A Scale-Out RDF Molecule Store for Improved Co-Identification, Querying and Inferencing

Andrew 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 implementation details for our MapReduce-based RDF molecule store describing: (a) graph decomposition into molecules; (b) SPARQL querying across molecules; and (c) molecule merging to retrieve the search results. 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. The results of performance evaluations based on this case study are presented and discussed.

Keywords: scalability, MapReduce, RDF molecules, distributed querying

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 or co-reference – identifying when entries across data sets 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 [27].

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 process massive amounts of RDF instance data (billions or trillions of triples).

In particular, Google's MapReduce architecture [10] provides a software framework 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, scalable and economical architecture. **Our aim is to investigate methods by which MapReduce 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]. Several frameworks exist to query and store data on top of Hadoop including Pig Latin[4], Hive[5] and HBase[6]. While these systems provide a query language, they are batch oriented. In order to achieve responsive, real-time querying, our architecture employs an approach similar to Nutch [20], a project that applies the Lucene search engine using Hadoop. Nutch's distributed search configuration allows real-time querying of indexed Web pages. Each search server uses its own local index, responds to queries and the issuing server combines these results. We propose an RDF-based architecture similar to Nutch 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 [11] enables 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" [11]. Hence RDF molecules:

[1] http://www.w3.org/2001/sw/hcls/
[2] http://hadoop.apache.org/
[3] http://aws.amazon.com/ec2
[4] http://research.yahoo.com/node/90
[5] http://research.yahoo.com/node/2104
[6] http://hadoop.apache.org/hbase/

- 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. It lacks:

- The ability to disambiguate a triple with two blank nodes (subject and object);
- A semantic representation (*e.g.,* structure) of the entity in the original graph;
- The ability to leverage certain efficiencies (*i.e.,* triple order) 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 Semantc Web, is the *object co-identification* [13] 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, this problem is widely recognized and not easily resolved. The difficulty is that key, large-scale protein databases such as UniProt, DIP [29], IntAct [19] and MPact [14] 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 [8, 30, 31]. In particular, previous attempts to standardize naming and identification (e.g., LSIDs [28]) have had limited beneficial impact [12]. We believe that inventing another naming convention or trying to reach a consensus will not solve the identification problem [15]. We reject the idea of creating yet another URI to create a co-reference bundle [18], instead we propose an identity reconciliation process for the "life sciences identifier problem" based on *blank nodes*.

This 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 [7], 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 interaction 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 [6] via a common model/ontology. The common model is the BioMANTA OWL-DL ontology[7] that was developed [9, 24] by reusing vocabularies from well-established ontologies/standards such as Gene Ontology [2], Cell Type ontology [3], BioPAX [4], PSI-MI [17], and the NCBI taxonomy. Using 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.

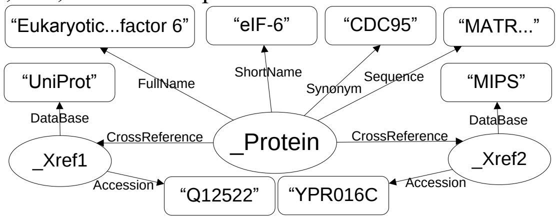

Figure 1. RDF triples about a yeast protein.

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, co-identification and semantic querying. The more specific objectives are to investigate and evaluate:

- 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;
- The adoption of RDF molecules for decomposing and distributing RDF graphs across computational nodes in the MapReduce architecture;

[7] http://biomanta.sourceforge.net/2007/07/biomanta_extension_02.owl

- The use of blank nodes to resolve the co-identification problem;
- Extensions to RDF molecules to overcome problems of ambiguity, data loss and inefficiency introduced by blank nodes.

In addition, the aim is to evaluate our proposed scale-out RDF Molecule Store using bio-molecular pathway datasets that have been integrated for the purposes of the BioMANTA project, described in Section 1.3.

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/merging, distributed loading and SPARQL querying. Finally, in Section 6 we present our conclusions and discuss future work.

