2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

MapReduce-based Algorithms for Managing Big RDF Graphs: State-of-the-Art Analysis, Paradigms, and Future Directions

Alfredo Cuzzocrea
DIA Department
University of Trieste
Trieste, Italy
alfredo.cuzzocrea@dia.units.it

Rajkumar Buyya
Cloud Computing and Distributed Systems Lab
Department of Computing and Information Systems
The University of Melbourne, Australia
rbuyya@unimelb.edu.au

Vincenzo Passanisi
R&D Department
Xenia Progetti
Catania, Italy
vpassanisi@xeniaprogetti.it

Giovanni Pilato
ICAR-CNR
Palermo, Italy
pilato@pa.icar.cnr.it

***Abstract*— Big RDF (Resource Description Framework) graphs, which populate the emerging *Semantic Web*, are the core data structure of the so-called *Big Web Data*, the "natural" transposition of *Big Data* on the Web. *Managing big RDF graphs* is gaining momentum, essentially due to the fact that this task represents the "baseline operation" of fortunate *Web big data analytics*. Here, it is required to access, manage and process *large-scale*, *million-node* (big) RDF graphs, thus dealing with severe *spatio-temporal complexity challenges*. A possible solution to this problem is represented by the so-called *MapReduce-model-based algorithms for managing big RDF graphs*, which try to exploit the computational power offered by the MapReduce processing model in order to tame the complexity above. In this so-depicted scientific context, this paper provides a critical survey on MapReduce-based algorithms for managing big RDF graphs, with analysis of state-of-the-art proposals, paradigms and trends, along with a comprehensive overview of future research trends in the investigated scientific area.**

Keywords: *Web Big Data; Big RDF Graphs; Large-Scale RDF Graphs; MapReduce-Model-based Algorithms; Web Big Data Analytics*

I. INTRODUCTION

Big Web Data (e.g., [1,2]) are a relevant class of *big data* (e.g., [43,42,3]) occurring on the emerging *Semantic Web*. Here, *big RDF (Resource Description Framework) graphs* (e.g., [4,5,6,30]) play the major role, since a relevant number of *Web entities* can be modeled in terms of graphs, also nicely capturing their inter-connections, and used to *represent and mine Web knowledge derived from big data* (e.g., [29]). This paradigm is actually applied to a wide collection of interesting cases, ranging from *social networks* to *biological data processing*, and so forth.

The phenomenon is strongly stirred-up by the proliferation of a rich family of *big data processing frameworks and platforms*, among which *MapReduce* [7] and *Hadoop* [8] are the reference solutions for *distributed programming over large-scale repositories* and the corresponding *computational platforms*, respectively (e.g., [49]). Indeed, based on these big data processing solutions, it is possible to design and develop algorithms and tools for managing Web big data, with possibilities that are far beyond classical *parallel and distributed processing methodologies*. Coupled with this exciting programming/deploying opportunity, the amenity of designing and devising *Web big data analytics* (e.g., [29]) that extract "actionable knowledge" from the Semantic Web is really leading the recent *Web scene*, with a plethora of successful applications among which *Web advertisement* is just a fortunate and well-known instance. Several real-life applications have confirmed the reliability and the effectiveness of using RDF graphs in various modern scenarios. To give an example, Tian and Patel [50] propose a tool for querying large graph-databases produced by large-scale scientific applications, where *approximate*, rather than exact, *graph matching* is required, mostly due to the noisy and incomplete nature of real graph datasets. The proposed technique for supporting approximate matching of large graph queries introduces a novel indexing method that incorporates *graph structural information* in a *hybrid index structure*, yet exposing high pruning power and linear size-scalability as the database size scales linearly.

In this so-delineated scenario, the issue of *effectively and efficiently managing big RDF graphs* is gaining momentum, essentially because big data analytics for the Web largely found on this computational task in the vest of "baseline operation". Indeed, a leading solution in this context is represented by the so-called *MapReduce-model-based algorithms for managing big RDF graphs and big RDF databases* (being the latter one the "natural" way of representing RDF graphs in memory). Basically, these approaches try to exploit the computational power of the MapReduce processing model in order to tame the *spatio-temporal complexity* inducted by accessing, managing and processing *large-scale*, *million-node* (big) RDF graphs. Recent studies (e.g., [51,52]) have already highlighted the suitability of the MapReduce model for RDF data processing, especially because RDF sub-graphs can be easily mapped on nodes of a distributed architecture on top of which MapReduce is running. Of course, this is not the only solution available in literature. A complete survey of other processing/programming models for big data processing is available in Kune *et al.* [49].

978-1-5090-6611-7/17 $31.00 © 2017 IEEE
DOI 10.1109/CCGRID.2017.109

Based on the current literature focusing on MapReduce-based algorithms for big RDF graph management, we can identify the following three main classes of algorithms:

- *MapReduce-based algorithms for big data processing*;
- *MapReduce-based algorithms for managing RDF graphs*;
- *MapReduce-based algorithms for managing RDF databases*.

