ISSN 2347 - 3983
Volume 7, No.7 July 2019
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter01772019.pdf
https://doi.org/10.30534/ijeter/2019/01772019

Meta-modeling of Big Data management layer

Allae Erraissi[1], Abdessamad Belangour[2]
[1,2]Laboratory of Information Technology and Modeling, Hassan II University, Faculty of sciences Ben M'Sik, Casablanca, Morocco, erraissi.allae@gmail.com

ABSTRACT

Nowadays, the magnitude of data generated daily through the technological environment has increased enormously. This massive amount of heterogeneous data led to the emergence of a large number of big data systems and technologies that share similar architectures but with different implementations. In essence, the common architecture is composed of many components: Data sources, Ingestion, Hadoop Storage, Hadoop Platform management, Visualization, Monitoring, and Security Layers. In our way for a unified abstract implementation, we proposed, in previous works, meta-models for data sources, ingestion, storage, and visualization layers. We also relied on our previous comparatives studies to define key concepts of management layer in Big Data. Thus, we shall present in this paper our meta-model for management layer in Big Data by applying techniques related to Model Driven Engineering. The main goal of this universal meta-modeling is to enable Big Data distribution providers to offer standard and unified solutions for a Big Data system.

Key words: Meta-model, Model Driven Engineering, Big Data, Management layer, MapReduce, Zookeeper, Hive, Pig.

1. INTRODUCTION

As evident, huge volume of data are generated daily by all kind of devices, programs, and Social networks. The data includes social media sites, email, communication, blogs, and videos and so on. The rapid increase in the amount of published information or data establishes a new dimension called Big Data and, hence, raises huge issues. For efficient and retrieval data management, Database engines, based on SQL standard, were created in the 1970s [8]. They can perform well only when they process small amounts of relational data. Therefore, these tools remain very limited in the face of data expansion in volume and complexity. Similarly, Massively Parallel Processing 'MPP' [2], initially created in the early 1980s, improved slightly performance indicators for complex data volumes. However, this processing could not be used for processing non-relational data with permanent expansion. Powerful tools are required to store and exploit this daily expanding data in order to provide simple and reliable processing of the data collected from users. Traditional modeling operators face their limitations in this challenge, as information multiplies in volume and complexity, something that currently can only be managed by non-relational modeling techniques. In fact, Hadoop MapReduce [3] is the most efficient and reliable processing technique, compared to SQL databases and MPP processing. It has many characteristics that differentiate it from the other data processing system. For instance, Hadoop has a performance proportional to the complexity of large data. It is not only an effective tool for solving massive data problems but also it is a concept that has changed the organization of large-scale processing systems. However, despite its success, this model has not yet reached its final appearance as a mature computer solution. On the contrary, it is a starting point for other perspectives. Hadoop can support other types of distributed programming paradigms, whose tasks will be deployed relying always on the MapReduce resource manager and application manager. Ultimately, the ecosystem of Hadoop is very rich. For example, there are notably higher level applications to process data in a formalism close to SQL using HIVE [4] (as in a relational database), and tools for importing external data into Hadoop distributed file system or exporting Hadoop data to the outside, ie Sqoop [30], Pig [29], etc. At this point, we deem it necessary to point out that in our earlier studies we have identified key concepts of Hadoop management through comparative studies of major Big Data distributions [5]. Our aim there was to outline the weakness and the strength of each distribution and thus collect the appropriate raw material necessary for the study. Yet, the following work is a progress report of our previous proposals for meta-models for layers: Data Sources, Ingestion [6], Storage [7,35], and Visualization [36]. It is also an extended version of our work that has already been published in the Proceeding of a Conference [1]. In this article, we shall propose a universal meta-model for Big Data management layer by applying techniques related to Model Driven Engineering 'MDE' [9]. These meta-models together with the previous ones, which we have proposed for the other layers, can be used as an independent cross-platform Domain Specific Language.

2. RELATED WORK

Big data is an important subject that has recently attracted the interest of academics in the field of information technology. It is extremely important since it can help to identify and extract valuable and relevant knowledge, which is a vital factor for the success and prosperity of companies and industries. Indeed, many researchers have worked on big data, particularly on its value chain and on its processing tools. To start with, Mohammed, Humbe, and Chowhan (2016) [10] aimed to provide an overview of the big data context. They