2 Previous Related Work

Apart from our own previous work [25], 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, 23] 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. YARS2 and SWSE use a similar "shared nothing" scale-out approach to achieve scalability [16] but they are not based on MapReduce.

Other work has suggested ways to increase the utility of MapReduce by adding a Merge stage to provide a relationally complete scale-out system [34]. A similar, but alternative idea is found in Yahoo's Pig Latin [26]. 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. Below we provide an analysis of four possible approaches to RDF graph decomposition (their relationship is visualized in Figure 2):

- **Concise Bounded Description** (CBD) [32] is a subgraph of triples about a particular resource *R* and a chain of triples with blank nodes consisting of matching object to subject nodes (ignoring the special case for reification). A drawback of CBD is that it only looks at subject nodes in RDF triples and a CBD created for a resource node may not include all of the information.
- **Minimum Self-contained Graphs** (MSG) [33] is a proposal for the decomposition of an RDF graph into self-contained subgraphs. Given an RDF triple, its corresponding MSG includes (a) the triple itself and recursively, (b) for all the blank nodes involved in the MSG so far, all the triples of MSGs containing these blank nodes. Compared to CBD, MSG looks for statements to be included in the MSG in both directions. Hence, it results in lossless decomposition.
- **RDF Molecules [11]** decomposes an RDF graph into a set of molecules. A naïve decomposition creates a set of molecules that do not share blank nodes, which is

equivalent to MSG. A functional decomposition takes into consideration the ontology to create finer molecules that may share a blank node.

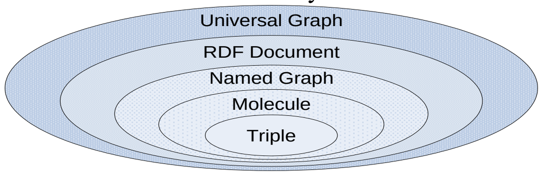

Figure 2. Granularity of RDF constructs including RDF Molecules (from [11]).

Note that an RDF document can be divided into Named Graphs [5] arbitrarily, hence a named graph is not necessarily less granular than RDF molecules.

Based on the above analysis, we believe that RDF molecules provide the best approach for our MapReduce RDF store as they ensure automated, unambiguous and lossless decomposition and an optimal level of granularity.

3 Extended RDF Molecules & Algorithms

In this section, we describe how we augment RDF molecules with hierarchy and ordering to alleviate certain drawbacks associated with the original molecules definition. We also present a number of important algorithms on graph decomposition and molecule merging.

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.

$$M = d(G,W) \qquad (1)$$
$$G' = m(M,W)$$

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 the 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.

The following graph and diagram in Figure 3 consists of 6 triples (in modified N-Triples format) that model a physical interaction between two proteins, represented as blank nodes.

```
[_:1 type ExperimentalObservation.]
[_:1 observedInteraction _:2.]
[_:2 participant _:3.]
[_:3 hasUniprotID "p32379".]
[_:2 participant _:4.]
[_:4 hasUniprotID "p46949".]
```

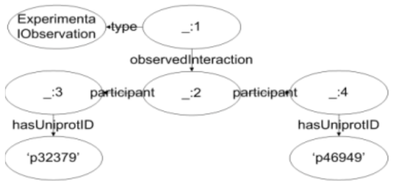

Figure 3. A simple RDF graph modeling a Protein-Protein Interaction (PPI).

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. Moreover, RDF molecules helps to maintain lean graphs as redundant blank nodes are identified and merged. This will become clearer later in Section 5 when we discuss molecule merging 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 left side of Figure 3, 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 subsection, we present our extensions of RDF molecules that mitigate these problems.

3.2.1 Hierarchies of Molecules

Formally, a molecule is recursively defined with the abstract syntax (in EBNF format) shown in Figure 4. A molecule has a (possibly empty) set of *root triples*, each of which has an optional *submolecule.* An example is shown in Figure 5, which is a molecule with two root triples and one of these triples, `_:1 observedInteraction _:2` has a submolecule. The lexicographically largest and most grounded triple of the set of root triples, as defined in Section 3.2.2, is called the *head triple.*

```
Molecule ::= { RootTriple [ Molecule ] }
RootTriple ::='RootTriple('   Triple   ')'
Triple ::='['   Subject   Predicate   Object   '.'   ']'
```

Figure 4. Abstract syntax of extended molecule.

