On the Support of Ad-Hoc Semantic Web Data Sharing

Jing Zhou, Kun Yang, Lei Shi, Zhongzhi Shi

On the Support of Ad-Hoc Semantic Web Data Sharing

Jing Zhou[1,2], Kun Yang[1], Lei Shi[1], and Zhongzhi Shi[2]

[1] School of Computer Science,
Communication University of China, Beijing, 100024, China
{zhoujing,yangkun,shilei_cs}@cuc.edu.cn
[2] The Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, 100190, China
shizz@ics.ict.ac.cn

Abstract. Sharing Semantic Web datasets provided by different publishers in a decentralized environment calls for efficient support from distributed computing technologies. Moreover, we argue that the highly dynamic ad-hoc settings that would be pervasive for Semantic Web data sharing among personal users in the future pose even more demanding challenges for enabling technologies. We propose an architecture that is based upon the peer-to-peer (P2P) paradigm for ad-hoc Semantic Web data sharing and identify the key technologies that underpin the implementation of the architecture. We anticipate that our (current and future) work will offer powerful support for sharing of Semantic Web data in a decentralized manner and becomes an indispensable and complementary approach to making the Semantic Web a reality.

Keywords: Semantic Web, RDF triples, storage organization, data sharing, decentralized query processing

1 The Semantic Web, Data, and Data Storage

The Semantic Web was intended to transform heterogeneous and distributed data into the form that machines can directly process and manipulate by providing the data with explicit semantic information, thus facilitating *sharing and reusing of data across applications*. To make the Semantic Web a reality, one important way is publishing a large amount of data encoded in standard formats on the Web. These data and the associations between them are both essential for supporting large scale data integration and the automated or semi-automated querying of data from disparate sources.

In the early days of the Web, documents were connected by means of hyperlinks, or *links*, and data was not provided independently of their representation in the documents. It is obvious that the ultimate objective of the Semantic Web calls for a shift of the associative linking from documents to data [7]–the Semantic Web turns into the Web of Linked Data whereby machines can explore and locate related data.

Virtually, in the Semantic Web one can create links between any Web data that is identified by Uniform Resource Identifier (URI) references with the Resource Description Framework (RDF) [13] being utilised to describe the association between the data that the link represents. The RDF is a standard model for exchanging data on the Web as well as a language for describing Web resources or anything that is identifiable on the Web. The basic RDF data model adopts triples in the form of (*subject, predicate, object*) to depict the attributes of Web resources. RDF triples are also referred to as RDF statements and stored in data stores known as *triplestores*.

A triplestore may contain millions of triples and is mainly responsible for not only storage but also efficient reasoning over and retrieving triples in it. Typically, triplestores come in three forms: (1) RDF triples are stored in relational database management systems (DBMSs), (2) an XML database is employed to store RDF data, or (3) a proprietary information repository is developed to accommodate RDF triples. Among others, the practice of building triplestores on top of relational DBMSs has gained wide adoption thanks to their powerful transaction management. The underlying relational DBMSs deal with the storage and organization of RDF triples in one of the following schemes: triple tables, property tables [19], and vertical partitioning [1]. For a more detailed discussion of individual schemes, interested readers may refer to [5].

2 The Development of Triplestores

Triplestores provide storage and management services to Semantic Web data and hence are able to support data sharing in specialized Semantic Web applications including Semantic Web search engines [10], personal information protection products [9], and wikis. In this section, we will present several typical systems and focus on their logical storage, data description, and query processing mechanisms in particular.

2.1 Single Machine Supported Triplestores

Sesame [3] was developed as a generic architecture for storage and querying of large quantities of Semantic Web data and it allowed the persistent storage of RDF data and schema information to be provided by any particular underlying storage devices. The Storage And Inference Layer (SAIL) was a single architectural layer of Sesame that offered RDF-specific methods to its clients and translated these methods to calls to its certain DBMS. A version of RQL (RDF Query Language) [12] was implemented in Sesame that supported querying of RDF and RDFS [2] documents at the semantic level.

