VELaSSCo

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

# D3.4 – Engine able to perform first-time visualizations and simple queries of the last results and over the unmodified domain or the transformations performed in D3.2 (EDM) & report

**Version 1.2**

Deliverable Information

| | |
|---|---|
| Grant Agreement no | 619439 |
| Web Site | http://www.velassco.eu/ |
| Related WP & Task: | WP3, T3.4 |
| Due date | September 30th, 2015 |
| Dissemination Level | PU |
| Nature | P |
| Author/s | Jochen Haenisch |
| Contributors | Ivan Martinez, Miguel Angel Tinte, Toan Nguyen, Alvaro Janda, Giuseppe Filippone, Andreas Dietrich, Miguel Pasenau, Abel Coll, Olav Liestøl |

Approvals

| | Name | Institution | Date | |
|---|---|---|---|---|
| Author | Jochen Haenisch | Jotne | 2015-09-30 | |
| Task Leader | Miguel Pasenau | CIMNE | 2015-09-30 | |
| WP Leader | Miguel Pasenau | CIMNE | 2015-09-30 | |
| Coordinator | Abel Coll | CIMNE | 2015-09-30 | |
| | | | | |

Change Log

| Version | Description of Change |
|---|---|
| Version 1.0 | Completed final draft for delivery to PO |
| Version 1.2 | Review by all the partners |

# Table of Contents

1 Introduction ____ 4
2 The VELaSSCo system architecture ____ 5
3 Simple queries implementation ____ 6
3.1 VQuery process in the architecture ____ 6
3.2 Engine layer implementation ____ 14
3.3 Query Manager Module Implementation ____ 15
3.4 Analytics Implementation ____ 16
3.5 Storage Implementations ____ 16
4 Tasks for the second release of this prototype ____ 24
5 Abbreviations and Definitions ____ 25
6 References ____ 30

# Table of figures

Figure 1: The VELaSSCo architecture used for the open source version of this prototype ..... 5
Figure 2: The VELaSSCo architecture used for the closed source version of this prototype ..... 5
Figure 3: Processes diagram of the VELaSSCo platform. ..... 7
Figure 4: Output messages of the TestClient as it connects to the VELaSSCo prototype 9
Figure 5: Output messages of the Engine Layer as it connects to the Storage module of the Data Layer, receives the VQueries UserLogin, GetResultsForVerticesID and UserLogout, executes them (for GetStatus and GetResultsForVerticesID the Query Manager access the Data Layer) and returns the results ..... 12
Figure 6: Output messages of the Data Layer as it connects to the stored data (in this case HBase tables), receives the status queries and the GetResultsForVerticesID from the Engine Layer, and issues a data scan of the HBase table 'Simulations_Data' ..... 14
Figure 7: I/O performance for the write operation on the Acuario cluster ..... 19
Figure 8: I/O performance for the read operation on the Acuario cluster ..... 20
Figure 9: Output from the EDM plug-in test program ..... 23

# Table of tables

Table 1: Table of acronyms ..... 25

# 1 Introduction

This is deliverable D3.4 with the title "Engine able to perform first-time visualizations and simple queries of the last results and over the unmodified domain or the transformations performed in D3.2 (EDM) & report".

It is the first of two prototypes and associated documents; this first one is delivered at month 21 of the project, the other one, D3.5, at month 24. The prototypes show how pre-computed results of simple queries can be requested, retrieved and presented by the VELaSSCo visualization tool. The second prototype is a refinement of the first one and includes lessons learnt.

This deliverable is the result of Task 3.3, which the Description of Work (DoW) describes as follows:

*"Development of distributed database system that efficiently executes simple users' queries (DEM &FEM)*

*— Subtask 3.3.1: design and implement the opening case: visualization tool connects to the system, provides information about the capabilities and the system, depending on the capabilities, returns a first view (geometric mesh) of the simulated model so that the user can move it, and zoom.*

*— Subtasks 3.3.2: provide first view of the results of the last time-step for doing a colour representation, vector visualization over the views (mesh) provided in subtasks 3.3.1 ."*

D3.4 relies on D3.2, "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". While D3.4 provides a global view of the Simple queries engine, D3.2 provides an insight view of some analytics functionalities and transformations like GetBoundary Mesh, GetBoundingBox, Discrete2Continuum and Calculate Multi-resolutions. These are integrated into an initial version of the entire VELaSSCo platform.

Based on this guidance by the DoW this document describes the implementation of the first prototype of simple queries, that is, the architecture and execution process aspects of the platform with its modules. The relevant issues for this purpose are discussed, and remaining work is identified.

The document covers the following topics in this sequence:

1) The VELaSSCo system architecture applied to this prototype;
2) Implementation aspects of the modules of the architecture;
3) Development items for the second release of this prototype.

This prototype and document and the results of Task 3.3 in general will serve Tasks 3.4 and 3.5 as input to implement DEM specific queries and to address not only simple, but also complex queries in prototypes based on the same VELaSSCo architecture.

## 2 The VELaSSCo system architecture

The current architectures for the two scenarios used to produce this prototype are depicted in the following figures (Figure 1 and Figure 2), which may have evolved slightly compared to earlier deliverables. Figure 1 depicts the architecture based on open source software, Figure 2 the one with Jotne's DBMS EDM.

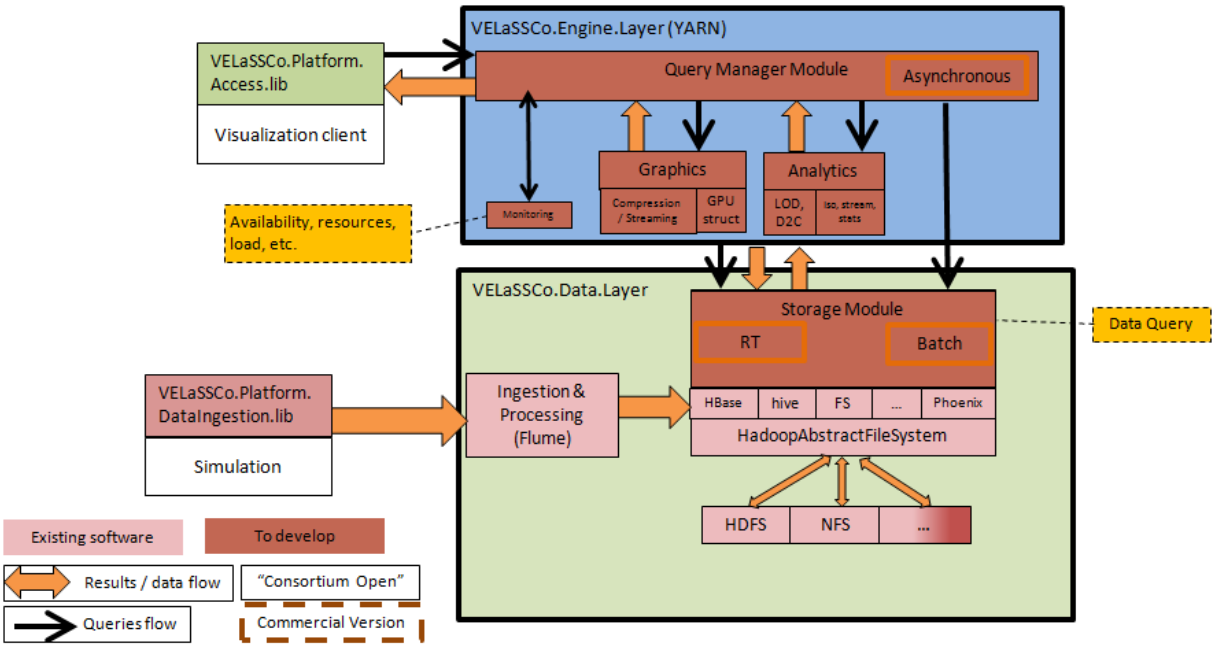


Figure 1: The VELaSSCo architecture used for the open source version of this prototype

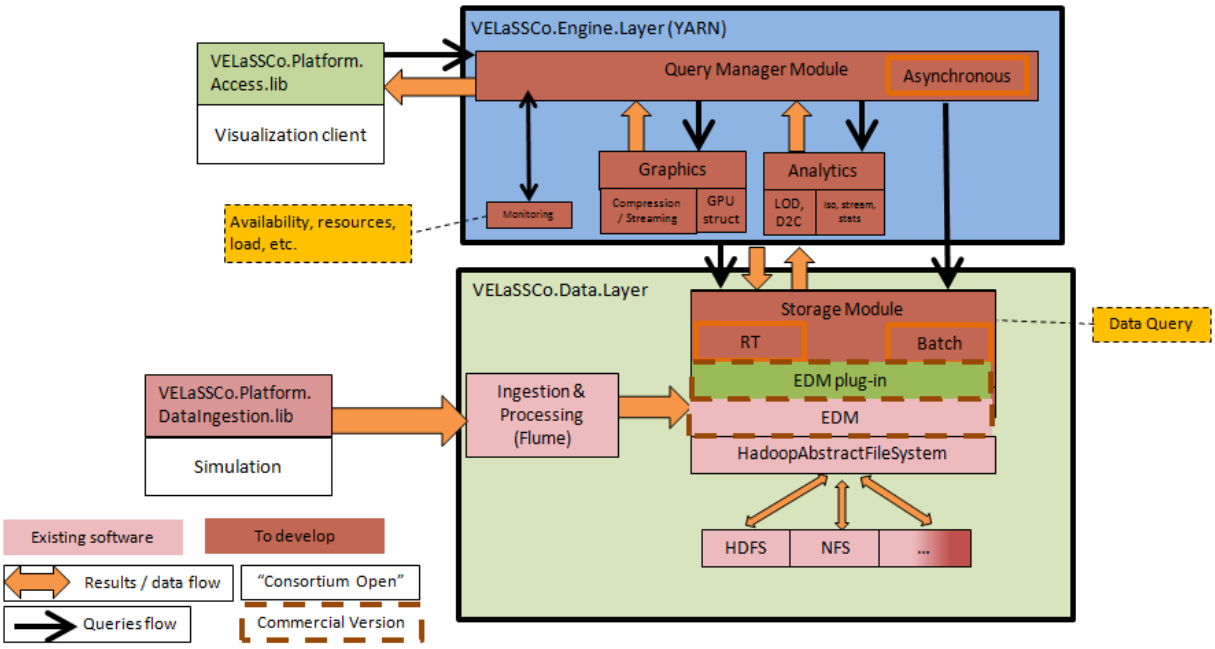


Figure 2: The VELaSSCo architecture used for the closed source version of this prototype

Deliverable D4.2 [53] relates in detail the modules of this architecture to the workflow involved in the simple VQueries

# 3 Simple queries implementation

The intention of this prototype is to provide a proof of feasibility for the architecture in general, rather than delivering a wide range of end-user functionality. Thus, focus has been to develop and run all modules in order to validate the interaction among the various modules as shown in Figure 1 and Figure 2; that is, query manager, analytics, graphics and real time storage retrieval. Tests included both DEM and FEM data. To ease debugging the data sets are still small; the extension to bigger data will be done in the second prototype.

VELaSSCo Queries (VQueries) are the means of traversing the architecture, providing end user functionality and, thus, connect visualization with data storage. The scope of the simple queries of the first VELaSSCo prototype is simulation data access and initial analysis queries, as listed below.

1. Session connection:
    a. User Authentication
    b. List of models for model selection:
        - Get Model Information Summary: name, number of nodes, elements, steps and results. Eventually the Bounding Box, if there is a thumbnail of the model, validation status, which are data not mandatory for the model, but may present some cases.
2. Model view:
    a. Get Surface / Line meshes / DEM particles
    b. Extract skin of Volume meshes, eventually store this mesh
    c. Eventually store a thumbnail of the model
    d. Eventually change the validation status of the model
        - c. and d. can be considered as metadata (d. needed in EDM-approach)
3. Results view:
    a. Get list of analyses, time steps, and results properties (name, type, …)
    b. Given a list of nodes or elements, get the results values
    c. Given a point in space, get the result interpolated considering the nodes of the element that contains the given point

## 3.1 VQuery process in the architecture

The architecture described in the previous section is illustrated in the process diagram in Figure 3.