As described in the previous section, a molecule in the original definition contains triples all of which are on a single level. We believe that the incorporation of hierarchies as shown above helps to capture the structure of the underlying RDF triples. Specifically, this allows us to determine equality of blank nodes based on context rather than on an internal identifier.

3.2.2 Ordering of Molecules

The ordering of molecules is determined by comparing the 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 a 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 ⊗ represents <, = or >. Molecule comparison can be extended to include comparing root triples and submolecules – this is used during graph merging and molecule subsumption.

Example. For the RDF graph shown in Figure 3, blank nodes `_:3` and `_:4` cannot be distinguished in the original molecule definition. Moreover, as RDF graphs capture semantic information, usually there is inherent structure about the information being captured. Hierarchical molecules allow the representation of this structure as well. Based on the extended molecule definition, the graph in Figure 3 is decomposed into the molecule shown in Figure 5. 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.

```
[ _:1 type ExperimentalObservation . ]
[ _:1 observedInteraction _:2 . ]
            [ _:2 participant _:3 . ]
                     [ _:3 hasUniprotID "p32379" . ]
            [ _:2 participant _:4 ]
                     [ _:4 hasUniprotID "p46949" . ]
```

Figure 5. RDF molecule decomposition of graph shown in Figure 3.

3.3 Graph Decomposition

We adapted the naïve decomposition algorithm, which computes connected components only through edges that connect two blank nodes, to decompose an RDF graph. In describing the decomposition algorithm, we make a distinction between *global graph* and *local graph*. Global graphs require the context of a molecule to uniquely identify a blank node. Local graphs use an internal, unique identifier for each blank node. The decomposition algorithm takes a local graph, which has blank nodes with internal, unique identifiers, and creates a set of molecules (a global graph) that uniquely identifies blank nodes based on their context within a molecule.

The naïve decomposition algorithm in Figure 6 works on local RDF graphs. A triple is grounded if none of its nodes are blank nodes. The top of a chain of linking triples is defined by matching blank subject nodes to blank object nodes. For example, in a chain of triples: `_:1 p _:2`, `_:2 p _:3`, `_:3 p _:4` the head is `_:1 p _:2`. There are three cases to consider when identifying submolecules:

- If molecule `M`'s head triple is a linking triple (both subject and object nodes are blank nodes) and the triple to add `T` has a subject that is equal to its object then the triple is added to the submolecule `SM`.
- If the identified submolecule `SM` contains a triple which links to the head of the current molecule `M` then the current molecule is added to the submolecule and the molecule used from then on is the submolecule. In other words, the contents of the molecule are added to the submolecule which becomes the molecule used from then on in future operations. If there are cycles in molecules, triple ordering is used to decide which molecule is the outer-most molecule.
- If the identified submolecule does not contain a triple which links to the current molecule then it is added to the current molecule.

In terms of computational complexity, the worst case is when all triples share, recursively, some blank nodes and they end up in one molecule with *n* levels (one triple at a level). Each triple is only added to a (sub)molecule once and is compared to the head triple once. Hence, a constant number of basic operations are performed for adding each triple and the time complexity of decomposition is *O(n)*.

```
AT = the set of added triples (initially empty).
LGT = a sorted set in descending order (defined above) of
    triples from a local graph.
FOR EACH Triple T from LGT not in AT
    If T is a Link Triple Get Head of Chain of Linking Triples
   Create a new molecule M, add T to M.
    Add T to AT.
     IF T is not Grounded THEN
         findEnclosedTriples(M).
     END IF
END FOR
findEnclosedTriples(M)
      T = the HeadTriple of M.
      BTS = a set of all triples which contain T's blank nodes.
      FOR EACH Triple BT from BTS not in AT
          Create a new molecule SM adding BT.
          Add BT to AT.
          findEnclosedTriples(SM)
          IF BT is a Link Triple THEN
             IF BT's object node equals T's subject node THEN
                Add M to SM.
                SM becomes M.
             ELSE
                 Add SM to M.
             END IF
           ELSE
             Add BT to M.
           END IF
     END FOR
     Add all triples found to the set AT.
END findEnclosedTriples
```
Figure 6. Pseudocode for naive graph decomposition.

