ISSN: 0974-5823 Vol. 6 No. 3 December, 2021

Storage, Distribution, and Query Processing RDF Data in Apache Spark and GraphX: A Review Survey

Wria Mohammed Salih Mohammed[1,2,*], Alaa Khalil Jumaa[1]
[1]Technical college of Informatics, Sulaimani Polytechnic University, Sulaimani 46001, Kurdistan Region, Iraq
[2]College of Base Education, University of Sulaimani, Street 1- Zone 501 Sulaimani, Kurdistan Region, Iraq
* Corresponding Authors

Abstract

The term "Semantic Web" refers to W3C's vision of the Web of linked. The goal of the Semantic Web is to make Internet data machine-readable. Resource Description Framework (RDF) and Web Ontology Language are used to enable the encoding of semantics with the data. RDF is the basic core of the semantic web which allows the machine to exploit and understand data. It is a model to interchange data on the World Wide Web. Also, it extends the structure of the web links by creating the relationship among things and links. SPARQL Protocol and RDF Query (SPARQL) is a query language for RDF.

The main challenge in the data processing and analysis field is how to store and process a huge amount of data efficiently. Storing Big data needs too much hard disk space and sometimes the processing time is not satisfied properly. As a result of this, big data technologies are invented, these techniques can be used to store and process massive data using distributed file systems. Some of Big data techniques are used for storing and managing huge amounts of data like Apache Hadoop and others which are used for processing these data like MapReduce and Apache Spark.

This paper attempted to obtain benefits from mixing both emergency research area Semantic web and Big data technologies, the real challenge is how to use RDF graph-based with Apache Spark GraphX. This paper also includes a state-of-art for RDF and Spark GraphX, and a survey on how to store, distribute and query RDF data using Spark GraphX.

Keywords : RDF, RDFS, Spark, SPARQL, GraphX, Semantic Web, Hadoop. HDFS, Big Data.

1- Introduction

RDF is a typical data model for the description of resources that are available on the Web and their relationship together. It is recommended by World Wide Web Consortium (W3C). RDF is a common model for exchanging data on the web and the Triple-store is an RDF database. RDF has three main associations which are Subject, Predicate, and Object. Furthermore, SPARQL is the most well-known query language for RDF, which is also defined by W3C, it can work on querying massive RDF data. RDF can have Big data characteristics including veracity, velocity, volume, and variety. It can see that RDF data is highly varied with plenty of different types of data from many sources. The increasing of the RDF volume brings the challenge to solve the issue of storing, querying, and distributing big RDF data [1]. Big data technologies solved the problem of storing and processing huge data using Apache Hadoop and MapReduce. One of the main drawbacks of MapReduce is, instead of storing data on memory, it stores it on the disk. As a result, it affects slow processing querying. On the other hand, Resilient Distributed Dataset (RDD) in Spark has solved this issue which allows storing in memory. Spark can analyze graph data, machine learning, stream processing, and so on [19][3].

The main goal of this survey is how to use semantic web datasets with Big data technologies. Particularly, using RDF data with Hadoop, Map-reduce, Spark, and GraphX. In their theoretical aspects, how can store, query, and process large-scale semantic web data in a distributed method. The remaining sections of this paper are arranged as follows: In section 2, there is an explanation of semantic web data including RDF, RDFS and SPARQL. It also has the structure of RDF graph and OWL. In section 3, it is an overview of Apache Spark including GraphX, it also includes why Spark is chosen in this paper rather than MapReduce. Section 4, is the survey about the usage of semantic web data with Apache Spark, MapReduce, and GraphX. In section 5, it has a discussion about the papers which are used for the survey, followed by section 6 which includes the conclusion and future direction.

