International Journal of Engineering &Technology, 8 (3) (2019) 288-294

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

RDFSpark: a new solution for querying massive RDF data using spark

Mouad Banane [1] *, Abdessamad Belangour [1]

[1] *Laboratory of Information Technology & Modeling, Hassan 2 University, Morocco*
**Corresponding author E-mail: mouadbanane@gmail.com*

Abstract

The invasion of semantic data, the rapid growth of RDF data has brought significant new challenges in the querying of RDF data. On the other hand, Apache Spark is an open source distributed computing framework, characterized by its speed as MapReduce, Big Data processing has never been easier. In last years MapReduce solves problems at scales unimaginable a few years ago. But like any other tool, it remains limited. Several research works propose the querying of large volumes of RDF data using MapReduce as H2RDF, We find Spark which is more powerful and robust than MapReduce, and it is 100 times faster than MapReduce. In this paper, we present RDFSpark: a new distributed RDF query management based on Spark to ensure scalability, fault tolerance, and performance to solve low-efficiency problems for RDF data query. n this approach, a SPARQL query is analyzed first by the Parser, and compiled before being passed through a series of optimization techniques, then it will be translated to a Spark program. The results of experiments conducted on huge volumes of RDF data show that RDFSpark has a high querying performance.

Keywords: *RDF; Spark; SPARQL; MapReduce; Big Data.*

1. Introduction

Data storage systems in RDF (or triple store) have shown limitations in terms of data integrity (transaction management), performance (response time of some queries) and scalability (volumetry). Thus, among the three axes that traditionally define Big Data: speed, volume, and variety (the "3V"), the first two characteristics are not yet achieved by these technologies and if the decentralization of data, at the very heart of the Web of data, could constitute part of a solution, it is to forget the problematic of resilience of the network and the necessity of data aggregation to interrogate them.
The very structure of the RDF model has revealed limitations on the management of the provenance of the different information and the contextualization of the triple: and if this point was present in the Tim Berners-Lee semantic Web roadmap, it still not resolved. Solutions have appeared but they are not entirely satisfactory. From this point of view, RDF 1.1 is a missed appointment, especially since at the same time the model of "property graph" that proposes a response to this limit has begun to prevail ... This model is today at the heart of all graph database technologies proposed by the major players in the sector: IBM, Microsoft, Google, not counting the newcomers: Datastax, Neo4j [1] or OrientDB. Thus, the graph model is doing well and, for good reason, it offers unparalleled flexibility in the manipulation of structured data and cross-referencing of heterogeneous data.

2. Related work

Many research efforts have been devoted to developing a large RDF data management system such as Jena-HBase [2], it is an evolutionary RDF based triplestore based on HBase[3] database storage which a basic management system column-oriented NoSQL data, Jena-HBase uses Jena for querying RDF data, another system also based on HBase is H2RDF [4] which is a fully distributed RDF data store, H2RDF combines the MapReduce Framework with a store distributed data NoSQL, so for storage this system uses HBase and for the processing of these data it uses MapReduce. we also mention that the H2RDF system has two features that allow efficient processing of simple SPARQL queries and also multiple joins on an unlimited number of triple RDF's through joining algorithms that perform joins based on the selectivity of the query for reduce the processing. Other researchers use for example the NoSQL database Cassandra [5] as: Ladwig and al. Providing with CumulusRDF [6] the nested key-values RDF data store and distributed as the underlying storage component for a linked data server, CumulusRDF provides functionality to process linked data through HTTP lookups. Or the authors of CumulusRDF have developed two index schemes for RDF to support both linked data retrieval and basic triple searching based on the model. The work [7] presents an overview of the RDF data management systems based on the NoSQL databases according to the different models: column-oriented, document-oriented, key-value oriented and graph-oriented. Cudre-Mauroux and al present [8] a comparison of NoSQL stores for RDF data processing. This work describes only four NoSQL stores that are: 4store, Jena-HBase, Hive-HBase, and CumulusRDF, this work also compares their key features through running standard RDF benchmark tests on a cloud infrastructure using two deployment modes. : On a single machine and distributed mode. A recent comparison [9] of RDF stores based on NoSQL, and re-

cently in [10] the authors propose RDF data storage technique, for efficient processing of SPARQL queries using two distributed computing engines Spark and Drill [11].
The outline of the paper is as follows: Section 2 exposes some existing related works in this topic. Section 3 describes Spark. Section 4 presents our main contribution. Section 5 evaluates and analyzes our approach with Lehigh University Benchmark. Finally. We conclude in Section 6.