defined the big data value chain in 5V. These authors focused on a classification based on five categories: data stores, content format, data sources, data processing and the staging of the data. Siddiqa and al. (2016) [11] focused on large data management by studying big data management techniques in storage, preprocessing, processing and security [37]. They also presented future directions for research in this field such as data integration and governance. As for Philip Chen and Zhang (2014) [12], they gave a brief overview of the problems of big data such as data mining, volume, variety, and velocity management in different areas including trade, administrations, and scientific research. They were also interested in the opportunities and challenges of storage, transmission, analysis, and visualization related to big data through current techniques and technologies. Big data and its processing tools are developing rapidly, reaching and affecting more and more domains. Skourletopoulos and al. (2017) [13] and Hashem (2015) [14] explored the opportunities, challenges, and techniques of big data and their relationship with cloud technologies, such as Big Data-As-A-Service. They identified some data analysis challenges such as scalability, availability, data integrity, and data transformation. As one of the main subjects of big data is the ability to manage constraints in real time, Liu, Iftikhar and Xie (2014) [15], Zheng and al. (2015) [16], Mohamed and Al-Jaroodi (2014) [17] addressed the topic by providing an overview of the current state of the art tools for dealing with big data. Liu, Iftikhar, and Xie (2014) [18] presented an analysis of open source real-time processing technologies with a focus on real-time architectures. Zheng and al. (2015) [16] discussed the challenges of big data and in particular those of real-time processing. They also presented a multilevel storage model and some deployment methods to meet the requirements of big data in real time and in heterogeneity. Mohamed and Al-Jaroodi (2014) [17] presented some technical challenges to real-time applications in big data. Besides, they provided a performance analysis and some big data requirements in real time. Other researchers have focused on the comparison of big data processing tools. Indeed, Lopez, Lobato, and Duarte (2016) [19] described, analyzed and compared three main open source distributed flow processing platforms such as Spark, Flink, and Storm. They provided experimental performance results focusing on throughput and parallelism in a threat detection application in network traffic. Urmila (2016) [20] introduced and compared Hive, Pig, and MapReduce for big data analysis. The comparison is based on the type of language: the user interface, the available algorithms and the scale of data supported in each tool.

According to this research and the research studies we have done on the Big Data world, we found that big data system contains several tools that allow the analysis and processing of massive data. This jungle of tools is grouped in several distributions depending on the provider of the solution (HortonWorks, Cloudera, Pivotal, etc.) [5]. Hence the need to standardize concepts through the application of techniques related to Model-driven Engineering 'MDE' become an urgent factor to efficiently manage a substantial amount of data. Correspondingly, in our previous works [5], we gave a detailed comparison of the top five big data solution providers. These comparatives studies along with the evaluation made by Forrester Wave [21] on the same Hadoop distributions helped us to define the key concepts of Management layer in a Big Data system. We also rely on two other comparative studies made by Robert D. Schneider [22] and V. Starostenkov [23] on the three HortonWorks, Cloudera, and MapR distributions.

In earlier works, we proposed meta-modeling of the layers of a Big Data system, namely: Data Sources, Ingestion [6], Storage [35], and visualization [36]. Yet, we shall propose in this paper a universal meta-modeling of Big Data Management layer. The main goal of this universal meta-modeling is to enable Big data distribution providers to offer standard and unified solutions for a Big Data system.

3. MAPREDUCE

3.1 MapReduce phases

MapReduce [3] refers to both the programming model and the framework originally developed by Google [25] for large-scale parallel data processing. Users only need to provide two functions, called map and reduce, and the system handles all the other issues related to parallelization, fault tolerance, data distribution, and load balancing. The map and reduce functions are set to key-value pairs. The map function consumes key-value input pairs and produces a (possibly empty) list of intermediate key-value pairs. Then, the framework groups the intermediate pairs by key and delivers each group (the key and all associated values) to the reduce function, which in turn produces a (possibly empty) list of key-value output pairs. The MapReduce framework runs parallel programs in a shared-nothing cluster. There are two types of processes, the workers, who execute the map and reduce tasks, and the master, which is in charge of controlling the execution of the workers. Typically, input and output data is stored in a distributed file system, for example, Google File System [26], which runs in the same nodes where MapReduce jobs are run. In a MapReduce job, the input is divided into M parts (splits), which are consumed by M map tasks, one per part. The output of the map tasks is partitioned according to the intermediate key in R fragments using a partitioning function, by default (hash (k2) mod R), which are then processed by R reduce tasks.

Figure 1 summarizes the execution steps of a Map/Reduce program that consist of:

- The Map Phase: Each Mapper (the node that executes the Map function) works on one or more pieces of the initial data that are in its node. According to the processing described by the Map function, Mappers produce results in pairs (key, value).
- The pairs (key, value) produced by the different Mappers are grouped and sorted according to their keys. Then, they are directed to the different Reduce

nodes so that all pairs that have the same key will be in the same Reduce node.

- Each Reducer processes the values associated with each key at a time.

This processing is fixed by the Reduce function written by the programmer.