Other single machine supported triplestores include 3store [8] that adopted a relational DBMS as its repository for storage and Kowari[3], an entirely Java-based transactional, permanent triplestore.

[3] See `http://www.xml.com/pub/a/2004/06/23/kowari.html`.

2.2 Centralised Triplestores

YARS2 [10] was a distributed system for managing large amounts of graph-structured RDF data. RDF triples were stored as quadruple (*subject, predicate, object, context*) with context indicating the URL of the data source for the contained triple. An inverted text index for keyword lookups and the sparse index-based quad indices were established for efficient evaluation of queries. For scalability reason, the quad index placement adopted established distributed hash table substrates that provided directed lookup to machines on which the quadruple of interest could be located. The Index Manager on individual machines provided network access to local indices such as keyword indices and quad indices. The Query Processor was responsible for creating and optimising the logical plan for answering queries. It then executed the plan over the network by sending lookup requests to and receiving response data from the remote Index Managers in a multi-threaded fashion.

4store [9] was implemented on a low-cost networked cluster with 9 servers and its hardware platform was built upon a shared-nothing architecture. RDF triples were represented as quads of (*model, subject, predicate, object*). All the data was divided into a number of non-overlapping segments according to the RIDs (Resource IDentifiers) that were calculated for the subject of any given RDF triple. Each Storage Node maintained one or more data segments. To allow the Processing Node to discover the data segments of interest, each Storage Node in a local network advertised its service type, contained dataset with a unique name, and the total number of data segments. Hence, the Processing Node could locate the Storage Nodes that host the desired data segments by checking the advertised messages and answer queries from the client.

Furthermore, DARQ [15] and [11] utilising Hadoop and MapReduce are among the systems that process RDF data queries in a centralised way.

2.3 Fully Distributed Triplestores

Piazza [6] was one the few unstructured P2P systems that supported data[4] management in Semantic Web applications. Each node provided source data with its schema, or only a schema (or ontology) to which schemas of other nodes could be mapped. Point-to-point mappings (between domain structures and document structures) were supported by Piazza whereby nodes could choose other nodes with which they would like to establish semantic connections. One of the roles that mappings in Piazza played was serving as storage descriptions that specified what data a node actually stored. A flooding-like technique was employed to process queries and the designers claimed that they focused mostly on obtaining semantically correct answers. The prototype system of Piazza comprised 25 nodes.

[4] At the time of writing of [6], most Semantic Web data were encoded in the XML format and hence Piazza mainly dealt with the issues arising from XML data management.

RDFPeers [4] was a distributed and scalable RDF repository. Each triple in RDFPeers was stored at three places in a multi-attribute addressable network (MAAN) by applying hash functions to the subject, predicate, and object values of the triple. Hence, data descriptions were given implicitly as identifiers in the one-dimensional modulo-2^m circular identifier space of MAAN. MAAN not only supported exact-match queries on single or multiple attributes but also could efficiently resolve disjunctive and range queries as well as conjunctive multi-attribute queries. RDFPeers was demonstrated to provide good scalability and fault resilience due to its roots in Chord [18].

S-RDF [20] was a fully decentralized P2P RDF repository that explored ways to eliminate resource and performance bottlenecks in large scale triplestores. Each node maintained its local RDF dataset. Triples in the dataset were further grouped into individual RDF data files according to the type of their subject. The Description Generator took an RDF data file as input to generate a description of the file, which could comprise up to several terms, with the help of ontologies. Considering the semantic relationship that might exist between RDF data files hosted by different nodes, S-RDF implemented a semantics-directed search protocol, mediated by topology reorganization, for efficiently locating triples of interest in a fully distributed fashion. Desired scalability was demonstrated through extensive simulations.

