VELaSSCo

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

D3.5 – 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 1.1

Deliverable Information

Grant Agreement no	619439
Web Site	http://www.velassco.eu/
Related WP & Task:	WP3, T3.4
Due date	December 31st, 2015
Dissemination Level	PU
Nature	P
Author/s	Jochen Haenisch
Contributors	Miguel Pasenau, Ivan Martinez, Abel Coll, Olav Liestøl

Approvals

	Name	Institution	Date	
Author	Jochen Haenisch	Jotne	2016-04-12	
Task Leader	Miguel Pasenau	CIMNE	2016-04-15	
WP Leader	Miguel Pasenau	CIMNE	2016-04-15	
Coordinator	Abel Coll	CIMNE	2016-04-15	

Change Log

Version	Description of Change
Version 1.0	Completed final draft for delivery to PO
Version 1.1	Corrected broken links as pointed out by the project reviewers

Table of Contents

Table of figures

Table of tables

1 Introduction

1.1 Scope of the deliverable

This is deliverable D3.5 with the following title and description: "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: Installation of a database management system for the WP4 visualization tool to answer simple queries based on the indexed pre-computed results. This includes prototype software and a report in two deliverables, three months apart.".

This report is the second in a sequence of two deliverables. The first deliverable, D3.4, described the initial version of the first project prototype; it was due and was delivered at month 21 of the project. The prototype shows how pre-computed results of simple queries can be requested, retrieved and presented by the VELaSSCo visualization tool. The scope of D3.4 and of this document is the role and the implementation of VELaSSCo Queries (VQueries) in the two database management systems in the overall architecture.

This deliverable, D3.5, describes modifications to the first prototype and includes lessons learnt; it was scheduled for month 24, after the evaluation of the platform by the user panel in an evaluation event.

D3.5 is slightly delayed because of delays in the development of the prototype and its deployment for the evaluation event. The latter was mainly caused by the loss of a key resource and Hadoop expert who could not be replaced in the expected time. Thus, also the evaluation event had to be postponed to 2016-02-04. D3.5, therefore, summarizes changes to the VELaSSCo platform that result from internal evaluations and from first-handed impressions by external users of the platform since D3.4. Several of the results of the evaluation event can first be included in an update to the version of the platform that will be used during the project review.

Correspondingly, the project team plans to deliver an update of D3.5 after the review. It will cover the modifications of the first prototype due to user feedback of the evaluation event and will be completed at the latest three months after the evaluation event, as originally planned, that is, before 2016-04-28.

1.2 Related DoW task

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 .”

1.3 Related deliverables

D3.5 relies on D3.3, the second version of “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 D3.5 provides a global view of the Simple queries engine, D3.3 provides an insight view of analytics functionalities and transformations like GetBoundary Mesh, GetBoundingBox, Discrete2Continuum and Calculate Multi-resolutions. These are integrated into the first prototype of the VELaSSCo platform.

1.4 Contents of the document

Based on the guidance by the DoW this document describes the implementation of the modifications of the first prototype of simple queries, that is, the architecture and the execution process aspects of the platform and its modules. Relevant issues are discussed and lessons learnt are reported.

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) lessons learnt for the continuous development of the VELaSSCo platform.

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

Lessons learnt are input to D1.4 and D1.6, which will update the project technical requirements and will guide further implementation.

2 Latest VELaSSCo system architecture

Figure 1 and Figure 2 depict the current system architectures of the two scenarios used to produce the revised prototype. They have been updated compared to D3.4. Figure 1 depicts the architecture based on open source software, Figure 2 the one with Jotne's DBMS EDM.