2- Semantic Web Data

The Semantic web was firstly mentioned by Tim Berners-Lee [4], it uses to make technologies to interpret the web of data, as well as, it provides a set of common interoperability data formats that can be used over the various platforms [5]. The term "Semantic Web" refers to W3C's sight for linked data for the web. Building data storage on the web can be enabled by semantic web technologies. In another word, the Semantic web is an extension of the current web, that makes data to have meaning. This improves the capability of people and computers to work together. One of the main purposes of web 3.0 is to define the method of connecting, exposing, and sharing data that use URIs (Uniform Resource Identifier) on the Web [8]. The Semantic web can write rules and build vocabularies. RDF, OWL, SPARQL, and SKOS are technologies that empowered linked data [1]. The Core of the semantic web is a generic and flexible language that allow to combination and represent data from different domain source. This language is called RDF. RDF is the linked data of the semantic web which store data in a triple format (subject, predicate, and object). The Predicate is the relationship between subject and object. Subject and predicate have to be URI, but the object can be either URI or literal as shown in Figure 2.

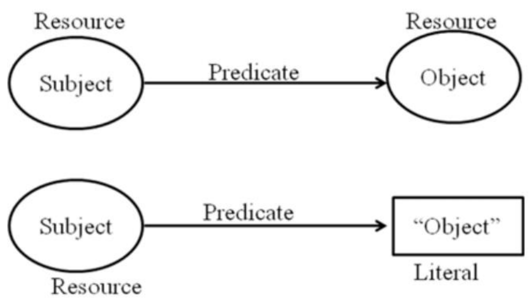

Figure 1: RDF Graph[15]

An RDF Graph can have different syntaxes such as N-Triple, RDF/XML, Notation-3, and Turtle. The RDF can be more structured and usable by adding some common vocabularies which are called RDF Schema (RDFS) vocabularies. OWL is also performing a function similar to RDFS, OWL is also including more common vocabularies [6][7]. Furthermore, the Resource Description Framework in Attribute (RDFa) is adding attributes to HTML which allows rich metadata to be embedded with the website. There are two types of RDFa formats, JSON-LD (JavaScript Object Notation for Linked Data) and Microdata [5].

The Semantic web architecture, which was built by Tim Berners-Lee, is illustrated in many types of research. But in brief, it has four layers shown in Figure 2 [8]:

- XML (eXtensible Markup Language), it represents data.

- RDF, shows the meaning of data.

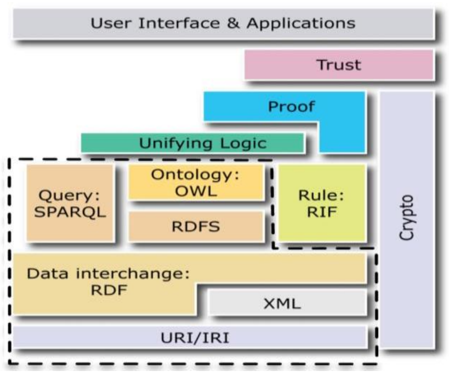

Figure 2: Semantic Web layers [8]

- Ontology, shows the proper common satisfaction on the meaning of the data.

Logical: it shows brilliant reason with the meaning of data. SPARQL is the well-known query language for RDF data in the simplest form. SPARQL query makes up of RDF triple patterns. It can query multiple datasets using complex queries [9].

3- Apache Spark and GraphX

Apache Spark is a cluster computing system based on in-memory storage. It is based on Apache Hadoop MapReduce algorithms, but dissimilar to MapReduce, Spark always stores intermediate and the output results of spark jobs in memory [10]. Spark has data abstraction called RDDs. It is a fault-tolerant collection of partitioned objects across a cluster of machines. Usually, any RDD provides parallel transformation (groupBy, join or filter) that could be detected to recover the RDD data [25]. RDD is a read-only data collection, it is an immutable dataset. It is enriching for having the operations to handle the data [13]. Spark has four common libraries, such as Spark SQL, Spark GraphX, Spark Streaming, and Spark ML-lib. It is capable of leveraging the Hadoop ecosystem, e.g., HDFS, YARN, HBase, S3 [11][14]. The ecosystem of Spark is described in Figure 3.