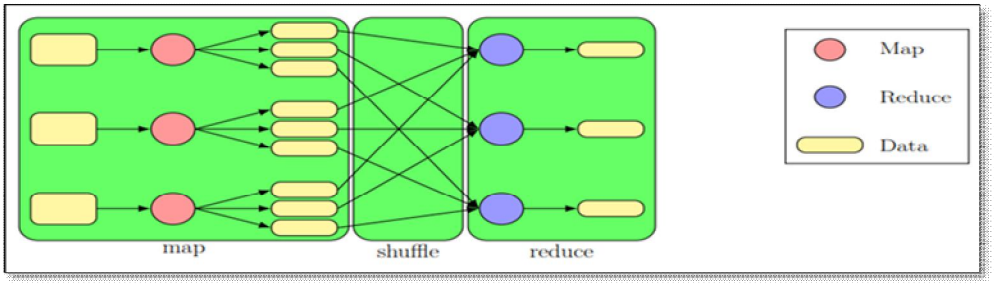

Figure 1: Typical structure of a MapReduce application [27].

3.2 Meta-model of MapReduce

This figure shows the meta-model that we proposed for the MapReduce process within Hadoop:

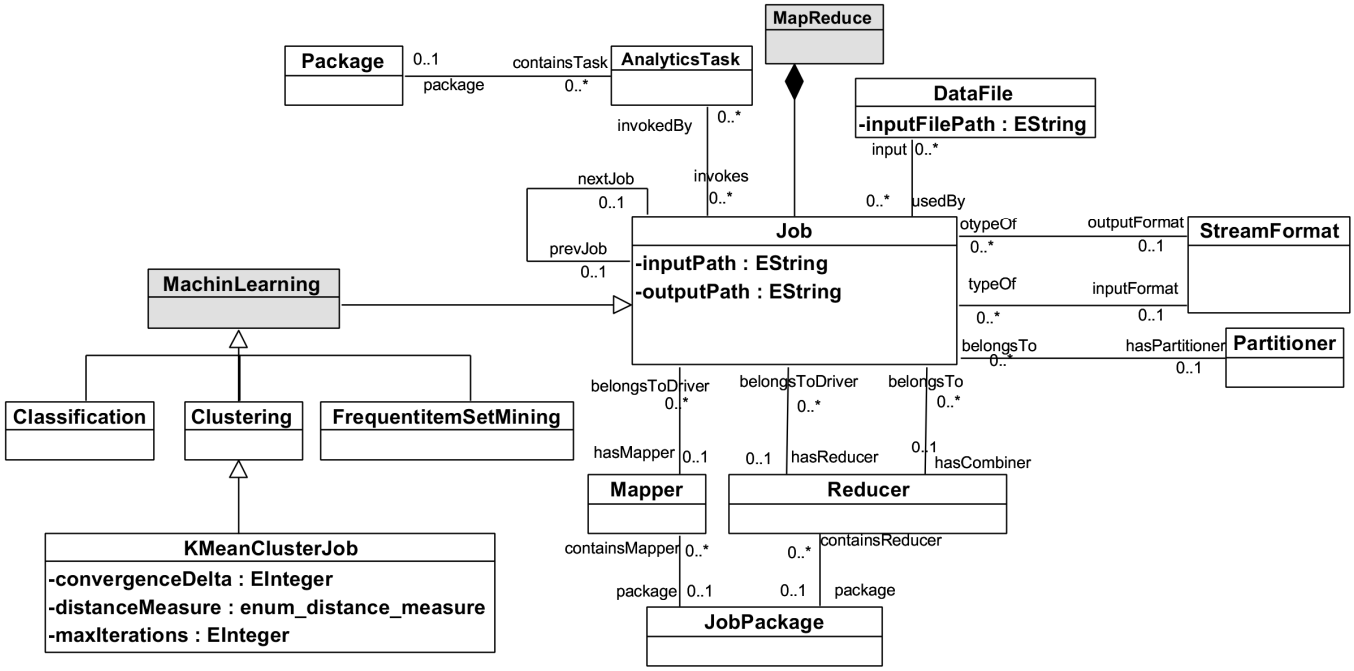

Figure 2: Meta-model of MapReduce.

Figure 2 presents the meta-model that we proposed to specify the program structure of the analysis applications at a higher level of abstraction. The AnalyticsTask meta-class presents the problem to be solved using the analysis. Typically, each scan task is divided into a set of small subtasks. These subtasks are modeled as Job. Each Job may need to refer to input data files. These input data files are modeled as DataFile with the file path. As a rule, several subtasks of a scan task must be executed sequentially. They are designed in a way that the output of a Job is used as input by a subsequent Job. This Job execution sequence is modeled with the nextJob association. Each Analysis Job uses a set of input files and generates output data in a set of output files. The path for input and output is specified using the properties of the Job inputPath and outputPath, respectively. The format of these input/output files may be different. For instance, data contained in files can be structured in vector form or as unstructured text, etc. StreamFormat specifies this format information. The default value for input/output files is the text. Each job has a mapper and may have a gearbox. Mapper transforms input records into intermediate records. It maps the key-value pairs into a set of intermediate key-value pairs. The Mapper class has many properties: keyInType, keyOutType, valueInType, and valueOutType. These properties specify the data types of the input key and the output key, the input data values, and the output data values, respectively. Combiners may optionally be used to reduce the amount of data transferred from Mapper to Reducer. They are