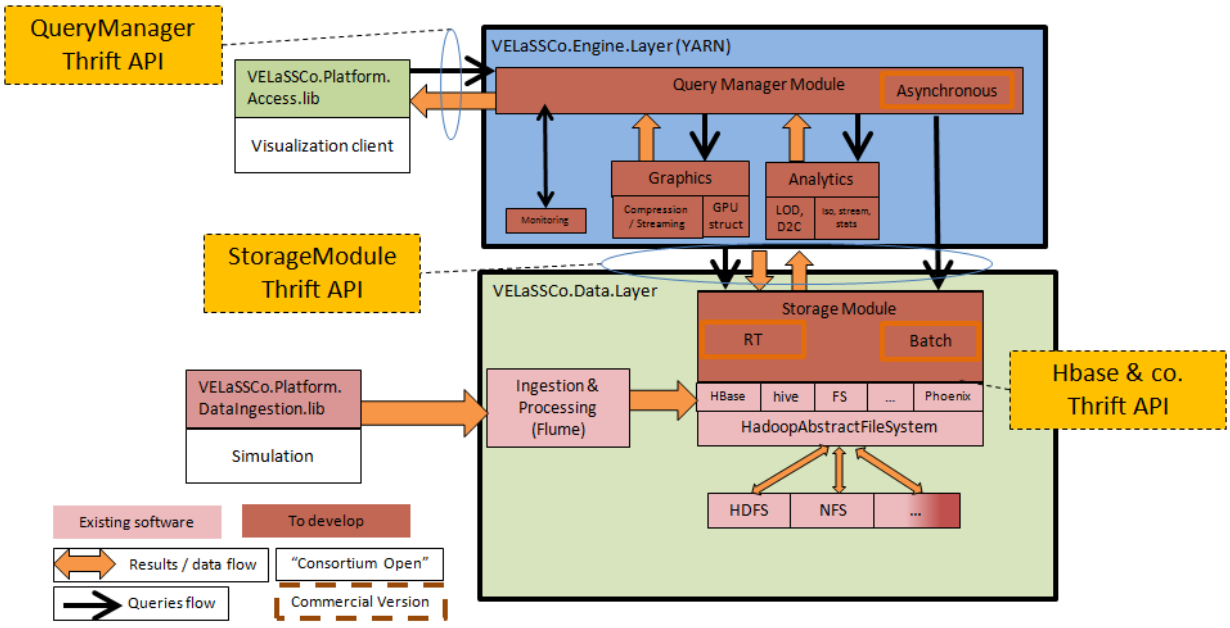

Figure 1 VELaSSCo prototype, open source architecture, version 2015-06-16

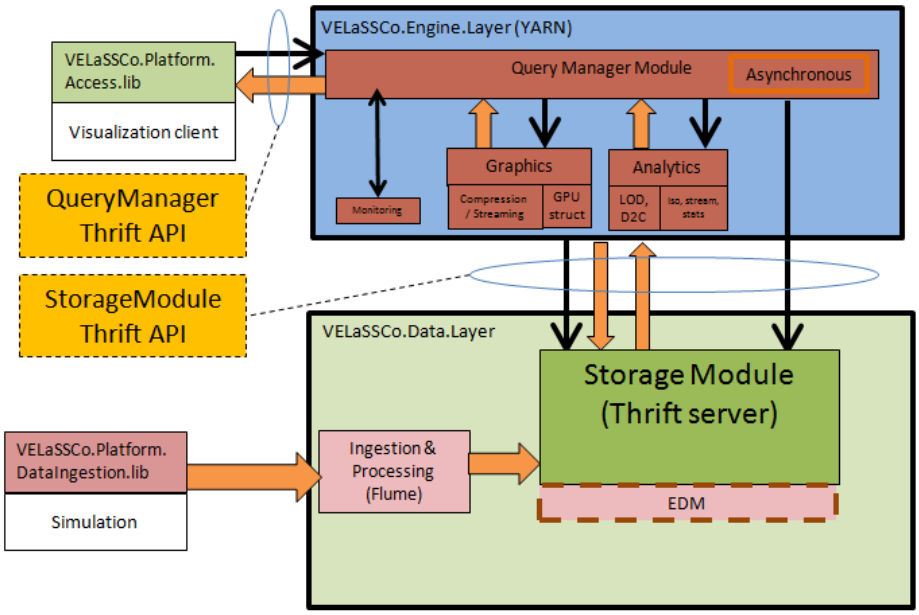

Figure 2: VELaSSCo prototype, closed source architecture, version 2015-10-21

Deliverable D4.2 [1] relates in detail the modules of this architecture to the workflow involved in the simple VQueries.