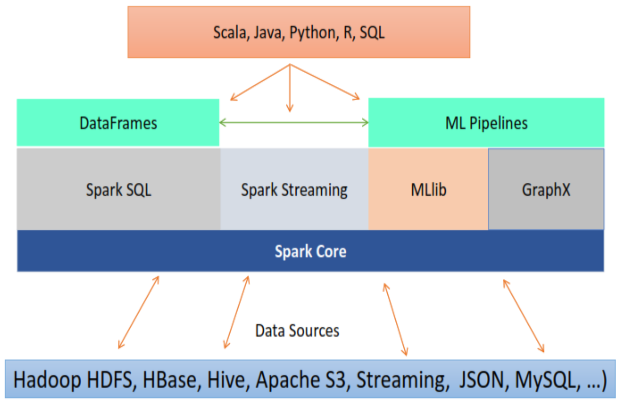
Scala, Java, Python, R, SQL
DataFrames
ML Pipelines
Spark SQL
Spark Streaming
MLlib
GraphX
Spark Core
Data Sources
Hadoop HDFS, HBase, Hive, Apache S3, Streaming, JSON, MySQL, ...)

Figure 3:Apache Spark Echo System [12].

In Apache Spark, data can be automatically divided by the master device on the cluster. Thus, the number of partitions can be chosen by users. It reduces the network transferring when it partitions data.

Many libraries can be used with Spark to analyze big data. Originally, Spark is written in Scala. However, another programming language can be used with Spark API such as Java, Python and R. Usually, using Scala with Spark can be more adequate because Spark can support Scala firstly and the size of the Scala code is less than another language in Spark [12].

GraphX is on the top of the Spark, it is a library to process graphs. It computes the model distribution to have common data structure processing including graph data or grouping and It works with in-memory RDDs [14]. GraphX makes up both directed adjacency structure and user-defined attributes associated with each edge and vertex. Usually, programs in GraphX show transformations from one to another from edges, vertices, or both in the context [16]. GraphX can have multiple graphs which can attach to each edge and vertex attribute. It also supports the computation of the graph because it has the basic set of operators including Join Vertices, Sub-graph, MapReduce Triplets, optimal transition Pregel API. It also has graphical analysis tasks and graphical algorithms including PageRank, triangle Counting [2].

Internally Graph consists of RDD collections: VertexRDD is a collection of vertices storing unique key properties. EdgeRDD is an edge collection that has the edge properties for the source and destination vertices. EdgeRDD and VertexRDD make GraphX supports many common challenges on graphs which are not easy to implement by other techniques. The same index data structures are used to collect edges and vertices. GraphX includes triplet view which is used for graph computation. A triplet has an edge with its sources and destination properties as shown in Figure 4. The neighborhood of vertices is built easily using triplets[22].

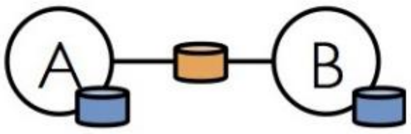

Figure 4: Triplet view [22]

Usually, any triplet consists of two functions: sendMsg and mergeMsg functions. SendMsg is a map function, it can apply on the triplet to send messages to destination or source vertices of the triplet. On the other hand, mergeMsg is a reduced function, it collects the message values which are sent by sendMsg functions. AggregateMessages can use both sendMsg and mergeMsg functions to build new VertexRDD [22].

4- Literature Survey

There are several researches using RDF in Big data technologies, particularly Apache Spark GraphX. In this section, the relevant recently published works are discussed and we showed how the Storage, Partition, and Queries operations are managed in those works.

In [17], the authors attempt to show a comprehensive study to compare four different SPARQL strategies, also different datasets and benchmark queries have been utilized with the different techniques with Spark. Firstly, SPARQL SQL is used which rewrites SPARQL into SQL queries and they are evaluated using Spark SQL engine. Secondly, SPARQL RDD uses Spark RDD with SPARQL with the large triple set. Thirdly, SPARQL DF is tabular data with specific relational operators. Finally, SPARQL Hybrid allows the query optimizer on data partitioning to mix local partitioning and broadcast joins. As a result, they received a result and they emphasize a hybrid query that improved the query performance in most cases.