3. Background

3.1. Big data

The explosion of digital traces in an ever more connected society has led to the development of big data or big data. These are opportunities but require overcoming many technical challenges. In addition, they mobilize methods which, in order to bring back useful information, bring together advanced technical skills in terms of digital data manipulation, computer programming and statistical calculation that go beyond the tools of many researchers. to the already old but still topical definition of big data around the notion of the 3V [12], a term referring to the English terms of volume, velocity and variety, which are all characteristics that would be endowed by big data. In this context, the volume (volume) underlines the mass of data particularly important, the speed (velocity) the fact that the process of data generation is very fast, often continuous, whereas the diversity (variety) refers to the sources and varied formats of the data collected, which is particularly indicative of their unattractive nature, not consistent with the direct application of analytical processing. To manage Big Data a new ecosystem has been developed named Hadoop [13].

3.2. Hadoo

(distributed file system) has become the technology systematically associated with the notion of data considered as massive because distributed. Largely developed by Google before becoming a project of the Apache Foundation, this technology meets the specific needs of data centers: to store considerable volumetrics by stacking thousands of low-cost computer cards and disks, rather than implement a supercomputer, while maintaining reliability by strong fault tolerance. The data is duplicated and, in case of failure, the processing is continued, a card replaced, the data automatically rebuilt, without having to shut down the system. This type of architecture generates an algorithmic cost. The nodes of these servers can only communicate in pairs (key, value) and the different stages of a process must be decomposed into basic functional steps like those called MapReduce [14]. For simple operations, for example counting words, URLs, basic statistics, this architecture is effective. A map step performs, in parallel, the counts at each node or executor (workers) of a computer cluster, the result is a set of pairs: a key (the word, the URL to be counted) associated with a result part. The identical keys are grouped in an intermediate sorting step (shuffle) within the same Reduce step which provides for each key the final result. In this architecture, the algorithms are said to be scalable scalable English if the execution time decreases linearly with the number of executors dedicated to the calculation. This is immediate for counting, averaging, this is not necessarily the case for complex iterative algorithms. The k-nearest neighbor method is not scalable, unlike non-supervised dynamic reallocation classification algorithms (e.g. Forgy, k-means) that operate by MapReduce step iterations. The example of Forgy's algorithm is very revealing.

- Initialization of the algorithm by definition of a function of distance and random designation of k centers.
- Until convergence:
- The Map step calculates, in parallel, the distances of each observation to the current k centers. Each observation (vector of values) is assigned to the nearest center (key). The pairs: (key or center number, vector values) are communicated to the Reduce step.
- An implicit intermediate step Shuffle addresses pairs of the same key at the same next step.
- For each key designating a group, the Reduce step calculates the new centroids, the average of the values of the variables of the individuals sharing the same class, that is to say, the same key value. The main problem of this implementation, the execution time saved by the parallelization of calculations is strongly penalized by the need to write and reread all the data between two iterations.

3.3. Spark

This is the main motivation behind the development of Spark [15] technology at the University of Berkeley. This layer of software over file management systems like Hadoop introduces the notion of the resilient distributed dataset (RDD) where each partition remains, if necessary, present in memory between two iterations to avoid rewriting and replay. This response well to the main constraints: massive data must not be moved and a result must be obtained by a single read operation. Technically, these RDDs are handled by commands in Java or Scala language but there are APIs (application programming interface) accepting commands in Python (PySpark) and R. Spark integrates many features divided into four modules: GRAPHX for the analysis of graphs or networks, streaming for processing and analysis of flows, SparkSQL for querying and managing databases of all types and the MLlib[16] library for the main learning algorithms. In full development, this environment includes (version 1.6) inconsistencies. SparkSQL generates and manages a new DataFrame data class (similar to R) but is not known to MLlib which will gradually be replaced by SparkML in future versions ... In the presentation that is made, Spark appears as a framework to connect most of the technologies developed as an Apache project. All types of files (JSON, CSV, RDDs ...) and structured or non-structured data types, all types of data flows (connected objects, tweets, e-mails ...) can thus be managed, aggregated by standard operations of selection, merge using commands that use SQL (structured query language) syntax before being used for modelizations.