As shown in Figure 3 the VQueries triggered by user interactions with the visualization client will be sent by the Access Library to the Query Manager Module (QMM) of the Engine Layer. The Query Manager Module is part of a Thrift server, which acts as the Engine Layer. The QMM passes, in this first prototype, most of the VQueries to the Storage module of the Data Layer, as most of the VQueries are data access queries.

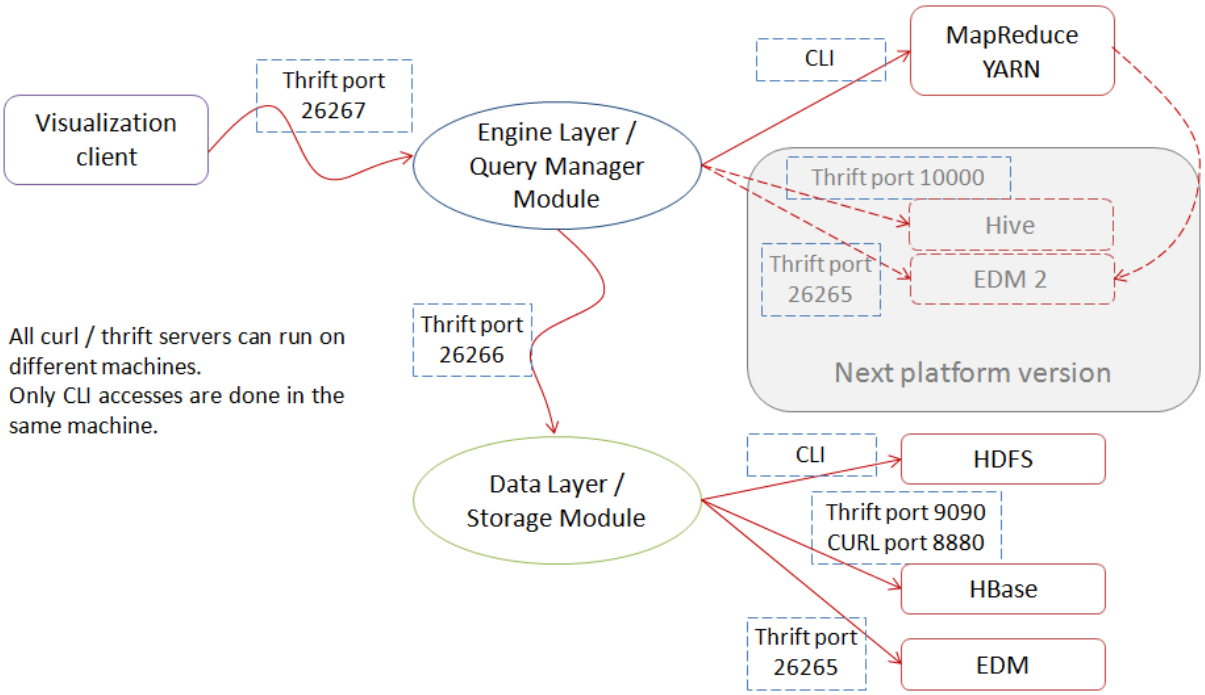


Figure 3: Processes diagram of the VELaSSCo platform.

A few complex VQueries, i.e., analytics transformations, are evaluated as Yarn CLI jobs by the Analytics module in the Engine Layer Thrift server. These jobs perform their significant work outside the engine layer; they are distributed over the Hadoop or EDM nodes.

The Storage module will be part of another Thrift server, the Data Layer. This will contact either the HBase Thrift server of the EDM Thrift server, depending on the scenario. In the first prototype, however, the Storage Module contacts the HBase REST server using the CURL library, for testing purposes.

Having separated Thrift server applications for the engine layer, data layer and HBase/EDM storage allows encapsulating the different functionalities and the different scenarios, making the core VELaSSCo functionalities independent of the choice of data storage solution.

As mentioned in D2.3 [54], the computing nodes from an HPC cluster are not usually accessible by the outside world. Access needs to be provided through a login node, which in turn may launch the computer simulations against the thousands of computing nodes of the cluster. With the chosen modular approach, the Engine Layer can be installed on the login node. It will provide external users running the visualization client on, e.g., an MS Windows desktop computer, access to the simulations stored in the internal nodes of the cluster without compromising security.

After downloading the source files of the platform and compiling them, the following five separate executables and one library become available:

- modules/DataLayer/**FlumeIngestion** → Data injector using Flume agents (described in D2.5 and D2.6)

- modules/DataLayer/Storage/bin/**Storage** → the storage module of the Data Layer responsible for the queries that access the data in HBase or EDM.
- modules/EngineLayer/QueryManager/bin/**QueryManager** → the query manager module of the Engine Layer, which will receive the VQueries from the Access Library of the Visualization Client and which will either forward them to the Storage module, execute them using Yarn CLI or handle them internally.
- modules/AccessLib/build/**libAccessLib.a** → library to be used by the Visualization Client to connect to the platform.
- modules/AccessLib/build/Test/**Test_Client** → a demo client, which uses the Access Library to connect to the platform.
- modules/AccessLib/build/Test/**Test_Server** → a demo server used to verify the connection of the Access Library.

An example of the communications among the three processes (Test_Client, QueryManager and Storage) is shown in the following output listings.

Further details of the implementations of the involved modules are given in the subsequent sections.

```
# launch Test_Client to connect to velassco platform at host 'pez001' with port '26267'
[miguel@pez001 modules]$ AccessLib/build/Test/Test_Client pez001 26267
Connecting to 'pez001:26267' ...
[VELaSSCo]
[VELaSSCo]
[VELaSSCo] UserLogin_Result:
[VELaSSCo]     result    : 0
[VELaSSCo]     sessionID : -3870262791008661227
[VELaSSCo]
[VELaSSCo] Query_Result:                    # GetResultForVerticesID for id = 0 … 25
[VELaSSCo]     result : 0
[VELaSSCo]     data   :
# Raw data:
0000000000000000: 32 36 20 33 0a 00 00 00 00 00 00 00 01 00 00  26 3............
0000000000000010: 00 00 00 00 00 02 00 00 00 00 00 00 00 03 00 00  ................
0000000000000020: 00 00 00 00 00 04 00 00 00 00 00 00 00 05 00 00  ................
0000000000000030: 00 00 00 00 00 06 00 00 00 00 00 00 00 07 00 00  ................
0000000000000040: 00 00 00 00 00 08 00 00 00 00 00 00 00 09 00 00  ................
0000000000000050: 00 00 00 00 00 0a 00 00 00 00 00 00 00 0b 00 00  ................
0000000000000060: 00 00 00 00 00 0c 00 00 00 00 00 00 00 0d 00 00  ................
0000000000000070: 00 00 00 00 00 0e 00 00 00 00 00 00 00 0f 00 00  ................
0000000000000080: 00 00 00 00 00 10 00 00 00 00 00 00 00 11 00 00  ................
0000000000000090: 00 00 00 00 00 12 00 00 00 00 00 00 00 13 00 00  ................
00000000000000a0: 00 00 00 00 00 14 00 00 00 00 00 00 00 15 00 00  ................
00000000000000b0: 00 00 00 00 00 16 00 00 00 00 00 00 00 17 00 00  ................
00000000000000c0: 00 00 00 00 00 18 00 00 00 00 00 00 00 18 00 00  ................
00000000000000d0: 00 00 00 00 00 af 0b 3f 38 9f 3a ba 3f 8a da 47  .......?8.:.?..G
00000000000000e0: b8 da d4 a2 3f 7b bf d1 8e 1b 7e d3 bf 2c 7e 53  ....?{....~..,~S
00000000000000f0: 58 a9 a0 ba 3f af 5c 6f 9b a9 10 d0 3f 89 b2 b7  X...?.\o....?...
0000000000000100: 94 f3 c5 d3 bf 67 b9 6c 74 ce 4f bd 3f 3f 42 83  .....g.lt.O.??B.
0000000000000110: 03 10 2d a0 3f db c1 88 7d 02 28 d4 bf 80 ee cb  ..-.?...}.(.....
0000000000000120: 99 ed 0a bd 3f 73 f2 22 13 f0 6b d0 3f ea b3 03  ....?s."..k.?...
0000000000000130: ae 2b 66 d4 bf d5 3e 1d 8f 19 a8 c0 3f b9 11 bb  .+f...>.....?...
0000000000000140: 5b fe e8 98 3f 5c c7 b8 e2 e2 a8 d5 bf ad da 35  [...?\.........5
0000000000000150: 21 ad 31 c0 3f 0c ad 4e ce 50 dc d0 3f 96 ea 02  !.1.?..N.P..?...
0000000000000160: 5e 66 d8 d5 bf 4f c9 39 b1 87 f6 d3 3f 64 6c 11  ^f...O.9....?dl.
0000000000000170: e6 b0 da 70 bf 18 93 fe 5e 0a 0f d8 bf 39 6f ad  ...p....^....9o.
0000000000000180: 40 3e 32 b3 bf 57 41 0c 74 ed 0b ca bf c3 d4 96  @>2..WA.t.......
0000000000000190: 3a c8 eb d1 bf 2e 00 8d d2 a5 7f c3 bf c8 ed 97  :...............
00000000000001a0: 4f 56 0c bb bf ec f7 c4 3a 55 be d2 bf de 04 df  OV......:U......
00000000000001b0: 34 7d 76 c4 3f 45 17 2f 71 3f 3b a7 bf 3f 19 e3  4}v.?E./q?;..?..
00000000000001c0: c3 ec 65 d1 bf bd c7 99 26 6c 3f c7 bf 57 5d 87  ..e.....&l?..W].
```

```
00000000000001d0: 6a 4a b2 c8 3f c3 f3 52 b1 31 af d1 bf 1b a0 34  jJ..?..R.1.....4
00000000000001e0: d4 28 24 cb bf 0e 2f 88 48 4d bb bc 3f d5 5c 6e  .($.../.HM..?.\n
00000000000001f0: 30 d4 61 d2 bf b3 2e c9 5c 74 0d b6 3f 32 f4 3e  0.a.....\t..?2.>
0000000000000200: 44 59 ae a9 3f fd f8 4b 8b fa 24 d1 bf dc bd dc  DY..?..K..$.....
0000000000000210: 27 47 01 d2 3f f9 f5 43 6c b0 70 d1 bf b3 0a 9b  'G..?..Cl.p.....
0000000000000220: 01 2e c8 d7 bf f8 8c 44 68 04 1b d8 bf 22 c6 6b  .......Dh....".k
0000000000000230: 5e d5 59 cf 3f 48 6a a1 64 72 6a d5 bf c0 52 a7  ^.Y.?Hj.drj...R.
0000000000000240: 4d e6 69 b6 3f 11 3b ae a1 2f 18 ab 3f 01 87 50  M.i.?.;../..?..P
0000000000000250: a5 66 0f d2 bf 48 98 58 cf ff 61 b9 3f 73 d9 e8  .f...H.X..a.?s..
0000000000000260: 9c 9f e2 cc 3f 9d f5 29 c7 64 71 d2 bf e1 d1 c6  ....?..).dq.....
0000000000000270: 11 6b f1 d2 3f ff 41 24 43 8e ad d1 bf 36 3d 28  .k..?.A$C....6=(
0000000000000280: 28 45 2b d8 bf f6 ec b9 4c 4d 82 cf bf 08 af 5d  (E+.....LM.....]
0000000000000290: da 70 58 c0 3f 21 92 21 c7 d6 33 d7 bf de c4 35  .pX.?!.!..3....5
00000000000002a0: e3 d9 8a b6 3f 47 1c b2 81 74 b1 a5 3f 7d 94 11  ....?G...t..?}..
00000000000002b0: 17 80 46 d2 bf 25 de 4b 2b cf cd b6 3f 8a 48 03  ..F..%.K+...?.H.
00000000000002c0: 2e 7e 09 a7 3f 63 b8 3a 00 e2 ae d2 bf 29 e0 e8  .~..?c.:.....)..
00000000000002d0: 74 6a 78 b8 3f f8 a9 2a 34 10 cb ce 3f 77 9d 0d  tjx.?..*4...?w..
00000000000002e0: f9 67 06 d3 bf 68 b1 14 c9 57 02 d4 3f 5f d1 ad  .g...h...W..?_..
00000000000002f0: d7 f4 a0 d0 bf db 6c ac c4 3c 2b d8 bf 1f a1 66  ......l..<+....f
0000000000000300: 48 15 c5 b7 3f 32 d1 7b be c1 72 a3 3f a4 a9 9e  H...?2.{..r.?...
0000000000000310: cc 3f fa d2 bf 54 3b 1e 8e 09 8c 80 bf e4 a3 c5  .?...T;.........
0000000000000320: 19 c3 9c c2 3f 39 42 06 f2 ec f2 d2 bf 54 3b 1e  ....?9B......T;.
0000000000000330: 8e 09 8c 80 bf e4 a3 c5 19 c3 9c c2 3f 39 42 06  ............?9B.
0000000000000340: f2 ec f2 d2 bf                                   .....

# after decoding the data:
Vertex: 0  ID: 0  Values: [ 0.102457 0.0367802 -0.304572 ]
Vertex: 1  ID: 1  Values: [ 0.104014 0.251017 -0.308957 ]
Vertex: 2  ID: 2  Values: [ 0.114499 0.0315938 -0.314942 ]
Vertex: 3  ID: 3  Values: [ 0.113448 0.256588 -0.318736 ]
Vertex: 4  ID: 4  Values: [ 0.13013 0.0243263 -0.338433 ]
Vertex: 5  ID: 5  Values: [ 0.126516 0.263447 -0.341333 ]
Vertex: 6  ID: 6  Values: [ 0.311922 -0.00411481 -0.375918 ]
Vertex: 7  ID: 7  Values: [ -0.0749854 -0.203489 -0.280016 ]
Vertex: 8  ID: 8  Values: [ -0.152333 -0.105657 -0.292867 ]
Vertex: 9  ID: 9  Values: [ 0.159866 -0.0453739 -0.271846 ]
Vertex: 10  ID: 10  Values: [ -0.181623 0.192941 -0.276318 ]
Vertex: 11  ID: 11  Values: [ -0.212041 0.112233 -0.287221 ]
Vertex: 12  ID: 12  Values: [ 0.0861428 0.0501583 -0.267882 ]
Vertex: 13  ID: 13  Values: [ 0.281328 -0.272503 -0.371593 ]
Vertex: 14  ID: 14  Values: [ -0.376649 0.244929 -0.334622 ]
Vertex: 15  ID: 15  Values: [ 0.0875534 0.0529189 -0.28219 ]
Vertex: 16  ID: 16  Values: [ 0.0991516 0.225666 -0.288171 ]
Vertex: 17  ID: 17  Values: [ 0.295985 -0.276218 -0.377641 ]
Vertex: 18  ID: 18  Values: [ -0.246164 0.127699 -0.362539 ]
Vertex: 19  ID: 19  Values: [ 0.0880562 0.0423695 -0.285553 ]
Vertex: 20  ID: 20  Values: [ 0.0890779 0.0449943 -0.291924 ]
Vertex: 21  ID: 21  Values: [ 0.0955874 0.240572 -0.297266 ]
Vertex: 22  ID: 22  Values: [ 0.312643 -0.259824 -0.377639 ]
Vertex: 23  ID: 23  Values: [ 0.092851 0.0379849 -0.296524 ]
Vertex: 24  ID: 24  Values: [ -0.0080796 0.145409 -0.296077 ]
Vertex: 25  ID: 24  Values: [ -0.0080796 0.145409 -0.296077 ]
[VELaSSCo]
[VELaSSCo] UserLogout_Result:
[VELaSSCo]     result : 0
[VELaSSCo]
[miguel@pez001 modules]$
```