3.4 Molecule Merging

The molecule store combines 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.

```
merge(m1, m2, blankNodeMap)
    LET blankNodeMap = findBlankNodeMap(m1, m2)
    create new molecule m3.
    replace the blank nodes in m2 with m1's using the blankNodeMap.
    add the root triples from m1 and m2 to the root triples of m3.
    FOR EACH root triple t1 in m3
       LET sm1 = m1.submolecule for t1.
       LET sm2 = m2.submolecule for t1.
       IF sm1 != null AND sm2 != null THEN
           sm3 = merge(sm1, sm2, blankNodeMap).
           add sm3 to m3 using the root triple t1.
       END IF
    END FOR
return m3
END merge
```
Figure 7. Pseudocode for molecule merging.

A mapping from the blank nodes in one molecule to another must first be determined. Given two molecules `m1` and `m2`, the blank node mapping procedure, `findBlankNodeMap`, tries to find a corresponding blank node in `m1` for each blank node in `m2`, respecting the hierarchies and ordering of both molecules. The complexity of the `findBlankNodeMap` procedure depends on the number of comparisons between triples of the two molecules. The hierarchies reduce the number of comparisons as comparisons are only made for submolecules on the same level. Without loss of generality, we assume that `m1` has fewer levels of submolecules than `m2`. Let the number of levels of `m1` be m, and the number of triples on level i be n_i^1. For the first m

levels, let the number of triples of molecule m2 be n_i^2. The complexity of the findBlankNodeMap algorithm is:

$$C_2^1 = n_1^1 * n_1^2 + \cdots + n_m^1 * n_m^2 = \sum_{i=1}^{m} n_1^i * n_2^i < \sum_{i=1}^{n} n_1^i * \sum\nolimits_{j=1}^{m} n_2^j = O(n^2) \qquad \textbf{(2)}$$

If the map returned by findBlankNodeMap is an empty map, the two molecules cannot be merged as there is no correspondence between the blank nodes they contain. On the other hand, if the map is non-empty (m1 subsumes m2), the merge procedure shown in Figure 7 merges the two molecules. The complexity of the merge algorithm is quadratic to the size of the larger molecule of the two (dominated by the findBlankNodeMap method) - with respects to the number of levels of submolecules, as it merges all the triples on the top level and iteratively merges all the submolecules.