intermediate reducers and they are modeled with the hasCombiner association whose target is Reducer. The Reducer class has properties named keyOutType and valueOutType which to specify the data types of the output. Each Job MapReduce has a default partitioner. Hadoop MapReduce also provides a mechanism to have a user-defined partitioning class. This is modeled as a Partitioner. To take advantage of parallel data processing capabilities, machine learning [28] algorithms must be implemented in the MapReduce paradigm. The meta-model shows representative classes of algorithms. These inherit the structure of the Job class. At this stage, we deem it necessary to note that MapReduce has a direct relationship with the other components of Hadoop management layer like Hive [4], Pig [29], Sqoop [30], Zookeeper [31], etc. In the next parts, we will propose meta-models for these components.

4. ZOOKEEPER

4.1 Definition

ZooKeeper [31] is a Hadoop cluster management solution. This solution allows coordinating the tasks of the services of a Hadoop cluster and provides Hadoop components with distribution capabilities. Zookeeper is an open-source project that provides services like the presentation of configuration information and gives a distributed synchronization. Zookeeper has ephemeral nodes representing different Region Server [32]. Master servers use these nodes to discover the available servers. In addition to availability, nodes are also used to track server failures or network partitions. Zookeeper also offers a coordination service as the client finds and communicates with the Region Server via Zookeeper.

4.2 Meta-model of Zookeeper

The following figure shows the meta-model we proposed for the Zookeeper:

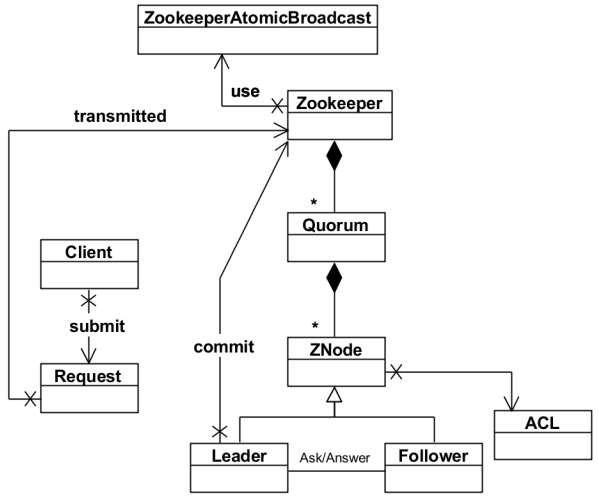

Figure 3: Meta-model of Zookeeper.

ZooKeeper uses several great ideas to ensure that node failure does not affect the system as a whole. In principle, it defines its own algorithm, the ZooKeeper Atomic Broadcast based on the algorithm of Paxos [33] that it improves. The Paxos algorithm guarantees a form of consensus, which is also the case at ZooKeeper.

The meta-model we proposed for ZooKeeper expresses the general behavior of ZooKeeper. We note that there are several major steps:

1. A customer submits a request.
2. A leader node is elected and takes care of the request. The election is based on a version number: the node with the highest version number is considered a leader.
3. The leader asks a set of followers (together called quorum) to help it to deal with the problem. Followers receive a request and respond to the leader. If the leader does not respond, a new one is chosen. In this way, we prevent the leader from becoming a SPOF (Single Point Of Failure).
4. The leader establishes a point-to-point connection with each element of the quorum and submits the request to them. There is an advantage in having a quorum available: if a quorum node does not respond, others can do it.
5. The quorum sends the result to the leader, which sends the end of processing (commit). The consensus is respected, the result is usable.

We note that the manipulation permissions of a ZNode are handled by the ACLs (AC for access control).

5. PIG

Pig [29] is a brick that allows the querying of Hadoop data from a scripting language (language that interprets the code line by line instead of making a compilation). Pig is based on a high-level language called PigLatin. It transforms data streams step by step either by running MapReduce programs in sequence or by using predefined methods such as averaging, minimum value, or by allowing the user to define his own methods called User Defined Functions (UDF).
Three steps in a typical Pig program:

- Loading: Loading HDFS data.
- Transformation: Translates the data to a list of Map and Reduce tasks, and the application of relational operators: FILTER, FOREACH, GROUP, UNION, etc.
- Unloading or Storage: Display the results on the screen or store them in a file.