Figure 4: Output messages of the TestClient as it connects to the VELaSSCo prototype

```
[miguel@pez001 modules]$ EngineLayer/QueryManager/bin/QueryManager
# connection to Data Layer and getting status
[EngineLayer] Connecting to Data Layer at pez001:26266
        ########## getStatus - 10 servers: 8 live and 2 dead.
        8 live servers: pez004:60020, pez009:60020, pez008:60020, pez001:60020, pez002:60020,
pez006:60020, pez005:60020, pez007:60020
        2 dead servers: node002, node001

# start listening, it also has a command line for some options:
```

```
[EngineLayer] listening on port26267
List of avaiblable cmd :
exit (or quit): stop the current application (enginelayer)
dl_exit (or dl_quit or dlq or dle): stop the dataLayer application
 ################################################

[EngineLayer] Starting VELaSSCo Server...
[EngineLayer]   using port: 26267
[EngineLayer]   before serving ...
[EngineLayer]
# received Query:
[EngineLayer] ----- UserLogin() -----
[EngineLayer]
[EngineLayer] Input:
[EngineLayer]   url      : pez001:26267
[EngineLayer]   name     : andreas
[EngineLayer]   password : 1234
[EngineLayer]
[EngineLayer] Output:
[EngineLayer]   result    : 0
[EngineLayer]   sessionID : -3870262791008661227
[EngineLayer]
# received Query:
[EngineLayer] ----- Query() -----
[EngineLayer]
[EngineLayer] Input:
[EngineLayer]   sessionID : -3870262791008661227
[EngineLayer]   query     :
{
  "name"       : "GetResultFromVerticesID",
  "modelID"    : "d94ca29be534ca1ed578e90123b7",
  "resultID"   : "MASS",
  "analysisID" : "DEM",
  "vertexIDs"  : [1,2,3,4,5,6,7],
  "timeStep"   : 10000
}

S -3870262791008661227
M d94ca29be534ca1ed578e90123b7
R MASS
A DEM
V [1,2,3,4,5,6,7]
T 10000
# get data from Data Layer / Storage Module
########## getStatus - 0 0.102457 0.0367802 -0.304572
1 0.104014 0.251017 -0.308957
2 0.114499 0.0315938 -0.314942
3 0.113448 0.256588 -0.318736
4 0.13013 0.0243263 -0.338433
5 0.126516 0.263447 -0.341333
6 0.311922 -0.00411481 -0.375918
7 -0.0749854 -0.203489 -0.280016
8 -0.152333 -0.105657 -0.292867
9 0.159866 -0.0453739 -0.271846
10 -0.181623 0.192941 -0.276318
11 -0.212041 0.112233 -0.287221
12 0.0861428 0.0501583 -0.267882
13 0.281328 -0.272503 -0.371593
14 -0.376649 0.244929 -0.334622
15 0.0875534 0.0529189 -0.28219
16 0.0991516 0.225666 -0.288171
17 0.295985 -0.276218 -0.377641
18 -0.246164 0.127699 -0.362539
19 0.0880562 0.0423695 -0.285553
20 0.0890779 0.0449943 -0.291924
21 0.0955874 0.240572 -0.297266
22 0.312643 -0.259824 -0.377639
23 0.092851 0.0379849 -0.296524
24 -0.0080796 0.145409 -0.296077
```

```
0 0.102457 0.0367802 -0.304572
1 0.104014 0.251017 -0.308957
2 0.114499 0.0315938 -0.314942
3 0.113448 0.256588 -0.318736
4 0.13013 0.0243263 -0.338433
5 0.126516 0.263447 -0.341333
6 0.311922 -0.00411481 -0.375918
7 -0.0749854 -0.203489 -0.280016
8 -0.152333 -0.105657 -0.292867
9 0.159866 -0.0453739 -0.271846
10 -0.181623 0.192941 -0.276318
11 -0.212041 0.112233 -0.287221
12 0.0861428 0.0501583 -0.267882
13 0.281328 -0.272503 -0.371593
14 -0.376649 0.244929 -0.334622
15 0.0875534 0.0529189 -0.28219
16 0.0991516 0.225666 -0.288171
17 0.295985 -0.276218 -0.377641
18 -0.246164 0.127699 -0.362539
19 0.0880562 0.0423695 -0.285553
20 0.0890779 0.0449943 -0.291924
21 0.0955874 0.240572 -0.297266
22 0.312643 -0.259824 -0.377639
23 0.092851 0.0379849 -0.296524
24 -0.0080796 0.145409 -0.296077

# encode data for the visualization client/Access library
[EngineLayer]
[EngineLayer] Output:
[EngineLayer]   result : 0
[EngineLayer]   data   :
0000000000000000: 32 36 20 33 0a 00 00 00 00 00 00 00 01 00 00  26 3...........
0000000000000010: 00 00 00 00 00 02 00 00 00 00 00 00 00 03 00 00  ................
0000000000000020: 00 00 00 00 00 04 00 00 00 00 00 00 00 05 00 00  ................
0000000000000030: 00 00 00 00 00 06 00 00 00 00 00 00 00 07 00 00  ................
0000000000000040: 00 00 00 00 00 08 00 00 00 00 00 00 00 09 00 00  ................
0000000000000050: 00 00 00 00 00 0a 00 00 00 00 00 00 00 0b 00 00  ................
0000000000000060: 00 00 00 00 00 0c 00 00 00 00 00 00 00 0d 00 00  ................
0000000000000070: 00 00 00 00 00 0e 00 00 00 00 00 00 00 0f 00 00  ................
0000000000000080: 00 00 00 00 00 10 00 00 00 00 00 00 00 11 00 00  ................
0000000000000090: 00 00 00 00 00 12 00 00 00 00 00 00 00 13 00 00  ................
00000000000000a0: 00 00 00 00 00 14 00 00 00 00 00 00 00 15 00 00  ................
00000000000000b0: 00 00 00 00 00 16 00 00 00 00 00 00 00 17 00 00  ................
00000000000000c0: 00 00 00 00 00 18 00 00 00 00 00 00 00 18 00 00  ................
00000000000000d0: 00 00 00 00 00 af 0b 3f 38 9f 3a ba 3f 8a da 47  .......?8.:.?..G
00000000000000e0: b8 da d4 a2 3f 7b bf d1 8e 1b 7e d3 bf 2c 7e 53  ....?{....~..,~S
00000000000000f0: 58 a9 a0 ba 3f af 5c 6f 9b a9 10 d0 3f 89 b2 b7  X...?.\o....?...
0000000000000100: 94 f3 c5 d3 bf 67 b9 6c 74 ce 4f bd 3f 3f 42 83  .....g.lt.O.??B.
0000000000000110: 03 10 2d a0 3f db c1 88 7d 02 28 d4 bf 80 ee cb  ..-.?...}.(.....
0000000000000120: 99 ed 0a bd 3f 73 f2 22 13 f0 6b d0 3f ea b3 03  ....?s."..k.?...
0000000000000130: ae 2b 66 d4 bf d5 3e 1d 8f 19 a8 c0 3f b9 11 bb  .+f...>.....?...
0000000000000140: 5b fe e8 98 3f 5c c7 b8 e2 e2 a8 d5 bf ad da 35  [...?\.........5
0000000000000150: 21 ad 31 c0 3f 0c ad 4e ce 50 dc d0 3f 96 ea 02  !.1.?..N.P..?...
0000000000000160: 5e 66 d8 d5 bf 4f c9 39 b1 87 f6 d3 3f 64 6c 11  ^f...O.9....?dl.
0000000000000170: e6 b0 da 70 bf 18 93 fe 5e 0a 0f d8 bf 39 6f ad  ...p....^....9o.
0000000000000180: 40 3e 32 b3 bf 57 41 0c 74 ed 0b ca bf c3 d4 96  @>2..WA.t.......
0000000000000190: 3a c8 eb d1 bf 2e 00 8d d2 a5 7f c3 bf c8 ed 97  :...............
00000000000001a0: 4f 56 0c bb bf ec f7 c4 3a 55 be d2 bf de 04 df  OV......:U......
00000000000001b0: 34 7d 76 c4 3f 45 17 2f 71 3f 3b a7 bf 3f 19 e3  4}v.?E./q?;..?..
00000000000001c0: c3 ec 65 d1 bf bd c7 99 26 6c 3f c7 bf 57 5d 87  ..e.....&l?..W].
00000000000001d0: 6a 4a b2 c8 3f c3 f3 52 b1 31 af d1 bf 1b a0 34  jJ..?..R.1.....4
00000000000001e0: d4 28 24 cb bf 0e 2f 88 48 4d bb bc 3f d5 5c 6e  .($.../.HM..?.\n
00000000000001f0: 30 d4 61 d2 bf b3 2e c9 5c 74 0d b6 3f 32 f4 3e  0.a.....\t..?2.>
0000000000000200: 44 59 ae a9 3f fd f8 4b 8b fa 24 d1 bf dc bd dc  DY..?..K..$.....
0000000000000210: 27 47 01 d2 3f f9 f5 43 6c b0 70 d1 bf b3 0a 9b  'G..?..Cl.p.....
0000000000000220: 01 2e c8 d7 bf f8 8c 44 68 04 1b d8 bf 22 c6 6b  .......Dh....".k
0000000000000230: 5e d5 59 cf 3f 48 6a a1 64 72 6a d5 bf c0 52 a7  ^.Y.?Hj.drj...R.
0000000000000240: 4d e6 69 b6 3f 11 3b ae a1 2f 18 ab 3f 01 87 50  M.i.?.;../..?..P
0000000000000250: a5 66 0f d2 bf 48 98 58 cf ff 61 b9 3f 73 d9 e8  .f...H.X..a.?s..
0000000000000260: 9c 9f e2 cc 3f 9d f5 29 c7 64 71 d2 bf e1 d1 c6  ....?..).dq.....
```

```
0000000000000270: 11 6b f1 d2 3f ff 41 24 43 8e ad d1 bf 36 3d 28  .k..?.A$C....6=(
0000000000000280: 28 45 2b d8 bf f6 ec b9 4c 4d 82 cf bf 08 af 5d  (E+.....LM.....]
0000000000000290: da 70 58 c0 3f 21 92 21 c7 d6 33 d7 bf de c4 35  .pX.?!.!..3....5
00000000000002a0: e3 d9 8a b6 3f 47 1c b2 81 74 b1 a5 3f 7d 94 11  ....?G...t..?}..
00000000000002b0: 17 80 46 d2 bf 25 de 4b 2b cf cd b6 3f 8a 48 03  ..F..%.K+...?.H.
00000000000002c0: 2e 7e 09 a7 3f 63 b8 3a 00 e2 ae d2 bf 29 e0 e8  .~..?c.:.....)..
00000000000002d0: 74 6a 78 b8 3f f8 a9 2a 34 10 cb ce 3f 77 9d 0d  tjx.?..*4...?w..
00000000000002e0: f9 67 06 d3 bf 68 b1 14 c9 57 02 d4 3f 5f d1 ad  .g...h...W..?_..
00000000000002f0: d7 f4 a0 d0 bf db 6c ac c4 3c 2b d8 bf 1f a1 66  ......l..<+....f
0000000000000300: 48 15 c5 b7 3f 32 d1 7b be c1 72 a3 3f a4 a9 9e  H...?2.{..r.?...
0000000000000310: cc 3f fa d2 bf 54 3b 1e 8e 09 8c 80 bf e4 a3 c5  .?...T;.........
0000000000000320: 19 c3 9c c2 3f 39 42 06 f2 ec f2 d2 bf 54 3b 1e  ....?9B......T;.
0000000000000330: 8e 09 8c 80 bf e4 a3 c5 19 c3 9c c2 3f 39 42 06  ............?9B.
0000000000000340: f2 ec f2 d2 bf                                   .....

[EngineLayer]
# received query:
[EngineLayer] ----- UserLogout() -----
[EngineLayer]
[EngineLayer] Input:
[EngineLayer]   sessionID : -3870262791008661227
[EngineLayer]
[EngineLayer] Output:
[EngineLayer]   result : 0
# ending query manager
quit
[miguel@pez001 modules]$
```