3.4. NoSQL

With the advent of web 2.0 and social networks, besides the problem of data storage and management by RDBMS, the constraints have become increasingly important and evoked solutions for scaling up especially scale-out and clustering allowed the birth of a new family of data engine called: Not Only SQL or "NoSQL, to bypass the limits of relational databases. The NoSQL storage mode allows you to manage large amounts of data that can be unstructured in distributed storage servers that are capable of scaling up at a lower cost.

3.5. Semantic web

The Semantic Web (more technically called the "Data Web") allows machines to understand semantics, the meaning of information on the Web. It extends the hyperlink network between conventional web pages through a structured data link network, allowing automated agents to more intelligently access different data sources on the Web and thereby perform tasks (research, learning, etc.) more accurate for users. The term was coined by Tim Berners-Lee, co-inventor of the Web and director of W3C, who oversees the development of semantic Web standard proposals. Most of the time, when we use the term Semantic Web, we talk about the different technologies behind it. Among the best known are RDF (Resource Description Framework) which corresponds to an information model, and data exchange formats in RDF to communicate between different applications (RDF / XML, RDF / JSON, N3, Turtle, N-Triples and others). In the semantic Web domain, the semantics of data is described by ontologies with languages intended to provide a formal description of concepts, terms, or relationships of any domain. These languages are RDFS (Resource Description Framework Schema) and OWL (Web Ontology Language). One of the main goals of the Semantic Web is to enable users to use the full potential of the Web, so they can find, share and combine information more easily. Today everyone can use forums, use social networks, chat, do research or even buy different products. Nevertheless, it would be better for the machine to do all this in the place of the man because currently, the machines need the man to perform these tasks. The main reason is that current web pages are designed to be readable by humans and not by machines. The semantic Web has the main objective that these same machines can perform all the tedious tasks alone such as searching for or associating information and acting on the Web itself.

3.6. RDF

RDF (Resource Description Framework), is a standard describing:

- resources, a resource that can be anything (people, places, animals, documents, concepts, etc.);
- the description of these resources by attributes and relationships;
- the framework containing a data model, languages and syntaxes.

Indeed, RDF is mainly a data model, since the data is structured by triplets (subject, predicate, object):

- subject: a resource;
- predicate: a property;
- object: a literal or a resource

For example:

- (Big Ben, is created, 1859)
- (Big Ben, is located, London)

With this, one can easily deduce that the structure created by this model is a directed graph, whose subjects and objects are the nodes and whose predicates are the oriented edges. Here is a small example describing the two triplets above:

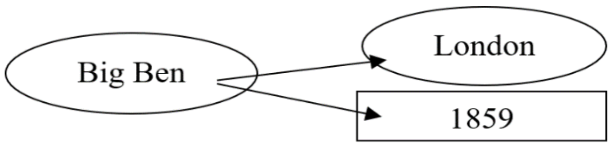

Fig. 1: Representation Via Triple Graphs.

The oval shapes represent resources and the rectangular shape represents a literal.

4. RDF spark architecture

To handle the growing size of RDF datasets that arrives up to billions of triple RDFs, we find the MapReduce Framework that Google developed for the parallel processing of very large datasets. Indeed writing programs with MapReduce is technically difficult and takes a lot of time. We use Spark because Spark allows applications on Hadoop Clusters to be executed up to 100 times faster in memory, 10 times faster on the disk, in addition to the "map" and "reduce" functions, Spark supports SQL queries and data flow and offers machine learning and graph-oriented processing functions. On the other hand, Apache Spark is an open source Big Data processing framework built to perform sophisticated and very fast analysis as MapReduce. We now present our approach which is a framework for translating SPARQL queries into Spark Jobs as an intermediate layer between SPARQL and Hadoop MapReduce. This abstraction layer ensures interoperability and compatibility to future Hadoop changes since it leaves our approach independent of the Hadoop version.

