# Scalable Reduction of Large Datasets to Interesting Subsets

Gregory Todd Williams, Jesse Weaver, Medha Atre, and James A. Hendler

Tetherless World Constellation, Rensselaer Polytechnic Institute, Troy, NY, USA
{willig4,weavej3,atrem,hendler}@cs.rpi.edu

**Abstract.** With a huge amount of RDF data available on the web, the ability to find and access relevant information is crucial. Traditional approaches to storing, querying, and inferencing fall short when faced with web-scale data. We present a system that combines the computational power of large clusters for enabling large-scale inferencing and data access with an efficient data structure for storing and querying this accessed data on a traditional personal computer or smaller embedded device. We present results of using this system to load the Billion Triples Challenge dataset, fully materialize RDFS inferences, and extract an "interesting" subset of the data using a large cluster, and further analyze the extracted data using a traditional personal computer.

## 1 Introduction

The semantic web continues to grow with vast amounts of RDF data becoming available. With so much data available on the web, it is important to be able to find, extract and use just the data that is relevant to a particular problem in a reasonable amount of time. Several factors make this difficult, including:

- **Domain**: The domain of interest may not be modeled in the same way across the web.
- **Inferences**: Deriving inferences from large datasets can be challenging due to the size of data and the lack of scalable inferencing tools.
- **Analysis**: Analysis of large datasets is difficult due to the time-consuming nature of interactive queries on such data.
- **Extraction**: Extracting specific subsets of interest from very large datasets can alleviate the challenge of analysis, but can also be a time consuming task despite often being simplistic in nature. Many tools require lengthy loading and indexing times which cannot be ammortized when used to support just a single query meant to extract a subset of data.

We propose a composition of approaches that addresses these issues. In particular, we present a system that makes use of a large cluster and/or supercomputer to allow full materialization of inferences over large datasets and efficient extraction of specific data subsets. We also make use of a compressed triplestore that may be used to directly answer interactive queries on a wide range of (non-cluster) systems. We evaluate this system on the Billion Triples Challenge 2009 (BTC2009) dataset[1].

In the following sections, we rely on a specific use case supported by the BTC2009 dataset. We consider a scenario in which a user desires to discover what people are mentioned in the dataset, and to retrieve specific facts about them. This involves discovering the resources that represent people as well as retrieving their names and email addresses.

The rest of the paper is organized as follows. Section 2 describes the core components of our system and briefly describes the relevant algorithms and system architectures that support them. In section 3 we evaluate our system using the use case described above, performing inferencing on the dataset, extracting relevant data, and loading the reduced dataset into a compressed triplestore. Finally, section 4 concludes the paper and discusses extensions we hope to pursue as future work.

## 2 Methodology

To address the issue of domain, we create a simple upper ontology to provide superclasses and superproperties to appropriate classes and properties in the BTC2009 dataset. Adding this upper ontology (along with some BTC2009-related ontologies to ensure that these ontologies are present), we then apply the parallel RDFS reasoning approach described in [1] to derive triples using terms from our upper ontology. Following reasoning, we perform a basic graph pattern query to extract our specific knowledge of interest (persons with names and email addresses) using the approach described in [2]. Finally, we take the reduced dataset and store it as a BitMat[3] for compressed storage as well as simple basic graph pattern querying.

### 2.1 Domain of Interest

For our domain of interest, we chose people and relevant information for people which is widely used in the semantic web. Information about people are represented differently in the BTC2009 dataset, including the use of the SIOC[2], DBpedia[3], AKT[4], and FOAF[5] ontologies.

We define the following prefixes for use throughout the paper:

- akt: <http://www.aktors.org/ontology/portal#>
- dbpo: <http://dbpedia.org/ontology/>
- dbpp: <http://dbpedia.org/property/>

[1] http://vmlion25.deri.ie/index.html
[2] Semantically-Interlinked Online Communities, http://sioc-project.org/ontology
[3] http://wiki.dbpedia.org/Ontology
[4] Advanced Knowledge Technologies, http://www.aktors.org/publications/ontology/
[5] Friend of a Friend, http://xmlns.com/foaf/spec/

- foaf: <http://xmlns.com/foaf/0.1/>
- rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- rdfs: <http://www.w3.org/2000/01/rdf-schema#>
- sioc: <http://rdfs.org/sioc/ns#>
- up: <http://www.cs.rpi.edu/~weavej3/btc2009#>

Note that "up" is the namespace for our upper ontology. We define our upper ontology as consisting of the following class and properties with their listed subclasses and subproperties:

- Subclasses of **up:Person**: akt:Person, dbpo:Person, foaf:Person, and sioc:User.
- Subproperties of **up:affiliated_with**: foaf:knows.
- Subproperties of **up:email**: akt:has-email-address, foaf:mbox, and sioc:email.
- Subproperties of **up:email_sha1**: foaf:mbox_sha1sum and sioc:email_sha1.
- Subproperties of **up:first_name**: akt:given-name, foaf:firstName, foaf:givenname, and sioc:first_name.
- Subproperties of **up:full_name**: akt:full-name, dbpp:name, foaf:name, and sioc:name.
- Subproperties of **up:last_name**: akt:family-name, foaf:family_name, foaf:surname, and sioc:last_name.
- Subproperties of **up:web_page**: foaf:page.