The following figure shows the meta-model that we proposed for PIG:

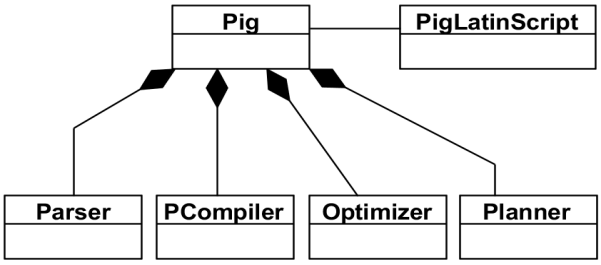

Figure 4: Meta-model of PIG.

6. HIVE

Hive [4] is a data query tool that allows the execution of SQL queries on the Hadoop cluster to analyze and aggregate data. The language used by Hive is named HiveQL. This is a visualization language only, thus wise only "Select" statements are supported for data manipulation. First, Facebook developed Hive, but Apache Software Foundation took it subsequently to make it Open Source under the name of Apache Hive [38]. The features of HIVE:

- Hive is at first familiar, fast, scalable, and extensible.
- It is not designated for transactional operations, but it can be said that it is closer to OLAP operations (OnLine Analysis Processing).
- Hive does not process data in real time,
- Its databases are not relational,
- Structure of the data in a well-known model: Tables, Columns, Lines, etc.

This figure presents the meta-model that we proposed for Hive:

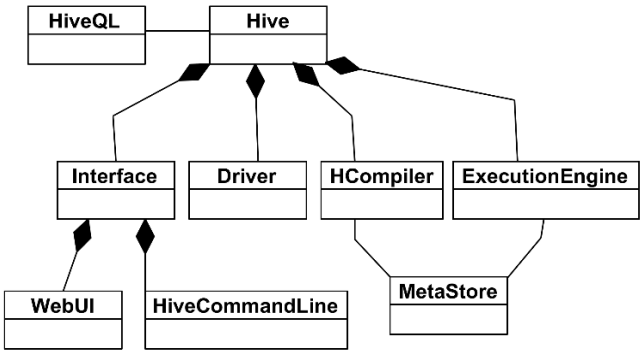

Figure 5: Meta-model of Hive.

Our meta-model shows that Hive is a software with a data warehouse infrastructure. It allows interactions between the user and the file system. The user interfaces that Hive supports are Hive WebUI, Hive command line, and Hive HD Insight (for Windows server). We represent them in our meta-model by the meta-class Interface, which consists of WebUI and HiveCommandLine. The MetaStore represents the database server chosen by Hive respectively to store table schemas, databases, table columns, their data types, and HDFS Mapping.

HiveQL is similar to SQL for querying schema information in the Meta Store. HiveQL Process Engine is one of the replacements for the traditional approach of the MapReduce program. Instead of writing the MapReduce program in Java, you can write a query for the MapReduce job and execute it. The ExecutionEngine meta-class represents the link between HiveQL Process Engine and MapReduce. It executes the queries and generates the same results as MapReduce.

7. SQOOP

Sqoop [30] is a brick for data integration. It allows the transfer of data between a cluster and a relational database. With Sqoop, the Apache Software Foundation connects Hadoop to databases and storage systems. As more and more companies use Hadoop to analyze large amounts of information, they realize that they may also need to transfer data between Hadoop and their existing databases, storage systems, and other databases. Volunteer developers are behind the development of a new connector to speed up these data exchanges. Consequently, they gained full support from the Apache Software Foundation (ASF). Indeed, the foundation, which supports the development of open source software, has promoted a tool, called Sqoop, which accelerates the transfer of data, to the rank of priority project. Figure 6 shows the usefulness of the Sqoop tool:

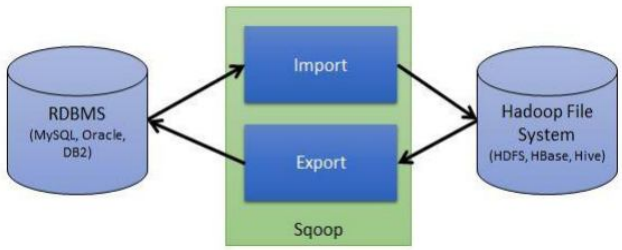

Figure 6: Sqoop utility [30].

The following figure 7 shows the meta-model that we proposed for Sqoop:

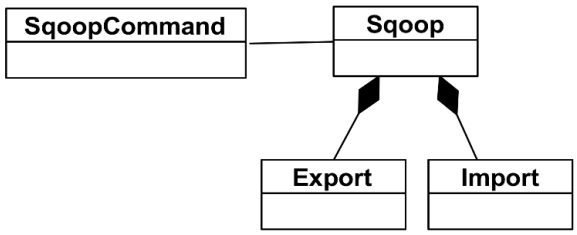

Figure 7: Meta-model of Sqoop.

