Cloud-based RDF data management

Zoi Kaoudi, Ioana Manolescu

Cloud-based RDF data management

Zoi Kaoudi*
IMIS, Athena Research Center
Athens, Greece
zoi.kaoudi@inria.fr

Ioana Manolescu
Inria
France
ioana.manolescu@inria.fr

ABSTRACT

The W3C's Resource Description Framework (or RDF, in short) is set to deliver many of the original semi-structured data promises: flexible structure, optional schema, and rich, flexible URIs as a basis for information sharing. Moreover, RDF is uniquely positioned to benefit from the efforts of scientific communities studying databases, knowledge representation, and Web technologies. As a consequence, numerous collections of RDF data are published, going from scientific data to general-purpose ontologies to open government data, in particular published as part of the Linked Data movement.

Managing such large volumes of RDF data is challenging, due to the sheer size, the heterogeneity, and the further complexity brought by RDF reasoning. To tackle the size challenge, distributed storage architectures are required. Cloud computing is an emerging paradigm massively adopted in many applications for the scalability, fault-tolerance and elasticity features it provides. This tutorial presents the challenges faced in order to efficiently handle massive amounts of RDF data in a cloud environment. We provide the necessary background, analyze and classify existing solutions, and discuss open problems and perspectives.

1. INTRODUCTION

During the past decade, many Semantic Web applications have been using the W3C's Resource Description Framework (or RDF, in short) [26] as their data model. RDF data is organized in graphs consisting of *triples* of the form (s, p, o), stating that the subject node s has the property edge p whose value is the object node o. A key concept for RDF is that of *URIs* or Unique Resource Identifiers; these can be used in either of the s, p and o positions to uniquely refer to some entity or concept. Literals (constants) are also allowed in the o position. RDF allows some limited form of incomplete information through *blank nodes*, standing for unknown constants or URIs; an RDF database may, for instance, state that the *author* of X is *Jane* while the *date* of X is *4/1/2011*, for a given, unknown resource X. This contrasts with standard relational databases where all attribute values are either constants or *null*.

RDF Schema (RDFS) [15] is the ontology language of RDF used for giving meaning to resources, grouping them into concepts and identifying the relationships between these concepts. If an RDF Schema is available, RDF semantics requires considering that the database consists not only of triples explicitly present in the store, but also of a set of *implicit triples* obtained through *reasoning based on an RDF Schema and the RDFS rules*. For instance, assume the RDF database contains the fact that the *studentRegistrationNo* of *Bob* is *12345*, whereas an RDF Schema states that only a *student* can have a *studentRegistrationNo*. Then, the fact that *Bob* is a *student* is implicitly present in the database, and a query asking for all *student* instances in the database must return *Bob*.

The proliferation of RDF-based applications has created the need for systems capable of efficiently storing and querying RDF data. The earliest systems developed within the Semantic Web community include Jena [49] and Sesame [16]. More recently, RDF-based stores have gained interest in the database community as well, as illustrated by the works [34, 1, 48]. However, these works focus mostly on RDF viewed as a relational database on which to evaluate conjunctive queries, and do not consider RDF-specific features such as those related to implicit data. Recently, commercial database management systems also started providing some support for RDF, e.g., IBM DB2 [14].

One could consider RDF as yet another graph model for semi-structured data [40], useful in certain application contexts [45] but too expensive to handle in the general case. However, arguably, RDF has the potential for being the most successful semi-structured data model ever, since it draws on a fundamental tenet of the World Wide Web, namely, that every resource is assigned a single URI, which everyone can use to describe it. RDF also draws on other widely adopted W3C specifications, such as XML for serialization and exchange, namespaces for the interoperability of vocabularies, ontology languages for describing knowledge etc. Thus, it is very well positioned as a format for supporting data exchange over the Web.

A particularly interesting class of applications comes from the Open Data concept that "*certain data should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control*"[1]. Open Data federates players of many roles, from organizations such as business and government aiming to demonstrate transparency and good (corporate) governance, to end users interested in consuming and producing data to share with the others, to aggregators that may build business models around warehousing, curating, and sharing this data [41]. In contrast with Open Data which designates a general philosophy, Linked Data refers to the *"recommended best practice for exposing, sharing, and con-*

*Work done while the author was at Inria Saclay.