Figure 5: Output messages of the Engine Layer as it connects to the Storage module of the Data Layer, receives the VQueries UserLogin, GetResultsForVerticesID and UserLogout, executes them (for GetStatus and GetResultsForVerticesID the Query Manager access the Data Layer) and returns the results

```
[miguel@pez001 modules]$ DataLayer/Storage/bin/Storage
# start listening:
[DataLayer] Storage Module: listening on port 26266
# received query ( GetStatus):
http://pez001:8880/status/cluster
**********
{"LiveNodes": …
**********
[VELaSSCo] 10 servers: 8 live and 2 dead.
        8 live servers: pez004:60020, pez009:60020, pez008:60020, pez001:60020, pez002:60020,
pez006:60020, pez005:60020, pez007:60020
        2 dead servers: node002, node001
# received query ( GetResultsForVrrticesID):
S -3870262791008661227
M d94ca29be534ca1ed578e90123b7
R MASS
A DEM
V {"id":[]}
T 10000
# Access HBase table Simulations_Data:
http://pez001:8880/Simulations_Data/*
**********
{"Row":[{"key":"NjQzOTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxNjU2NDM1MzczODY1MzkzMDMxMzIzMzYyMzc2MzM4NjM2NTQ
0NDU0ZDMxMzAzMDMwMzAzMDMx","Cell":[{"column":"TTpjMl8w","timestamp":1441900812512,"$":"MC4xMDI0NTcgMC4w
MzY3ODAyIC0wLjMwNDU3Mg=="},{"column":"TTpjMl8x","timestamp":1441900812512,"$":"MC4xMDQwMTQgMC4yNTEwMTcg
LTAuMzA4OTU3"},{"column":"TTptMmNuXzA=","timestamp":1441900812512,"$":"MSAyIDA="},{"column":"TTptMmNuXz
E=","timestamp":1441900812512,"$":"MiA5IDE="},{"column":"TTptMmdwXzA=","timestamp":1441900812512,"$":"Z
W1wdHk="},{"column":"TTptMmdwXzE=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"UjpyNG5y","time
stamp":1441900812372,"$":"Mg=="},{"column":"UjpyNHZsXzA=","timestamp":1441900812372,"$":"Ny44OTAxNyA3NC
40MDcyIC0yOTUuMDY1"},{"column":"UjpyNHZsXzE=","timestamp":1441900812372,"$":"NS41NDUyNCAtODMuMzMyNCAxNT
IuNDQx"}]},{"key":"NjQzOTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxNjU2NDM1MzczODY1MzkzMDMxMzIzMzYyMzc2MzM4NjM2
NTQ0NDU0ZDMxMzEzMDMwMzAzMDMx","Cell":[{"column":"TTpjMl8w","timestamp":1441900812512,"$":"MC4xMTQ0OTkgM
C4wMzE1OTM4IC0wLjMxNDk0Mg=="},{"column":"TTpjMl8x","timestamp":1441900812512,"$":"MC4xMTM0NDggMC4yNTY1O
DggLTAuMzE4NzM2"},{"column":"TTptMmNuXzA=","timestamp":1441900812512,"$":"MSAyIDA="},{"column":"TTptMmN
```