Sqoop Import is the process of taking the data from a relational database management system and putting it in Hadoop, Sqoop Export is the process of taking the data from Hadoop and putting it back into a database management system relational. Sqoop can handle both these processes using the Sqoop Import and Sqoop Export functions.

8. GENERIC META-MODEL OF HADOOP MANAGEMENT LAYER

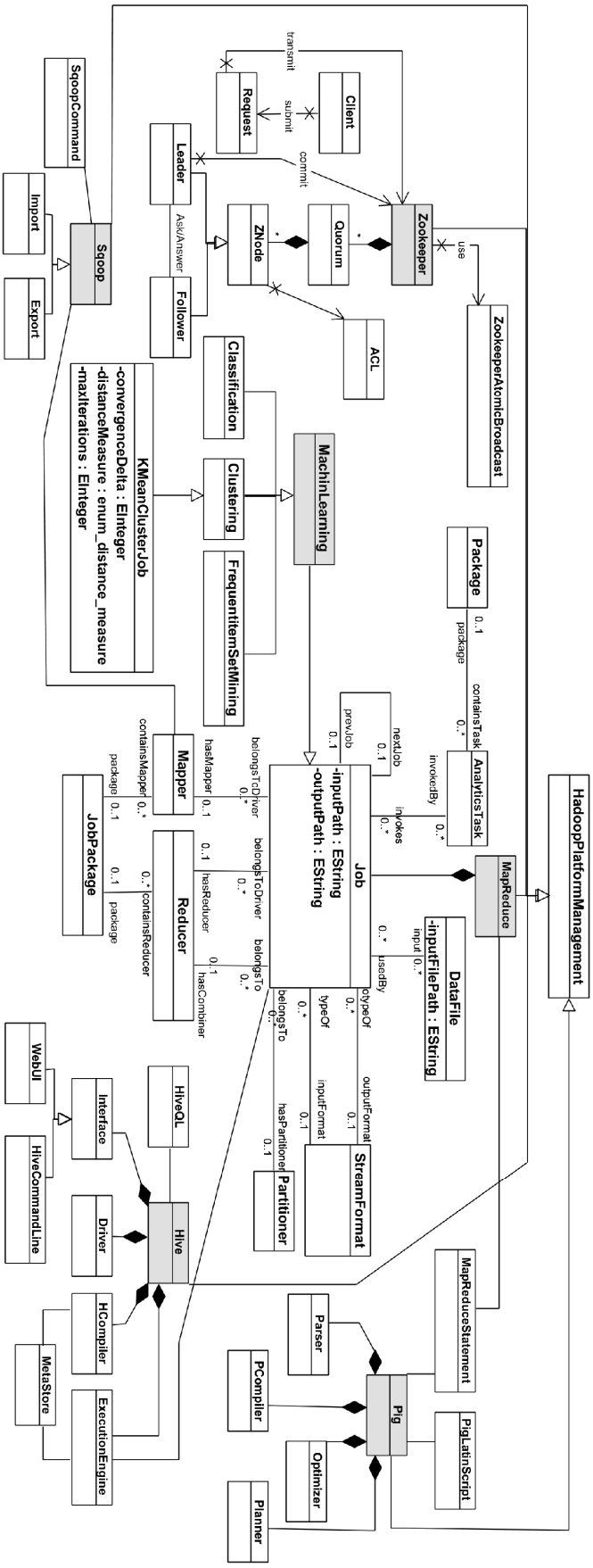

Figure 8: Generic meta-model of Hadoop management layer.

After the creation of meta-models for Hadoop management layer, our next step in our future studies will focus on the creation of models respecting these meta-models. Subsequently, we shall define the transformation rules between these meta-models using the transformation language ATL (Atlas Transformation Language) [34].

9. DISCUSSION AND PERSPECTIVES

In the age of the web giants, all our actions generate digital traces. We find a huge amount of data generated daily that is hugely related to personal life and that can be exploited in different areas. There is also internally generated data, called offline data, which is created by the operations of organizations. The processing of this massive data plays a key role in decision-making. Big data has become essential in today's world. Therefore, this work relies more particularly on our three research studies that we have already done in the world of Big Data and its different solutions. We have found that there are several distributions that can handle Big Data (HortonWorks, Pivotal HD, IBM Big Insights, etc.). Each distribution provider designs a solution in its own way without respecting standard references such as meta-models, the fact that caused the diversity of solutions and the non-interoperability between the different solutions. In our research project, we apply techniques related to the engineering of the models to propose a universal meta-modeling including all layers of the architecture of a Big Data system. After the creation of these meta-models, in the next step, we shall work on the creation of models respecting these meta-models. Then we shall define the transformation rules between these meta-models using the transformation language ATL (Atlas Transformation Language) [24]. These meta-models are platform independent according to Model Driven Architecture pattern [9], which describes the structures of Data Sources, Ingestion, and Hadoop Storage independently from any specific platform.