[1] `http://en.wikipedia.org/wiki/Open_data`

necting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF" [12]. In practice, Open and Linked data are frequently combined to facilitate data sharing, interpretation, and exploitation [33].

All these applications have led to numerous RDF data collections currently available on the Web. To exploit such large data volumes, distributed architectures are necessary. Past works on distributed RDF query processing and reasoning have relied on peer-to-peer platforms [30, 31] or clustered architectures [21, 25, 35].

Currently, the cloud computing paradigm is massively adopted for the scalability to very large data volumes and the fault-tolerance and elastic resource allocation that it provides. Recently, interest in massively parallel processing has been renewed by the MapReduce proposal [20] and many follow-up works, which aim at solving large-volume data management tasks based in a cloud environment. For these reasons, cloud-based stores are an interesting avenue to explore for handling very large volumes of RDF data.

This tutorial has two objectives.

- First, we will introduce the audience to the basics of RDF data management, including storage, query processing, reasoning and updating. We will achieve this based on the well-known concepts of semistructured data and existing works on RDF processing of the data management community.
- Second, we will present a classification of the existing architectures and tools for handling large volumes of RDF data in a cloud environment, compare their approaches and algorithms and discuss the respective trade-offs. Finally, we plan to draw a list of problems we currently find open and outline promising avenues to answer them.

Previous presentation The proposed tutorial has also been presented as an 1.5-hour seminar in ICDE 2013, under the title "Triples in the clouds". In the present proposal, we revisit the classification of the existing systems, update the material to reflect recent influential proposals and put more emphasis on massively parallel RDF query processing techniques.

2. TUTORIAL OUTLINE

Our proposed tutorial is structured as follows.

2.1 Semistructured data and RDF

We will start by briefly recalling the main principles of *semistructured data* [40, 37], a general concept pioneering many works on complex data management in the database community. The goal is to position RDF as one of the most popular models for semistructured data management currently around, while also acknowledging the contributions previously laid out for the more general model to which it can be traced.

We will provide the necessary *background on RDF processing*. This includes the basic concepts of RDF and RDFS, including their semantics [26]. We will then focus on the Basic Graph Pattern (BGP) queries of SPARQL [38], its *conjunctive fragment* allowing to express the core Select-Project-Join database queries. We will introduce the formal semantics of a BGP query, taking into account also the *implicit* data in an RDF database by the presence of an RDFS (or other flavor of) schema.

We will then consider the *core data management issues* raised by RDF processing, namely: storing (in row- and column-oriented stores), indexing, evaluating and ordering joins, query pattern selectivity estimation, updating, and the impact of reasoning. We will devote particular attention to a pedagogic introduction of the issues related to *reasoning*, since they are often ignored in mainstream database works. We will illustrate with detailed examples and introduce the various techniques proposed in the literature to handle implicit data [2, 31] and their performance trade-offs [24].

At the end of this segment, the audience will have a good grasp of the RDF model and RDF data management issues.

2.2 Cloud-based data management

Interest in massively parallel processing has been renewed recently since the emergence of the MapReduce proposal [20] and its open source implementation Hadoop [8]. MapReduce has become popular in various computer-science fields as it provides a simple programming paradigm which frees the developer from the burden of handling the issues of parallelization, scalability, load balancing and fault-tolerance.

Although MapReduce was first mostly intended for data analysis tasks, it has also started to be used in query processing tasks. However, MapReduce provides simple primitives and more complex operations such as joins are not directly supported. In this part of the tutorial, we will recall the basics of MapReduce and outline the main existing strategies for processing joins in a MapReduce framework [13, 5, 4], since joins are at the heart of SPARQL BGP query processing.

RDFS reasoning can be assimilated to deductive databases and therefore recursive query processing techniques are pertinent. Therefore, we will highlight the connections between cloud-based RDFS reasoning and recursive query processing on top of MapReduce [3, 17].

This part of the tutorial will have introduced the audience to the basic primitives available in the cloud, the strong advantages of MapReduce and the difficulties that remain to be solved for high-level, declarative management of complex data such as RDF.