uXzE=","timestamp":1441900812512,"$":"MiA5IDE="},{"column":"TTptMmdwXzA=","timestamp":1441900812512,"$"
:"ZW1wdHk="},{"column":"TTptMmdwXzE=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"UjpyNG5y","t
imestamp":1441900812372,"$":"Mg=="},{"column":"UjpyNHZsXzA=","timestamp":1441900812372,"$":"MTkuNDc3MiA
4MS43MTc1IC0zMzYuMjY2"},{"column":"UjpyNHZsXzE=","timestamp":1441900812372,"$":"MTEuNjkyMSAtNzUuNjc5MSA
xMTEuMTA2"}]},{"key":"NjQzOTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxNjU2NDM1MzczODY1MzkzMDMxMzIzMzYyMzc2MzM4N
jM2NTQ0NDU0ZDMxMzIzMDMwMzAzMDMx","Cell":[{"column":"TTpjMl8w","timestamp":1441900812512,"$":"MC4xMzAxMy
AwLjAyNDMyNjMgLTAuMzM4NDMz"},{"column":"TTpjMl8x","timestamp":1441900812512,"$":"MC4xMjY1MTYgMC4yNjM0ND
cgLTAuMzQxMzMz"},{"column":"TTptMmNuXzA=","timestamp":1441900812512,"$":"MSAyIDA="},{"column":"TTptMmNu
XzE=","timestamp":1441900812512,"$":"MiA5IDE="},{"column":"TTptMmdwXzA=","timestamp":1441900812512,"$":
"ZW1wdHk="},{"column":"TTptMmdwXzE=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"UjpyNG5y","ti
mestamp":1441900812372,"$":"Mg=="},{"column":"UjpyNHZsXzA=","timestamp":1441900812372,"$":"NzMuMjg4IDcw
LjgwMDggLTQ3NS4yNDE="},{"column":"UjpyNHZsXzE=","timestamp":1441900812372,"$":"LTM4LjgyNDMgLTEyMy42MzEg
MjMwLjQ4NQ=="}]},{"key":"NjQzOTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxNjU2NDM1MzczODY1MzkzMDMxMzIzMzYyMzc2Mz
M4NjM2NTQ0NDU0ZDMxMzQzMDMwMzAzMDMx","Cell":[{"column":"TTpjMl8w","timestamp":1441900812512,"$":"MC4zMTE
5MjIgLTAuMDA0MTE0ODEgLTAuMzc1OTE4"},{"column":"TTptMmNuXzA=","timestamp":1441900812512,"$":"NSAyIDA="},
{"column":"TTptMmdwXzA=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"UjpyNG5y","timestamp":144
1900812372,"$":"MQ=="},{"column":"UjpyNHZsXzA=","timestamp":1441900812372,"$":"MjU1LjAwMyAtMzcyLjUwNiAt
MTYxLjMyOA=="}]},{"key":"NjQzOTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxNjU2NDM1MzczODY1MzkzMDMxMzIzMzYyMzc2Mz
M4NjM2NTQ0NDU0ZDMyMzAzMDMwMzAzMQ==","Cell":[{"column":"TTpjMl8w","timestamp":1441900812512,"$":"LTAuMDc
0OTg1NCAtMC4yMDM0ODkgLTAuMjgwMDE2"},{"column":"TTptMmNuXzA=","timestamp":1441900812512,"$":"NiA3IDA="},
{"column":"TTptMmdwXzA=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"UjpyNG5y","timestamp":144
1900812372,"$":"MQ=="},{"column":"UjpyNHZsXzA=","timestamp":1441900812372,"$":"LTUwMS40NDMgMzc4LjUzMyAt
MjUxMy44MQ=="}]},{"key":"NjQzOTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxNjU2NDM1MzczODY1MzkzMDMxMzIzMzYyMzc2Mz
M4NjM2NTQ0NDU0ZDMzMzAzMDMwMzAzMQ==","Cell":[{"column":"TTpjMl8w","timestamp":1441900812512,"$":"LTAuMTU
yMzMzIC0wLjEwNTY1NyAtMC4yOTI4Njc="},{"column":"TTpjMl8x","timestamp":1441900812512,"$":"MC4xNTk4NjYgLTA
uMDQ1MzczOSAtMC4yNzE4NDY="},{"column":"TTpjMl8y","timestamp":1441900812512,"$":"LTAuMTgxNjIzIDAuMTkyOTQ
xIC0wLjI3NjMxOA=="},{"column":"TTptMmNuXzA=","timestamp":1441900812512,"$":"NiA0IDA="},{"column":"TTptM
mNuXzE=","timestamp":1441900812512,"$":"MSA1IDE="},{"column":"TTptMmNuXzI=","timestamp":1441900812512,"
$":"MyA4IDI="},{"column":"TTptMmdwXzA=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"TTptMmdwXz
E=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"TTptMmdwXzI=","timestamp":1441900812512,"$":"Z
W1wdHk="},{"column":"UjpyNG5y","timestamp":1441900812372,"$":"Mw=="},{"column":"UjpyNHZsXzA=","timestam
p":1441900812372,"$":"MTQxLjI2NyAtMzM0LjkyOCAtMTExMi41"},{"column":"UjpyNHZsXzE=","timestamp":144190081
2372,"$":"LTg2LjM5MDggMjguNTk4MiAtNTAyLjUzOA=="},{"column":"UjpyNHZsXzI=","timestamp":1441900812372,"$"
:"MTc4LjA0MiAtMjA3LjE0NyAtODExLjQzOA=="}]},{"key":"NjQzOTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxNjU2NDM1Mzcz
ODY1MzkzMDMxMzIzMzYyMzc2MzM4NjM2NTQ0NDU0ZDM1MzAzMDMwMzAzMQ==","Cell":[{"column":"TTpjMl8w","timestamp":
1441900812512,"$":"LTAuMjEyMDQxIDAuMTEyMjMzIC0wLjI4NzIyMQ=="},{"column":"TTpjMl8x","timestamp":14419008
12512,"$":"MC4wODYxNDI4IDAuMDUwMTU4MyAtMC4yNjc4ODI="},{"column":"TTptMmNuXzA=","timestamp":144190081251
2,"$":"NCAzIDA="},{"column":"TTptMmNuXzE=","timestamp":1441900812512,"$":"MSAyIDE="},{"column":"TTptMmd
wXzA=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"TTptMmdwXzE=","timestamp":1441900812512,"$"
:"ZW1wdHk="},{"column":"UjpyNG5y","timestamp":1441900812372,"$":"Mg=="},{"column":"UjpyNHZsXzA=","times
tamp":1441900812372,"$":"LTY4LjQ2MDYgLTM0LjEyNSAxMzguNTIx"},{"column":"UjpyNHZsXzE=","timestamp":144190
0812372,"$":"LTEuOTM1NTkgLTEwLjMzMTggMjMuMDQ3OA=="}]},{"key":"NjQzOTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxN
jU2NDM1MzczODY1MzkzMDMxMzIzMzYyMzc2MzM4NjM2NTQ0NDU0ZDM2MzAzMDMwMzAzMQ==","Cell":[{"column":"TTpjMl8w","
timestamp":1441900812512,"$":"MC4yODEzMjggLTAuMjcyNTAzIC0wLjM3MTU5Mw=="},{"column":"TTpjMl8x","timestam
p":1441900812512,"$":"LTAuMzc2NjQ5IDAuMjQ0OTI5IC0wLjMzNDYyMg=="},{"column":"TTpjMl8y","timestamp":14419
00812512,"$":"MC4wODc1NTM0IDAuMDUyOTE4OSAtMC4yODIxOQ=="},{"column":"TTpjMl8z","timestamp":1441900812512
,"$":"MC4wOTkxNTE2IDAuMjI1NjY2IC0wLjI4ODE3MQ=="},{"column":"TTptMmNuXzA=","timestamp":1441900812512,"$"
:"NyA1IDA="},{"column":"TTptMmNuXzE=","timestamp":1441900812512,"$":"NCA4IDE="},{"column":"TTptMmNuXzI=
","timestamp":1441900812512,"$":"MSAyIDI="},{"column":"TTptMmNuXzM=","timestamp":1441900812512,"$":"OSA
yIDM="},{"column":"TTptMmdwXzA=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"TTptMmdwXzE=","ti
mestamp":1441900812512,"$":"ZW1wdHk="},{"column":"TTptMmdwXzI=","timestamp":1441900812512,"$":"ZW1wdHk=
"},{"column":"TTptMmdwXzM=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"UjpyNG5y","timestamp":
1441900812372,"$":"NA=="},{"column":"UjpyNHZsXzA=","timestamp":1441900812372,"$":"LTY5MC40ODMgLTU5My42M
DkgNDMuNTg3Ng=="},{"column":"UjpyNHZsXzE=","timestamp":1441900812372,"$":"MzMuMzU3NSAtNjUwLjEyNCAyMjQuM
zU0"},{"column":"UjpyNHZsXzI=","timestamp":1441900812372,"$":"LTAuMjk5OTQ2IC0xNC42NTEyIC00My42MzY2"},{"
column":"UjpyNHZsXzM=","timestamp":1441900812372,"$":"MTMuNjk2NCAxODAuODUyIC01NjQuMDE4"}]},{"key":"NjQz
OTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxNjU2NDM1MzczODY1MzkzMDMxMzIzMzYyMzc2MzM4NjM2NTQ0NDU0ZDM3MzAzMDMwMzA
zMQ==","Cell":[{"column":"TTpjMl8w","timestamp":1441900812512,"$":"MC4yOTU5ODUgLTAuMjc2MjE4IC0wLjM3NzY0
MQ=="},{"column":"TTpjMl8x","timestamp":1441900812512,"$":"LTAuMjQ2MTY0IDAuMTI3Njk5IC0wLjM2MjUzOQ=="},{
"column":"TTpjMl8y","timestamp":1441900812512,"$":"MC4wODgwNTYyIDAuMDQyMzY5NSAtMC4yODU1NTM="},{"column"
:"TTptMmNuXzA=","timestamp":1441900812512,"$":"NyA1IDA="},{"column":"TTptMmNuXzE=","timestamp":14419008
12512,"$":"NCAzIDE="},{"column":"TTptMmNuXzI=","timestamp":1441900812512,"$":"MSAyIDI="},{"column":"TTp
tMmdwXzA=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"TTptMmdwXzE=","timestamp":1441900812512
,"$":"ZW1wdHk="},{"column":"TTptMmdwXzI=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"UjpyNG5y
","timestamp":1441900812372,"$":"Mw=="},{"column":"UjpyNHZsXzA=","timestamp":1441900812372,"$":"LTcyLjQ
zODUgLTEyOS4zNjUgNzQuMDg3"},{"column":"UjpyNHZsXzE=","timestamp":1441900812372,"$":"LTkyLjExMzIgLTcuMDY
zMTMgNTUuODUxNg=="},{"column":"UjpyNHZsXzI=","timestamp":1441900812372,"$":"LTMuMzY0ODggLTEwMS4wMTMgLTI
5MC44MTQ="}]},{"key":"NjQzOTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxNjU2NDM1MzczODY1MzkzMDMxMzIzMzYyMzc2MzM4N
jM2NTQ0NDU0ZDM4MzAzMDMwMzAzMQ==","Cell":[{"column":"TTpjMl8w","timestamp":1441900812512,"$":"MC4wODkwNz
c5IDAuMDQ0OTk0MyAtMC4yOTE5MjQ="},{"column":"TTpjMl8x","timestamp":1441900812512,"$":"MC4wOTU1ODc0IDAuMj

```
QwNTcyIC0wLjI5NzI2Ng=="},{"column":"TTptMmNuXzA=","timestamp":1441900812512,"$":"MSAyIDA="},{"column":"
TTptMmNuXzE=","timestamp":1441900812512,"$":"MiA5IDE="},{"column":"TTptMmdwXzA=","timestamp":1441900812
512,"$":"ZW1wdHk="},{"column":"TTptMmdwXzE=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"UjpyN
G5y","timestamp":1441900812372,"$":"Mg=="},{"column":"UjpyNHZsXzA=","timestamp":1441900812372,"$":"MC4w
MDQwODcyOCAwLjEyNjQxMyAwLjAyNjk0NDM="},{"column":"UjpyNHZsXzE=","timestamp":1441900812372,"$":"LTYuMjc4
OTggLTEzNi40MjYgMzU2LjEz"}]},{"key":"NjQzOTM0NjM2MTMyMzk2MjY1MzUzMzM0NjM2MTMxNjU2NDM1MzczODY1MzkzMDMxMz
IzMzYyMzc2MzM4NjM2NTQ0NDU0ZDM5MzAzMDMwMzAzMQ==","Cell":[{"column":"TTpjMl8w","timestamp":1441900812512,
"$":"MC4zMTI2NDMgLTAuMjU5ODI0IC0wLjM3NzYzOQ=="},{"column":"TTpjMl8x","timestamp":1441900812512,"$":"MC4
wOTI4NTEgMC4wMzc5ODQ5IC0wLjI5NjUyNA=="},{"column":"TTpjMl8y","timestamp":1441900812512,"$":"LTAuMDA4MDc
5NiAwLjE0NTQwOSAtMC4yOTYwNzc="},{"column":"TTptMmNuXzA=","timestamp":1441900812512,"$":"NyA1IDA="},{"co
lumn":"TTptMmNuXzE=","timestamp":1441900812512,"$":"MSAyIDE="},{"column":"TTptMmNuXzI=","timestamp":144
1900812512,"$":"MyAyIDI="},{"column":"TTptMmdwXzA=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column"
:"TTptMmdwXzE=","timestamp":1441900812512,"$":"ZW1wdHk="},{"column":"TTptMmdwXzI=","timestamp":14419008
12512,"$":"ZW1wdHk="},{"column":"UjpyNG5y","timestamp":1441900812372,"$":"Mw=="},{"column":"UjpyNHZsXzA
=","timestamp":1441900812372,"$":"LTEzLjU0NzMgLTEwLjY2OTcgNC42NTc5"},{"column":"UjpyNHZsXzE=","timestam
p":1441900812372,"$":"LTMuMTQwODcgLTc1LjM5MzkgLTE1Mi45MDQ="},{"column":"UjpyNHZsXzI=","timestamp":14419
00812372,"$":"LTMxLjQwNTMgLTYuMDQ3MTEgLTQ5LjQ5MzI="}]}]}
**********
List Of vertices : {"id":[]}

# next Query Manager connection:
statusDL
...
```


Figure 6: Output messages of the Data Layer as it connects to the stored data (in this case HBase tables), receives the status queries and the GetResultsForVerticesID from the Engine Layer, and issues a data scan of the HBase table 'Simulations_Data'

## 3.2 Engine layer implementation

The Query Manager Module (QMM) is the core module of the platform; it manages both simple and complex VQueries. It is composed of a Thrift server (to receive queries from the visualization clients) and some classes. Its main objective is to parse a query, validate it and verify the access permissions, and decompose it into an adapted workflow (described in the VQueries forms, see Deliverable D3.1 [55]). This decomposition triggers operation functions that can communicate with the analytics and graphics modules of the platform, and even with the storage module. Another task of the QMM is to find and use the best decomposition path for a query. It is also in charge of triggering long-term jobs and short-term jobs (costly or inexpensive jobs in terms of time) to do asynchronous computations, e.g., retrieve data from a coarse model or from a full resolution model.

The Query Manager is part of the engine layer, which, generally speaking, performs the computations required for the transfer of information from storage to visualization. The goal of the Query Manager is to provide communication services to the clients of the platform, and to dispatch queries to the correct modules.

In addition to the Query Manager, the engine layer consists of the following modules:

- Monitoring,
- Analytics,
- Graphics.

The subsections, below, describe their interrelationships with the Query Manager.

The Monitoring Module is in charge of the management of the platform; it aggregates log files and extracts the log status of the whole platform. It works closely with all the other modules of the platform. When a user asks for information, a query is sent from the QMM to the monitoring module, and the information produced is returned to the client.

A Thrift server handles the communication between the QMM and the Visualization Client. The server handles queries that were specified in the VQueries descriptions forms (see D3.1 [55]). This Thrift server listens to messages sent by the visualization client and parses them in order to execute the correct query. A valid query is decomposed into operations, and sent to the specific module: monitoring, graphics, analytics or storage module. The QMM also supports asynchronous queries in order to fit with the real-time requirements of the platform, e.g., to apply preliminary visualization queries on a simplified data set, while completing the computation on the full resolution data set. This module is also in charge of checking if a pre-computed result exists in the storage layer. This step is helpful to reduce the computation time of complex queries.