10. CONCLUSION

Many scientific fields are now facing a deluge of data. One of the approaches proposed to allow the processing of such volumes is the MapReduce programming paradigm introduced by Google, and the various other tools constitute the management layer in the overall architecture of a Big Data system. In this paper, we have studied this layer, which is very useful for the proper functioning of a Big Data system. We have also applied techniques related to Model Driven Engineering (MDE) to propose a universal meta-modeling for Big Data management layer. These meta-models together, which we have suggested for the other layers, can be used as an independent cross-platform Domain Specific Language.

REFERENCES

1. Erraissi, A., and A. Belangour. « **Meta-modeling of Zookeeper and MapReduce processing** ». In 2018 *International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, 1–5, 2018. https://doi.org/10.1109/ICECOCS.2018.8610630
2. Inmon, W. H., and Daniel Linstedt. « 2.1 - **A Brief History of Big Data** ». In Data Architecture: a Primer for the Data Scientist, édité par W. H. Inmon et Daniel Linstedt, 45-48. Boston: Morgan Kaufmann, 2015. https://doi.org/10.1016/B978-0-12-802044-9.00008-8.
3. Blokdyk, Gerardus. **MapReduce Complete Self-Assessment** Guide. CreateSpace Independent Publishing Platform, 2017.
4. Dayong Du. **Apache Hive Essentials: Essential techniques to help you process, and get unique insights from, big data,** 2nd Edition eBook: Dayong Du: Gateway.
5. Allae Erraissi, Abdessamad Belangour, and Abderrahim Tragha, "**Digging into Hadoop-based Big Data Architectures,**" *Int. J. Comput. Sci. Issues IJCSI*, vol. 14, no. 6, pp. 52–59, Nov. 2017.
6. Erraissi, A., & Belangour, A. (2018). **Data sources and ingestion big data layers: meta-modeling of key concepts and features.** *International Journal of Engineering & Technology*, 7(4), 3607-3612.
7. Erraissi A., Belangour A. (2019) **Capturing Hadoop Storage Big Data Layer Meta-Concepts.** In: Ezziyyani M. (eds) *Advanced Intelligent Systems for Sustainable Development (AI2SD'2018). AI2SD 2018. Advances in Intelligent Systems and Computing*, vol 915. Springer, Cham
8. Codd, Edgar F. (June 1970). "**A Relational Model of Data for Large Shared Data Banks**". *Communications of the ACM.* 13 (6): 377–87. CiteSeerX 10.1.1.88.646. doi:10.1145/362384.362685
9. Royer, Jean-Claude, and Hugo Arboleda. **Model-Driven and Software Product Line Engineering**. 1st Edition. London, UK : Hoboken, NJ, USA: Wiley-ISTE, 2012.
10. Mohammed, A. F., V. T. Humbe, and S. S. Chowhan. 2016. "**A Review of Big Data Environment and Its Related Technologies**." *In 2016 International Conference on Information Communication and Embedded Systems (ICICES)*, 1–5.
11. Siddiqa, Aisha, Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Mohsen Marjani, Shahabuddin Shamshirband, Abdullah Gani, and Fariza Nasaruddin. 2016. "**A Survey of Big Data Management: Taxonomy and State-of-the-Art.**" *Journal of Network and Computer Applications* 71 (August): 151–66.
12. Philip Chen, C. L., and Chun-Yang Zhang. 2014. "**Data-Intensive Applications, Challenges, Techniques and Technologies: A Survey on Big Data**." *Information Sciences* 275 (August): 314–47.