The Semantic Web allows the formalization, publication, and linking of the vocabularies used in these descriptions. These developments enable applications to use Web data more effectively by recognizing the different types of resources and links they encounter, and by exploiting the meaning and reasoning behind them, in the following, we cite the benefits of the use of Semantic Web technologies in Big Data.

1) Identify resources on the Web

The Web of data connects data sources of varying degrees of reliance on the classical architecture of the Web: i) Universal Identifiers (URIs) to name on the Internet. Web any resource; ii) a protocol (HTTP for Hypertext Transfer Protocol) for, from an address (URL for Universal Resource Locator), interrogating a resource and obtaining a representation thereof; iii) Hypertext Markup Language (HTML) to represent and communicate these representations. The Web of data does not necessarily exchange HTML documents, but data in general and in different formats. It is no longer just pages that are linked on the Web, but arbitrary resource identifiers. When an identifier is consulted, the servers respond by providing data describing the resource without it necessarily being on the Web (e.g. a car, an animal species, a protein, an author ....). The term "Web of Data" therefore emphasizes the possibility of opening our data silos of all sizes, from our address book to the huge genomic databases, and to exchange, link and compose them. according to our needs.

2) Describe the resources in a global data graph

Representing these data requires models, structures, formats, and languages. RDF (for Resource Description Framework) is to the Web of data what HTML is to Web documentary: the language that allows to represent and connect data about resources. RDF respects the architecture of the Web, including URIs to identify the resources and relationships described. Such descriptions can come from any source on the Web and be merged with others. The term Global Giant Graph sometimes refers to this web of world-wide data woven by thousands of descriptions distributed on the Web declaring links between nodes identified by URIs. Besides, RDF also provides a data model that serves as a foundation for other standards. Thus, above RDF, the SPARQL recommendation mainly provides three tools for data exchange: i) a language for querying and modifying RDF graphs; (ii) formats for the results of a query or change; iii) a protocol for submitting a request to a remote server and receiving the results, especially over HTTP. For example, on the DBpedia site, you can ask in SPARQL all the URIs of the resources named "Casablanca" in English. From the received identifiers, one can again interrogate the site to have additional data and thus to pass from related data in related data as one would pass from page to page.

3) Vocabulary and Formal Knowledge

Different types of models are designed to provide vocabularies for describing our world on the Web (such as ontologies or thesauri). By interrogating and reasoning about these computer models, it is possible to improve existing functionalities and to propose new ones. Above RDF stands the stack of schema languages, with increasing expressiveness and computational cost: the higher up the stack, the more logical definitions of the vocabulary can accurately capture the structures and meaning of the vocabulary. data, but also the reasoning on these schemes is expensive in terms of complexity and computation time. The first level says "light schemas" is that of RDFS (RDF Schema) for declaring and naming resource classes (such as books, movies, people ...) and their properties (like the author, the actor, title ...) and organize these types in hierarchies. Above RDFS, the OWL (Ontology Web Language) recommendation allows to formally represent definitions and is organized into several more or less extended expressivity fragments, which allow additional deductions in return for longer computation times. In the continuity of the Web of data, the "Semantic Web" thus emphasizes the possibility of exchanging the schemas of our data and the associated semantics. Formalized and published according to standards, these models make it possible to enrich the range of automatic processes that can be applied to data. By opening the data and their models, the Web of data and the Semantic Web open the whole of the uses that it is possible to make of it.

5. Experiments

In order to demonstrate how the RDFSpark can be a scalable RDF system, we describe an experiment. We discuss selected systems; in addition, we describe the experiment conditions, present the results, and discuss them. In this experiment, we wanted to evaluate the scalability of RDFSpark for this reason, we used the Lehigh University Benchmark (LUBM) [17], and we used four instances of this Benchmark LUBM1, LUBM2, LUBM3, LUBM4. (Table 1). We are using a cluster of 6 machines, each of which is equipped with an Intel Xeon CPU E5-2640 v4 @2.40GHz and 124GB RAM. Each machine has 20 physical CPU cores. However, we use the cluster with a hyper-threading option enabled, which gives an operating system effectively 40 cores for resource allocation. All machines in this cluster are connected to a switch using 10 Gigabit Ethernet. We give to Spark and YARN 100GB of RAM and 36 cores at each machine. We leave some fraction of the memory (remaining 24GB) and 4 CPU cores for the operating system.