3 Revision summary

The intention of this modified first prototype is to incorporate feedback from evaluations of the initial version of the first prototype. Focus is still not to deliver a wide range of end-user functionality. Instead, 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. Evaluations included both DEM and FEM data.

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. Finally, the queries listed below have been implemented.

In order to have a working prototype so that it can be evaluated before the review meeting several compromises have been taken and not all the originally planned VQueries, listed in D3.4 and other deliverables, have been implemented.
The list of Vqueries implemented in the current version of the platform are:

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.
2. Model view:
    a. Get DEM particles
    b. Extract the bounding box of the model
    c. Extract skin of Volume meshes, i.e. set of tetrahedrons
3. Results view:
    a. Get list of analyses, time steps, and results properties (name, type, …)
    b. Given a list of nodes (vertices), get the results values for these nodes

The following analytics Vqueries are implemented and described in deliverables D3.2 and D3.3:

1. GetBoundingBox,
2. GetBoundaryOfAMesh and
3. DiscreteToContinuum transformation.

3.1 Modifications forecasted in D3.4

D3.4 predicted that the modified release of the first VELaSSCo prototype would address the following tasks. The list below indicates to which degree this has happened:

1) EDM plug-in
    a. D3.4: “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.”
        i. Done; large data sets are split and ingested into different databases.
    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.”
        i. Done; see section 3.4.2 .
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

        For the second release, such tool might be linked to the VELaSSCO HBase table.”

        i. Apache Phoenix has not been applied to the modifications of the first release, but is still up for consideration for the second release. Apache Phoenix is a relational database layer over HBase delivered as a client-embedded JDBC driver targeting low latency queries over HBase data. Apache Phoenix takes your SQL query, compiles it into a series of HBase scans, and orchestrates the running of those scans to produce regular JDBC result sets. The table metadata is stored in an HBase table and versioned, such that snapshot queries over prior versions will automatically use the correct schema. Direct use of the HBase API, along with coprocessors and custom filters, results in performance of

approximately milliseconds for small queries, or seconds for tens of millions of rows.

3.2 Modifications triggered by internal evaluations

3.2.1 GetResultsFromVerticesID

Given a list of vertices id's, this VQuery is used to retrieve result values for them.

Issue

The execution of Vquery GetResultsFromVerticesID showed poor performance especially when it returned single values, which was needed for the evaluation event use cases that showed single node results or their evolution over time, Node evolution graph.

Resolution

1) The function code was optimized by including a special path for single values.
2) Measurements showed that hand-over of huge amount values through Thrift is significantly faster if this happens by binary (BLOB) data.
3) Operating systems and types of compilers with their different options have additional impact on performance and need to be carefully tested and synchronized.

Issue

GetResultsFromVerticesID showed also poor performance when retrieving and returning very many values, such as result values for the round 400,000 vertices on a boundary mesh of the FEM telescope model.

Resolution

1) The list of vertex identifiers is not passed between the QueryManager and the Visualisation Client as a JSON list of comma separated integers any more, but as a JSON base64 encoded binary array.
2) The returned data is compressed between the QueryManager and the AccessLibrary integrated in the Visualization Client; this is further discussed in D3.3.
3) For the second prototype it will be considered to pass all arguments in binary format. However, as this would require rewriting of Vqueries in AccessLib and in the QueryManager, this measure will depend on the result of an effort estimation.

Issue

GetResultsFromVerticesID performed badly when applied to the node evolution graph use cases.

Resolution

1) In addition to the above mentioned generic measures of single value retrieval, the number of steps of the step list for creating the node evolution graph was reduced to five.
2) A specific path will be added to the query function to get the result values for a single vertex along several time-steps.

3.2.2 GetMeshDrawData

This VQuery is used in the prototype to retrieve the sphere elements of the DEM model to be rendered in the visualization client. It will also support other element types (like line and triangles) in further implementations, for the final version of the VELaSSCo platform.

Issue

The query did not perform fast enough.