2.3 State-of-the-art cloud-based RDF systems

The core part of our tutorial will be a comprehensive classification of existing architectures and systems for handling RDF data within a cloud. We will explore the most recent advances of RDF data management in the cloud as well as in parallel/distributed architectures that were not necessarily intended for the cloud, but can easily be deployed therein. We will present the main principles used for: (*i*) organizing the data store, (*ii*) processing conjunctive queries and (*iii*) handling implicit data through reasoning.

A first classification of existing platforms can be made according to their underlying data storage facilities:

- systems relying on a distributed file system, such as HDFS, for warehousing RDF data;
- systems which use existing "NoSQL" key-value stores [19] as back-ends for storing and indexing RDF data;
- systems warehousing RDF data in a "federation" of single-site (centralized) RDF stores, one on each node;
- hybrid systems that use a combination of the above.

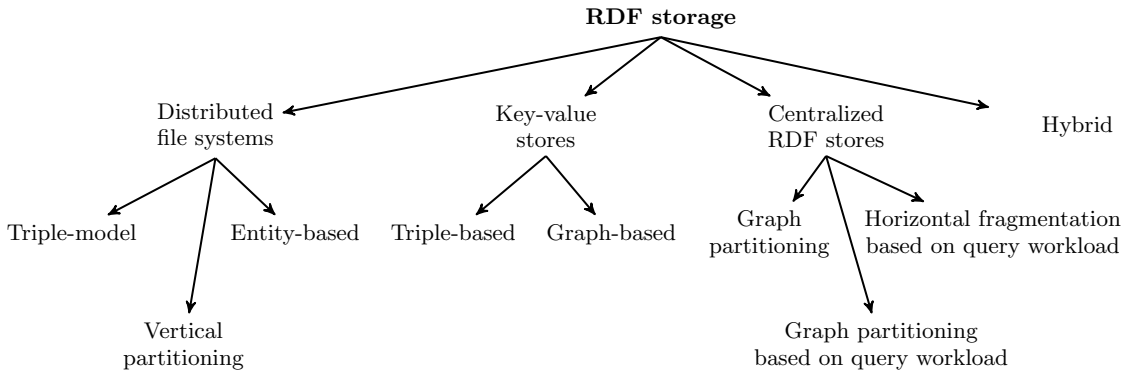

Figure 1: Taxonomy of storage schemes.

Figure 1 shows a more detailed taxonomy of the storage facilities being used in the state-of-the-art, and on which our tutorial will be based. The first core category comprises platforms such as those described in [23, 29, 42]. These systems are built to make the most out of the parallel processing capacities provided by the underlying MapReduce paradigm. However they may be seen disadvantaged from the perspective of the data store, given that they do not have efficient fine-grained data stores to rely on. Representatives of the second category include systems such as Rya [39] which uses Apache Accumulo [6], CumulusRDF [32] based on Apache Cassandra [7], Stratustore [44] which relies on Amazon's SimpleDB [11], and H_2RDF [36], built on top of HBase [9]. These systems benefit from the efficient and fine-grained storage and retrieval of the key-value stores, however suffer in more complex functionalities such as joins. Within the third category, centralized RDF stores distributed among multiple nodes are used to exploit the parallelization offered by the multiple RDF store instances such as in [22, 27, 28]. Finally, in [10, 18] a mixed approach is used with raw data residing in Amazon's storage service (S3), a file index built in Amazon's key-value store and the query answering to be done by a centralized RDF store.

A second important angle of analysis of cloud-based RDF platforms comes from their strategy for processing SPARQL queries. From this perspective, we identify the following main classes:

- systems following a relational-style query processing strategy;
- systems using graph exploration techniques based on the graph structure of the data.

While few works use graph techniques, many are based on relational processing strategies. Works in this category can be classified according to the taxonomy of Figure 2. Data access paths are tightly coupled with the underlying storage facility and we categorize them accordingly: Systems such as [29, 36, 42, 51] use distributed file systems to access the data, [39, 44] rely on key-value stores and typically implementing their own join operators, while works such as [10, 18, 28] take advantage of existing centralized RDF stores. Join evaluation can be classified to works that either use MapReduce or perform the join out of MapReduce, often at a single site.

Trinity.RDF [50], which is built on top of on Microsoft's graph-based in-memory store [43], is the only work that we know so far that belongs to the second category above. It uses graph exploration instead of relational-style joins.