Table 1: A Datasets Summary

Dataset	Number of Triples	Size
LUBM1	1,316,993	0.11GB
LUBM2	6,890,933	0.583GB
LUBM3	133.000.000	22 GB
LUBM4	1,334,000,000	213 GB

We compare the performance of our system with three other systems: Jena-HBase, a horizontally expandable triple RDF store that uses HBase for storage, CumulusRDF, the Cassandra-based RDF triplestore proposed by Ladwig et al, which uses an indexing scheme based on the NoSQL model, and H2RDF which is also based on HBase. The code Jena-HBase and H2RDF are open-source and we used the default configurations for the code.

Table 2: Query Execution Time on Lubm1

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Jena-HBase	324	6549	325	480	339	576	236	754	9766
H2RDF	430	7857	560	543	587	550	321	506	11276
CumulusRDF	308	5801	311	452	305	496	376	467	8744
RDFSpark	165	2765	176	265	187	298	188	345	3870

Table 3: Query Execution Time on LUBM2

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Jena-HBase	205	2495	175	207	180	1498	185	322	3874
H2RDF	831	7857	481	543	507	563	377	509	2800
CumulusRDF	308	5801	311	851	305	2492	341	752	8732
RDFSpark	414	3458	219	407	289	479	250	447	3332

Table 4: Query Execution Time on LUBM3

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Jena-HBase	324	6549	325	480	339	576	236	754	9766
H2RDF	430	7857	560	543	587	550	321	506	11276
CumulusRDF	308	5801	311	452	305	496	376	467	8744
RDFSpark	165	2765	176	265	187	298	188	345	3870

Table 5: Query Execution Time on LUBM4

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Jena-HBase	324	6549	325	480	339	576	236	754	9766
H2RDF	430	7857	560	543	587	550	321	506	11276
CumulusRDF	308	5801	311	452	305	496	376	467	8744
RDFSpark	165	2765	176	265	187	298	188	345	3870

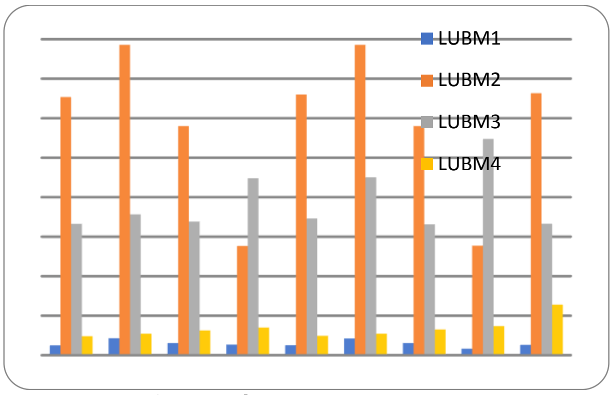

Fig. 2: Loading Time LUBM Queries.

For the Load Time: the evaluation shows that the load time increases linearly with the size of the data, as expected. The RDFSpark loading process uses Spark Streaming provided by Apache Spark to speed up the acquisition. The acquisition process is parallelized between the servers, using as much as possible all the servers.

Figures 2,3,4 and 5 illustrate the results of the execution of the nine LUBM Benchmark queries, according to the four datasets used LUBM1, LUBM2, LUBM3, and LUBM4.

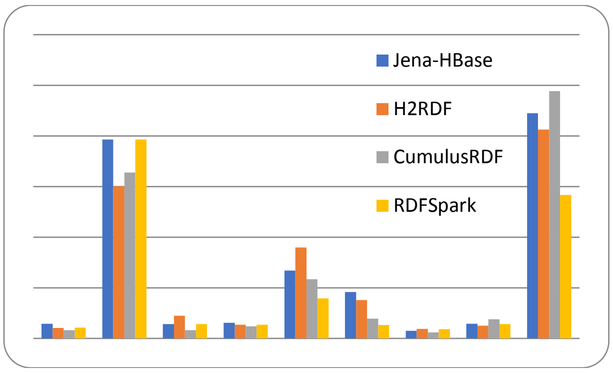

Fig. 3: LUBM1 Queries Execution Time.