The first one represents a class of algorithms devoted to the general goal of processing big data on the Web, with a clear reference to RDF data. The second one represents a class of algorithms specifically devoted to the issue of managing RDF graphs. Finally, the third one represents a class of algorithms that focus on the issue of managing RDF databases definitely.

Inspired by the motivations above, in this paper we propose a critical survey on MapReduce-based algorithms for managing big RDF graphs, with analysis of state-of-the-art proposals, paradigms and trends. We complete our analytical contributions by means of a comprehensive overview of future research trends in the investigated scientific area.

The rest of the paper is organized as follows. In Section 2, we provide definitions, concepts and examples on RDF graphs. Section 3 focuses on MapReduce-based algorithms for big data processing. In Section 4, we move the attention on MapReduce-based algorithms for managing RDF graphs. Section 5 considers MapReduce-based algorithms for managing RDF databases. In Section 6, we draw future research directions in the context of MapReduce-based algorithms for big RDF graphs. Finally, Section 7 concludes the paper.

II. RDF GRAPHS: FUNDAMENTALS AND EXAMPLES

Since RDF graphs play a central role in our research, in this Section we focus in a greater detail on definitions, concepts and examples of such very important data structure of the popular Semantic Web. RDF is a Semantic Web data model oriented to represent *information on resources available on the Web*. To this end, a critical point is represented by the fact that RDF model *metadata* about such Web resources, which can be defined as data that describe other data. Thanks to metadata, RDF can easily implements some very important functionalities over the Semantic Web, such as: resource representation, querying resource on the Web, discovering resources on the Web, resource indexing, resource integration, interoperability among resources and applications, and so forth. RDF is primarily meant for managing resources on the Web for applications, not for end-users, as to support *querying the Semantic Web* (e.g., [20]), *Semantic Web interoperability* (e.g., [21]), *complex applications* (e.g., [22]), and so forth.

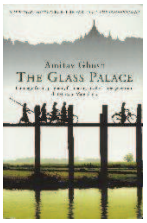
Looking into more details, RDF is founded on the main assertion that resources are identified on the Web via suitable *Web identifiers* like *Uniform Resource Identifiers* (URI) and they are described in terms of *property-value* pairs via *statements* that specify these properties and values. In an RDF statement, the *subject* identifies the resource (to be described), the *predicate* models the property of the subject, and the *object* reports the value of that property. Therefore, each RDF resource on the Web is modeled in terms of the triple ⟨*subject*, *predicate*, *object*⟩. This way, since Web resources are typically connected among them, the Semantic Web represented via RDF easily generates the so-called RDF graphs. In such graphs, RDF models statements describing Web resources via nodes and arcs. In particular, in this graph-ware model, a statement is represented by: (*i*) a node for the subject; (*ii*) a node for the object; (*iii*) an arc for the predicate, which is directed from the node modeling the subject to the node modeling the object. In terms of the popular *relational model*, an RDF statement can be represented like a tuple ⟨*subject*, *predicate*, *object*⟩ and, as a consequence, an RDF graph can be naturally represented via a *relational schema* storing information on subjects, predicates and objects of the corresponding Web resources described by the RDF statements modeled by that graph (e.g., [23]).

Figure 1, borrowed from Herman [24], shows an example where information on the Web regarding the book "*The Glass Palace*", by A. Ghosh, occur in two different sites in the vest of different media, one about the English version of the book, and the other one about the French version of the book. According to this, the two different Web resources are represented by two different relational schemas (see Figure 1). Figure 2, instead, shows the corresponding RDF graphs for the two different Web resources, respectively.

As highlighted before, one of the goal of RDF graphs is allowing the easily integration of correlated Web resources. This is, indeed, a typical case of Web resource integration because the two RDF graphs, even if different, describe two different media of the same resource. It is natural, as a consequence, to integrate the two RDF graphs in a unique (RDF) graph, even taking advantage from the "natural" topological nature of graphs, as shown in Figure 3. In particular, the integration process is driven by recognizing the same URI that identifies the book resource.

Another nice property of the RDF graph data model that makes it particularly suitable to Semantic Web is its "open" nature, meaning that any RDF graph can be easily integrated with *external knowledge* on the Web about resources identified by the URIs it models. For instance, a common practice is that of integrating RDF graph with Web knowledge represented by *Wikipedia* [25] via ad-hoc tools (e.g., *DBpedia* [26]). This contributes to move the actual Web towards the Web of data and knowledge. Figure 4 shows the RDF graph of Figure 3 integrated with knowledge extracted from Wikipedia via DBpedia.