Having toured the issues of RDF storing and querying through the above techniques, we will show how reasoning is handled in these cloud-based systems. From this angle, the options are as follows:

- pre-compute and materialize all implicit triples;
- compute the necessary implicit triples at query time;
- some hybrid approach among the two above, with some implicit data computed statically and some at query time.

Most recent proposals on RDF reasoning in cloud environments come from the Semantic Web community [46, 47] but do not integrate it with the querying phase. The only work from the above that injects reasoning within RDF query processing is [29].

This part of the tutorial will provide a principled categorization of existing works and point out the strengths and limitations of various proposals.

2.4 Open issues

In the final part of the tutorial we will draw a list of problems we currently find open and which we believe that will attract significant interest in the near future. These range from RDF query optimization and RDF updates in a cloud environment, as well as RDF view materialization and sharing of intermediate results in the distributed environment of the cloud.

Targeted Audience The intended audience consists of database students and researchers with an interest in RDF data management, cloud-based data processing, or both. We will provide brief self-sufficient introductions to both these areas to enable non-specialists to follow. In particular, while RDF querying has been well explored in database-oriented works, we will devote some time to the issues involved in RDFS reasoning, which have not been considered in these works.

Duration The proposed tutorial is intended for a length of 3 hours. It can also be shortened to 1.5 hours, by omitting

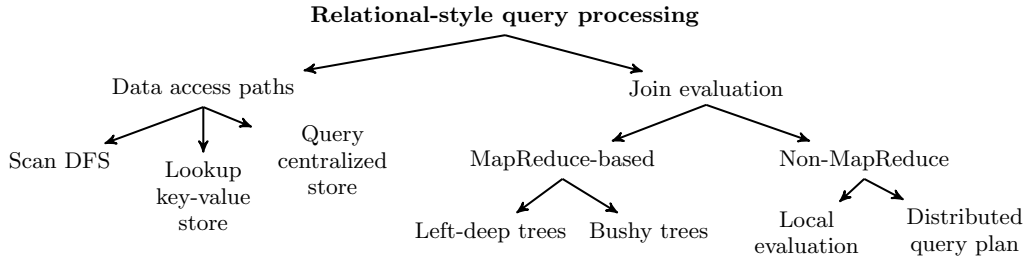

Figure 2: Taxonomy of relational-style query processing strategies.

the connection with semi-structured data and RDF indexing, shortening the presentation of cloud-based data storage, and possibly omitting the details of join evaluation. The shortened tutorial will thus feature reduced versions of Sections 2.1 and 2.2, while preserving most of the content from the core part, corresponding to Section 2.3.

Acknowledgments We have studied the material on which the tutorial is based, while collaborating with many colleagues. We wish to thank in particular François Goasdoué, Alexandra Roatiş and Jorge Quiané-Ruiz for many insightful discussions.

3. PRESENTERS

Zoi Kaoudi received her PhD from the National and Kapodistrian University of Athens in 2011, after graduating from the school of Electrical and Computer Engineering of the National Technical University of Athens. In 2011 she joined the OAK team of Inria Saclay as a postdoctoral researcher for two years. She is now a research associate at the Institute for the Management of Information Systems of the Athena Research Center in Athens, Greece. Her research interests include cloud-based management of Web data, distributed RDF query processing and reasoning. Personal webpage: `http://pages.saclay.inria.fr/zoi.kaoudi/`

Ioana Manolescu received her PhD from Inria and Université de Versailles Saint-Quentin in 2001, after graduating from Ecole Normale Supérieure in Paris. Ioana has been a post-doc at Politecnico di Milano, Italy, then she joined Inria where she is now senior researcher and the leader of the OAK team, specialized in database optimization techniques for complex, large data. Her research interests include algebraic optimizations, adaptive storage and efficient management of semantically rich data and cloud-based management of Web data. She has previously presented tutorials in VLDB (2003), ICDE (2005 and 2008), and the EDBT summer school (2002 and 2004). Personal webpage: `http://pages.saclay.inria.fr/ioana.manolescu/`

4. REFERENCES

[1] D. J. Abadi, A. Marcus, S. Madden, and K. Hollenbach. SW-Store: a vertically partitioned DBMS for Semantic Web data management. *VLDB J.*, 18(2):385–406, 2009.