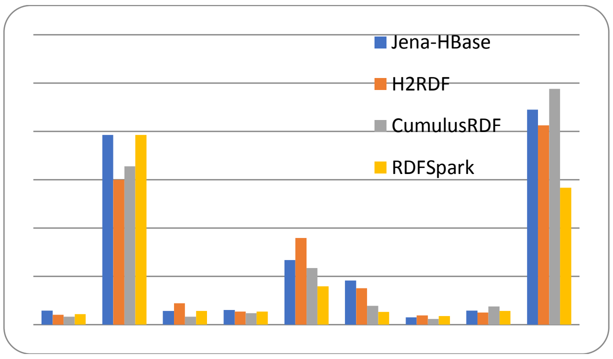

Fig. 4: LUBM2 Queries Execution Time.

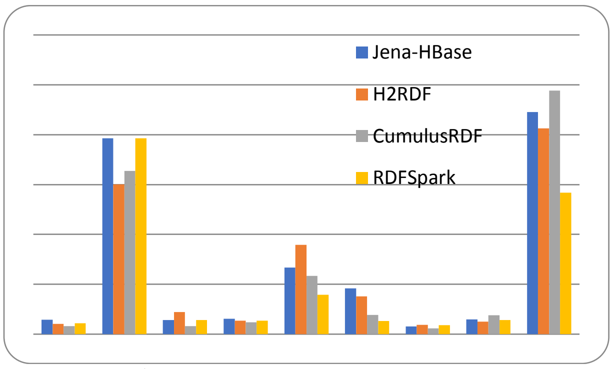

Fig. 5: LUBM3 Queries Execution Time.

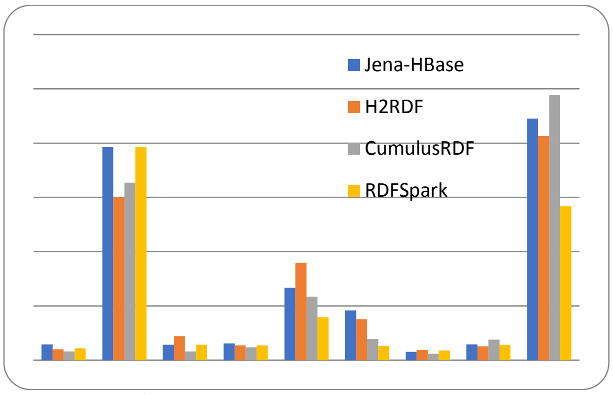

Fig. 6: LUBM4 Queries Execution Time.

From these results, our experiments firstly confirm the efficiency of RDFSpark, and for the processing time of a SPARQL query T with respect to the size of the data S, we notice that there is an almost linear scalability of this factor T / S. Thanks to the advantages of Spark, these evaluations show that RDFSpark is an efficient system for handling complex SPARQL[17, 18] queries against the processing of these queries by using MapReduce or a query language of NoSQL databases. Another interesting point is the step of optimizing RDFSpark, the series of optimization techniques reduces both the amount of data to be processed and the processing time of the corresponding SPARQL request.

6. Conclusion

Semantic Web applications generate huge amounts of data every day. RDF databases or triplestores are not scalable, on the other hand, the big data systems guarantee us these scalability options, high data availability, and the high performance of the system, from where helped to integrate the Semantic Web technologies in a Big Data environment. In this paper, we have seen the features of Apache Spark in data processing and analysis. Spark positions itself against traditional MapReduce implementations like Apache Hadoop or we have proposed RDFSpark, a new approach for scalable execution of Apache Spark-based SPARQL queries to applications based on extracting information from very large RDF data volumes. To do this, we designed and implemented a translation of SPARQL queries to Spark program. The resulting Spark program is translated into a MapReduce job sequence and run in parallel on a Hadoop cluster. It is also possible to combine the Spark processing types with Spark SQL, Spark Machine Learning and Spark Streaming, thanks to different Spark integration modes and adapters.

