16:41:49 INFO mapreduce.Job: Counters: 59
        File System Counters
                FILE: Number of bytes read=341071304
                FILE: Number of bytes written=513192109
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=199
                HDFS: Number of bytes written=272176840
                HDFS: Number of read operations=5
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Rack-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=223672
                Total time spent by all reduces in occupied slots (ms)=45482
                Total time spent by all map tasks (ms)=111836
```

```
                Total time spent by all reduce tasks (ms)=22741
                Total vcore-seconds taken by all map tasks=111836
                Total vcore-seconds taken by all reduce tasks=22741
                Total megabyte-seconds taken by all map tasks=229040128
                Total megabyte-seconds taken by all reduce tasks=46573568
        Map-Reduce Framework
                Map input records=128
                Map output records=4487780
                Map output bytes=161560080
(…)
     status: Ok
boundary mesh has 783984 triangles and 391984 vertices.
in GraphicsModule::getInstance
[VELaSSCo-EL] 2016.02.16 16:43:03
[VELaSSCo-EL] 2016.02.16 16:43:03 Output:
[VELaSSCo-EL] 2016.02.16 16:43:03   result : 0
[VELaSSCo-EL] 2016.02.16 16:43:03   boundary_mesh = ( 34495087 bytes)
[VELaSSCo-EL] 2016.02.16 16:43:03   data   :
0000000000000000: 4e 75 6d 62 65 72 4f 66 56 65 72 74 69 63 65 73  NumberOfVertices
0000000000000010: 3a 20 33 39 31 39 38 34 0a 4e 75 6d 62 65 72 4f  : 391984.NumberO
0000000000000020: 66 46 61 63 65 73 3a 20 37 38 33 39 38 34 0a 01  fFaces: 783984..
0000000000000030: 00 00 00 00 00 00 00 4e 25 03 40 11 7d c2 40 3a  .......N%.@.}.@:
0000000000000040: 95 0c 00 65 78 9c 40 d9 ee 1e a0 3b ad 97 40 02  ...ex.@....;..@.
0000000000000050: 00 00 00 00 00 00 00 59 6d fe 5f a7 7d c2 40 5b  .......Ym._.}.@[
0000000000000060: 7c 0a 80 c1 8d 9c 40 10 75 1f 80 d4 c4 97 40 03  |.....@.u.....@.
0000000000000070: 00 00 00 00 00 00 00 17 9f 02 60 a6 80 c2 40 b0  ..........`...@.

[VELaSSCo-EL] 2016.02.16 16:43:03   --> result size: 34495091 Bytes.

[VELaSSCo-EL] 2016.02.16 16:43:03   --> scaled size: 32.8971 MBytes to send back.
```

Listing 1: Log of the GetBoundaryOfAMesh analytics which corresponds to two yarn jobs: one to calculate the skin triangles, and a second to retrieve the coordinates of this mesh so that the visualization client can draw it.