Scalable Semantics – the Silver Lining of Cloud Computing

Andre Newman, Yuan-Fang Li and Jane Hunter

School of ITEE, The University of Queensland
4072 Queensland, Australia
{anewman,liyf,jane}@itee.uq.edu.au

Abstract

Semantic inferencing and querying across large-scale RDF triple stores is notoriously slow. Our objective is to expedite this process by employing Google's MapReduce framework to implement scale-out distributed querying and reasoning. This approach requires RDF graphs to be decomposed into smaller units that are distributed across computational nodes. RDF Molecules appear to offer an ideal approach – providing an intermediate level of granularity between RDF graphs and triples. However, the original RDF molecule definition has inherent limitations that will adversely affect performance. In this paper, we propose a number of extensions to RDF molecules (hierarchy and ordering) to overcome these limitations. We then present some implementation details for our MapReduce-based RDF molecule store. Finally we evaluate the benefits of our approach in the context of the BioMANTA project – an application that requires integration and querying across large-scale protein-protein interaction datasets.

Keywords: RDF, RDF molecules, MapReduce, distributed processing, data integration

1. Introduction

Semantic Web technologies such as RDF, OWL and SPARQL offer significant potential as technologies designed to support the integration of and reasoning across heterogeneous, disparate data sources. The widespread adoption of these technologies is being driven by the need to answer complex queries that demand the integration and processing of multiple related, but disparate, multidisciplinary datasets. Datasets from disciplines including environmental sciences, biological sciences, social sciences, life sciences and health care sciences have been employing these technologies to facilitate data correlation, integration and reasoning.

However, despite the widespread adoption of RDF, OWL and SPARQL within many disciplines and applications, there remain two major challenges to the seamless integration of large-scale distributed datasets:

1. Efficient scalable RDF querying and reasoning;
2. Object co-identification – identifying when entries across datasets are the same.

These issues are particularly problematic within the life sciences domain that typically involves multiple, large-scale datasets generated by independent organizations and communities. The W3C's Semantic Web Health Care and Life Sciences Interest Group (HCLSIG) [1] recently identified co-identification and poor reasoning performance as two of the greatest challenges to the adoption of Semantic Web technologies in the life sciences [21].

1.1. Distributed, Real-time Processing of Large-scale RDF Data

Large-scale data integration places high demands on processing, storage and querying. Distributed processing in a clustered environment offers a low cost, high performance approach to processing massive amounts of RDF instance data (billions or trillions of triples).

Cloud computing, in particular Google's MapReduce architecture [7],provides mechanisms to support distributed processing over extremely large datasets using a cluster of commodity-grade hardware. Data is broken into smaller units and each compute node processes its local copy of data. These results are then combined to obtain the complete answer. MapReduce has been successfully deployed within Google on a number of large-scale tasks including the indexing of web pages and has been shown to be a highly reliable,

[1] http://www.w3.org/2001/sw/hcls/

978-0-7695-3535-7/08 $25.00 © 2008 IEEE
DOI 10.1109/eScience.2008.23

scalable and economical architecture. **Our aim is to investigate methods by which Map-Reduce could be used to expedite querying and reasoning over large-scale RDF triple stores.**

Hadoop[2] is an open-source software platform that implements the MapReduce architecture. It is used by Yahoo!, IBM, Facebook and Amazon and can be used on Amazon's Elastic Compute Cloud (EC2)[3]. We propose a Hadoop-based, distributed RDF molecule store that:

- Breaks an RDF graph into smaller units that can be distributed and indexed across nodes in a cluster;
- Queries each node in the cluster using SPARQL as the query language;
- Merges the query results from each node to generate the search results.

RDF molecules [8] enable lossless decomposition and merging of RDF graphs, while maintaining RDF semantics, suitable for distributed processing in a MapReduce architecture. They provide "the finest components in which an RDF graph can be decomposed without loss of information" [8]. Hence RDF molecules:

- Provide a method to enable distribution of RDF graphs across compute nodes;
- Enable query results to be aggregated from many nodes;
- Provide a minimal dataset to synchronize graph modifications;
- Provide a way to differentiate blank nodes based on their "context".

In the original definition of RDF molecules, an RDF graph is decomposed into a set of molecules each consisting of a set of triples. However this original design lacks (a) the ability to disambiguate a triple with two blank nodes (subject and object); (b) representation of the structure of triples from the original graph; and (c) the ability to leverage certain efficiencies that are available.