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 and RDF molecule distribution; and
- SPARQL querying across the 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 [24], 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.

```
[_: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 8. The molecule corresponding to a simplified yeast protein.

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 8 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.

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.

The RDF graph that is decomposed into RDF molecules follows the structure of the ontology effectively leading to functional decomposition. They structurally represent the relationships between experiments, interactions and proteins. All of the SPARQL queries that will be issued are expected to match this structure.

4.2 Indexing and SPARQL Querying

Each node in the cluster contains a local, persistent store designed to merge new data and respond to SPARQL queries. Our indexing scheme consists of the permutations of RDF triples (spo), the molecule ID (m) and the parent molecule ID (d), (spomd, posmd and ospmd). A fourth index (dmspo) is used find RDF molecules efficiently. 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.

As RDF molecules are distributed across compute nodes in a cluster, the SPARQL queries also need to be broken down and executed against local indices. The partial answers are then aggregated and form the final answer. With knowledge of the structure of the molecules in the distributed molecule store, SPARQL queries can be written in a way that can be easily broken down to queries about individual molecules.

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 [21]. 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 graph contains triples that have chaining blank nodes, e.g., `_:1 p1 _:2, _:2 p2 _:3, _:3 p3 _:4`. For example, the table on the left shows that Jena takes 0.05 seconds to perform the graph equivalence test where the chain depth is 3 and number of chains 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 secs).

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

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

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 graph size is at least 1,000 (i.e., 100 chains and chain depth of 10), 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, 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

In the MapReduce framework, *tasks* are units of execution that perform certain computation. In our project, MapReduce tasks are used to populate the distributed RDF molecule store. For example, in one MapReduce task the *map* task converts data files from PSI-MI format to RDF molecule graphs. The *reduce* task collects the generated molecules and puts them in the persistent graph. A series of tests were performed to evaluate the loading time of the distributed molecule store. On a 3-node cluster, a MapReduce task was run multiple times using 10 input files (a total of 4,015,778 triples and 222,419 molecules).

Table 2. Time measurement of various MapReduce tasks.

Task no.	# triples	# molecules	Time (second)
1	363,308	10,387	165
2	1,164,446	73,357	829
3	1,727,754	83,744	895
4	2,488,024	138,675	1,784
5	2,851,332	149,062	1,789
6	3,652,470	212,032	1,883
7	4,015,778	222,419	1,994

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.
- Tasks 4, 5, 6 and 7 all take roughly the same amount of time to complete, although there is a significant difference in the sizes of the tasks.

The above performance characteristics are due to the nature of the MapReduce framework, in which the *map* and *reduce* phases execute in sequence. Hence, 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 9 gives a more intuitive view of the running time of the different tasks. The graph shows that the MapReduce RDF molecule store maintains performance, by balancing the load, as the number of input files increase.

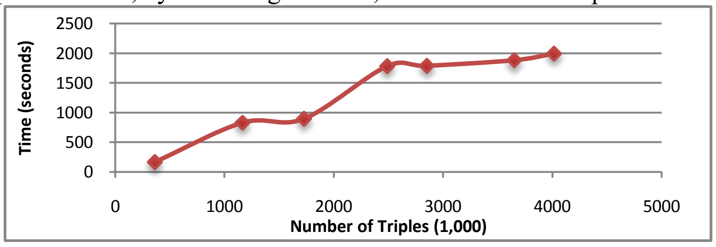

Figure 9. Time measurement of MapReduce conversion task

The distributed RDF molecule store takes up around 0.5 GB disk space per million triples. This is due to the fact that more indexing information is maintained for RDF molecules and no compression or other space-saving optimizations have been applied at this time. Previous modeling [22] has shown the response time of Nutch is essentially constant as the number of servers reaches 2000 nodes 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.

The 3-node cluster does not take full advantage of the MapReduce framework. The results suffer from communication overhead and node balancing. We expect that a larger cluster will amortize these overheads.

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 greater numbers of triples are loaded into the store, as the computation power increases, we can expect much better performance relative to traditional RDF triple stores.

6 Conclusions

In the future, we expect to continue developing the disk-based, distributed, processing and querying environment using the extended RDF molecules. This environment will greatly enhance our ability to query and reason across large amounts of data efficiently. Currently, we decompose a graph according to blank node connectedness but it could be extended to create molecules based on the connectedness of grounded triples (triples without blank nodes) with matching subjects and objects as well.