[2] S. Abiteboul, I. Manolescu, P. Rigaux, M.-C. Rousset, and P. Senellart. *Web Data Management and Distribution*. Cambridge University Press, Dec 2011.

[3] F. N. Afrati, V. R. Borkar, M. J. Carey, N. Polyzotis, and J. D. Ullman. Map-Reduce Extensions and Recursive Queries. In *EDBT*, 2011.

[4] F. N. Afrati and J. D. Ullman. Optimizing Joins in a Map-Reduce Environment. In *EDBT*, 2010.

[5] F. N. Afrati and J. D. Ullman. Optimizing Multiway Joins in a Map-Reduce Environment. *IEEE Trans. Knowl. Data Eng.*, 23(9), 2011.

[6] Apache Accumulo. `http://accumulo.apache.org/`, 2012.

[7] Apache Cassandra. `http://cassandra.apache.org/`, 2012.

[8] Apache Hadoop. `http://hadoop.apache.org/`, 2012.

[9] Apache HBase. `http://hbase.apache.org/`, 2012.

[10] A. Aranda-Andújar, F. Bugiotti, J. Camacho-Rodríguez, D. Colazzo, F. Goasdoué, Z. Kaoudi, and I. Manolescu. Amada: Web Data Repositories in the Amazon Cloud (demo). In *CIKM*, 2012.

[11] Amazon Web Services. `http://aws.amazon.com/`, 2012.

[12] T. Berners-Lee. Linked data - design issues. `http://www.w3.org/DesignIssues/LinkedData.html`, 2006.

[13] S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J. Shekita, and Y. Tian. A Comparison of Join Algorithms for Log Processing in MapReduce. In *SIGMOD*, 2010.

[14] M. A. Bornea, J. Dolby, A. Kementsietsidis, K. Srinivas, P. Dantressangle, O. Udrea, and B. Bhattacharjee. Building an efficient RDF store over a relational database. In *SIGMOD*, 2013.

[15] D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, W3C Recommendation, 2004.

[16] J. Broekstra and A. Kampman. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *1st International Semantic Web Conference (ISWC)*, volume 2342 of *LNCS*, 2002.

[17] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst. The HaLoop Approach to Large-Scale Iterative Data Analysis. *VLDB J.*, 21(2), 2012.

[18] F. Bugiotti, F. Goasdoué, Z. Kaoudi, and I. Manolescu. RDF Data Management in the Amazon Cloud. In *DanaC Workshop (in conjunction with EDBT)*, 2012.

[19] R. Cattell. Scalable SQL and NoSQL data stores. *SIGMOD Record*, 39(4):12–27, May 2011.

[20] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, 2004.

[21] O. Erling and I. Mikhailov. RDF Support in the Virtuoso DBMS. *Networked Knowledge - Networked Media*, 2009.

[22] L. Galarraga, K. Hose, and R. Schenkel. Partout: A Distributed Engine for Efficient RDF Processing. Technical Report: CoRR abs/1212.5636, 2012.

[23] F. Goasdoué, Z. Kaoudi, I. Manolescu, J. Quiané-Ruiz, and S. Zampetakis. CliqueSquare: efficient Hadoop-based RDF query processing. In *Bases de Données Avancées (BDA)*, 2013.

[24] F. Goasdoué, I. Manolescu, and A. Roatis. Efficient query answering against dynamic RDF databases. In *EDBT*, 2013.

[25] S. Harris, N. Lamb, and N. Shadbolt. 4store: The Design and Implementation of a Clustered RDF Store. In *5th International Workshop on Scalable Semantic Web Knowledge Base Systems*, 2009.

[26] P. Hayes. RDF Semantics. W3C Recommendation, February 2004. `http://www.w3.org/TR/rdf-mt/`.

[27] K. Hose and R. Schenkel. WARP: Workload-Aware Replication and Partitioning for RDF. In *DESWEB Workshop (in conjunction with ICDE)*, 2013.

[28] J. Huang, D. J. Abadi, and K. Ren. Scalable SPARQL Querying of Large RDF Graphs. *PVLDB*, 4(11):1123–1134, 2011.