We propose extending the original RDF molecule definition by adding hierarchy and ordering to mitigate the above drawbacks. By having hierarchical, nested molecules, triples with two blank nodes can be differentiated according to their "context" in the enclosing molecule. Moreover, the merging of molecules can be made more efficient by imposing a lexicographical and "groundedness" ordering over triples.

1.2. The Co-Identification Problem and Blank Nodes

One of the key challenges for the Semantic Web is the *object co-identification* [10]problem. Data integration is made more complicated because different data sources often use different naming conventions for the same object. A mechanism is needed to identify two equivalent objects and to map between their identifiers. Within the Life Sciences domain, this problem is widely recognized and not easily resolved. The difficulty is that key, large-scale protein databases such as UniProt, DIP [23], IntAct [14] and MPact [11] each employ different naming conventions – both for proteins and their various attributes. A single protein may be annotated with a variety of properties including different accession IDs, labels, its genomic sequence, the host organism, publication information, etc. A protein may participate in interactions with another protein in observed experiments, and be documented in a variety of databases. The harmonization of such databases and their respective ontologies is a significant research challenge for the Semantic Web community and has been the focus of a number of research projects [5, 24, 25]. In particular, previous attempts to standardize naming and identification (e.g., LSIDs [22]) have had limited beneficial impact [9]. We believe that inventing another naming convention or trying to reach a consensus will not solve the identification problem[12]. We reject the idea of creating yet another URI to create a co-reference bundle [13], instead we propose an identity reconciliation process for the "life sciences identifier problem" based on RDF *blank nodes*.

Our approach uses RDF blank nodes to represent real-world entities. A blank node is used to represent a specific protein; and the properties of this protein, including various identifiers from different databases, are modeled as triples with this blank node as the subject. Although RDF blank nodes have previously been demonstrated to provide a useful approach to the object co-identification problem [4], they also introduce a number of associated problems that arise during RDF graph decomposition and merging. The most significant problem is that RDF blank nodes are only uniquely identifiable within their enclosing graph - they are not globally addressable. The implication is that breaking down an RDF graph that contains blank nodes will incur loss of information. Overcoming this problem will require a number of extensions to RDF molecules that are described in detail in Section 3.

1.3. The BioMANTA project

BioMANTA is a collaborative project between Pfizer Research and the University of Queensland that is applying Semantic Web technologies to the modeling of biological pathways and protein-protein interac-

[2] http://hadoop.apache.org/

[3] http://aws.amazon.com/ec2

tion data. It aims to enable *in silico* drug discovery and development by identifying candidate therapeutic targets through the analysis of integrated datasets that relate molecular interactions and biochemical pathways to physiological effects such as toxicology and gene-disease associations. As such, BioMANTA is integrating data from protein datasets such as MPact, DIP, IntAct and MINT [3] via a common model/ontology, the BioMANTA OWL-DL ontology[6, 19]. Conforming to the BioMANTA ontology, protein datasets are converted to RDF instances and stored in a distributed RDF triple store where they are available for subsequent analysis and querying.

Figure 1 below shows RDF triples about a yeast protein with UniProt ID "Q12522", together with other information such as host species, genomic sequence, external references, etc., that are compliant with our BioMANTA ontology.

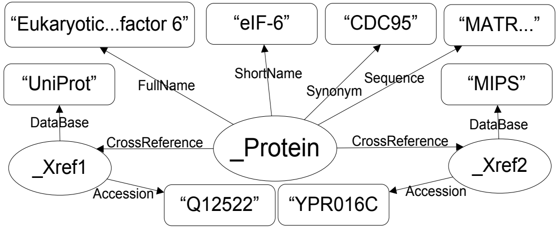

Figure 1. RDF triples about a yeast protein.

The molecular biologists, with whom we are collaborating, want rapid responses to queries such as "Show me all the human kinases expressed in the liver that are strongly inhibited by at least two compounds and are localized to the nucleus". Such queries are potentially very slow to execute as they involve many joins and may generate an RDF graph that exceeds available memory. Consequently, the BioMANTA project provides us with an ideal testbed application and end-user group for evaluating our *Scale-Out RDF Molecule Store*.