Resolution

1) Returned values will be compressed as described in section 3.2.3 to speed up transmission times.
2) The algorithm that retrieves required data from the set of simulation data may be improved.
3) Specific support for lines and triangles may be added.
4) An analysis will be performed to detect the other bottleneck locations.

3.2.3 Data transmission

Issue

The transmission of data from the UEDIN HPC cluster to remote visualization clients took unreasonably much time.

Resolution

1) The data will be compressed using zlib or lzop by the StorageModule on the HPC side and uncompressed by the QueryManager in the machine with the visualisation software; this is further discussed in D3.3

3.3 Impact of the Eddie HPC cluster

After the delivery of D3.4, at the end of M24 the EDDIE HPC cluster in Edinburgh was put in commission for the project. The open source solution, which is being developed in CIMNE's Acuario cluster, was also deployed in Eddie which brought some modifications of the VELaSSCo platform and its Query engine. The closed solution stayed on Acuario due to its MS Windows dependency. The evaluation event was performed using EDDIE.

In spite of good support from UEDIN, the transition caused some delays. The entire infrastructure needed to be reinstalled, and new routines needed to be established.

Also the closed source solution will move to Eddie, as soon as the EDM database has been ported to Linux.

3.4 Changes to the storage implementations

Both the open and the closed storage implementations have been updated continuously, especially to improve query performance.

The architecture described in the previous section, Figure 1 and Figure 2, is illustrated in the process diagram in Figure 3.

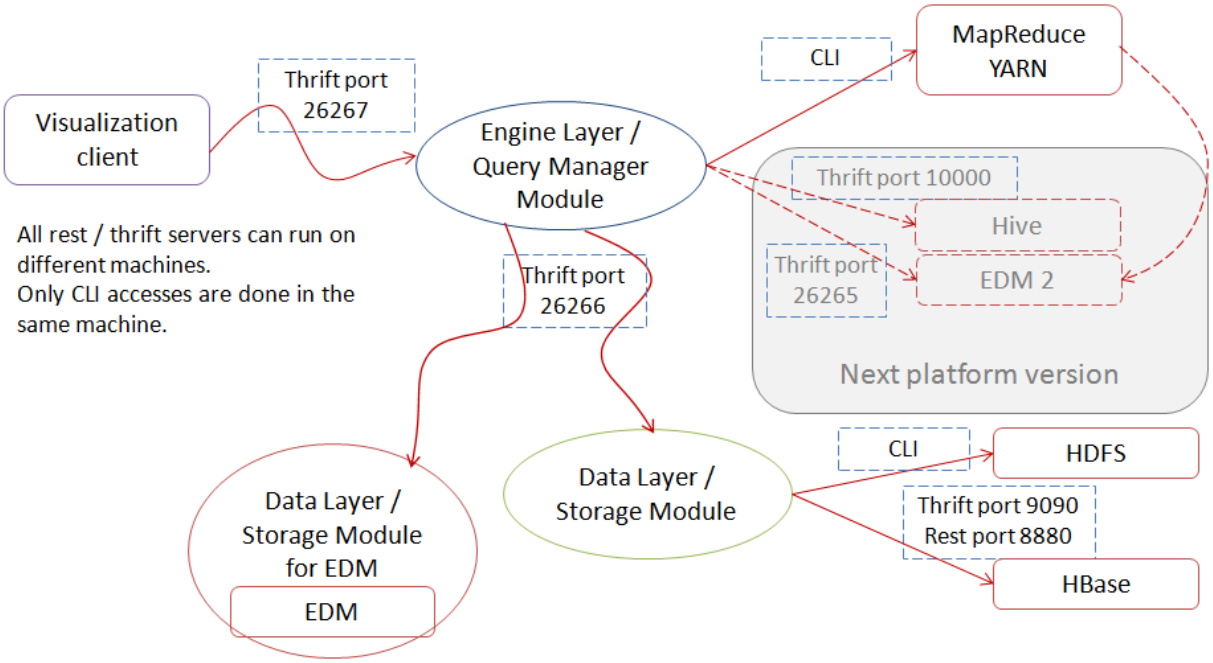

Figure 3: Process diagram of the VELaSSCo platform. Depending on the launched version platform, open or closed, the Query Manager Module will communicate with the Storage Module + HBase or with the Storage Module + EDM.

