A software infrastructure for big data analytics

Sandro Fiore[1,2*], Ian Foster[3], Dean N. Williams[4], Giovanni Aloisio[1,2]
[1]Centro Euro-Mediterraneo sui Cambiamenti Climatici, Italy
[2]Università del Salento, Lecce, Italy
[3]Computation Institute, Argonne National Lab and University of Chicago, Chicago, IL 60637, USA
[4]Lawrence Livermore National Laboratory, Livermore, CA, USA
[*]sandro.fiore@unisalento.it

1. Introduction

The Ophidia project [1] is a research effort facing big data analytics challenges in multiple scientific domains like climate, bioinformatics, astrophysics, etc. It aims at addressing scientific use cases related to the analysis and mining of large volumes of multidimensional data [2-4].

In this work we discuss the software architecture stack designed and implemented in Ophidia for the management of scientific datasets on High Performance Computing machines taking into account the key challenges towards scientific big data analytics at exascale [5,6] presented at the BDEC Charlestone meeting in [7]. The software infrastructure is discussed in detail, from level-0 (the datacube store) to level-3 (the front end).

2. The Ophidia software infrastructure

In terms of software infrastructure, Ophidia provides a layered structure consisting of the *datacube store*, the *analytics framework*, the *analytics workflow management* and, finally the *front-end*. All of these layers expose a set of interfaces (API).

The *datacube store* (level-0) provides the API to perform I/O tasks on the storage system. It exploits an internal storage model jointly with a hierarchical data organization to manage "datacubes" (the abstraction used by the Ophidia system to manage n-dimensional data) [8]. To address scalability and performance, from a physical point of view, a datacube is associated to multiple units called *fragments* (implemented as a key-value store), which are distributed across multiple I/O servers. Depending on the target storage device a fragment can be stored in a relational table, a file, an object or a memory structure. For each storage device a specific plugin of the datacube store API is implemented. As a direct consequence, the platform is able to manage at the same time multiple datacubes stored on heterogeneous storage devices and with different I/O servers (RDBMS, object stores, in-memory servers, file servers, etc.). The current release of Ophidia exploits a well-known RDBMS as I/O server (MySQL); yet, an in-memory I/O server (able to store, query and process in-memory fragments) is being tested and will be soon released. This means the system is able to manage both persistent and transient data cubes. By loading in-memory an entire datacube, Ophidia allows complex data-driven workflows to be entirely executed in memory, drastically reducing the time needed for the analysis. Along with a *load* interface, a *store* one is needed as well, to move an in-memory datacube to a persistent storage device (for instance at the end of an in-memory workflow). As mentioned before, the datacube store exploits the datacube abstraction to represent multidimensional datasets in the system. This implies the need to provide file-system like primitives to get the list of existing datacubes, their size, ownership, etc. (system metadata). In such a context, each datacube is identified by a Digital Object Identifier (DOI), which represents a persistent reference for the datacube. Yet, due to the key role of metadata in the scientific contexts, the datacube store must provide a native support for scientific metadata management (for instance in terms of provenance, facet-based search, etc.). All of these features extend the file-system concept, moving the datacube store toward an *n-dimensional (facet-based) objects space*.

The *analytics framework* (level-1) is key in the entire software stack. It is responsible for atomically processing and manipulating data cubes, by providing a common way to run distributive tasks (operators) on large set of fragments. At this level the system provides a set of datacube oriented operators to perform sub-setting (slicing/dicing), time series analysis, data reduction (e.g. by aggregation), rollup/drill-down, merge, split, etc. All of the operators are plugins of the analytics framework, which means they have to comply with a well-known set of interfaces. To address flexibility and support both complex and simple operators, three interfaces have been defined as mandatory and four ones as non-mandatory. The API reflects the following general tasks:

- *setup/clean-up*, to initialize and clean-up the environment;
- *init/destroy*, to setup/finalise the operator execution;
- *distribution/reduction*, to perform a data distribution before and a data reduction after, the execution of an operator;
- *execution*, to run the core part of the operator.

Each Ophidia operator (i) is a dynamic library of the framework, (ii) implements (some/all of) the seven interfaces, (iii) is dynamically loaded at runtime by the framework to perform a specific data analytics task, (iv) can be sequential, parallel or distributed, (v) can be *data*- *metadata*- or *system*- oriented depending on its main focus, (vi) does not manage any general information related to monitoring, logging and bookkeeping.

Some operators apply *array-based* data analysis through a set of *primitives*. So far, about 80 primitives have been implemented to perform data sub-setting, data aggregation (i.e. max, min, avg), array concatenation, algebraic expressions, predicates evaluation, etc. Integer, float, double and complex data types are supported. Multiple plugins can be also nested to implement more complex tasks. *Compression* routines (in particular zlib, xz, lzo) are also available as array-based primitives.