In [18], they proposed Spark which has faster processing of RDF dataset. They used SQL queries on RDF data for evaluation. As well as, they gave two approaches to partition data on the cluster. This approach is used to decrease the number of intermediate results and transmit data over the network clusters.

On the other hand, in [19] the researchers proposed a method for investigating SPARQL queries using the GraphX analytic tools. They used tools to create a form of subject-predicate-object which is a graph-like structure. Also, they used looping edges for evaluating the properties of the data stored on nodes. In contrast, they try to reduce the loop edge to run faster and reduce sending messages. Furthermore, they try to change the query plan generator. As a result, they try to analyze the triple patterns using common SPARQL keywords such as FILTER, OPTIONAL, and so on.

In [20], the researchers proposed Subset Property Table (SPT) which is a relational partitioning scheme, this method is used to address the issues of query efficiency, query patterns, and reducing loading with pre-processing data. Also, the proposed system can reduce query input and join operation by clustering further partitions more than the existing Property Table method. They combined both Subset Property Table with Vertical Partition methods for storing RDF datasets.

In [21], Albahli used Apache Spark to load and query massive data, also, he attempted to evaluate the performance of queries using selection, sorting, filtering multiple attributes. The queries using distributed SPARQL on Spark GraphX are performed and he could find which stages of the query pipeline can be improved. This pipeline can include a load Graph, pattern of Basic Graph, and calculation. The main purpose of Albanhli's paper is to decrease the time of the loading graph, as a result, whole processing reduces the time.

In [22], the algorithm of matching the sub-graph is suitable with SPARQL queries is proposed. The researchers used large datasets to show the scalability of the system. Their algorithm shows RDF data as a graph using RDDs in GraphX. This can handle large amounts of RDF datasets and GraphX can partition the dataset. Each vertex has three properties, such as label, mach_track table (M_T), and an endFlag. As a result, the subject/object has value provided by the label. endFlag is set to true if the paths are made from a sequence of match BGP triples. Endflag detects if the vertex is located at the end of a path.

In [23], the authors used Apache HBase as one of the bases to distribute and store graph-based RDF data. Spark context is also proposed by this paper, and they experience the LUBM benchmark to show the originality of their work. In the first step, Two RDFS rules (SubClassOf and SubPropertyOf) are stored in the master node. In the second step, there are four important steps:

1- The key-value of RDD is initialized using accessing IR data models from HBase, this is called RDD(IR).
2- the operator of Map and Filter are used to provide the IR model. This can match at least one SubPropertyOf rule and run.
3- filter and map operator are called to execute (SubClassOf, Domain, and Range) of RDFS rules to obtain new rdf:type.
4- three above step results are mixed to obtain the final reasonable results.
The architecture of framework is explained in figure 3.

[24] introduces SparkRDF and elastic discrete to process semantic graphs with memory distribution. It affects to decrease high input/ output communications for partitioning platform. This technique uses the SPARQL query on Spark. All the results are distributed in memory to speed up the process of joining. This is effective to the search space and memory overhead. It divides the graph of RDF into multi-layer sub-graphs based on relations and classes.

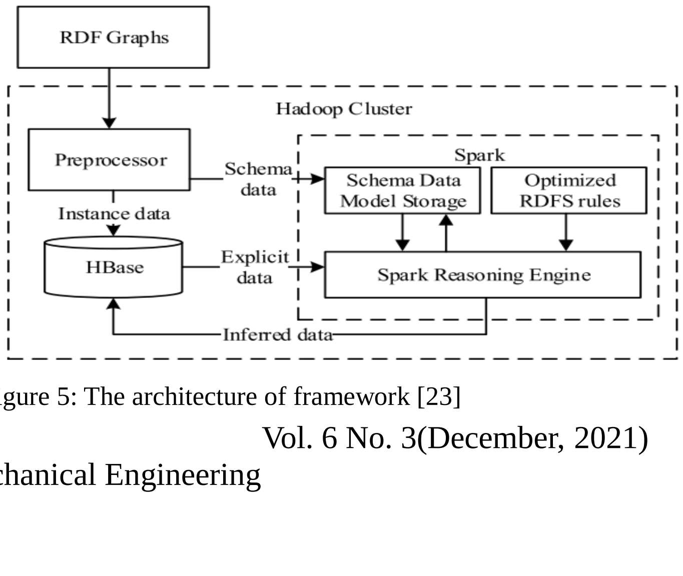

Figure 5: The architecture of framework [23]