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.

According to the changes in the architecture, now the Query Manager will communicate with the Storage Module (Hbase) of the open platform or with the Storage Module (EDM) of the closed platform, but not with both at the same time. Now the Query Manager needs to be aware of the type of Storage Module it is connecting to. In the previous version of the architecture, the Query Manager did not need to be aware of the nature of the Storage Module.

3.4.1 Open source (HBase)

During the implementation process of 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. Several mechanisms have been implemented in the Query engine to support the different table definitions and the different table sets used by the partners of the consortium. These mechanism has been implemented both in the simple queries and the analytics queries, and are described in D3.3.

3.4.2 Closed source (EDM)

The “EDM database plug-in for the VELaSSCo platform” has, since D3.4, changed name to “EDM Storage Module for the VELaSSCo platform”. It is designed with a master process that communicates with the rest of the platform via the Thrift interface now described in "https://projects-ext.igd.fraunhofer.de/svn/VELASSCO/trunk/modules/Thrift/VELaSSCo/VELaSSCoSM.Thrift". The acronym SM in VELaSSCoSM indicates that it is a definition for the Storage Module. The interface definition is common for the Open and the Closed source version of the Storage Module.

The Jotne design of the approach for parallel execution in a HPC cluster is now complete, and the first prototype is running and can execute several Storage Module operations. A sketch of the design is depicted in Figure 4.

The EDM Storage Module is a process that receives operations from the Query Manager via the VELaSSCoSM Thrift interface. The incoming operations are there translated to EDM type queries that are executed in parallel on many EDM application servers. A new remote query, edmiRemoteExecuteCppMethod was developed. It makes it possible for client programs, in this case the EDM Storage Module, to execute C++ methods on EDM application servers. These methods are implemented and deployed in .dll modules.

The parallel processing approach is also newly developed; it is based on the EDMcluster concept. It allows the EDM Storage Module (ESM) to host information about all EDM databases and database servers of the VELaSSCo platform in a local database on the machine(s) where the ESM executes; see Figure 4: Architecture of the EDM Storage Module.

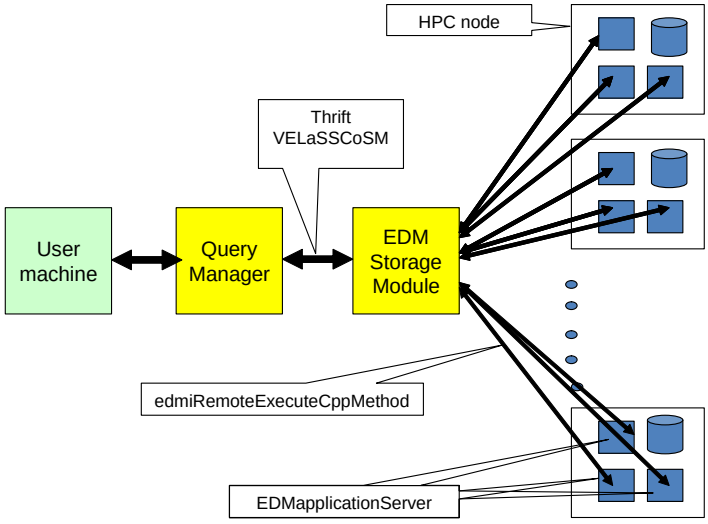

Figure 4: Architecture of the EDM Storage Module

The database schema of the EDMcluster is shown in Figure 5, below. The central concept is the ClusterModel. It represents a VELaSSCo model and is divided into many separate EDM datasets (data models).