1.4. Objectives

The high-level objectives of the work described in this paper are to investigate solutions to the problems of: the ability for an RDF molecule store to decompose, merge and process RDF data, co-identification and semantic querying. The more specific objectives are to investigate and evaluate: (a) methods by which the MapReduce scale-out architecture can be used to improve the performance of semantic querying and inferencing over large-scale RDF triples; (b) the adoption of RDF molecules for decomposing and distributing of RDF graphs across computational nodes in the architecture; (c) the use of blank nodes to resolve the co-identification problem; and (d) extensions to RDF molecules to overcome problems of ambiguity, data loss and inefficiency introduced by blank nodes.

The remainder of the paper is organized as follows. In Section 2 we discuss related work. Section 3 provides a description of our extensions to RDF molecules to support hierarchy and ordering. Section 4 describes the system implementation using BioMANTA data and describes: (a) graph decomposition into molecules; (b) SPARQL querying across molecules; and (c) molecule merging to construct new RDF graphs based on queries. In Section 5, we present the initial results of the system's performance of graph decomposition and merging, distributed loading and SPARQL querying. Finally, we present our conclusions in Section 6.

2. Previous Related Works

Apart from our own previous work [18], there have also been a number of similar or related approaches to support scalable semantic querying across large RDF triple stores. Abadi *et al.* and Muster [1, 17]investigated improving RDF query performance through the use of column databases that vertically partition the data. This approach improves query performance for certain types of data and uses a very similar indexing approach to our proposal but does not take advantage of multiple compute nodes in a cluster.

Other work has suggested ways to increase the utility of MapReduce by adding a Merge stage to provide a relationally complete scale-out system [28]. A similar, but alternative idea is found in Yahoo's Pig Latin [20]. Both of these systems could be used to store and process RDF by treating RDF as tuples. The drawback with both of these approaches is that they are "batch" oriented and not real-time.

In addition, there has been previous relevant work in the area of RDF graph decomposition. Proposals such as Named Graphs [2], Concise Bounded Description (CBD) [26] and Minimum Self-contained Graphs (MSG) [27] all attempt to decompose a graph into smaller units. Figure 2 visualizes their relationship. However, they all have their respective disadvantages. In comparison, RDF molecules[8]provide the best approach for our MapReduce RDF store as they ensure automated, unambiguous and lossless decomposition and an optimal level of granularity.

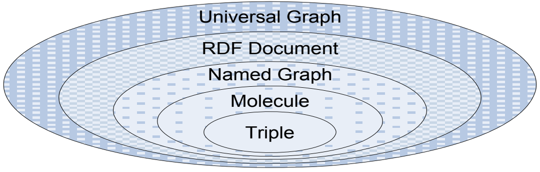

Figure 2. Granularity of RDF constructs including RDF Molecules (from [8]).

3. Extended RDF Molecules

In this section, we describe how we augment RDF molecules with hierarchy and ordering to alleviate certain drawbacks associated with the original molecules definition.

3.1. RDF Molecules

Formally, given an RDF graph G and a background ontology W, a pair of operators (d, m) is defined for decomposition and merging.

$$\begin{aligned} M &= d(G,W) \\ G' &= m(M,W) \end{aligned} \qquad (1)$$

Where, M is the set of molecules as the result of decomposition of G with regards to W using decomposition operator d. The merging operator m merges M back to an equivalent graph G', also with respect to the background ontology W. The set of molecules M are mutually independent in the sense that no blank node is shared among them. Hence, they can be individually processed and later merged to construct the RDF graph G' losslessly.

Two types of decomposition were defined: *naïve decomposition*, in which no background ontology is consulted; and *functional decomposition*, in which an OWL ontology is queried for functional dependency between nodes.

The following graph and diagram in Figure 3 consists of 6 triples (in N3 format) that model a physical interaction between two proteins (:_3 and _:4), represented as blank nodes.

```
{_:1 type ExperimentalObservation}
{_:1 observedInteraction _:2}
{_:2 participant _:3}
{_:3 hasUniprotID ‘p32379’}
{_:2 participant _:4}
{_:4 hasUniprotID ‘p46949’}
```

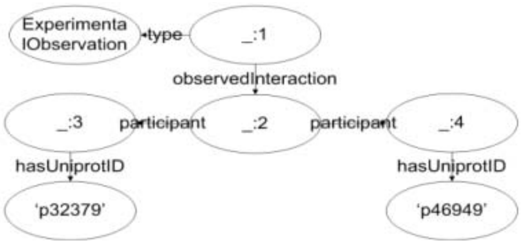

Figure 3. A simple RDF graph modeling a Protein-Protein Interaction (PPI).