In [25], they introduce SPARQL on Spark with GraphX (S2X) which is Hadoop based SPARQL query, they try to apply the matching of the SPARQL basic graph pattern which is graph-parallel while other functions are executed in a data-parallel method. This work tries to combine both data-parallel and a graph-parallel to query with SPARQL on the Hadoop which has RDF. This introduces the RDF property graph for GraphX, as well as, they built the algorithm of vertex-centric to basic graph pattern (BGP) match.

The authors in [25] attempted to propose an efficient method of distribution to match the query of answer sub-graph that uses MapReduce on RDF graphs. They use two techniques to improve their algorithms, one of them is RDF property filtering which is used to decrease intermediate results. And another algorithm is to improve the query performance using postponing the Cartesian product operations. These two techniques are used in an algorithm which is called StarMR which is a scalable and efficient distributed algorithm which is based on star decomposition.

Storage

The storage can be a relational database using the intermediate result to compile the RDF to a relational database, for instance, [18] uses MySQL with Spark which are works on the master node. The master node controls the clusters and the worker process the data distribution on the cluster. Furthermore, a new storage schema can be built using mixing Property Table with Vertical Partition as proposed in [20]. Also, it can be seen that existing data sources can be used, for example [21] used the DBPedia dataset, they used DBPedia for massive datasets, and for the few nodes graph, they created manually. They converted the DBPedia dataset which is Turtle format into N-Triple format experiments.

In [23] Hadoop cluster using HBase are be used for storing large-scaled RDF data storage environment. Additionally, all file indexes with joins can be stored using the HDFS technique [24]. The data can be arranged to store in a distributed manner, [26] stored in distributed adjacency list, they can find the neighbor N information for any vertex v. the vertices shows in the position of a subject which can be stored in the first column. For instance, Baghdad has a neighbor which is {<country, Iraq>}

Data partition

It can be seen that in [18], there is a way to reduce the network communication among the nodes of the cluster, they implemented a partitioning approach using Apache Spark to lessen the network communication among the cluster. This approach is appropriate to apply to massive RDF data. The clustering can be divided into two different partitioning techniques, one of them is irrelevant filed which can be partitioned, and another is used on fields that can be used in SQL queries.

On the other hand, two approaches can be combined to build new data partitioning technique, for instance, in [20] both Vertical Partitioning and Property Table are mixed. As a result, new data partitioning is produced. Also, the query can be partitioned for better performance, [21] the query can be distributed for instance SPARQL. This query processing can be divided into three main stages. First, loading the graphs, patterning graphs, and finally is the calculation of the result. Loading of graphs takes around 65% of the execution time.

[23] uses HBase for storage and partitioning according to the structure of graph-based. This also has a distribution of RDFS data in memory using SPARK RDD techniques.[24] created a vertical partitioning based on classes and relations which is called MESG (Multi-layer Elastic SubGraph). This connects predicate indexes to class indexes. As a result, Triple Pattern (TP) has a smaller index file, and it made it easy to query with SPARQL.

Querying

In [18], query SQL on RDF dataset is applied, the necessity of clustering has become common. But also, SPARQL can compile into Spark SQL for instance. The author in [20] create a compiler that applied using Flex and Bison which can transfer the SPARQL query to Spark SQL.

Furthermore, in [21] SPARQL is used with Spark and the author compared it with Jena results. As a result, they can find that it improves query performances, and reduces the time of loading data. They divided querying into three types, the first of them is a simple query and the second is an aggregation function on the dataset to compute the average, frequency and, so on, and the third one is querying with joins.

Also, [22] is based on subgraph matching and they used Basic Graph Pattern (BGP) queries. However, [23] used RDFS rules to map and filter operators to obtain a reasonable result as a query.