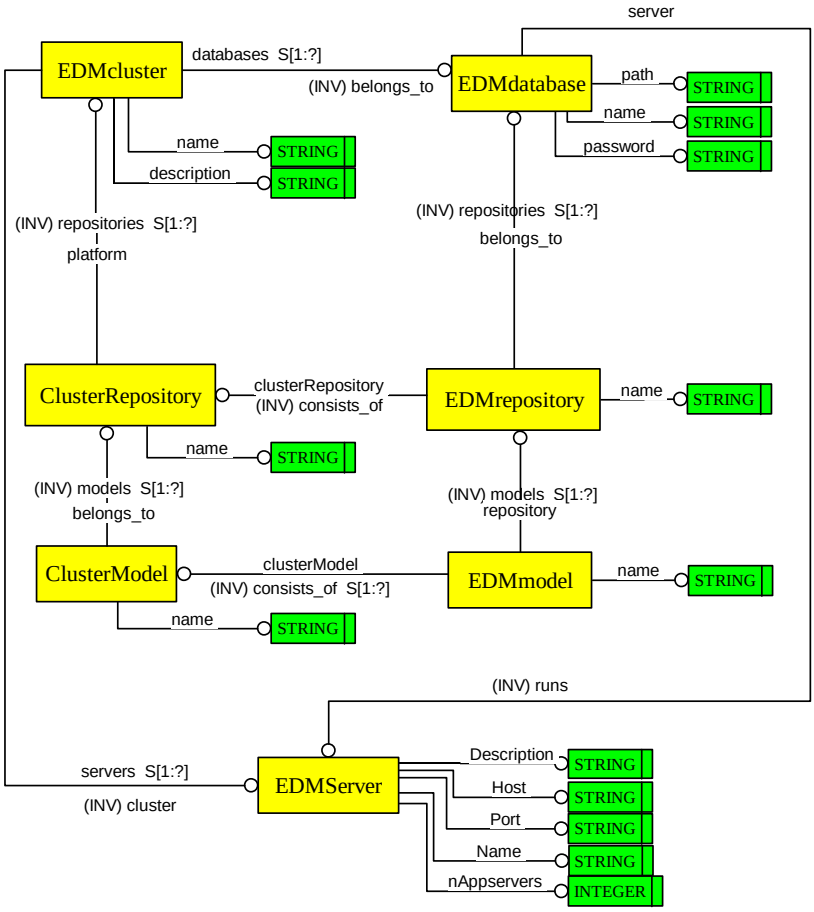

Figure 5: EDMcluster schema

A Storage Module operation is executed by ESM in a stepwise algorithm. If we, for example, look at the operation `GetListOfVerticesFromMesh`, it has the following steps:

1. The input parameter ModelID specifies which ClusterModel shall be queried. It is checked that it is a legal ClusterModel identifier and that the user has access to it.

2. ESM will, for all the physical EDMmodels that the VELaSSCo/ClusterModel consists of, create a set of input parameters to the C++ method `GetListOfVerticesFromMesh` that will be executed on EDM application servers with access to the data models. ESM uses OpenMP, an open standard for concurrent programming, to parallelize the execution on the different EDM application servers. ESM creates one local thread for each EDM application server. If, for example, the execution of `GetListOfVerticesFromMesh` can utilize 20 EDM application servers, ESM will create 20 working threads. Below you can see the loop in `GetListOfVerticesFromMesh` where the execution of sub-queries starts in parallel.

```
#pragma omp parallel for
for (int i = 0; i < nOfSubmodels; i++) {
   EDMexecution *e = subQueries->getElementp(i);
   // e contains information about the execution of one subquery
   // create return value object for this subquery, retVal
   nodeRvGetListOfVerticesFromMesh *retVal =
          new(e->ema)nodeRvGetListOfVerticesFromMesh(e->ema, NULL);
   e->returnValues = retVal;
   ExecuteRemoteCppMethod(e, "GetListOfVerticesFromMesh", inParams, &errorFound);
}
```

3. As mentioned before, a new function for executing queries/methods on the application server, edmiRemoteExecuteCppMethod has been developed. It includes a rich set of methods to transfer input parameters and return values between the client program and the server side plug-in. To be accepted as an EDM application server plug-in it must obey certain rules. The two most important ones are that there must exist

> 1) a method named dll_main that for a specified method name starts the correct method and
>
> 2) a method named dll_alloc that the system executes to get memory for input parameters.