2.4 Cloud Computing-based Triplestores

SemaPlorer [16], which was awarded the first prize at the 2008 Semantic Web Challenge Billion Triples Track, was built upon the cloud infrastructure. Semantic data from sources including DBPedia, GeoNames, WordNet, personal FOAF files, and Flickr, was integrated and leveraged in the SemaPlorer application. SemaPlorer allowed users to enjoy, in real-time, blended browsing of an interesting area in different context views. A set of 25 RDF stores in SemaPlorer was hosted on the virtual machines of Amazon's Elastic Computing Cloud (EC2), and the EC2 virtual machine images and the semantic datasets were stored by Amazon's Simple Storage Service (S3). SemaPlorer employed Networked Graphs [17] to integrate semantically heterogeneous data. In theory, any distributed data sources can be integrated into the data infrastructure of SemaPlorer in an ad-hoc manner on a very important premise, that is, such data sources can be located by querying SPARQL (SPARQL Protocol And RDF Query Language) endpoints [14]. However, the same premise does not hold in an ad-hoc settings since peers have no knowledge about the specific locations of these data sources in the absence of a fully distributed query forwarding and processing mechanism.

2.5 Summary

In brief, triplestores that were intended to support data sharing and reusing have shifted from centralised to distributed architecture over time. The inherent reasons for this are manifold. To begin with, every single machine is unable to accommodate all existing (and coming) Semantic Web data and process them in

memory. Then, copying large amounts of distributed data to one machine and processing them in a centralised manner will inevitably result in issues such as increased network traffic, deteriorated query efficiency, and even infrigement of copyrights sometimes. Furthermore, personal users of the future sharing Semantic Web data would seemingly appear the same way that Web users of today share music, video, and image files; the distinguishing features of Semantic Web data, including being structured and given well-defined associations, however, urge us to come up with new distributed computing technologies. An ad-hoc setting in which Semantic Web data sharing should be supported certainly brings about more challenges.

3 Semantic Web Data Sharing – What is Still Missing?

Hall pointed out in [7] that the great success of the early Web is attributed in part to the power of the network effect, that is, more people will join a network to get the benefits as its value increases and meanwhile people contribute more information to the network, thus further increasing its value. The network effect is expected to exhibit in the Semantic Web and promote its success. Hence, research communities and organizations were encouraged to publish and share data that can be mapped onto URIs and links to other data, thus increasing the value of the data and the Semantic Web. Ever since, triplestores maintained by individual publishers for storing RDF data from particular domains came into existence. These data stores provide generic features such as data access, retrieval, and deletion. Meanhile, system designers manage to increase the scalability and robustness of the data stores through enabling techniques to deal with the ever-increasing volume of Semantic Web data.

For Semantic Web data sharing among different publishers, querying is an indispensable mechanism. Regardless of the paradigm in which RDF data is stored (centralised or distributed), existing query mechanisms are rather simple and most assume that the target data is within two hops away, that is, the data can be simply located by directly interrogating some central directory node[5].

For more complex scenarios, for example a fully distributed ad-hoc environment in which the target data may be more than two hops away, Semantic Web researchers have yet to supply an efficient querying solution. Approaches from the P2P computing community, in the meantime, generally do not take into account the inherent semantic associations between Semantic Web data, and hence the quality of the query results they yield has room for improvement.

Another important factor that makes supporting ad-hoc Semantic Web data sharing a pressing issue, relates to the paradigm in which people will share RDF data on their personal computers sometime in the future. As RDF converters are becoming available for many kinds of application data, we anticipate that

[5] Triplestores built upon distributed hash tables are an exception to this but still not applicable to solving our problem because of the data placement issue, that is, any single RDF triple needs to be placed at some location. In our scenario, the tripletore does not have the issue, which excludes the use of distributed hash tables.

large amounts of RDF data will be generated in personal computers. These data might be references to literature (e.g. BibTex), information in spreadsheets (e.g. Excel), or even data from the GPS (Global Positioning System) receiver. They would be carried around and shared with others at will–just like what we do with document files, music files, or video files in our computers. In most cases, Semantic Web data sharing among personal computers will typically occur in an ad-hoc environment[6] where querying becomes much more complicated in the absence of a central directory node.

Though it has long been recognised and followed that publishing and sharing large amounts of RDF data on the Web is an important approach to making the Semantic Web a reality, we argue that providing efficient techniques and technologies to support ad-hoc Semantic Web data sharing, which is currently missing from the overall approach, is a complementary and indispensable solution the importance of which should never be underestimated. In an ad-hoc environment, Semantic Web data will only be shared in a desired way if efficient mechanisms for describing, manipulating, querying, and analysing data are developed.