[24] Resilient Discreet SubGraph (RDSG) collects index files and intermediate results (IR), it distributed memory and query computed on large clusters easily using the basic operations such as RDSG_Filter, RDSG_Join, RDSG_Preparation, and RDSG_Gen. Querying on RDF graph using subgraph matching is used by [26]. this is performed by a distributed algorithm based on a decomposition star called StarMR.

5- Discussion

Apache Hadoop can be used to store big data in a distributed file system. However, the processing of the data using Hadoop MapReduce is not satisfactory as Spark. Because Apache Spark runs 100x faster than Hadoop MapReduce [27]. Nowadays, researchers mostly concentrated on Spark more than Hadoop MapReduce, for instance, in [18] Apache Spark is used for processing RDF data, they used two different methods for distribution data on the cluster.

On the other hand, when a combination between RDF graph triples and Apache Spark, there are several issues that come up, for instance, scalability, query efficiency, query optimization time, the query which can work for all types of query patterns, decreasing the time of data loading and pre-processing [20][27]. For instance, in [17], they use SPAQL with Spark including SPARQL SQL, SPARQL RDD, and SPARQL DF. SQL, RDD, and DF, which are parts of Spark, are used with SPARQL. However, these techniques have limitations, for instance, SPARQL RDD has less efficiency when the broadcast join is cheaper. It reads the whole dataset for a triple when there is a big dataset with a small joining. Furthermore, SPARQL DF has a problem with partitioning, because it is difficult to distribute subject, predicate, and object. As a result, SPARQL Hybrid, which is proposed by [17], allows mixing joins and query optimization for existing distributed data. Even though, it can be seen that the distribution of the big graph data can be done in GraphX using different files for edges and nodes, for instance, [19]collects the nodes with their properties and store them in a single file. As well, all edges are stored in another file. There is another method for distributing RDF data by making Property Table in subsets, and this mixes with the Vertical Partitioning method to having a new storage schema, this method is also translated SPARQL into Spark SQL, Spark SQL can be used for structured datasets [20].

Another major problem with the traditional data storage on a single machine is not accurate enough, because it has a limitation of processing time and hardware resources. That is why cluster processing is significantly important. Even though with using SPARQL query or analysis, linear SPARQL query has a limitation, it means when semantic data has SPARQL query with a single node processing, it cannot process whole data at once, it needs distribution process, that is why Apache Spark GraphX library has distribution processing, and it has three separated stages, including graph loading, Basic Graph Pattern and Result calculation [21]. In contrast, SPARQL with Basic Graph Pattern according to [22] has some limitations such as restricting the position of query variables to subjects and objects.

SPARQL over GraphX is a new graph processing that is developed over Spark. GraphX can combine the beneficial aspects from the combination between graph-parallel and data-parallel [22]. In [22], Basic Graph Pattern has been used for querying, however, it has limitations for instance the position of query variables has restricting because it only has subjects or objects.

6- Conclusion and Future direction

Semantic web data continuously produces huge volumes of diverse and heterogeneous data. On the other hand, due to the data volume issue, traditional Semantic web technologies cannot be stored or processed inefficient method. As a result, storing, distributing, and processing RDF data remains a challenging task.

The semantic web already satisfied the characteristics of big data technologies. This paper provides a state-of-art about Semantic web data and big data technologies, particularly, RDF, SPARQL, Hadoop, Spark and GraphX. Also, it provides a survey from recent papers which have recent development in combining both emergence research areas. This paper can be concluded with follows:

It can be noticed from the reviewed papers that SPARQL can be used mostly for querying big RDF data. Furthermore, there are several common SPARQL keywords, such as FILTER, OPTIONAL and so on, which are used for graph/sub-graph pattern matching. In SparkRDF, SPARQL is used, as well as, the distribution of SPARQL can be used to reduce loading time. Furthermore, Hybrid SPARQL is also utilized to optimize queries on existing distributed data. It can be noticed that there is a compiler to convert SPARQL to SQL for Spark. Another challenge of using the Semantic web with big data, it works on using reducing intermediate results. It is possible to uses the rules of RDF can be distributed, it works on initializing the key-value of RDD. The future direction from this survey is to reduce time loading and improve performance and query processing. This can use GraphX, which is based on Spark, with RDF graph-based data which has pattern graph matching using SPARQL as the main query language for RDF data. Also, the file can be stored using Apache Hadoop and GraphX. Vertex and Nodes of the GraphX can be dedicated for each part of RDF data.