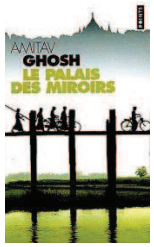
It is worth noticing that, when RDF graphs are processed, applications can easily extract knowledge from them via so-called *RDF query language* [27] or other AI-inspired approaches (e.g., [28]). In addition to this, mining such graphs is critical for a wide range of Web applications. To give an example, traversal paths can be used as "basic" procedures of powerful *analytics over Web Big Data* (e.g., [29]). This further gives solid motivation to our research.



ID	Author	Title	Publisher	Year
ISBN 0-00-6511409-X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

ID	Publisher's name	City
id_qpr	Harper Collins	London



	ID	Titre	Traducteur	Original
2	ISBN 2020286682	Le Palais des Miroirs	$A12$	ISBN 0-00-6511409-X

	ID	Auteur
7	ISBN 0-00-6511409-X	$A11$

	Nom
11	Ghosh, Amitav
12	Besse, Christianne

Figure 1. Different Web Media for the Same Book (left) and their Corresponding Relational Schemas (right) [24].

III. MapReduce-Based Algorithms for Big Data Processing

This Section focuses on algorithms devoted to the general goal of processing big data on the Web, with a clear reference to RDF data.

Dittrich and Quiané-Ruiz [9] put light on main issues and challenges of big data processing over MapReduce, by highlighting actual data management solutions that found over this computational platform. Several aspects are touched, including *job optimization*, *physical data organization*, *data layouts*, *indexes*, and so forth. Finally, a comparative analysis between Hadoop-MapReduce and *Parallel DBMS* is provided, by highlighting their similarities and differences.

Chen *et al.* [10] *investigates interactive analytical processing in big data systems*, with particular emphasis over the case of MapReduce, even considering the industrial applicative setting. In more detail, authors capture and model typical *big data processing workloads*, which are characterized by several small, short and increasingly interactive jobs, which clearly contrast with the large, long-running batch jobs for which MapReduce was originally designed. In line with this trend, authors provide an empirical analysis of MapReduce traces from several business-critical deployments inside Facebook and Cloudera instances in a wide range of applications scenarios ranging from e-commerce to telecommunication systems, from social media to large-scale retail systems, and so forth. As a result of their empirical analysis, authors provide a new class of MapReduce workloads that are composed by *interactive analysis components* making heavy use of query-like programming frameworks on top of MapReduce.

Papailiou *et al.* [30] introduce and experimentally assess *H_2RDF*, an innovative, *fully-distributed RDF framework* that *combines MapReduce with NoSQL (distributed) data stores*. With respect to the state-of-the-art contributions, *H_2RDF* allows two distinctive characteristics to be achieved: (*i*) high efficiency in both *simple* and *multi-join SPARQL queries* [31] via joins that execute on the basis of query selectivity; (*ii*) alternatively-available centralized and MapReduce-based join execution that ensures higher speed-up during query evaluation. The system is targeted to billions of RDF triples even with small commodity Clouds, and its performance over that of comparative methods is experimentally proven.

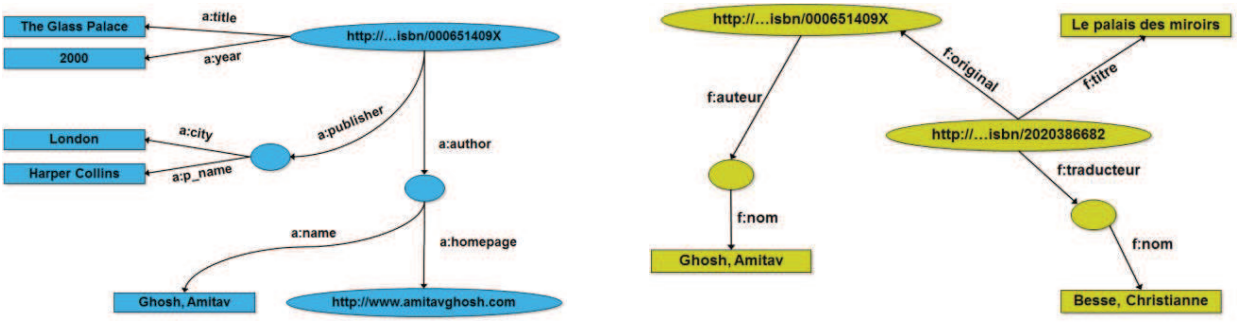


Figure 2. The Two RDF Graphs Corresponding to the Two Different Web Media for the Same Book Reported in Figure 1 [24].

Przyjaciel-Zablocki *et al.* [32] similarly focus the attention on the problem of *efficiently supporting SPARQL join queries over big data in Cloud environments via MapReduce*. The novelty of this research effort relies on an implementation of the *distributed sort-merge join algorithm* on top of MapReduce where the join is computed during the map phase completely. The problem of *cascaded executions* that may derive from executing multiple joins is addressed during the reduce phase by ensuring that the "right-hand" side of the join is always pre-sorted on the required attributes. Experiments show an excellent performance over comparative approaches.