The naïve decomposition results in a single molecule consisting of all the above triples since they are connected by blank nodes. This process maintains existing RDF semantics such as reification, containers, collections and blank nodes as existential variables. The later is demonstrated in Section 5 where molecule merging is demonstrated as a way to maintain lean versions of RDF graphs.

3.2. Extensions to RDF Molecules

RDF molecules have a number of inherent limitations that need to be overcome for efficient merging and decomposition. As the top figure of Figure 3 shows the absence of hierarchy in the original RDF molecule definition makes it difficult or even impossible to distinguish triples `{_:2 participant _:3}` and `{_:2 participant _:4}`. Moreover, the absence of ordering prevents certain important performance benefits including rapid retrieval of triples. In the following subsections, we present our extensions of RDF molecules that mitigate these problems.

3.2.1. Hierarchies of Molecules. Extended with hierarchies, a molecule consists of the following components: (a) a *head triple*, which is the lexicographically largest and most grounded triple, as defined in Section 3.2.2, from the set of *root triples;* (b) *root triples*, the set of *head triples* of submolecules; and (c) a number of *submolecules* (optional), where each of the submolecules has a head triple in turn.

3.2.2. Ordering of Molecules. The ordering of molecules is determined by comparison of their head triples. The ordering of two triples is based on the comparison of their nodes in turn. If subject nodes are equal, predicate nodes are compared. If predicate nodes are equal as well, object nodes are finally compared.

For two nodes, the lexicographical ordering is determined by the following rules,

- Node type - Blank node < URI reference node < Literal node
- Node value - String comparison of node values (“a”< “b”< “c”…)

The comparison of two molecules is based on the comparison of their head triples. For molecules *molecule*$_1$ and molecule$_2$and their head triples t_1 and t_2, *molecule*$_1$ $\otimes$ *molecule*$_2$ iff $t_1 \otimes t_2$, where the symbol $\otimes$represents<, = or>. Molecule comparison can be extended to include comparing root triples and submolecules – this is used during graph merging and molecule subsumption.

```
 { _:1 type ExperimentalObservation }
 { _:1 observedInteraction _:2 }
           { _:2 participant _:3 }
                     { _:3 hasUniprotID ‘p32379’ }
           { _:2 participant _:4 }
                     {  :4 hasUniprotID ‘p46949’’ }
```

Figure 4. RDF molecule decomposition of graph shown in Figure 3.

Example. Based on the extended molecule definition, the graph in Figure 3 is decomposed into the molecule shown in Figure 4. Note that this molecule has three hierarchies and the second root triple contains two submolecules. The blank nodes (_:3 and _:4) in these two submolecules are distinguishable because of the hierarchies.

4. Implementation Details

In this section, we describe the actual testbed system (Section 4.1) and the system components that we have implemented, including graph decomposition, RDF molecule merging and SPARQL querying across RDF molecules.

4.1. The BioMANTA Testbed

For the purpose of the BioMANTA project, we initially selected datasets from DIP, IntAct, MINT and MPact. In our previous work [19], we developed an integration process to (a) represent the datasets as RDF instances compliant with the BioMANTA ontology and (b) integrate the PPI RDF instances to form new RDF graphs based on UniProt IDs and genomic sequences of proteins, which are represented as RDF blank nodes. The integrated RDF graphs were subsequently decomposed into molecules, distributed across the molecule store and queried.

In protein-protein interaction (PPI) networks, a protein has a number of identifiers, external references, a genomic sequence string, and a host organism. The protein may also participate in interactions with other proteins. As discussed in Section 2, blank nodes are used to represent proteins, interactions, external references, etc. Hence, each protein and all of its associated information will belong to a single molecule, as shown in Figure 5, which illustrates the corresponding RDF molecule of the triples in Figure 1. It shows the lexicographical and "groundedness" ordering of the triples and differentiates_:Xref1 and _:Xref2 based on hierarchy.

```
_:Protein FullName "Eukaryotic…"
_:Protein Sequence "MATR…"
_:Protein ShortName "eLF-6"
_:Protein Synonym "CDC95"
_:Protein CrossReference _:Xref1
     _:Xref1 Accession "Q12522"
     _:Xref1 Database "UniProt"
_:Protein CrossReference _:Xref2
     _:Xref1 Accession "YPR016C"
     _:Xref1 Database "MIPS"
```

Figure 5. The RDF molecule corresponding to a simplified yeast protein shown in Figure 1.