6- References

[1] "Semantic Web - W3C." https://www.w3.org/standards/semanticweb/ (accessed Oct. 13, 2021).

[2] R. C. Maheshwar and D. Haritha, "Survey on high-performance analytics of bigdata with apache spark," *Proc. 2016 Int. Conf. Adv. Commun. Control Comput. Technol. ICACCCT 2016*, no. 978, pp. 721–725, 2017, doi: 10.1109/ICACCCT.2016.7831734.

[3] M. Banane and A. Belangour, "Towards a new scalable big data system semantic web applied on mobile learning," *Int. J. Interact. Mob. Technol.*, vol. 14, no. 1, pp. 126–140, 2020, doi: 10.3991/ijim.v14i01.10922.

[4] TIM BERNERS-LEE; JAMES HENDLER; ORA LASSILA, "The Semantic Web," *Circ. Linguist. Apl. a la Comun.*, vol. 73, no. May, pp. 303–314, 2001, doi: 10.5209/CLAC.59071.

[5] S. Kanza and J. G. Frey, "A new wave of innovation in Semantic web tools for drug discovery," *Expert Opin. Drug Discov.*, vol. 14, no. 5, pp. 433–444, 2019, doi: 10.1080/17460441.2019.1586880.

[6] P. Pauwels, S. Zhang, and Y. C. Lee, "Semantic web technologies in AEC industry: A literature overview," *Autom. Constr.*, vol. 73, pp. 145–165, 2017, doi: 10.1016/j.autcon.2016.10.003.

[7] S. A. Khan and R. Bhatti, "Semantic Web and ontology-based applications for digital libraries: An investigation from LIS professionals in Pakistan," *Electron. Libr.*, vol. 36, no. 5, pp. 826–841, 2018, doi: 10.1108/EL-08-2017-0168.

[8] A. Bakhouyi, R. Dehbi, M. Banane, and M. Talea, *A Semantic Web Solution for Enhancing the Interoperability of E-Learning Systems by Using Next Generation of SCORM Specifications*, vol. 1102 AISC. Springer International Publishing, 2020.

[9] I. Abdelaziz, R. Harbi, Z. Khayyat, and P. Kalnis, "A survey and experimental comparison of distributed SPARQL engines for very large RDF data," *Proc. VLDB Endow.*, vol. 10, no. 13, pp. 2049–2060, 2017, doi: 10.14778/3151106.3151109.

[10] J. Fu, J. Sun, and K. Wang, "SPARK-A Big Data Processing Platform for Machine Learning," *Proc. - 2016 Int. Conf. Ind. Informatics - Comput. Technol. Intell. Technol. Ind. Inf. Integr. ICIICII 2016*, pp. 48–51, 2017, doi: 10.1109/ICIICII.2016.0023.

[11] J. Yu, Z. Zhang, and M. Sarwat, "Spatial data management in apache spark: the GeoSpark perspective and beyond," *Geoinformatica*, vol. 23, no. 1, pp. 37–78, 2019, doi: 10.1007/s10707-018-0330-9.

[12] H. K. Omar and A. K. Jumaa, "Big Data Analysis Using Apache Spark MLlib and Hadoop HDFS with Scala and

Java," *Kurdistan J. Appl. Res.*, vol. 4, no. 1, pp. 7–14, 2019, doi: 10.24017/science.2019.1.2.

[13] Z. Han and Y. Zhang, "Spark: A Big Data Processing Platform Based on Memory Computing," *Proc. - Int. Symp. Parallel Archit. Algorithms Program. PAAP*, vol. 2016-Janua, pp. 172–176, 2016, doi: 10.1109/PAAP.2015.41.