Finally, at the mere system-side, several works focus the attention on very interesting applicative settings, such as the case of supporting MapReduce-based big data processing on *multi-GPU systems* (e.g., [11]), and the case of efficiently supporting *GIS polygon overlay computation* with MapReduce for *spatial big data processing* (e.g., [12]).

IV. MapReduce-Based Algorithms for Managing RDF Graphs

This Section concerns with algorithms specifically devoted to the issue of managing RDF graphs.

Choi *et al.* [13] focus the attention on the problem of *effectively and efficiently supporting scalable storage and retrieval of large volumes of in-memory-representations of RDF graphs* by exploiting a combination of MapReduce and HBase. The solution conveys to the so-called *RDFChain* framework, whose storage schema reflects *all* the possible join query patterns, thus providing a reduced number of storage accesses on the basis of the joins embedded in the target queries. In addition to this, a *cost-based map-side join* of RDFChain is provided. This allows to reduce the number of *map jobs* significantly.

Kim *et al.* [14] consider the research challenge of *optimizing RDF graph pattern matching on MapReduce*. Authors correctly recognize that the MapReduce computation model provides limited *static optimization techniques* used in relational DBMS, such as indexing and cost-based optimizations, hence it is mandatory to investigate *dynamic optimization techniques* for join-intensive tasks on MapReduce. With this idea in mind, authors extend their previous *Nested Triple Group data model and Algebra* (NTGA) for efficient graph pattern query processing in the Cloud [15] and achieve the definition of a *scan-sharing technique* that is used to optimize the processing of graph patterns with repeated properties. Specifically, the proposed scan-sharing technique eliminates the need for repeated scanning of input relations when properties are used repeatedly in graph patterns.

Finally, Zhang *et al.* [16] propose *a cost-model-based RDF join processing solution using MapReduce* to minimize the query responding time as much as possible. In particular, authors focus the attention on *SPARQL queries in a shared-nothing environment on top of MapReduce*, and propose a technique based on which they (1) first transform a SPARQL query into a sequence of MapReduce jobs, and (2) then create a novel index structure, called *All Possible Join tree* (*APJ-tree*) for reducing the searching space inducted by the optimal execution plan of a set of MapReduce jobs. To speed up the join processing, they employ *hybrid join* and Bloom filters for performance optimization. Authors complete their analytical contributions by means of an extensive set of experiments on real data sets that prove the effectiveness of the proposed cost model.

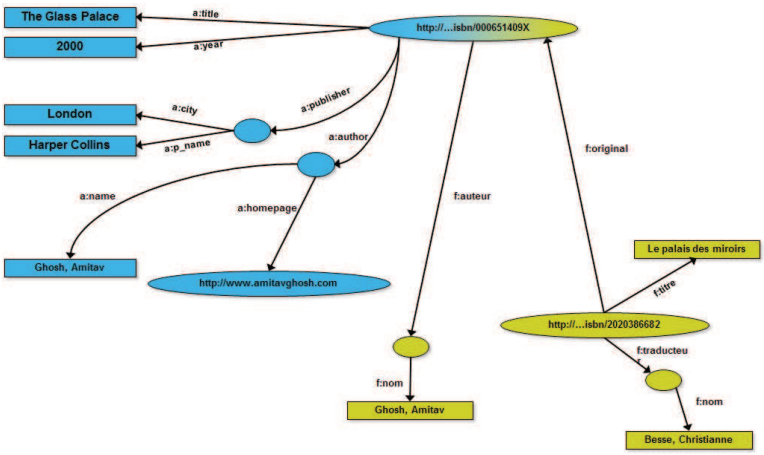


Figure 3. RDF Graph Obtained via Integrating the Two RDF Graphs of Figure 2 [24].

V. MapReduce-Based Algorithms for Managing RDF Databases

The relevance of RDF query processing via MapReduce has been highlighted by several studies (e.g., [33, 34, 35]). Basically, these works focus the attention on RDF data management architectures and systems designed for Cloud environments, with a special focus to large-scale RDF processing. Approaches investigated here mainly consider several special tasks that can be performed on the Web via RDF, such as *Web data publishing* and *Web data exchanging*. Within this scientific niche, this Section considers algorithms that focus on the issue of managing RDF databases.

Gergatsoulis *et al.* [36] focus the attention on querying linked data based on MapReduce. Linked data can easily be modeled via RDF databases. In particular, authors consider linked data that are arbitrarily partitioned and distributed over a set of Cloud nodes, such that input queries are decomposed into a set of suitable sub-queries, each one involving data stored in a specific node. The proposed method is two-step in nature. In the first step, sub-queries are executed on the respective node in an isolated manner. In the second step, intermediate results are combined in order to obtain the final answer to the input query. The innovation introduced by this research effort consists in the fact that the proposed query algorithm is independent on all the parameters of the model, i.e. linked data partitioning, local data storage, query decomposition mechanism, local query algorithm.