4 An Architecture for Ad-Hoc Semantic Web Data Sharing

In this section, we will present the main components in an architecture for ad-hoc Semantic Web data sharing based on the P2P paradigm. This is followed by describing the way that all the components in the architecture collaborate to resolve a query for RDF data in an ad-hoc scenario.

4.1 Building an Architecture on Top of Chord

Our decentralized RDF data repository consists of many individual nodes or peers (each representing a user and her client program) and extends Chord [18] with RDF-specific retrieval techniques. We assume that some of the nodes are willing to host indices for other nodes and self-organize into a Chord ring. Nodes that are reluctant to do so will need to be attached to one of the nodes on the ring. In Fig. 1 we show a peer network of five nodes[7] in a 4-bit identifier space. Node identifiers N1, N4, N7, N12, and N15 correspond to actual nodes that are willing to host indices for other peers. In the meantime, node identifiers D1, D2, and D3 represent three actual nodes that are attached to N12 and share information about their RDF data with N12. Hence, N12 becomes associated with all RDF data shared by D1, D2, and D3. For instance, if N4 stores in its finger table an item regarding a pointer to D3 which stores specific RDF triples, the item will only involve node identifier N12 rather than D3.

[6] This is very much like the way that Internet users share music and video files in a P2P fashion.

[7] Nodes that are attached to any node on the Chord ring are not taken into account.

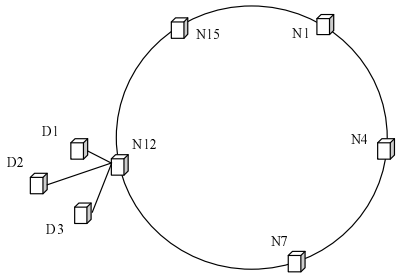

Fig. 1. A peer network of five nodes in a 4-bit identifier space

Each node has an IP address by which it may be contacted. The overall index will be spread across the nodes on the ring. Unlike RDFPeers [4], our system applies hash functions to the subject (s), predicate (p), object (o), subject and predicate (sp), predicate and object (po), and subject and object (so) types of an RDF triple and stores the information (a pointer to the node, for instance) about the node that shares the triple at six places on a Chord ring.

As shown in Fig. 2, each node on the Chord ring is comprised of seven components: the RDF Converter, SPARQL Query Engine, RDF Data Repository, Pre- and Post-processor, Query Rewriter, Query Forwarder, and Result Merger.

Both the RDF Converter and the SPARQL Query Engine can be obtained by downloading from the Web and some coding is needed for enabling them to interact with other components in the architecture. The RDF Data Repository, or the triplestore, is assumed to be implemented by XML databases, and then is mainly responsible for data storage.

The functionality of the other components is described as follows.

- The Pre- and Post-processor accepts incoming queries from either the external application or any other peer in the P2P network. It directs queries for RDF triples to the Query Rewriter for further processing. When the targe RDF triples are finally obtained, the result will be transmitted to the Pre- and Post-processor for proper presentation to the query originator.
- The Query Rewriter is responsible for reformulating the queries from the Pre- and Post-processor whenever needed. For instance, on the receipt of a query that specifies a set of disjunctive ranges for attribute (subject, predicate, or object) values, the Query Rewriter may create several sub-queries for each attribute value if the targe triples are hosted by remote nodes. It may also transform a query into the SPARQL format for resolving by the SPARQL Query Engine.
- The Query Forwarder receives queries from the Query Rewriter and, according to the query forwarding protocol, will pass on the queries to corresponding target nodes.
- The Result Merger collects and compiles query results regarding all the remote nodes that may contain targe RDF triples and then returns the result