Our molecular biologist collaborators identified a set of queries that may reveal previously unrecognized protein-protein interactions. For instance, the query "Find all yeast protein-protein interactions that are known to be localized to the endosomal system" helps biologists to filter protein-protein interactions (PPIs) integrated across the Gene ontology, the NCBI taxonomy and PPI datasets. Given the size of the PPI data and associated datasets (well over 1 billion triples), only a distributed processing environment is capable of integrating and querying on this scale.

4.2. Indexing and Querying

Each node in the cluster contains a local, persistent store designed to merge new data and respond to SPARQL queries. Our indexing scheme takes each permutation of an RDF triple (subject, predicate and object) and adds a molecule ID (spom, posm, ospm). Two other indices consisting of a molecule ID (m), parent molecule ID (i), and triple (imspo and spoim) are used to recreate the structure of an RDF molecule. A Molecules ID uniquely identifies the set of root triples for the molecule with the ID (0 indicates that the triple is not part of a molecule). A Parent ID indicates the molecule ID of the containing molecule (or 0 if it is not a submolecule). This supports efficient addition, retrieval and removal of molecules in the molecule store. An RDF Molecule API allows molecules to be added, removed, and found and an adaptor provides an RDF API and SPARQL query functionality.

We have implemented both an in-memory and an on-disk SPARQL query engine with RDF molecules based on the open-source JRDF[4] project. Graph matching is performed locally and answers are combined to provide the final query result. Future development of additional index adaptors would allow query engines from other RDF triple stores to be reused.

4.3. Graph Decomposition and Molecule Merging

In our approach, we adopted the naïve decomposition algorithm for its simplicity, efficiency and robustness. This algorithm computes connected components only through edges that connect two blank nodes. Given an RDF graph, the naïve decomposition algorithm decomposes it into a set of RDF molecules, which do not share blank nodes and are therefore mutually independent.

The molecule store merges two molecules if one molecule contains all the properties (or more) of another molecule. In this way, as more molecules are added, redundant molecules are removed (or never added) allowing results from multiple nodes from a query to be merged.

The computational complexity of naïve decomposition and molecule merging are both *O(n)*.For brevity reasons, the detailed analysis is not presented.

[4] http://jrdf.sourceforge.net/

5. Evaluation Results

We have implemented both the in-memory local version and on-disk Hadoop version of the RDF molecule store. In this section, we provide initial performance evaluation results for the critical steps in our methodology: RDF graph decomposition, RDF molecule merging and SPARQL querying.

5.1. Graph Decomposition and RDF Molecule Merging

The graph decomposition and merging algorithms described in the previous section are critical components of the distributed RDF molecule store. In this subsection, we evaluate the performance of these algorithms by comparing it with Jena [15]. Applied sequentially, the two algorithms can decompose an RDF graph into a set of RDF molecules, and then merge them back to form an equivalent graph. Jena is, to the best of our knowledge, the only RDF triple store that provides similar functionality by performing graph equivalence testing.

A set of RDF graphs was created for comparison and the time taken to determine equivalence was measured. The graphs contain triples that have chaining blank nodes, e.g., `_:1 p1 _:2`, `_:2 p2 _:3`, `_:3 p3 _:4`. For example, Table 1 (a) below shows that Jena takes 0.05 seconds to perform the graph equivalence test when the chain depth is 3 and chain size is 10 (total graph size is 30). Note that DNF stands for "Did Not Finish" (> 900 seconds).

Table 1. Time measurement of Jena and molecule on graph equivalence (in seconds).

Jena	Depth=3	5	10	20
Chain size =10	0.05	0.07	0.1	0.3
100	0.2	0.4	1.8	9.2
1000	13.1	37.7	197.7	DNF
10000	DNF	DNF	DNF	DNF

(a) Time measurement for Jena.

(b)

Molecule	Depth=3	5	10	20
Chain size=10	0.06	0.09	0.1	0.2
100	0.2	0.3	0.4	0.7
1000	0.9	1.3	2.5	5.0
10000	7.7	13.0	26.4	57.4

(b) Time measurement for RDF molecules.