SPARQL query processing over RDF data has attracted a lot of attention from the research community as well. In this context, several approaches have been proposed recently. Schätzle *et al.* [37] address the problem of *supporting SPARQL queries over very large RDF datasets* via mapping SPARQL statements over *PigLatin* [38] programs. These programs are finally executed in terms of a series of MapReduce jobs on top of a Hadoop cluster. The study also provides an interesting experimental trade-off analysis on top of a popular benchmark dataset. Nie *et al.* [39] also investigate *scalability issues of processing SPARQL queries over Web-scale RDF knowledge bases*. To this end, authors propose an innovative partitioning method particularly suitable to RDF data that introduces the nice amenity of providing *effective indexing schemes* for supporting efficient SPARQL query processing over MapReduce. Still in this area, Du *et al.* [40] and Punnoose *et al.* [41] provide anatomy and main functionalities of *a semantic data analytical engine* and *a Cloud-enabled RDF triple store*, respectively. Both embed several points of research innovation over state-of-the-art proposals.

Urbani *et al.* [17] study how to apply *compression paradigms* to obtain *scalable RDF processing with MapReduce*. Authors point the specific case of Semantic Web, where *many billions of statements*, which are released using the RDF data model, exist. Therefore, they propose to apply a novel data compression technique in order to tame the severity of data size, and thus design *a set of distributed MapReduce-based algorithms* to efficiently compress and decompress a large amount of RDF data. In particular, they apply a *dictionary encoding technique* that maintains the structure of the (Web) data. The solution is implemented as a prototype using the Hadoop framework, and its performance is evaluated over a large amount of data and scales linearly on both input size and number of nodes.

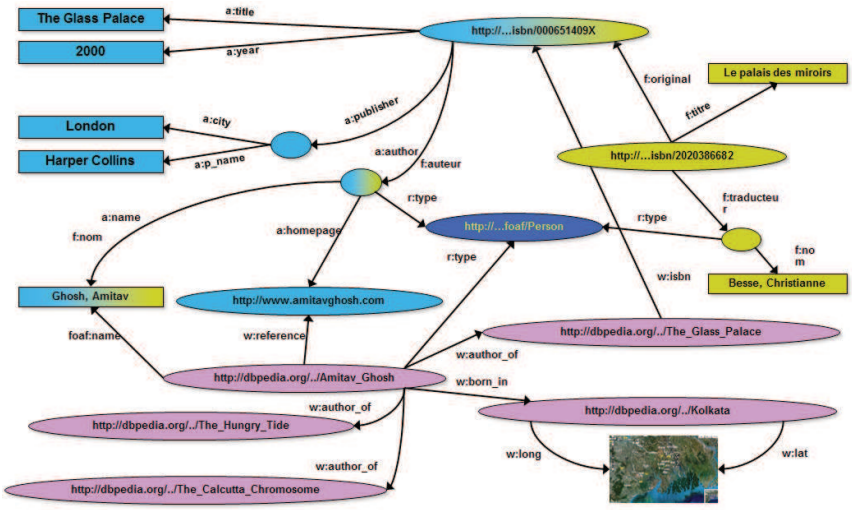


Figure 4. RDF Graph Obtained via Integrating the RDF Graph of Figure 3 with Wikipedia Knowledge Extracted via DBpedia [24].

Still in the context of optimization issues, Ravindra and Anyanwu [18] propose *a novel nested data model for representing intermediate data* of RDF processing concisely using *nesting-aware dataflow operators* that allow for lazy and partial un-nesting strategies. Indeed, authors correctly recognize that, during RDF data processing, intermediate results of joins with multi-valued attributes or relationships contain redundant sub-tuples due to repetition of single-valued attributes. As a consequence, the amount of redundant content is high for real-world multi-valued relationships in typical data-intensive instances such as social networks or biological datasets. Unfortunately, in MapReduce-based platforms, redundancy in intermediate results contributes avoidable costs to the overall I/O, sorting, and network transfer overhead of join-intensive workloads due to longer workflows. Therefore, in order to deal with this challenge, the proposed approach reduces the overall I/O and network footprint of a workflow by concisely representing intermediate results during most of a workflow's execution, until complete un-nesting is absolutely necessary.

Finally, Ravindra and Anyanwu again [19] focus the attention on query optimization aspects of massive *data warehouses* over RDF data. In this context, a relevant problem is represented by the evaluation of join queries over RDF data *when triple patterns with unbound properties occur*. A clear example to this case is represented by edges with "don't" care

labels. As a consequence, when evaluating such queries using relational joins, intermediate results contain *high redundancy* that is empathized by such unbound properties. In order to face-off this problem, authors propose *an algebraic optimization technique that interprets unbound-property queries on MapReduce via using a non-relational algebra based on a TripleGroup data model*. Novel logical and physical operators and query rewriting rules for interpreting unbound-property queries using the TripleGroup-based data model and algebra are formally introduced. The proposed framework is implemented on top of *Apache Pig* [44], and evaluated using both synthetic and real-word benchmark datasets and demonstrated the benefits of the framework.