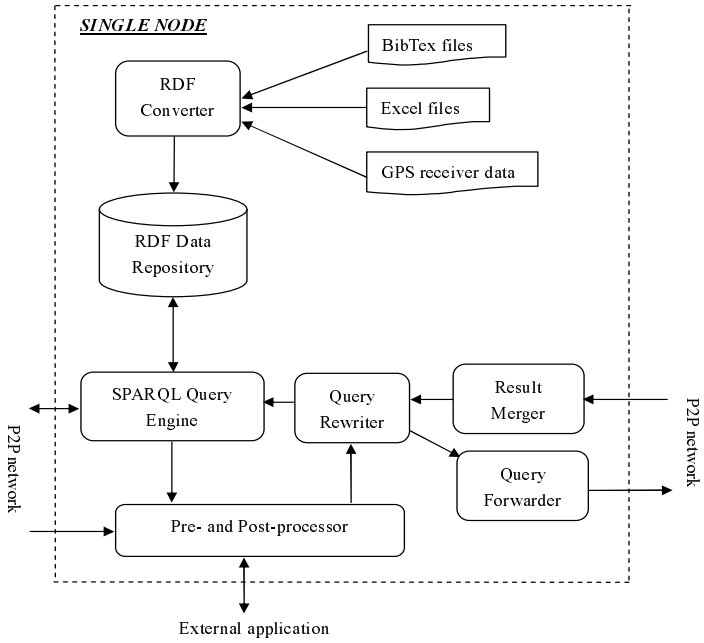

Fig. 2. Single node in ad-hoc Semantic Web data sharing system

to the Query Rewriter. The latter will rewrite the original query in the SPARQL format with explicit information on the named graphs that will be interrogated by the SPARQL query.

4.2 Resolving a Query for RDF Data in an Ad-Hoc Scenario

Data preparation and distributed index building On any single node, application-specific data from personal files can be transformed into RDF triples by the RDF Converter in batches and then inserted into the RDF Data Repository. To support efficient queries on decentralized RDF triples, we exploit the overlay structure of Chord to build a distributed index for locating these triples. We store pointers to the node that maintains a specific RDF triple six times, each obtained by applying hash functions to the class type of the subject, predicate, object, subject and predicate, subject and object, and predicate and object of the triple. Each pointer will be stored at the successor node of the hash key of the corresponding attribute type combination.

Decentralized query processing Generally, resolving a query for remote RDF triples in the proposed network is divided into two phases. In the first phase[8], a query from the external application is submitted to a single node

[8] Note that during the first phase, all queries (except the original one) processed by the Query Rewriter, Query Forwarder, and Result Merger are meant to discover the

which will instruct its Pre- and Post-processor to accept the incoming query. The query is further passed onto the Query Rewriter which will reformulate the query when needed and then send it to the Query Forwarder. The Query Forwarder determines the target node to which the query should be routed. When the Result Merger obtains the query result about the potential target nodes in the network, it will provide the information to the Query Rewriter. The Query Rewriter retrieves the original query, reformulates it in the SPARQL format using the information from the Result Merger, and submits it to the SPARQL Query Engine. In the second phase, the SPARQL Query Engine resolves the SPARQL query by querying against all relevant named graphs. The SPARQL query result containing the RDF data of interest is returned to the Pre- and Post-processor for presentation to the external application.

5 Conclusions and Future Work

We proposed in this position paper a P2P architecture for Semantic Web data sharing in an ad-hoc environment. To fully support the implementation of the architecture, we also need to address the following issues in the future work: (1) decentralized query processing mechanisms that resolve exact-match queries, disjunctive queries, conjunctive queries, and range queries for RDF data and (2) enhancement techniques for RDF data query performance by taking advantage of the semantic relationship between Semantic Web data and by dynamically changing the network topology based on the local knowledge of peer nodes.

Acknowledgments. The first author is grateful for the discussion with Prof. Gregor v. Bochmann from Ottawa University, Canada. This work is funded by China Postdoctoral Science Foundation (No.20100470557). We would also like to acknowledge the input of the National Natural Science Foundation of China (No. 61035003, 60933004, 60970088, 60903141, 61072085, 61103198), National High-tech R&D Program of China (863 Program) (No.2012AA011003), National Science and Technology Support Program2012BA107B02).