13. Skourletopoulos, Georgios, Constandinos X. Mavromoustakis, George Mastorakis, Jordi Mongay Batalla, Ciprian Dobre, Spyros Panagiotakis, and Evangelos Pallis. 2017. "**Big Data and Cloud Computing: A Survey of the State-of-the-Art and Research Challenges.**" *In Advances in Mobile Cloud Computing and Big Data in the 5G Era*, edited by Constandinos X. Mavromoustakis, George Mastorakis, and Ciprian Dobre, 23–41. Studies in Big Data 22. Springer International Publishing.
14. Hashem, Ibrahim Abaker Targio, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. 2015. "**The Rise of 'big Data' on Cloud Computing: Review and Open Research Issues.**" *Information Systems* 47 (January): 98–115. https://doi.org/10.1016/j.is.2014.07.006.
15. Liu, Xiufeng, Nadeem Iftikhar, and Xike Xie. 2014. "**Survey of Real-Time Processing Systems for Big Data**." In *Proceedings of the 18th International Database Engineering & Applications Symposium*, 356–361. *IDEAS* '14. New York, NY, USA: ACM. https://doi.org/10.1145/2628194.2628251.
16. Zheng, Z, P Wang, J Liu, and S Sun. 2015. "**Real-Time Big Data Processing Framework: Challenges and Solutions.**" *Applied Mathematics and Information Sciences* 9 (January): 3169–90.
17. Mohamed, N., and J. Al-Jaroodi. 2014. "**Real-Time Big Data Analytics: Applications and Challenges**." In 2014 *International Conference on High Performance Computing Simulation (HPCS)*, 305–10. https://doi.org/10.1109/HPCSim.2014.6903700.
18. Liu, Xiufeng, Nadeem Iftikhar, and Xike Xie. 2014. "**Survey of Real-Time Processing Systems for Big Data**." In *Proceedings of the 18th International Database Engineering & Applications Symposium*, 356–361. IDEAS '14. New York, NY, USA: ACM. https://doi.org/10.1145/2628194.2628251.
19. Lopez, M. A., A. G. P. Lobato, and O. C. M. B. Duarte. 2016. "**A Performance Comparison of Open-Source Stream Processing Platforms.**" *In 2016 IEEE Global Communications Conference (GLOBECOM)*, 1–6. https://doi.org/10.1109/GLOCOM.2016.7841533.
20. Urmila, R. 2016. "**Big Data Analysis: Comparision of Hadoop MapReduce, Pig and Hive Dr. Urmila R. Pol Assistant Professor, Department of Computer Science, Shivaji University, Kolhapur, India**" Vol. 5, Issue 6, June 2016 Copyright to IJIRSET.
21. Read, W., Report, T., & Takeaways, K. (2016). **The Forrester WaveTM: Big Data Hadoop Distributions**, Q1 2016.
22. R. D. Schneider, "**HADOOP BUYER'S GUIDE**," 2014.
23. V. Starostenkov, R. Senior, and D. Developer, "**Hadoop Distributions** ".
24. Mouad Banane and Belangour, A « **RDFMongo: A MongoDB Distributed and Scalable RDF Management System Based on Meta-Model** ». *International Journal of Advanced Trends in Computer Science and Engineering* 8, n° 3 (25 June 2019): 734-41. https://doi.org/10.30534/ijatcse/2019/62832019
25. J. Dean and S. Ghemawat, « **MapReduce: simplified data processing on large clusters** », in *6th Symposium on Operating System Design and Implementation*, 2004, pp. 137–150.
26. S. Ghemawat, H. Gobioff, and S.-T. Leung, « **The Google file system** », in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, vol. 37, 2003, pp. 29–43. https://doi.org/10.1145/1165389.945450
27. Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. **Mining of Massive Datasets**. 2 edition. Cambridge: Cambridge University Press, 2014. https://doi.org/10.1017/CBO9781139924801
28. Theobald, Oliver. **Machine Learning For Absolute Beginners: A Plain English Introduction**. Independently published, 2018.
29. Gates, Alan, and Daniel Dai. **Programming Pig: Dataflow Scripting with Hadoop**. 2 edition. O'Reilly Media, 2016.
30. Ting, Kathleen, and Jarek Jarcec Cecho. Apache Sqoop Cookbook: **Unlocking Hadoop for Your Relational Database.** 1 edition. Sebastopol, CA: O'Reilly Media, 2013.
31. Bagai, Chandan. **Characterizing & Improving the General Performance of Apache Zookeeper: Sub-Project of Apache Hadoop.** LAP AMBERT Academic Publishing, 2016.
32. Vohra, Deepak. **Apache HBase Primer.** 1st ed. edition. New York, NY: Apress, 2016. https://doi.org/10.1007/978-1-4842-2424-3
33. L. Lamport, "**Paxos Made Simple**," 2001.
34. "**ATL: Atlas Transformation Language Specification of the ATL Virtual Machine**."
35. Erraissi Allae, and Abdessamad Belangour. « **Hadoop Storage Big Data Layer: Meta-Modeling of Key Concepts and Features** ». *International Journal of Advanced Trends in Computer Science and Engineering* 8, n° 3 (2019): 646-53. https://doi.org/10.30534/ijatcse/2019/49832019
36. Erraissi Allae, and Abdessamad Belangour. « **Meta-Modeling of Big Data visualization layer using On-Line Analytical Processing (OLAP)** ». *International Journal of Advanced Trends in Computer Science and Engineering* 8, n° 4 (2019).
37. A, Aditya Rahman, and Gusman Dharma. « **Master Data Management Maturity Assessment : A Case Study of a Pasar Rebo Public Hospital** ». *International Journal of Emerging Trends in Engineering Research* 7, no 5 (5 June 2019): 15-20. https://doi.org/10.30534/ijeter/2019/02752019
38. Banane, M., & Belangour, A. (2019). **New Approach based on Model Driven Engineering for Processing Complex SPARQL Queries on Hive**. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 10(4). https://doi.org/10.14569/IJACSA.2019.0100474