The Graphics Module (in this first prototype version of the Platform) is in charge of formatting stored data into a GPU friendly format. The QMM sends a query related to the result data, and the Graphics Module extracts the requested subset of information, and converts it into a GPU friendly format. The main strategy here is to store data into GPU Vertex arrays or Vertex Buffer Objects (VBO) and send the information to the client through the QMM. In the final version of the Platform this module will integrate more functionalities, like streaming visualization, navigation and multi-resolution.

All queries need to access the data layer to retrieve the raw data required to reply to an end-user request. The data layer contains as its main constituent the VELaSSCo Storage Module. The VELaSSCo platform may come with HBase storage (this is referred to as the open source architecture, see Figure 1 and 3.5.1), or with EDM storage (this is referred to as the closed source architecture, see Figure 2 and 3.5.2).

## 3.3 Query Manager Module Implementation

As shown in Figure 3, the visualization client will translate the user interactions to VQueries by means of the Access Library, and send them to the Engine Layer of the VELaSSCo platform.

As already mentioned, the Engine Layer is a Thrift server application that consists of several modules: the Query Manager, Monitoring, Analytics and Graphics Modules. It also forwards some data queries directly to the Data Layer Thrift server application.

The implementation in this first prototype is a simple proof of concept (POC) of the Remote Procedure Call (RPC) communication between the Engine Layer and the Data Layer, which is also a Thrift server application. The communication to the Analytics and Graphics module is also done using the Command Line Interface (CLI).

As mentioned in deliverable D3.1 [55], a VQuery is decomposed into several operations. The Query Manager Module is responsible for the execution of the received VQuery and their operations. These operations may include:

- Data queries that will be forwarded to the Data Layer using the Thrift protocol;
- Analytics operations that in this first prototype are performed through the CLI;
- Graphics operations, that in this first prototype just forward VQuery results data to the Visualization client in the appropriate format.

An example of the QMM process is in the output messages of Figure 5: Output messages of the Engine Layer as it connects to the Storage module of the Data Layer, receives the VQueries UserLogin, GetResultsForVerticesID and UserLogout, executes them (for GetStatus and GetResultsForVerticesID the Query Manager access the Data Layer) and returns the results.

## 3.4 Analytics Implementation

The implementation of the Analytics Module is described in D3.2 “Pre-computed, or on-demand computed, transformations stored in HPC: Engine to create multi-resolution models & co. from simulation data” [52]. In the first prototype the Analytics operations will be executed as YARN jobs using the CLI interface, such as calculating a bounding box, the boundary of a mesh, a cut plane, discrete to continuum transformations, and simplified meshes. The results of these operations are read and returned to the visualization client; they will eventually be stored in the Storage system for reuse by subsequent calls (see D3.2).

## 3.5 Storage Implementations

In this section, the Data Layer and its current state in the first prototype are detailed with focus on how simple queries retrieve data in both the open and closed architectures. Various implementation choices are discussed, as well as performance considerations. Essential implementation aspects are covered to present the more detailed technical aspects of interest to the project development.

The Data Layer is composed of different constituents, some of which already exist, and others, which are under development. This layer includes also some standard tools like Flume, Hadoop with HDFS, HBase, Jotne’s EDM and the *storage module*, a component specifically developed for this platform. This layer could be extended in the future in order to meet additional requirements, e.g., near real-time access.

### 3.5.1 Open source (HBase)

For the open source architecture the data layer includes both the source code of the storage module and the VELaSSCo Thrift Interface Description Language (IDL) file that defines the service API (Application Programming Interface) used by the Engine Layer, and some external libraries (a base64 coder/decoder and a C Json library). In order to simplify the initial prototype, we selected to implement only the HBase framework, but in future versions we will add a Hive query solution. The main file starts a Thrift threaded server

based on the Velassco IDL file. This server is in charge of listening to RPC queries performed by the Query Manager Module (QMM). These queries are described by operation forms available on the Alfresco document management platform (see also D3.1 [55]).

As an example, we specify here an operation that is a part of the first VQuery. This operation is called op-22-100: getResultFromVerticesID, which is declared in the Thrift IDL file and called by the QMM. This operation is decomposed into two components (functions): getResultOnVertices and checkIfAllVerticesArePresent. These two functions are implemented in the Storage Module. The storage module is a singleton instance with some functions. Also these functions are defined by the forms available on the Alfresco document management platform.

The first implemented version used the REST (Representational State Transfer) API of HBase. The HBase REST API has a specific feature: it can perform a select operation using a wildcard character '*' for the last part of the Rowkey, allowing the selection of several rows from the HBase table. The interaction with HBase is performed using the CURL library. The REST API is the software architectural style of the World Wide Web and, thus, the HBase access queries are based on formatted URLs (Uniform Resource Locator). This first query is shown below (though based on an old description of Rowkey):

```
string cmd = "http://pez001:8880/";
cmd += "Simulations_Data";
cmd += "/";
std::stringstream key;
key << "0x";
key << modelID;
key << analysisID;
key << timeStep;
key << resultID;
key << "*";
```

The last part of the query uses a wildcard character in order to select the correct element. Then we set all the necessary items to open a HTTP connection, i.e., set CURL options, and open the connection. The query is performed (curl_easy_perform), and we free the CURL object. The result of the query is stored in a variable set by the curl_easy_setopt. Then the buffered data is filtered using a FEM- or DEM-specific methodology. The resulting information is returned to the checkIfAllVerticesArePresent function. This function is in charge of adding information related to missing elements to the result message, i.e., a message containing the IDs of all the missing nodes.

All the information stored concerning the output of this function is formatted using a JSON format, selected because the HBase REST API returns JSON data (or XML). To manage JSON data, we use a C library named cJSON. The formatted message, which contains information related to vertices and missing vertices can now be returned to the engine layer through the RPC call.

### *3.5.1.1 Hadoop HDFS*

This chapter discusses the implementation of the HDFS in an HPC system, as well as the proposed integration of EDM (at the lower level of the platform). We also examine the configuration of the cluster (on local storage and network storage), that is, the I/O tests performed to evaluate the possible use of a network storage system.

The choice of HDFS instead of a traditional storage has some advantages. This solution provides an abstract communication layer, which enables the platform to communicate natively with most foreign file systems (FS) without modifications.

In the early days of the project, we discussed how to integrate EDM at the lower level of the platform and use it as a basic storage system. The main idea was to reduce the development of the EDM integration, and enable the integration of EDM at the Hadoop level. In many papers, databases have been integrated at the bottom level, to have the benefit of the distributiveness of Hadoop (for both computations and storage). However, the EDM database is object-oriented, a design that supports EXPRESS-based data such as AP209 files. The low level integration would remove this market value of EDM by reducing EDM to a "simple file system". Thus, EDM is now being integrated at a slightly higher level, which maintains the object-oriented nature of the DataBase Management System (DBMS). However, EDM will be kept separate from the Analytics Module, as the Analytics Module is common for HBase and EDM. The Hadoop cluster will provide storage for EDM through the NFS gateway into HDFS. The main idea with this approach is to use the standard EDM read and write operations to manipulate data in HDFS.

For the current implementation in the Acuario cluster, located at the CIMNE premises, each node owns one or two small local hard drives. Traditional HPC systems are based on a long term networked storage system, and a small sized short-term local storage. On the UEDIN HPC cluster, most of the storage is provided by network storage; the update of the cluster to assign dedicated storage disks to processing nodes is in progress. Usually it is non-trivial to request changes to the architecture of a HPC centre, but the IT administrators of the UEDIN cluster liked the progressive expansion plan proposed by VELaSSCo, i.e., first extend the local storage on a few nodes, monitor their performance, and gradually increase the number of extended nodes until reaching the 100 extended nodes.

Before having access to the UEDIN cluster to perform tests there, we made an evaluation of the Acuario cluster using an existing tool.

The application TestDFSIO is helpful to stress HDFS and discover bottlenecks in the cluster. This tool is used to give a first overview of the available I/O performance of the cluster. The only restriction here was to run a write operation before a read operation. Our evaluation script is located in: "/localfs/home/velassco/measure-io". This script run write and read operations using MapReduce jobs on a specified number of files. The number of generated files can be set in the nbFiles variable, and the fileSize variable is used to specify the file sizes. The final variable is the number of tests (in the example, five tests are run). We ran

this evaluation on two configurations: the current Acuario cluster composed of seven nodes using their local storage, and another cluster specially configured to store data in the CIMNE network storage system. The result of read and write operations are presented below (each figure uses a logarithmic scale).

In Figure 7 and Figure 8, we show the I/O performance reach by the Hadoop cluster on the local (gray and yellow) or network storage (blue and orange). The performance test could not be completed due to the limited bandwidth and space of the network storage. To avoid external interferences for the local storage, the test were run during the night, while for the network storage the test were run during week-ends. The local storage is faster than the network storage by a factor of at least 12. These performance tests will be repeated on the evaluation and deployment cluster at UEDIN.

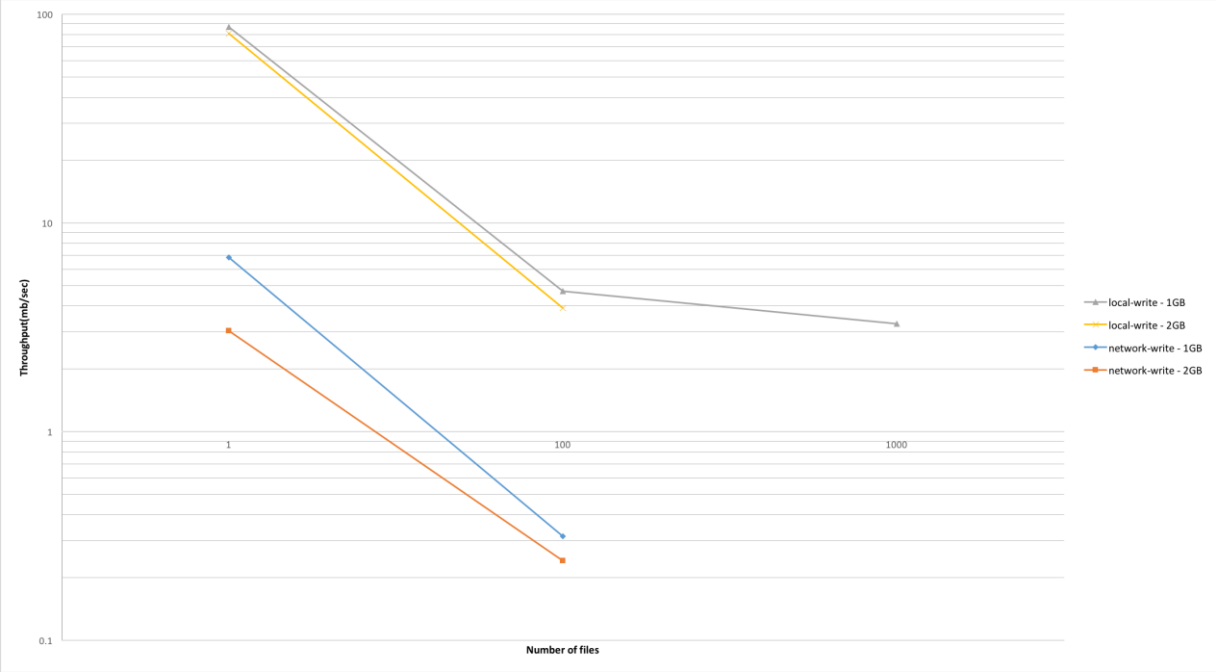


Figure 7: I/O performance for the write operation on the Acuario cluster

In the case of read operations, the difference between the local and network storage is less important than for the write operations. In the current architecture we only use seven nodes for the Hadoop cluster. In the future UEDIN cluster, more nodes will be used and a more important bottleneck might appear.