The RDF molecule approach is faster as the number of chains reaches 100. The RDF molecule implementation gives consistently superior performance as both the number of chains and chain depth increase. When chain depth is at least 10 and number of chains is at least 100 (i.e., graph size is at least 1,000), the molecule implementation performs orders of magnitude better than Jena, with Jena not being able to determine equivalence for graph sizes over 20,000. Also note that with the increase of chain size and depth the performance of molecule implementation exhibits linear degradation, which is in line with our complexity analysis of the algorithms.

5.2. MapReduce Performance

MapReduce tasks are used to populate the distributed RDF molecule store. The *map* task converts data files from PSI-MI format to individual RDF molecule graphs. The *reduce* task collects generated molecules and puts them in the persistent, distributed molecule store. A series of tests were performed to evaluate the loading time of the distributed molecule store on both a 2-node and a 3-node cluster (individual nodes have identical setup).

Note that small clusters like this do not take full advantage of the MapReduce framework as performance suffers from communication overhead and node balancing. The experiments were performed for testing purposes only. A larger production cluster is being setup and we expect these overheads be amortized, and as a result, much better performance.

The MapReduce tasks were run multiple times using 10 input files (a total of 4,015,778 triples and 222,419 molecules). Table 2 summarizes the dataset sizes and performance of the various tasks on the two clusters. Note that the last two columns represent the time taken (in seconds) on the 2-node and 3-node cluster, respectively. A number of observations are worth discussing:

- Tasks 2 and 3 take roughly the same time, despite the fact that task 3 handles 48% more triples and 14% more molecules.
- There is a 100% increase in the time taken from task 3 to task 4, although task 4 only handles 44% more triples.

On the 3-node cluster, tasks 4, 5, 6 and 7 take comparable amount of time to complete, although there is a significant increase in the sizes of the tasks.

Table 2. Time measurements of various MapReduce tasks on two clusters.

Task no.	# triples	# molecules	2-node cluster	3-node cluster
1	363,308	10,387	201	165
2	1,164,446	73,357	899	829
3	1,727,754	83,744	995	895
4	2,488,024	138,675	1872	1,784
5	2,851,332	149,062	2041	1,789
6	3,652,470	212,032	2098	1,883
7	4,015,778	222,419	2692	1,994

The above performance characteristics are due to the nature of the MapReduce framework, in which the *map* and *reduce* phases execute in sequence: no *reduce* task can start unless all *map* tasks have been finished. Therefore, a very large single input file in the map phase in tasks 4, 5, 6 and 7 dominated their running time. Preprocessing of input files to break them into smaller chunks can help to bring down the time taken by the *map* phase. Figure 6 gives a more intuitive view of the running time of the different tasks. Horizontal axis represents the number of triples (in 1,000) and the vertical axis represents time (in seconds).

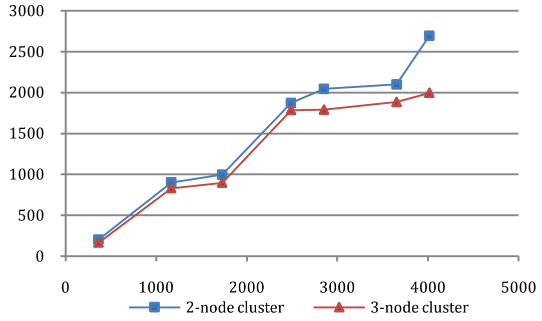

Figure 6. Time measurement of MapReduce conversion tasks.

As shown in Figure 6, with the increase of data size, the 3-node cluster shows greater scalability. When triple number exceeds 2 million, the 3-node cluster exhibits constant rate of slowdown whereas the 2-node cluster slowdowns considerably when processing 4 million triples.

Previous modeling on Nutch [16] showed that the response time is essentially constant as the number of servers reaches 2000 with up to 40 GB of data per node. We expect that our implementation of the on-disk, distributed RDF molecule store will conservatively reach 160 billion triples with a similar setup. Improving indexing efficiency will easily boost its capacity.

5.3. SPARQL Query Responses

As mentioned in Section 4.2, our SPARQL query engine has been developed by adapting the indexing structure of our RDF molecule store so that it is compatible with the indexing structure of the JRDF triple store. Hence, comparable query performance and memory usage is expected. We ran the same SPARQL query (see below) over the RDF molecule store and the JRDF triple store using an RDF graph describing yeast PPIs obtained and translated from the IntAct dataset.