References

1. Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K.: Scalable Semantic Web Data Management Using Vertical Partitioning. In: VLDB'07 the 33rd International Conference on Very Large Data Bases, pp.411–422. VLDB Endowment (2007)
2. Brickley, D. and Guha, R.: Rdf vocabulary description language 1.0: Rdf schema. World Wide Web Consortium, `http://www.w3.org/TR/rdf-schema/` (2004)
3. Broekstra, J., Kampman, A., and van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: the 2nd International Semantic Web Conference, pp.54–68. Springer-Verlag London, UK (2002)

potential peers that may contain the RDF data of interest. The target RDF data is obtained in the second phase by issuing a SPARQL query.

4. Cai, M. and Frank, M.: Rdfpeers: a scalable distributed repository based on a structured peer-to-peer network. In: the 13th international conference on World Wide Web, pp.650–657. ACM New York, NY, USA (2004)
5. Du, X. Y., Wang, Y., and Lv, B.: Research and Development on Semantic Web Data Management. J. Softw. 20(11), 2950–2964. Science Press, China (2009)
6. Halevy, A. Y., Ives, Z. G., Mork, P., and Tatarinov, I.: Piazza: Data management infrastructure for Semantic Web applications. In: the 12th International Conference on World Wide Web, pp.556–567. ACM New York, NY, USA (2003)
7. Hall, W.: The Ever Evolving Web: The Power of Networks. International Journal of Communication 5, 651–664 (2011)
8. Harris, S. and Gibbins, N.: 3store: Efficient bulk RDF storage. In: the 1st International Workshop on Practical and Scalable Semantic Web Systems, pp.1–15 (2003)
9. Harris, S., Lamb, N., and Shadbolt, N.: 4store: the Design and Implementation of a Clustered RDF store. In: the 5th International Workshop on Scalable Semantic Web Knowledge Base Systems, pp.94–109 (2009)
10. Harth, A., Umbrich, J., Hogan, A., and Decker, S.: YARS2: A Federated Repository for Querying Graph Structured Data from the Web. In: ISWC'07/ASWC'07: the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference, pp.211–224. Springer-Verlag Berlin, Heidelberg (2007)
11. Husain, M. F., Doshi, P., Khan, L., and Thuraisingham, B.: Storage and retrieval of large RDF graph using Hadoop and MapReduce. In: the 1st International Conference on Cloud Computing, pp.680–686. Springer-Verlag Berlin, Heidelberg (2009)
12. Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., and Scholl, M.: RQL: A Declarative Query Language for RDF. In: the 11th International Conference on World Wide Web (WWW '02), pp.592–603. ACM New York, NY, USA (2002)
13. Klyne, G. and Carroll, J. J. Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report, W3C Recommendation (2004)
14. Prudhommeaux, E. and Seaborne, A.: Sparql query language for rdf. W3C Recommendation, `http://www.w3.org/TR/rdf-sparql-query/` (2004)
15. Quilitz, B. and Leser, U.: Querying Distributed RDF Data Sources with SPARQL. In: the 5th European semantic web conference on the semantic web: research and applications, pp.524–538. Springer-Verlag Berlin, Heidelberg (2008)
16. Schenk, S., Saathoff, C., Staab, S., and Scherp, A.: SemaPlorer-Interactive Semantic Exploration of Data and Media based on a Federated Cloud Infrastructure. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 7(4), 298–304. Elsevier Science Publishers B. V. Amsterdam, The Netherlands (2009).
17. Schenk, S. and Staab, S.: Networked Graphs: A Declarative Mechanism for SPARQL Rules, SPARQL Views and RDF Data Integration on the Web. In: the 17th International Conference on World Wide Web, pp.585–594. ACM New York, NY, USA (2008)
18. Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for Internet applications. In: the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp.149–160. ACM New York, NY, USA (2001)
19. Wilkinson, K., Sayers, C., Kuno, H. A., and Reynolds, D.: Efficient RDF Storage and Retrieval in Jena 2. In: the 1st International Workshop on Semantic Web and Databases, pp.131–150 (2003)
20. Zhou, J., Hall, W., and Roure, D. D.: Building a distributed infrastructure for scalable triple stores. J. Comput. Sci. Tech. 24(3), 447–462 (2009)