The HDFS cluster has some configuration parameters. Here, we use only a subset of all the available parameters. These parameters are presented below:

Core-site.xml:

hadoop.tmp.dir (refers to the temporary storage of Hadoop)
fs.defaultFS (refers the the access method of the hdfs cluster)
hadoop.proxyuser.velassco.hosts (needed for the NFS gateway)

hadoop.proxyuser.velassco.groups (needed for the NFS gateway)

hdfs-site.xml:

dfs.namenode.name.dir (refers to the namenode storage repository of the cluster)
dfs.datanode.data.dir (refers to the datanode storage repository of the cluster)
dfs.webhdfs.enabled (enables the web api of HDFS)

Some other files have been updated (mr-site and yarn-site), but they concern the execution engine. They will be presented later on.

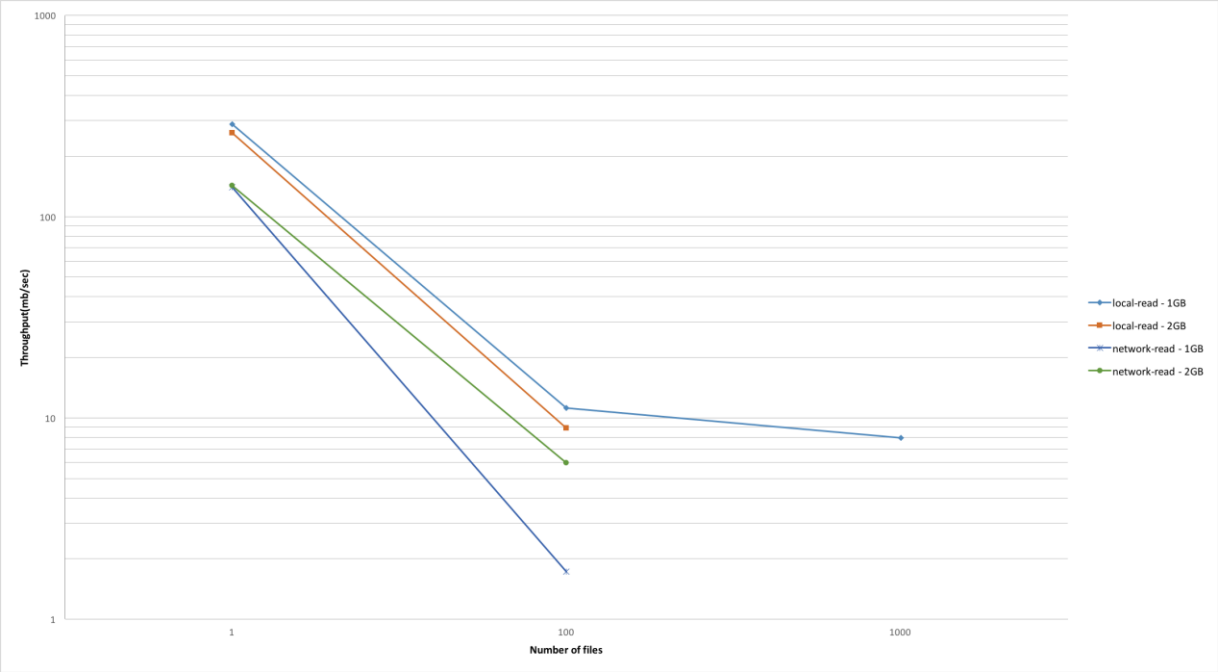


Figure 8: I/O performance for the read operation on the Acuario cluster

For the network storage, it is necessary to update the configuration file. However, due to some access restrictions it is not possible to write data into the same directory, and the Hadoop starting script uses the current configuration of the Hadoop cluster. Thus, to configure network storage, the best solution is to use a common symbolic link for the data storage. On each node, it is necessary to create a symbolic link for each node to a network folder. An example of this solution can be found in: /localfs/home/velassco/link. On each node two folders, DN and NN, are linked to the network directories: /home/blange/data/pez<xx>/DN and NN. This solution increases the complexity of the deployment of a Hadoop cluster on a centralized storage system.

### *3.5.1.2 HBase*

The open source storage module is implemented using HBase. It is an open source BigTable implementation of Google. It stores data into a tabular format, and provides a fast and efficient key indexing method. HBase can be plugged into different storage systems, and most Hadoop components can use it. Specific developments considering HBase are planned

in VELaSSCo. So far, most of the discussions were focused on the definition of the table structure, which has now been fixed. The remaining discussions are related to reference key generation.

The selected table structure is not the most compact alternative, but supports some useful filtering operations on the data. These filters will be used to reduce the sizes of extracted data sets.

To query an HBase table, different solutions can be embedded into an external application: the Java API for a JAVA application, a Rest API or a Thrift API, which also provides the necessary interfaces for C and C++.

In the current development, we have mainly focused on the Thrift API, which provides most of the necessary features for the access to the information stored in the table. Here, the usage of this API is discussed; the REST API was discussed in the Storage Module section. To perform the queries, we mainly use the getRow function, when the rowkey is known. We perform some scan with filtering when we need to extract a larger subsets of the data.

For the configuration of HBase, all the necessary files are located in:

/localfs/home/velassco/common/hbase/conf

For this application, the configuration is centralized in one file: 'hbase-site.xml'. In this file, the configuration concerns:

- hbase.rootdir : to specifcy the location of HBase data in the HDFS storage
- hbase.cluster.distributed: to enable the distributiveness of the system
- hbase.zookeeper.quorum: to specify the zookeeper server
- hbase.master.info.bindAddress: to specify the master node of HBase
- hbase.replication: to enable replication.

In this context, we also specify a list of 'hbase data nodes' in the 'regionservers' file, to use the distributiveness of the computation and storage.

### 3.5.2 Closed source (EDM)

The EDM database plug-in for the VELaSSCo platform is a server process that implements the services defined in the VELaSSCo Storage Module Thrift IDL file. This file is located in "https://projects-ext.igd.fraunhofer.de/svn/VELASSCO/trunk/modules/Thrift/VELaSSCo/velassco.Thrift".

Three different data injectors were developed for the first prototype:

- Thrift based server for injecting DEM data. Currently one DEM time step is ingested in one transaction. This is not a solution for real data because of the data size. ATOS will propose "smaller" injector methods for use in a Flume based injector.

- File based DEM injection of the Alfresco dataset "Documents->Data->DEM small examples->DEM test data for Particle schema AP209".
- File based injection of FEM data. The implementation was developed in collaboration with CIMNE. As part of this task an EDM database schema was defined that reflects the structure of the CIMNE data.

The prototype has the following characteristics:

- Data models are read into a memory cache when the Plug-in server is started.
- Data structures for supporting search are built when the Plug-in server is started.
- Each EDM based VELaSSCo database plug-in server process (later referred to as the EDM plug-in) has its own database. Each database can only be opened by one EDM plug-in process. Objects in an EDM database cannot link directly to objects in other EDM databases. This means that the simulation domains that are stored in EDM must be "self contained" and shall not have references to other domains.
- The EDM plug-in process is single threaded. This means that if two parallel queries on the VELaSSCo platform need access to the same EDM plug-in database at the same time, the two queries will be executed sequentially and not in parallel.

The following queries are implemented:

- GetElementOfPointsInSpace
- GetBoundaryOfLocalMesh
- UserLogin
- UserLogout.

Start up commands/parameters for the EDM plug-in process must be given on the command line with parameters in the following sequence:

1. Communication port
2. Database folder
3. Database name
4. Database password.

For the first prototype, the EDM plug-in server runs on one Windows machine with one EDM database.

A test program for the EDM plug-in, named Plug_in_tests is implemented. The program tests a set of queries in sequence. Below is a sample of output from the test program:

```
--->GetElementOfPointsInSpace  - 1000 random points:
Return status: OK
Comments:
No errors in returned data detected.

--->GetBoundaryOfLocalMesh with empty model name:
Return status: Error
Comments: Model does not exist.
Expected error message returned
```

--->GetBoundaryOfLocalMesh:
Return status: OK
Comments:

Enter a character to stop the program.

Figure 9: Output from the EDM plug-in test program

# 4 Tasks for the second release of this prototype

The second release of the VELaSSCo prototype will address the following tasks:

1) EDM plug-in
    a. The data ingestion via Flume will be adapted to load large data sets via several transactions so that each transaction handles a relatively small amount of data.
    b. The second prototype of the EDM plug-in will be a server that implements the Thrift interface between the VELaSSCo engine layer and the data layer. It will manage all EDM plug-in slave servers that are running on the HPC nodes. It will have one master database that holds the information about all EDM plug-in slave servers in the platform. To execute a user query the master will start queries on all relevant slave servers in parallel.
2) Hive and Phoenix
    a. The use of additional tools will be examined for alternative access solutions. Two examples are Hive (proposed by INRIA) and Phoenix (proposed by ATOS). Each of these tools may help to reduce development costs of the platform. For example, Hive might express a MapReduce job using a SQL query. The tool uses a table stored in the HDFS system, but can also be linked to an HBase table. It provides access APIs for integration with the following external tools:
        i. Command Line Interface
        ii. JDBC
        iii. Python
        iv. PHP
        v. ODBC driver
        vi. Thrift
    b. For the second release, such tool might be linked to the VELaSSCO HBase table.

# 5 Abbreviations and Definitions

Table 1: Table of acronyms

| Term | Definition |
|---|---|
| 2D | Two (2) dimensional |
| 3D | Three (3) dimensional |
| AIA | Aerospace Industries Association |
| AIM | Application Interpreted Model (ISO 10303) |
| Analysis Type | Type of simulation of the physical phenomena. For instance "Time analysis" of a process simulation, "Frequency analysis" of a resonant frequency analysis of a structure, in a casting simulation there can be two analyses: "Filling process", in which the amount of air bubbles are to be simulated, and "Cooling process", in which the residual stresses are simulated. In the VELaSSCo project we will have usually a single analysis, but there should be the possibility to handle several analyses.<br>ISO 10303: (entity Classification_assignment) |
| AP | Application Protocol (ISO 10303) |
| AP209 | ISO 10303-239 |
| API | Application Programming Interface |
| ARM | Application Reference Model (ISO 10303) |
| ASD | AeroSpace and Defence Industries Association of Europe |
| CAD | Computer Aided Design |
| Characteristic | Abstraction of a property of an object or of a set of objects (ISO 1087-1 [8]) |
| Class | Category or division of things based on one or more criteria for inclusion and exclusion (ISO 15926-1 [10]) |
| CLI | Command Line Interface |
| Component | Is the minimal conceptual entity of the VELaSSCo platform. |
| Concept | A human understanding of an object unit of knowledge created by a unique combination of characteristics (ISO 1087-1 [8]) |
| Converter | Software that imports, transforms, loads, merges and exports data from one system to another. A converter is a device the purpose of which is to convert attributes of one device or system to those of an otherwise incompatible device or system. |
| Coordinates | x, y and z values that represent a point in space. The z coordinate may not be present in data and should be handled as z = 0 .<br>ISO 10303: A Cartesian_point is a type of Point that defines a point by a list of up to 3 cartesian coordinates (entity Cartesian_point). |