VI. Future Research Directions

There are a number of research directions to be considered by future efforts in the context of big RDF graph management. In the following, we report on some among the most noticeable ones. Corresponding solutions should be, of-course, devised on top of suitable MapReduce/Hadoop infrastructures.

Storage Solutions for Large-Scale Big RDF Graphs: In-memory representation of large-scale big RDF graphs constitutes an open problem. In this context, ***a*** *column-based approach is* one of the promising research directions.

Indexing Big RDF Graphs: *Indexing data structures* improve query processing on big RDF graphs. Building *cost-effective* indexing data structures is a classical open problem that gets worse when big RDF graphs are considered. Several opportunities such as *cluster indexing solutions* arise here.

Fragmentation/Partition Paradigms for Big RDF Graph Management: In fully-distributed big RDF graph management, *fragmentation/partition paradigms* can really improve the overall throughput of the reference computational infrastructure. This because, according to the fully-distributed paradigm, large-scale big RDF graphs are *systematically fragmented* into smaller RDF graphs and *distributed* on top of a Cloud architecture, so that smaller computations on smaller (sub-)graphs are possible on different Cloud nodes. *Merging intermediate results* is still an open, deriving problem.

Flexible Query Languages for Big RDF Graphs: Extending the capabilities of Web big data analytics can be possible via introducing *flexible query languages for big RDF graphs*. This means enriching actual RDF query language syntax by adding novel features such as *preference query processing*, *approximate query answering*, *query expansion methods*, and so forth, specifically tailored to big RDF graphs.

Integration with NoSQL Databases: NoSQL databases can be considered as the "natural" repositories for big RDF graphs, due to their intrinsic characteristic of being prone to support *high-performance query processing*, *large-scale transaction management*, *and support for long-running computations*. All the latter ones are important requirements for effective and efficient big RDF graph management.

Uncertain Big RDF Graph Management: It is well-known that real-life applications and systems are characterized by *imprecise and uncertain data*. Big RDF graphs can be of such a nature too, as originated by a plethora of scientific applications like *bio-informatics tools*, *genomic computing platforms*, *micro-array data processing components*, and so forth, which all *naturally* introduce imprecision and uncertainty in data. This calls for modes, techniques and algorithms that should be capable of efficiently managing uncertain big RDF graphs still in the presence of these challenging factors (e.g., providing *complete* answers over *incomplete* big RDF sub-graphs). It is important to note that this topic is widely investigated in literature (e.g., [47,48]), hence interesting correlations can inspire future work.

Privacy-Preserving Big RDF Graphs Management: *Protecting the privacy of big RDF graphs* while managing them will become critical for next-generation Cloud applications and systems that process such graphs. In this context, several solutions can be considered, such as approaches devoted to *RDF data obfuscation* (e.g., *node obfuscation*) or to *generalization/abstraction methodologies* (e.g., sub-graphs generalizing other sub-graphs via *isomorphism*).

Big RDF Graph Analytics: Devising suitable *big RDF graph analytics* is another hot topic in the context of big RDF graph management, also following similar initiatives developed in other related scientific contexts (e.g., [45,46]). Indeed, this imposes us to provide effective and efficient support for low-level operations that can be applied to the underlying big RDF graphs directly, such as *query processing*, *keyword-based searching*, *sub-graph merging*, and so forth.

VII. Conclusions

Starting from the emerging need for effective and efficient algorithms for big RDF graph management, as required by modern Cloud applications and systems, in this paper we have presented a critical survey on MapReduce-based algorithms for managing big RDF graphs, with analysis of state-of-the-art proposals, paradigms and trends. Big RDF graphs on the Web have been described deeply, and several examples on their usage in modern *Big Web Intelligence* scenarios have been provided. We have also provided a comprehensive overview of future research trends in the investigated scientific area. Our final aim is that, overall, this contribution will represent a precious milestone on the exciting research road towards achieving effective and efficient MapReduce-based algorithms for supporting big RDF graph management.

References

[1] Zhong, N.; Yau, S.S.; Ma, J.; Shimojo, S; Just, M.; Hu, B.; Wang, G.; Oiwa, K.; Anzai, Y. Brain Informatics-Based Big Data and the Wisdom Web of Things. *IEEE Intelligent Systems*, vol. 30, no. 5, 2015, pp. 2–7.

[2] Lane, J.; Kim, H.J. Big Data: Web-Crawling and Analysing Financial News using RapidMiner. *International Journal of Business Information Systems*, vol. 19, no. 1, 2015, pp. 41–57.

[3] Cuzzocrea, A.; Saccà, D.; Ullman, J.D. Big Data: A Research Agenda. In *IDEAS*, 2013, pp. 198–203.