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 index and 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. Attempts to standardize naming conventions have had limited impact. RDF blank nodes, on the other hand, 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 distribution/decomposition 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. Hierarchy and ordering provides structural information, efficient processing and 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 isomorphism algorithm and obtained promising results. We also ran SPARQL queries over the RDF molecule store and found it comparable 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.

Acknowledgements

We would like to acknowledge Pfizer for funding the research described in this paper and to thank Abdul Alabri, Melissa Davis, Imran Khan and Muhammad Shoaib B. Sehgal for their contributions and ideas.

References

1. Abadi, D.J., et al. *Scalable Semantic Web Data Management Using Vertical Partitioning*. in *VLDB 2007*. 2007. University of Vienna, Austria.
2. Ashburner, M., et al., *Gene Ontology: tool for the unification of biology.* Nature Genetics, 2000. **25**: pp. 25-29.
3. Bard, J., S.Y. Rhee and M. Ashburner, *An Ontology for Cell Types.* Genome Biology, 2005. **6**(2).
4. BioPAX Wokrgroup, *BioPAX – Biological Pathways Exchange LanguageLevel 2, Version 1.0 Documentation*. 2005, BioPAX.
5. Carroll, J.J., et al., *Named Graphs, Provenance and Trust*. In Proceedings of the 14th International Conference on World Wide Web, pp. 613-622. ACM, Chiba, Japan, (2005).
6. Chatr-aryamontri, A., et al., *MINT: the Molecular INTeraction database.* Nucleic Acids Res, 2007. **35**(Database issue): pp. 572-574.
7. 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), pp. 873-879. Melbourne, Australia, (2005).
8. Cheung, K.-H., et al., *YeastHub: a semantic web use case for integrating data in the life sciences domain.* Bioinformatics, 2005. **21**(Supp. 1): pp. 85-96.
9. 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), Kyoto, Japan, (2008).
10. 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, pp. 137--150. USENIX Association, San Francisco, CA, (2004).
11. Ding, L., et al., *Tracking RDF Graph Provenance using RDF Molecules*. Techical Report, 2005, TR-CS-05-06, UMBC.
12. Good, B.M. and M.D. Wilkinson, *The Life Sciences Semantic Web is Full of Creeps!* Briefings in Bioinformatics, 2006. **7**(3): pp. 275-286.
13. Guha, R. *Object co-identification on the Semantic Web*. in *13th World Wide Web Conference*. 2004. New York, USA.
14. Güldener, U., et al., *MPact: the MIPS protein interaction resource on yeast.* Nucleic Acids Res, 2006. **34**(Database issue): pp. 436-441.
15. 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.
16. Harth, A., et al., *YARS2: A Federated Repository for Searching and Querying Graph Structured Data*. 2007, DERI Galway, Ireland.
17. Hermjakob, H., et al., *The HUPO PSI's Molecular Interaction format—a community standard for the representation of protein interaction data.* Nat Biotechnol, 2004. **22**(2): pp. 177-83.

18. 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) Tenerife, Spain, (2008).
19. Kerrien, S., et al., *IntAct--open source resource for molecular interaction data.* Nucleic Acids Res, 2007. **35**(Database issue): pp. D561-5.
20. Khare, R., et al., *Nutch: A flexible and scalable open-source web search engine*. 2004: CommerceNet Labs Technical Report 04.
21. McBride, B., *Jena: a semantic Web toolkit.* IEEE Internet Computing, 2002. **6**(6): pp. 55-59.
22. 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.
23. Muster, P., *Quantitative and Qualitative Evaluation of a SPARQL Front-End for MonetDB*, in *Department of Informatics*. 2007, University of Zurich: Zurich.
24. Newman, A., et al., *BioMANTA Ontology: The Integration of Protein-Protein Interaction Data*. In Interdisciplinary Ontology Conference (InterOntology08 Tokyo), Tokyo, Japan, (2008).
25. 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) at the 17th International Conference on World Wide Web (WWW'08), Beijing, China, (2008).
26. 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, ACM, Vancouver, Canada, (2008).
27. Ruttenberg, A., et al., *Advancing Translational Research with the Semantic Web.* BMC Bioinformatics, 2007. **8**(Suppl 3).
28. Salamone, S., *LSID: An Informatics Lifesaver*. 2004, Bio-ITWorld, http://www.bio-itworld.com/archive/011204/lsid.html.
29. Salwinski, L., et al., *The Database of Interacting Proteins: 2004 update.* Nucleic Acids Res, 2004. **32**(Database issue): pp. D449-51.
30. 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), pp. 533-548. Springer, Innsbruck, Austria, (2007).
31. Stephens, S., A. Morales and M. Quinlan, *Applying Semantic Web Technologies to Drug Safety Determination.* IEEE Intelligent Systems, 2006. **21**(1): pp. 82-88.
32. Stickler, P., *CBD - Concise Bounded Description*. 2005, W3C Member Submission, http://www.w3.org/Submission/CBD/.
33. 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, pp. 1020-1021. ACM, Chiba, Japan, (2005).
34. 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, pp. 1029-1040. Beijing, China, (2007).