| Term | Definition |
|---|---|
| COTS | Commercial off-the-shelf |
| Data | Representation of information in a formal manner suitable for communication, interpretation, or processing by human beings or computers (ISO 10303-1 [6]) |
| Data query | Queries performed over the data of the model / simulation data. These can be Hive-queries, HBase-queries, HDFS-queries or EDM-queries. |
| DBMS | Database management system |
| EDIG | Engineering Data Interoperability Working Group (AIA) |
| EDM | EXPRESS Data Manager™ |
| EDMS | EDMSupervisor™ |
| Element | Basic geometric elements that represent a piece of spatial region to be simulated. In VELaSSCo elements can be points, circles, spheres or complex-particles or lines, triangles, quadrilaterals, tetrahedrons, hexahedrons, prisms, pyramids in their linear or quadratic forms.<br>ISO 10303: An Element is a basic building block of a Fea_model. It defines the mathematical relationship between the finite element nodes. An Element may be either a Curve_element or a Directionally_explicit_element, or an Explicit_element, or a Point_element, or a Substructure_element, or a Surface_element, or a Volume_element (entity Element). |
| EPMT | Jotne EPM Technology AS |
| EXPRESS | Data modelling language, defined in ISO 10303-11 |
| EXPRESS-X | Data manipulation language, defined in ISO 10303-14 |
| Flume | Is a distributed data collection tool, which was designed to aggregate and move large amounts of streamed data. |
| FS | File System |
| Gauss Points / Integration points | Points defined in elements using natural coordinates ( relatives to the element). A simulation which uses gauss points will, usually, use the same gauss points definition for all elements of the same type. Depending on the element type the definition changes. |
| GUI | Graphical user interface |
| Hadoop | Is a framework, which allows the distribution of processing and storage across clusters of computers. |
| HBase | Is distributed big data storage based on the big table proposal. |
| HDFS | Hadoop File System - Is a JAVA-based file system, which provides a high scalability and reliability for data storage. |
| ICT | Information & Computer Technology |

| Term | Definition |
|---|---|
| Information | Knowledge concerning objects, such as facts, events, things, processes, or ideas, including concepts, that within a certain context has a particular meaning (ISO/IEC 2382-1 [7]) |
| Information | Facts, concepts, or instructions (ISO 10303-1 [6]) |
| ISO | International Organization for Standardization |
| ISO 10303 | Industrial automation systems and integration - Product data representation and exchange |
| ISO 10303-11 | Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual |
| ISO 10303-21 | Industrial automation systems and integration - Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure |
| ISO 10303-209 | Industrial automation systems and integration - Product data representation and exchange - Part 209: Application protocol: Multidisciplinary analysis and design |
| Layer | Conceptual decomposition of the platform. In VELaSSCo we have identified 4 layers: one related to users, one for the computational engine, one for the storage, one for the simulation. |
| Mesh | A subdivision, discretization, of the simulated domain (region of space) into basic geometric elements like triangles, tetrahedrons or spheres. This includes both particles, surface and volume mesh. Volume meshes may represent both solids and volume data, for instance, air surrounding a racing car, water flowing through a pipe, etc.<br>ISO 10303: arrangement of cells with connectivity between the cells defined by the possession of common cell faces or cell edges (entity Mesh). |
| Model | A specific simulation case data set, including geometry, conditions, materials, mesh and results, i.e. the simulation data. It also refers to the geometry of the problem to be simulated.<br>ISO 10303: a container within which related entity instances exist (SDAI, ISO 10303-22) |
| Module | Is a specific functionality in the proposed architecture of VELaSSCo. Boxes in the global schema represent it. |
| Node/Vertex | A point in space with an identification number ( ID). In VELaSSCo the term vertex will be used for traditional FEM nodes. On static meshes, i.e. global meshes that do not change along the whole simulation analysis, the ID will be unique. On dynamic meshes, i.e. meshes defined at each time-step of the analysis, the ID will be unique for the meshes defined at one particular time-step. |

| Term | Definition |
|---|---|
| | ISO 10303: A node is a discretisation point for the field variables of the finite element analysis model (entity Node). |
| Object | Anything perceivable or conceivable (ISO 1087-1 [8]) |
| Operation | Is a part of a module. It is composed by a set of components. |
| P21 | ISO 10303-21, |
| P28 | ISO/DIS 10303-28e2 |
| PDM | Product Data Management |
| POC | Proof of Concept |
| QMM | Query Manager Module |
| Repository | ISO STEP: an identifiable data storage facility (SDAI, ISO 10303-22) |
| REST | Representational State Transfer |
| Result | May refer to the outcome of the simulation program including mesh and results defined over the mesh (simulation data); or to the results values defined over the vertices of the mesh or over the integration points, or gauss points, defined on the elements of the mesh. Results value(s) are one or several double floating point numbers depending on the result type. For the VELaSSCo platform we focus on Scalar, Vector, Matrix 2D and Matrix 3D types.<br>ISO 10303: the instances of Model_property_distribution that result from a simulation (entities Simulation_run and Model_property_distribution). |
| RPC | Remote Procedure Call |
| R&D | Research & development |
| SDAI | Standard Data Access Interface (ISO 10303-22) |
| Simulation | Program that solves equations using a discretization of the domain and generates results. These results may include meshes and result values defined over these meshes.<br>ISO 10303: A Simulation_run is an individual activity that simulates a Numerical_model (entity Simulation_run). |
| Simulation data | Outcome of the simulation program including mesh and results defined over the mesh.<br>ISO 10303: (see Mesh, Result etc.) |
| Step | An analysis can have several steps, for instance "frequency analysis" may have different steps, which represent different simulated frequencies. A "Time analysis" will have several time-steps. In the VELaSSCo project, for simplification purposes, we will mention time-steps to refer to the steps of an analysis.<br>ISO 10303: An Fe_analysis_control_step is a single step in a Fe_analysis. A Fe_analysis_control_step may be either a |

| Term | Definition |
|---|---|
| | Modes_and_frequencies_control_step or a Static_control_step. (entity Fe_analysis_control_step) |
| STEP | Standard for the Exchange of Product Model Data |
| TO | Technical Officer of the European Commission |
| Validation | The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements. [12] Confirms that the system meets the requirements defined in the user requirement document |
| Verification | The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase [12]. Checks that each component meets its specific requirement, usually as defined in the design document |
| Vquery (VQ) | Functionality offered/exposed by the QueryManager for the AccessLibrary to call depending on the visualization client demands. For instance, if the user wants to visualize a contour fill (coloured areas) of a certain result/step/analysis, the visualization client will generate one or more VQxxx-GetMeshData and VQxxx-GetResultsForVerticesID. VQxxx names are temporary. |
| XML | Extensible Mark-up Language |

# 6 References

[1] http://hadoop.apache.org/docs/r1.1.2/cluster_setup.html
[2] http://edureka.co/blog/hadoop-cluster-configuration-files/
[3] Benslimane, Z. (2013). Optimizing Hadoop Parameters Based on the Application Resource Consumption.
[4] https://hadoop.apache.org/docs/r2.3.0/hadoop-project-dist/hadoop-hdfs/HdfsNfsGateway.html
[5] https://hadoop.apache.org/docs/r2.3.0/hadoop-project-dist/hadoop-common/core-default.xml
[6] https://hadoop.apache.org/docs/r2.3.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml
[7] Enis Söztutar, HBase and HDFS, Understanding file system usage in HBase. http://fr.slideshare.net/enissoz/hbase-and-hdfs-understanding-filesystem-usage
[8] http://hbase.apache.org/book.html#external_apis
[9] http://hbase.apache.org/book.html#config.files
[10]http://www-01.ibm.com/support/knowledgecenter/api/content/nl/fr/SSPT3X_3.0.0/com.ibm.swg.im.infosphere.biginsights.analyze.doc/doc/bigsql_TuneHbase.html
[11]https://cwiki.apache.org/confluence/display/Hive/HiveODBC
[12]https://cwiki.apache.org/confluence/display/Hive/HiveClient#HiveClient-Thrift
[13]https://cwiki.apache.org/confluence/display/Hive/HBaseIntegration
[14]http://fr.hortonworks.com/blog/hbase-hive-better-together/
[15]http://hortonworks.com/blog/hbase-via-hive-part-1/
[16]http://www.qubole.com/blog/big-data/hive-best-practices/
[17]http://www.quora.com/What-are-the-best-practices-around-designing-tables-on-Hive-Also-are-there-any-tips-tricks-on-improving-hive-performance-I-understand-that-there-may-not-be-any-silver-bullets-but-any-pointers-would-be-very-helpful
[18]http://www.idryman.org/blog/2014/03/05/hadoop-performance-tuning-best-practices/
[19]http://fr.slideshare.net/Hadoop_Summit/w-235phall1pandey
[20]http://fr.slideshare.net/Hadoop_Summit/w-1205p230-aradhakrishnan-v3
[21]http://fr.hortonworks.com/blog/5-ways-make-hive-queries-run-faster/
[22]http://fr.hortonworks.com/hadoop-tutorial/real-time-data-ingestion-hbase-hive-using-storm-bolt/
[23]http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/ResourceManagerHA.html
[24]http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/TimelineServer.html
[25]http://fr.slideshare.net/vgogate/hadoop-configuration-performance-tuning
[26]http://fr.slideshare.net/Hadoop_Summit/w-525hall1shenv2
[27]https://hadoop.apache.org/docs/r2.3.0/hadoop-yarn/hadoop-yarn-site/WritingYarnApplications.html
[28]https://hadoop.apache.org/docs/r2.6.0/hadoop-mapreduce-client/hadoop-mapreduce-client-hs/HistoryServerRest.html#Response_Examples

[29]http://www.cloudera.com/content/cloudera/en/documentation/archives/cloudera-manager-4/v4-5-2/Cloudera-Manager-Enterprise-Edition-User-Guide/cmeeug_topic_2_3.html
[30]http://blog.sequenceiq.com/blog/2014/10/07/hadoop-monitoring/
[31]http://hbase.apache.org/0.94/book/ops.monitoring.html
[32]http://hbase.apache.org/0.94/book/hbase_metrics.html
[33]http://blog.cloudera.com/blog/2012/11/introducing-hannibal-a-tool-for-hbase-region-monitoring/
[34]http://blog.cloudera.com/blog/2009/03/hadoop-metrics/
[35]http://java.dzone.com/articles/enabling-jmx-monitoring-hadoop
[36]http://fr.slideshare.net/miloveme/hive-performance-monitoring-tool
[37]AIA EDIG Guidebook - Aerospace Industry Guideline for Implementing Interoperability Standards for Engineering Data.
[38]ISO 10303-209:2014, Industrial automation systems and integration -- Product data representation and exchange -- Part 209: Application protocol: Multidisciplinary analysis and design
[39]EU R&D project Generic Engineering Analysis Model (GEM) http://cordis.europa.eu/project/rcn/21959_en.html
[40]CFD General Notation System (CGNS), http://www.grc.nasa.gov/WWW/cgns/CGNS_docs_rel31/
[41]Volvo Aero EAR-model, http://www.powershow.com/view/146e66-NTY4M/The_EARmodel_Fundamentals_and_IAS_Implementation_Volvo_Aero_powerpoint_ppt_presentation
[42]AP209 on the web, http://www.ap209.org
[43]ISO 10303-21:2002, Industrial automation systems and integration -- Product data representation and exchange -- Part 21: Implementation methods: Clear text encoding of the exchange structure
[44]ISO 10303-11:2004, Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual
[45]ISO 10303-22:1998, Industrial automation systems and integration -- Product data representation and exchange -- Part 22: Implementation methods: Standard data access interface
[46]Business Objects for Industrial Data Standards, Keith A. Hunten P.E. and Allison Barnard Feeney, DETC2011-47965, ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, August 29-31, 2011, Washington, DC, USA
[47]ISO 10303-110:2011, Industrial automation systems and integration -- Product data representation and exchange -- Part 110: Integrated application resource: Mesh-based computational fluid dynamics
[48]Recommended Practices for AP 209, PDES, Inc. ME007.01.00, revised version of 2002-04-23

[49]Geometric Founding and Associativity in ISO 10303-209, K. A. Hunten, P.E., Revision B, 2001-02-15
[50]VELaSSCo D1.3 – Technical requirements, 2014-03-31
[51]EDMassist VOLUME IV: EDMinterface™ Application Development Guide and Binding Reference, http://edmserver.epmtech.jotne.com/EDMassist/WebHelp/EDMassist.htm
[52]VELaSSCo D3.2 – 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, 2015-09-30
[53]VELaSSCo D4.2 – First Prototype of the High Performance Visualization / Scalable Visual Analysis Client(s), 2015-09-30
[54]VELaSSCo D2.3 – HPC cloud infrastructure specification document suitable to the needs of e-Science, 2014-12-31
[55]VELaSSCo D3.1 – Query framework implementation in the project database system & report, 2015-04-30