References

[1] Vukotic, Aleksa, Nicki Watt, Tareq Abedrabbo, Dominic Fox, and Jonas Partner. Neo4j in action. Manning Publications Co., 2014.
[2] Khadilkar, Vaibhav, Murat Kantarcioglu, Bhavani Thuraisingham, and Paolo Castagna. "Jena-HBase: A distributed, scalable and efficient RDF triple store." In Proceedings of the 11th International Semantic Web Conference Posters & Demonstrations Track, ISWC-PD, vol. 12, pp. 85-88. 2012.
[3] George, Lars. HBase: the definitive guide: random access to your planet-size data. " O'Reilly Media, Inc.", 2011.
[4] Papailiou, Nikolaos, Ioannis Konstantinou, Dimitrios Tsoumakos, and Nectarios Koziris. "H2RDF: adaptive query processing on RDF data in the cloud." In Proceedings of the 21st International Conference on World Wide Web, pp. 397-400. ACM, 2012. https://doi.org/10.1145/2187980.2188058.
[5] Hewitt, Eben. Cassandra: the definitive guide. " O'Reilly Media, Inc.", 2010.
[6] Ladwig, Günter, and Andreas Harth. "CumulusRDF: linked data management on nested key-value stores." In The 7th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2011), vol. 30. 2011.
[7] Banane, Mouad, Abdessamad Belangour, and Labriji El Houssine. "Storing RDF data into big data NoSQL databases." In First International Conference on Real Time Intelligent Systems, pp. 69-78. Springer, Cham, 2017. https://doi.org/10.1007/978-3-319-91337-7_7.
[8] Cudré-Mauroux, Philippe, Iliya Enchev, Sever Fundatureanu, Paul Groth, Albert Haque, Andreas Harth, Felix Leif Keppmann, Daniel Miranker, Juan F. Sequeda, and Marcin Wylot. "NoSQL databases for RDF: an empirical evaluation." In International Semantic Web Conference, pp. 310-325. Springer, Berlin, Heidelberg, 2013. https://doi.org/10.1007/978-3-642-41338-4_20.
[9] Mouad Banane and Abdessamad Belangour. "An Evaluation and Comparative study of massive RDF Data management approaches based on Big Data Technologies". International Journal of Emerging Trends in Engineering Research. Volume 7 No. 7 (2019). https://doi.org/10.30534/ijeter/2019/03772019.
[10] Banane, Mouad, and Abdessamad Belangour. "A Survey on RDF Data Store Based on NoSQL Systems for the Semantic Web Applications." In International Conference on Advanced Intelligent Systems for Sustainable Development, pp. 444-451. Springer, Cham, 2018. https://doi.org/10.1007/978-3-030-11928-7_40.
[11] Hassan, M., & Bansal, S. K. (2018). RDF data storage tech-niques for efficient SPARQL query processing using distributed computation engines. Proceedings - 2018 IEEE 19th International Conference on Information Reuse and Integration for Data Science, IRI 2018, 323–330. https://doi.org/10.1109/IRI.2018.00056.
[12] Pedrycz, Witold, and Shyi-Ming Chen, eds. Information granularity, big data, and computational intelligence. Vol. 8. Springer, 2014. https://doi.org/10.1007/978-3-319-08254-7.
[13] White, Tom. Hadoop: The definitive guide. " O'Reilly Media, Inc.",
[14] Lyubimov, Dmitriy, and Andrew Palumbo. Apache Mahout: Beyond MapReduce. CreateSpace Independent Publishing Platform, Zaharia, Matei, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Spark: Cluster computing with working sets." HotCloud 10, no. 10-10 (2010): 95.
[15] Meng, Xiangrui, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman et al. "Mllib: Machine learning in apache spark." The Journal of Machine Learning Research 17, no. 1 (2016): 1235-1241.
[16] Guo, Yuanbo, Zhengxiang Pan, and Jeff Heflin. "LUBM: A benchmark for OWL knowledge base systems." Web Semantics: Science, Services and Agents on the World Wide Web 3, no. 2-3 (2005): 158-182. https://doi.org/10.1016/j.websem.2005.06.005.

[17] Banane, Mouad, and Abdessamad Belangour. "New Approach based on Model Driven Engineering for Processing Complex SPARQL Queries on Hive." International Journal of Advanced Computer Science and Applications (IJACSA) 10, no. 4 (2019). https://doi.org/10.14569/IJACSA.2019.0100474.

[18] Mouad Banane, and Abdessamad Belangour. « 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 (2019): 734 – 741. https://doi.org/10.30534/ijatcse/2019/62832019.