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX biopax:  <http://www.biopax.org/release/biopax-level2.owl#>
PREFIX biomanta:
<http://biomanta.sourceforge.net/2007/07/biomanta_extension_02.owl#>
PREFIX ncbi: <http://biomanta.sourceforge.net/2007/10/ncbi_taxo.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?name ?id
WHERE {
  ?x  rdf:type biopax:physicalEntity .
  ?x  biomanta:fromNCBISpecies ncbi:ncbi_taxo_4932_ind .
  ?x  biomanta:hasPrimaryRef ?y .
  ?y  biopax:DB ?db .
  FILTER (str(?db) = "uniprotkb"^^xsd:string)
  ?y  biopax:ID ?id .
  FILTER (str(?id) = "o13516"^^xsd:string)
  ?x  biomanta:hasFullName ?name .
}
```

The RDF graph contains 79,360 triples. The SPARQL query finds all instances of the class “physicalEntity”, with an NCBI taxonomy ID 4932 and a UniProt ID “o13516”. The query time (~18 seconds) is almost the same for the two stores. As the cluster becomes larger, we can expect much better performance relative to traditional RDF triple stores.

6. Conclusions

Efficient querying and inferencing across large-scale integrated datasets drawn from many sources is a challenge facing many communities. Semantic Web technologies such as RDF, OWL and SPARQL are ideal candidates for the task of data integration as they offer open, unambiguous and extensible solutions. Distributed processing paradigms such as MapReduce have demonstrated economic and practical ways to process massive amounts (petabytes) of data. The combination of MapReduce and Semantic Web technologies appears to offer a perfect solution to the problem of large-scale heterogeneous data integration, querying and reasoning.

The co-identification problem introduces additional complications. RDF blank nodes provide a novel way of referring to equivalent entities without creating new names. However, they introduce complications when attempting to distribute RDF graphs across a MapReduce architecture.

RDF graphs provide too coarse a granularity for effective processing, as the context of an entire graph is needed to disambiguate RDF blank nodes. A finer granularity is required to support the distributed integration and processing of RDF data. RDF molecules provide a finer grained solution to the semantic integration and decomposition/distribution problem and enables MapReduce processing. We developed optimized algorithms to losslessly decompose an RDF graph into a set of smaller “molecules” and subsequently merge them. This process revealed that the presence of RDF blank nodes can cause problems of data loss, integrity loss, ambiguity and slow performance. Consequently, we extended the definition of RDF molecules to include hierarchy and ordering, which provide structural information, efficient processing, data integrity checking and most importantly makes it possible to disambiguate blank nodes within a single molecule.

Critical algorithms for decomposing an RDF graph and merging RDF molecules have also been described, implemented and evaluated. We compared RDF graph decomposition and merging with Jena’s graph equivalence checking algorithm and obtained promising results. We also ran SPARQL queries over the RDF molecule store and observed comparable performance to the JRDF triple store for moderate numbers of RDF triples. As greater numbers of triples are loaded into the scale-out RDF molecule store and as the size of the computational cluster grows, we can expect the performance to increase relative to traditional RDF triple stores.

Besides MapReduce processing, extended RDF molecules can be used for other purposes as well. As molecules capture the inherent structure of information, they can be used for effective visualization of information regarding a particular resource. Molecules can also be used to efficiently construct RDF graphs dynamically. RDF molecules can also be used as *context* to disambiguate RDF blank nodes.

7. References

[1] Abadi, D.J., et al. *Scalable Semantic Web Data Management Using Vertical Partitioning*. in *VLDB 2007*. 2007. University of Vienna, Austria.

[2] Carroll, J.J., et al., *Named Graphs, Provenance and Trust*, in *Proceedings of the 14th International Conference on World Wide Web*. 2005, ACM: Chiba, Japan. p. 613-622.