[4] Curé, O.; Naacke, H.; Randriamalala, T.; Amann, B. LiteMat: A Scalable, Cost-Efficient Inference Encoding Scheme for Large RDF Graphs. In *Big Data*, 2015, pp.1823–1830.

[5] Anyanwu, K.; Ravindra, P.; Kim, H.S. Algebraic Optimization of RDF Graph Pattern Queries on MapReduce. In *Large Scale and Big Data, Springer*, 2014, pp. 183–228.

[6] Lee, K.; Liu, L. Scaling Queries over Big RDF Graphs with Semantic Hash Partitioning. *PVLDB*, vol. 6, no. 14, 2013, pp. 1894–1905.

[7] Dean, J.; Ghemawat, S. MapReduce: Simplified Data processing on Large Clusters. *Communications of the ACM*, vol. 51, no. 1, 2008, pp. 107–113.

[8] Hadoop. http://wiki.apache.org/hadoop

[9] Dittrich, J.; Quiané-Ruiz, J.-A. Efficient Big Data Processing in Hadoop MapReduce. *PVLDB*, vol. 5, no. 12, 2012, pp. 2014–2015.

[10] Chen, Y.; Alspaugh, S; Katz, R.H. Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads. *PVLDB*, vol. 5, no. 12, 2012, pp. 1802–1813.

[11] Jiang, H.; Chen, Y.; Qiao, Z.; Weng, T.-H.; Li, K.-C. Scaling Up MapReduce-based Big Data Processing on Multi-GPU Systems. *Cluster Computing*, vol. 18, no. 1, 2015, pp. 369–383.

[12] Wang, Y.; Liu, Z.; Liao, H.; Li, C. Improving the Performance of GIS Polygon Overlay Computation with MapReduce for Spatial Big Data Processing. *Cluster Computing*, vol. 18, no. 2, 2015, pp. 507–516.

[13] Choi, P.; Jung, J.; Lee, K.-H. RDFChain: Chain Centric Storage for Scalable Join Processing of RDF Graphs using MapReduce and HBase. In *ISWC (Posters & Demos)*, 2013, pp. 249–252.

[14] Kim, H.S.; Ravindra, P.; Anyanwu, K. Scan-Sharing for Optimizing RDF Graph Pattern Matching on MapReduce. In *CLOUD*, 2012, pp.139–146.

[15] Ravindra, P.; Kim, H.S.; Anyanwu, K. An Intermediate Algebra for Optimizing RDF Graph Pattern Matching on MapReduce. In *ESWC*, 2011, pp. 46–61.

[16] Zhang, X.; Chen, L.; Wang, M. Towards Efficient Join Processing over Large RDF Graph Using MapReduce. In *SSDBM*, 2012, pp. 250–259.

[17] Urbani, J.; Maassen, J.; Drost, N.; Seinstra, F.J.; Bal, H.E. Scalable RDF Data Compression with MapReduce. *Concurrency and Computation: Practice and Experience*, vol. 25, no. 1, 2013, pp. 24–39.

[18] Ravindra, P.; Anyanwu, K. Nesting Strategies for Enabling Nimble MapReduce Dataflows for Large RDF Data. *Int. J. Semantic Web Inf. Syst.*, vol. 10, no. 1, 2014, pp. 1–26.

[19] Ravindra, P.; Anyanwu, K. Scaling Unbound-Property Queries on Big RDF Data Warehouses using MapReduce. In *EDBT*, 2015, pp. 169–180.

[20] Broekstra, J.; Kampman, A.; van Harmelen, F. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *ISWC*, 2002, pp. 54–68.

[21] Decker, S.; Melnik, S.; van Harmelen, F; Fensel, D.; Klein, M.C.A.; Broekstra, J.; Erdmann, M.; Horrocks, I.. The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing*, vol. 4, no. 5, 2000, pp. 63–74.

[22] Beckett, D.J. The Design and Implementation of the Redland RDF Application Framework. *Computer Networks*, vol. 39, no. 5, 2002, pp. 577–588.

[23] Huang, J., Abadi, D.J., Ren, K. Scalable SPARQL Querying of Large RDF Graphs. *PVLDB*, vol. 4, no. 11, 2011, pp. 1123–1134.

[24] Herman. I. Introduction to Semantic Web Technologies. In *SemTech*, 2010, http://www.w3.org/2010/Talks/0622-SemTech-IH/ – material redistributed under the Creative Common License (http://creativecommons.org/licenses/by-nd/3.0/)

[25] Wikipedia, https://www.wikipedia.org/

[26] DBpedia, http://wiki.dbpedia.org/

[27] W3C. RDQL - A Query Language for RDF - W3C Member Submission 9 January 2004. http://www.w3.org/Submission/RDQL/

[28] Gabrilovich, E.; Markovitch, S. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *IJCAI*, 2007, pp. 1606–1611.

[29] Chandramouli, N.; Goldstein, J.; Duan, S. Temporal Analytics on Big Data for Web Advertising. In *ICDE*, 2012, pp. 90–101.