[14] M. Assefi, E. Behravesh, G. Liu, and A. P. Tafti, "Big data machine learning using apache spark MLlib," *Proc. - 2017 IEEE Int. Conf. Big Data, Big Data 2017*, vol. 2018-Janua, pp. 3492–3498, 2017, doi: 10.1109/BigData.2017.8258338.

[15] A. Clara Kanmani, V. Suma, and N. Guruprasad, "Hybrid data model of PACE and quadruple: An efficient data model for cloud computing," *Int. J. Comput. Aided Eng. Technol.*, vol. 13, no. 1–2, pp. 73–100, 2020, doi: 10.1504/IJCAET.2020.108108.

[16] I. S. Reynold S. Xin, Joseph E. Gonzalez, Michael J. Franklin, "GraphX: A Resilient Distributed Graph System on Spark," 2013.

[17] H. Naacke, B. Amann, and O. Curé, "SPARQL graph pattern processing with apache spark," *5th Int. Work. Graph Data Manag. Exp. Syst. GRADES 2017 - Co-located with SIGMOD/PODS 2017*, pp. 1–7, 2017, doi: 10.1145/3078447.3078448.

[18] A. H. Atashkar, N. Ghadiri, and M. Joodaki, "Linked data partitioning for RDF processing on Apache Spark," *2017 3rd Int. Conf. Web Res. ICWR 2017*, pp. 73–77, 2017, doi: 10.1109/ICWR.2017.7959308.

[19] G. Gombos, G. Racz, and A. Kiss, "Spar(k)ql: SPARQL evaluation method on spark GraphX," *Proc. - 2016 4th Int. Conf. Futur. Internet Things Cloud Work. W-FiCloud 2016*, pp. 188–193, 2016, doi: 10.1109/W-FiCloud.2016.48.

[20] M. Hassan and S. K. Bansal, "Data Partitioning Scheme for Efficient Distributed RDF Querying Using Apache Spark," *Data Partitioning Scheme Effic. Distrib. RDF Querying Using Apache Spark Mahmudul*, pp. 24–31, 2019, doi: 10.1109/ICOSC.2019.8665614.

[21] S. Albahli, "Efficient distributed SPARQL queries on Apache Spark," *Effic. Distrib. SPARQL Queries Apache Spark*, vol. 10, no. 8, pp. 564–568, 2019, doi: 10.14569/ijacsa.2019.0100874.

[22] B. Kassaie, "SPARQL over GraphX," no. April, 2017, [Online]. Available: http://arxiv.org/abs/1701.03091.

[23] R. Li, Q. Zhang, H. Wang, and G. Wang, "Distributed RDFS Rules Reasoning for Large-Scaled RDF Graphs Using Spark," *Proc. Int. Conf. Serv. Sci. ICSS*, vol. 0, pp. 158–162, 2016, doi: 10.1109/ICSS.2016.28.

[24] X. Chen, H. Chen, N. Zhang, and S. Zhang, "SparkRDF: Elastic discreted RDF graph processing engine with distributed memory," *Proc. - 2015 IEEE/WIC/ACM Int. Jt. Conf. Web Intell. Intell. Agent Technol. WI-IAT 2015*, vol. 1, pp. 292–300, 2016, doi: 10.1109/WI-IAT.2015.186.

[25] F. B. B, M. Mehrotra, and H. Vo, "S2X: Graph-Parallel Querying of RDF with GraphX," *Int. Work. Data Manag. Anal. Med. Healthc.*, vol. 9579, no. 2015, pp. 155–168, 2015, doi: 10.1007/978-3-319-41576-5.

[26] X. Wang *et al.*, "Efficient Subgraph Matching on Large RDF Graphs Using MapReduce," *Data Sci. Eng.*, vol. 4, no. 1, pp. 24–43, 2019, doi: 10.1007/s41019-019-0090-z.

[27] S. M. Deshpande and R. S. Shirsath, "Ranking of Product on Big Data using Apache Spark," vol. 4, no. cPGCON, pp. 8001–8006, 2017.