[29] M. Husain, J. McGlothlin, M. M. Masud, L. Khan, and B. M. Thuraisingham. Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing. *IEEE TKDE*, 2011.

[30] Z. Kaoudi, K. Kyzirakos, and M. Koubarakis. SPARQL Query Optimization on Top of DHTs. In *ISWC*, 2010.

[31] Z. Kaoudi, I. Miliaraki, and M. Koubarakis. RDFS Reasoning and Query Answering on Top of DHTs. In *ISWC*, 2008.

[32] G. Ladwig and A. Harth. CumulusRDF: Linked Data Management on Nested Key-Value Stores. In *SSWS*, 2011.

[33] State of the LOD cloud. Available from: `http://www4.wiwiss.fu-berlin.de/lodcloud/state/`, 2011.

[34] T. Neumann and G. Weikum. The RDF-3X Engine for Scalable Management of RDF Data. *VLDBJ*, 19(1), 2010.

[35] A. Owens, A. Seaborne, N. Gibbins, and M. Schraefel. Clustered TDB: A Clustered Triple Store for Jena. Technical Report, 2008.

[36] N. Papailiou, I. Konstantinou, D. Tsoumakos, P. Karras, and N. Koziris. H_2RDF+: High-performance Distributed Joins over Large-scale RDF Graphs. In *IEEE BigData*, 2013.

[37] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *ICDE*, 1995.

[38] E. Prud'hommeaux and A. Seaborn. SPARQL Query Language for RDF. W3C Recommendation, `http://www.w3.org/TR/rdf-sparql-query/`, 2008.

[39] R. Punnoose, A. Crainiceanu, and D. Rapp. Rya: A Scalable RDF Triple Store for the Clouds. In *1st International Workshop on Cloud Intelligence (in conjunction with VLDB)*, 2012.

[40] D. Quass, J. Widom, R. Goldman, K. Haas, Q. Luo, J. McHugh, S. Nestorov, A. Rajaraman, H. Rivero, S. Abiteboul, J. Ullman, and J. Wiener. Lore: a lightweight object repository for semistructured data. In *SIGMOD*, 1996.

[41] G. Raschia, M. Theobald, and I. Manolescu. Proceedings of the first international workshop on open data (WOD). 2012.

[42] K. Rohloff and R. E. Schantz. High-Performance, Massively Scalable Distributed Systems using the MapReduce Software Framework: the SHARD Triple-Store. In *Programming Support Innovations for Emerging Distributed Applications*, 2010.

[43] B. Shao, H. Wang, and Y. Li. The Trinity Graph Engine. Technical report, `http://research.microsoft.com/pubs/161291/trinity.pdf`, 2012.

[44] R. Stein and V. Zacharias. RDF On Cloud Number Nine. In *Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic*, May 2010.

[45] S. Trißl and U. Leser. Fast and practical indexing and querying of very large graphs. In *SIGMOD*, 2007.

[46] J. Urbani, S. Kotoulas, E. Oren, and F. van Harmelen. Scalable Distributed Reasoning using MapReduce. In *8th International Semantic Web Conference (ISWC)*, 2009.

[47] J. Urbani, F. van Harmelen, S. Schlobach, and H. Bal. QueryPIE: Backward Reasoning for OWL Horst over Very Large Knowledge Bases. In *10th International Semantic Web Conference (ISWC)*, 2011.

[48] C. Weiss, P. Karras, and A. Bernstein. Hexastore: sextuple indexing for semantic web data management. *PVLDB*, 1(1):1008–1019, 2008.

[49] K. Wilkinson, C. Sayers, H. A. Kuno, and D. Raynolds. Efficient RDF Storage and Retrieval in Jena2. In *1st International Workshop on Semantic Web and Databases (SWDB), in conjunction with VLDB*, 2003.

[50] K. Zeng, J. Yang, H. Wang, B. Shao, and Z. Wang. A Distributed Graph Engine for Web Scale RDF Data. In *PVLDB 2013*.

[51] X. Zhang, L. Chen, Y. Tong, and M. Wang. EAGRE: Towards Scalable I/O Efficient SPARQL Query Evaluation on the Cloud. In *ICDE*, 2013.