[30] Papailiou, N.; Tsoumakos, D.; Konstantinou, I.; Karras, P.; Koziris, N. H2RDF: Adaptive Query Processing on RDF Data in the Cloud. In *WWW*, 2012, pp. 397–400.

[31] W3C. SPARQL 1.1 Overview - W3C Recommendation 21 March 2013. http://www.w3.org/TR/sparql11-overview/

[32] Przyjaciel-Zablocki, M.; Schätzle, A.; Skaley, E.; Hornung, T.; Lausen, G. Map-Side Merge Joins for Scalable SPARQL BGP Processing. In *CloudCom*, 2013, pp. 631–638.

[33] Kaoudi, Z.; Manolescu, I. RDF in the Clouds: A Survey. *VLDB J.*, vol. 24, no. 1, 2015, pp. 67–91.

[34] Rohloff, K.; Schantz, R.E. High-Performance, Massively Scalable Distributed Systems Using the MapReduce Software Framework: The SHARD Triple-Store. In *PSI EtA*, 2010, art. no. 4

[35] Ladwig, G.; Harth, A. CumulusRDF: Linked Data Management on Nested Key-Value Stores. In *SSWS*, 2011, pp. 30–42.

[36] Gergatsoulis, M.; Nomikos, C.; Kalogeros, E.; Damigos, M. An Algorithm for Querying Linked Data Using Map-Reduce. In *Globe*, 2013, pp. 51–62.

[37] Schätzle, A.; Przyjaciel-Zablocki, M.; Lausen, G. PigSPARQL: Mapping SPARQL to Pig Latin. In *SWIM*, 2011, art. no. 4

[38] Olston, C.; Reed, B.; Srivastava, U.; Kumar, R.; Tomkins, A. Pig Latin: A Not-So-Foreign Language for Data Processing. In *SIGMOD*, 2008, pp. 1099–1110.

[39] Nie, Z.; Du, F.; Chen, Y.; Du, C.; Xu, L. Efficient SPARQL Query Processing in MapReduce through Data Partitioning and Indexing. In *APWeb*, 2012, pp. 628–635.

[40] Du, J.-H.; Wang, H.; Ni, Y.; Yu, Y. HadoopRDF: A Scalable Semantic Data Analytical Engine. In *ICIC*, vol. 2, 2012, pp. 633–641.

[41] Punnoose, R.; Crainiceanu, A.; Rapp, D. Rya: A Scalable RDF Triple Store for the Clouds. In *Cloud-I*, 2012, art. 4.

[42] Agrawal, D.; Das, S.; El Abbadi, A. Big Data and Cloud Computing: Current State and Future Opportunities. In *EDBT*, 2011, pp. 530–533.

[43] Cohen, J.; Dolan, B.; Dunlap, M.; Hellerstein, J.M.; Welton, C. MAD Skills: New Analysis Practices for Big Data. *PVLDB*, vol. 2, no. 2, 2009, pp. 1481–1492.

[44] Apache Pig, https://pig.apache.org/

[45] Cuzzocrea, A.; Bellatreche, L.; Song, I.-Y. Data Warehousing and OLAP over Big Data: Current Challenges and Future Research Directions. In *DOLAP*, 2013, pp. 67–70.

[46] Cuzzocrea, A. Analytics over Big Data: Exploring the Convergence of DataWarehousing, OLAP and Data-Intensive Cloud Infrastructures. In *COMPSAC*, 2013, pp. 481–483.

[47] Leung, C.K.-S.; Cuzzocrea, A.; Jiang, F. Discovering Frequent Patterns from Uncertain Data Streams with Time-Fading and Landmark Models. *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, vol. 8, 2013, pp. 174–196.

[48] Cuzzocrea, A.; Leung, C.K.-S.; MacKinnon, R.K. Mining Constrained Frequent Itemsets from Distributed Uncertain Data. *Future Generation Computer Systems*, vol. 37, 2014, pp. 117–126.

[49] Kune, R.; Konugurthi, P.; Agarwal, A.; Rao, C.R.; Buyya, R. The Anatomy of Big Data Computing. *Software: Practice & Experience*, vol. 46, no. 1, 2016, pp. 79–105.

[50] Tian, Y.; Patel, J.M. TALE: A Tool for Approximate Large Graph Matching. In *ICDE*, 2008, pp. 963–972.

[51] Mammo, M.; Bansal, S.K. Distributed SPARQL over Big RDF Data: A Comparative Analysis Using Presto and MapReduce. In *BigData Congress*, 2015, pp. 33–40.

[52] Schätzle, A.; Przyjaciel-Zablocki, M.; Hornung, T.; Lausen, G. Large-Scale RDF Processing with MapReduce. In *S. Sakr, M. Gaber (eds.) "Large Scale and Big Data", Auerbach Publications*, 2014, pp. 151–182.