[3] Chatr-aryamontri, A., et al., *MINT: the Molecular INTeraction database.* Nucleic Acids Res, 2007. **35**(Database issue): p. 572-574.

[4] Chen, H., Z. Wu, and Y. Mao, *RDF-Based Ontology View for Relational Schema Mediation in Semantic Web*, in *9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2005)*. 2005: Melbourne, Australia. p. 873-879.

[5] Cheung, K.-H., et al., *YeastHub: a semantic web use case for integrating data in the life sciences domain.* Bioinformatics, 2005. **21**(Supp. 1): p. 85-96.

[6] Davis, M., et al., *Integrating Hierarchical Controlled Vocabularies with OWL Ontology: A Case Study from the Domain of Molecule Interactions*, in *6th Asia Pacific Bioinformatics Conference (APBC08)*. 2008: Kyoto, Japan.

[7] Dean, J. and S. Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation*. 2004, USENIX Association: San Francisco, CA. p. 137--150.

[8] Ding, L., et al., *Tracking RDF Graph Provenance using RDF Molecules*. 2005, UMBC.

[9] Good, B.M. and M.D. Wilkinson, *The Life Sciences Semantic Web is Full of Creeps!* Briefings in Bioinformatics, 2006. **7**(3): p. 275-286.

[10] Guha, R. *Object co-identification on the Semantic Web*. in *13th World Wide Web Conference*. 2004. New York, USA.

[11] Güldener, U., et al., *MPact: the MIPS protein interaction resource on yeast.* Nucleic Acids Res, 2006. **34**(Database issue): p. 436-441.

[12] Halpin, H., *Identity, Reference, and Meaning on the Web.* Proceedings of the Workshop on Identity, Meaning and the Web (IMW06) at WWW2006, Edinburgh, Scotland, 2006.

[13] Jaffri, A., H. Glaser, and I.C. Millard, *Managing URI Synonymity to Enable Consistent Reference on the Semantic Web*, in *1st International Workshop on Identity and Reference on the Semantic Web (IRSW2008)* 2008: Tenerife, Spain.

[14] Kerrien, S., et al., *IntAct--open source resource for molecular interaction data.* Nucleic Acids Res, 2007. **35**(Database issue): p. D561-5.

[15] McBride, B., *Jena: a semantic Web toolkit.* IEEE Internet Computing, 2002. **6**(6): p. 55-59.

[16] Moreira, J.E., et al. *Scalability of the Nutch Search Engine*. in *Proceedings of the 21st Annual International Conference on Supercomputing*. 2007. Seattle, Washington: ACM Press.

[17] Muster, P., *Quantitative and Qualitative Evaluation of a SPARQL Front-End for MonetDB*, in *Department of Informatics*. 2007, University of Zurich: Zurich.

[18] Newman, A., et al., *A Scale-Out RDF Molecule Store for Distributed Processing of Biomedical Data*, in *Semantic Web for Health Care and Life Sciences Workshop (HCLS'08)*. 2008: Beijing, China.

[19] Newman, A., et al., *BioMANTA Ontology: The Integration of Protein-Protein Interaction Data*, in *Interdisciplinary Ontology Conference (InterOntology08 Tokyo)*. 2008: Tokyo, Japan.

[20] Olston, C., et al., *Pig Latin: A Not-So-Foreign Language for Data Processing*, in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. 2008, ACM: Vancouver, Canada.

[21] Ruttenberg, A., et al., *Advancing Translational Research with the Semantic Web.* BMC Bioinformatics, 2007. **8**(Suppl 3).

[22] Salamone, S., *LSID: An Informatics Lifesaver*. 2004.

[23] Salwinski, L., et al., *The Database of Interacting Proteins: 2004 update.* Nucleic Acids Res, 2004. **32**(Database issue): p. D449-51.

[24] Schroeter, R. and J. Hunter, *Annotating Relationships Between Multiple Mixed-Media Digital Objects by Extending Annotea*, in *Proceedings of the 4th European Semantic Web Conference (ESWC 2007)*. 2007, Springer: Innsbruck, Austria. p. 533-548.

[25] Stephens, S., A. Morales, and M. Quinlan, *Applying Semantic Web Technologies to Drug Safety Determination.* IEEE Intelligent Systems, 2006. **21**(1): p. 82-88.

[26] Stickler, P., *CBD - Concise Bounded Description*. 2005.

[27] Tummarello, G., et al., *Signing Individual Fragments of an RDF Graph*, in *Special interest tracks and posters of the 14th international conference on World Wide Web*. 2005, ACM: Chiba, Japan. p. 1020-1021.

[28] Yang, H.-c., et al., *Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters*, in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. 2007: Beijing, China. p. 1029-1040.