The *analytics workflow management* layer (level-2) provides the proper interface to manage workflows in Ophidia. This layer is responsible for scheduling, submitting (on the *analytics framework*) and monitoring the workflows submitted by the users through the Ophidia front-end. It properly takes into account data dependencies and priorities. By applying a well-defined set of algebraic equivalence transformation rules it also optimizes the workflow execution. A workflow can be submitted as a query using the Ophidia query language (a mix of procedural and declarative statements).

A key feature offered by the layer is the opportunity to define *massive jobs*. Basically the same task (or workflow) can be applied to a set of datacubes as a whole. This can be achieved by providing a single job jointly with an input argument defining a search & discovery query rather than a single datacube. As a result, a set of independent jobs are submitted in parallel to the analytics framework taking as input all the datacubes matching the filter criteria. The massive jobs functionality is being used at CMCC to derive products from raw datasets. Such a feature is very powerful as it allows through a well-defined search & discovery query to enable thousands of analytics tasks on hundreds of terabytes of data at once.

Finally, the Ophidia *front-end* (level-3) provides multiple interfaces to submit analytics workflows. The first one that has been implemented is WS-I compliant. In this regard, several clients are already available (a CLI to submit a single job, a terminal-like environment to manage more complex user sessions and a powerful web-based application embedding all the framework capabilities). However, the WS-I interface is not the only one currently available. Indeed, a GSI/VOMS enabled interface is going to be finalized, in order to make the system fully interoperable both with Globus and gLite based grid environments (CMCC is working on defining a EMI-based component to be tested in the European Grid Initiative context). Moreover, to support geosciences use cases, an OGC-Web Processing Service (WPS) interface is also part of the future roadmap.

3. Conclusions

In this work a general overview about the Ophidia software infrastructure (from level-0, the datacube store, to level-3, the front end) has been presented. The system is currently being tested at the Euro-Mediterranean Center on Climate Change (CMCC), on CMIP5 [9] data in NetCDF format, Climate and Forecast (CF) convention compliant. The software stack has been deployed at CMCC on 24-nodes (16-cores/node) of the Athena HPC cluster. Preliminary experimental results have been published in [10].

References

[1]S. Fiore, A. D'Anca, C. Palazzo, I. Foster, D. N. Williams, G. Aloisio, "Ophidia: toward big data analytics for eScience", Proceedings of the International Conference on Computational Science (ICCS) 2013, Procedia Elsevier, Barcelona, June 5-7, 2013, pp. 567-576.

[2] S. Fiore and G. Aloisio, Special section: Data management for eScience. *Future Generation Computer System* 27(3): 290-291 (2011).

[3] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber. 2005. Scientific data management in the coming decade. *SIGMOD Rec.* 34, 4 (December 2005), 34-41. http://doi.acm.org/10.1145/1107499.1107503

[4] J. Chen, A. Choudhary, S. Feldman, B. Hendrickson, C.R. Johnson, R. Mount, V. Sarkar, V. White, D. Williams. "Synergistic Challenges in Data- Intensive Science and Exascale Computing," DOE ASCAC Data Subcommittee Report, Department of Energy Office of Science, March, 2013.

[5] J. Dongarra, P. Beckman, et al., The International Exascale Software Project roadmap. International *J. High Performance Computing Apps.* 25, no. 1, 3-60 (2011), ISSN 1094-3420 doi: 10.1177/1094342010391989.

[6] G. Aloisio and S. Fiore, Towards exascale distributed data management, International *J. of High Performance Computing Apps.* 23, no. 4, 398-400 (2009) doi: 10.1177/1094342009347702.

[7] G. Aloisio, S. Fiore, Ian Foster, D. Williams , "Scientific big data analytics challenges at large scale", Big Data and Extreme-scale Computing (BDEC), April 30 to May 01, Charleston, South Carolina, USA (position paper).

[8] J. Han and M. Kamber, *Data mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2005.

[9] K.E. Taylor, R. J. Stouffer, and G. A. Meehl 2012 An overview of CMIP5 and the experiment design, Bulletin of the American Meteorological Society 93, no. 4, 485-498 (2012), doi:10.1175/BAMS-D-11-00094.1, http://journals.ametsoc.org/doi/abs/10.1175/BAMS-D-11-00094.1.

[10] S. Fiore, C. Palazzo, A. D'Anca, I. Foster, D. N. Williams, G. Aloisio, "A big data analytics framework for scientific data management", Workshop on "Big Data and Science: Infrastructure and Services", IEEE International Conference on BigData 2013, October 6-9, 2013, Santa Clara, USA, pp.1-8.