In addition there must be a dll_free and dll_version method.

Our example, `GetListOfVerticesFromMesh`, and the other Vqueries are implemented in C++, the plug-in concept described above and the C++ Express API. The C++ Express API is an API where the EXPRESS-based objects in the database can be handled by generated methods; it is an early binding API. The API has also a container concept that is designed to handle huge amounts of data effectively in a concurrent programming environment.

Below you can see the most central part of the `GetListOfVerticesFromMesh` method: The execution starts by finding the specified mesh object in the database. The code below initially declares a container for the returned vertices. See the comments for explanation of the rest of the code.

```
Container<EDMVD::Vertex> vertices(dllMa, 0x10000);

Iterator<fem::Node*, fem::entityType> nodeIter(thMesh->getNodes(), theModel);
// Loop through the nodes of the mesh and create one record pr. node
for (fem::Node *n = nodeIter.first(); n; n = nodeIter.next()) {
   EDMVD::Vertex *v = vertices.createNext();  // create a new record in the container
   // and then move the node attributes to the record
   v->id = n->get_id(); v->x = n->get_x(); v->y = n->get_y(); v->z = n->get_z();
}
// then link the vertices container to the return value attribute vertices
retVal->vertices->putContainer(&vertices);
// the vertices container will be automatically transferred back to the client program
```

4. When all queries on the HPC nodes are finished, ESM will build the result data for return to the Query Manager via the Thrift channel in single thread mode. In the case of `GetListOfVerticesFromMesh`, duplicated vertices are filtered out before storing the data in the result set. We think it is possible to improve the performance of this step by using OpenMP, now with the number of threads equal to the number of cores in the machine.

Currently the following Storage Module operations are implemented in the closed source database:

1. UserLogin
2. GetListOfModelNames
3. UserLogout
4. GetResultFromVerticesID
5. GetCoordinatesAndElementsFromMesh
6. GetElementOfPointsInSpace
7. GetBoundaryOfLocalMesh
8. GetListOfModelNames
9. FindModel
10. GetListOfAnalyses
11. GetListOfTimeSteps
12. GetListOfResultsFromTimeStepAndAnalysis
13. GetListOfVerticesFromMesh
14. GetListOfMeshes

For operations that have one version for FEM data and another one for DEM data, only the FEM version has been implemented.

Ingestion of FEM and DEM data is still done by C++ programs that read text files and store the data in the database. With the new edmiRemoteExecuteCppMethod it will be possible to move the ingestion program to an EDM application server plug-in. This will result in highly parallel data ingestion methods. An even more interesting option is to offer a FEM/DEM/AP209 ingestion library where the solvers can send data directly to the visualization database.

4 Lessons learnt

The first prototype has achieved its key functionality. Thus, the VELaSSCo platform is a success. However, the project has encountered more performance issues than expected.

Applying Big Data tools to scientific simulations data, analytics and graphics requirements is not as straightforward as initially planned. Setting up the VELaSSCo platform required constant review of implementation aspects, such as, architecture, data structures and communication, adaptation and usage of standard big-data frameworks, programming and deployment. Experts from both fields, Big Data and engineering, needed and will continuously need to work closely together to harvest the benefits of Big Data technology. We have learnt a lot from each other already.

The VELaSSCo platform is a complex system that consists of many modules and accordingly many interfaces. Many of these are not under the control of the partners, but have been developed by external communities. The first prototype has shown that the hand-over of data among these modules requires careful design, measurements and testing; example: binary bulk data vs. complex data structures. This is especially true for a system that will handle large amounts of data.

Big data requires query functions that are specifically designed for a task; example: special paths in GetResultsFromVerticesID. Generic algorithms will mostly not be performing fast enough.

Some modules in the architecture may not be fit for purpose. Replacements will need to be considered; example: use Spark rather than YARN.

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. 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 . 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. 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. 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. 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. 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. 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. 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. 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] VELaSSCo D3.2 – Pre-computed, or on-demand computed, transformations stored in HPC, September 2015

[2] VELaSSCo 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, April 2016

[3] VELaSSCo 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, 2015-09-30