The upper ontology, along with other BTC2009-related ontologies, were added to the BTC2009 dataset to allow inferencing to derive triples using terms in our upper ontology.

## 2.2 Inferencing over Large Dataset

Using the approach by Weaver and Hendler[1], we produced the (finite) RDFS closure of the entire BTC2009 dataset. Breaking down BTC2009 into separate chunks (such that the union of the chunks create the whole BTC2009 dataset), individually computing the RDFS closure of these chunks will result in the same triples as computing the RDFS closure of the entire dataset. This allows us to materialize the RDFS closure in parallel.

For this work, we excluded the following "less interesting" RDFS entailment rules[6]:

- Rule *lg*: generating unique blank nodes for all literals.
- Rule *rdfs1*: all literals are type rdfs:Literal.
- Rule *rdfs4a/b*: everything is an rdfs:Resource.
- Rule *6*: each property is a subproperty of itself.
- Rule *8*: each class is a subclass of rdfs:Resource.
- Rule *10*: each class is a subclass of itself.

The greatest benefit of RDFS reasoning (for our purposes) is type inference from class hierarchy, assertion inference from property hierarchy, and type inference from domains and ranges of properties. This allows us to efficiently generate inferences that use terms from our upper ontology.

[6] http://www.w3.org/TR/2004/REC-rdf-mt-20040210/#rules

### 2.3 Reducing the Dataset

Once the RDFS closure of the dataset is produced, we need to extract only the knowledge in which we are interested. Utilizing the parallel RDF query approach by Weaver and Williams[2], we can efficiently perform this query over the entire RDFS closure of the BTC2009 dataset to extract just the information we need, greatly reducing the number of triples for later storage and analysis.

This reduction process makes use of the parallel nature of clusters to achieve much higher loading rate than would otherwise be possible. Each processor in the cluster is responsible for loading a single contiguous chunk of the input data and maintaining local, in-memory indexes over this data. Once this parallel loading process is complete, query execution occurs immediately.

During query evaluation, our system makes use of a parallel hash join to redistribute intermediate results across processors in the cluster based on the bound values of the join variables present in the query. This distribution of data across the cluster ensures that intermediate results that can produce a result as part of a join are colocated. The joins that comprise the query occur sequentially, with each individual join making use of the entire cluster. When all joins have been completed, each processor in the cluster writes its final results to disk, making the (now greatly reduced in size) data available for use in a more traditional, non-cluster-based system.

### 2.4 Analysis of Reduced Dataset

Having extracted only the knowledge in which we are interested, we now wish to store the data in such a way that it does not occupy much disk space and in such a way that we can analyze it. We take advantage of Atre and Hendler's BitMat system[3] which represents triples as a compressed, 3-dimensional bit matrix that can be directly and efficiently queried using simple basic graph patterns. This allows us to reduce space consumption on disk as well as perform more specific queries on the dataset of interest **without** employing a cluster of machines.

## 3 Evaluation

Below, we provide the technical and performance details of using our system to extract data from the BTC2009 dataset and execute queries using the BitMat system. Run times for the individual phases are summarized in Table 1.

| | |
|---|---|
| Inferencing | 349s |
| Extracting/Reducing | 940s |
| BitMat Creation | 25s |
| **TOTAL** | **1,314s (∼22m)** |

**Table 1.** Times for Individual Phases

## 3.1 Inferencing over Large Dataset

Inferencing was done on the Opteron blade cluster at Rensselaer Polytechnic Institute's (RPI) Computational Center for Nanotechnology Innovations[7] (CCNI) using only the large memory machines. Each machine is an IBM LS21 blade server running RedHat Workstation 4 Update 5 with two dual-core 2.6 GHz AMD Opteron processors with gigabit ethernet and infiniband interconnects and system memory of 16 GB. We read and write files to/from the large General Parallel File System[8] (GPFS).

As mentioned, according to [1], the RDFS closures of different sets of data is the same as the RDFS closure for all the sets of data together. Since the BTC2009 dataset is provided as 116 separate files, we computed the RDFS closure of each file separately to get the RDFS closure for the entire dataset, using 32 processors for each file. Although we could have computed the RDFS closure of the entire BTC2009 dataset all at once, doing so would require a large number of processors. Due to queuing contention on the cluster, allocating such a large number of processors is infeasible.

If all the jobs had been run in parallel, it would require 32 * 116 = 3,712 processors. The average time for the full materialization computation takes on average 32.5 seconds per processor. However, high data skew in the BTC2009 dataset results in a minimum processor time of 5.3 seconds and a maximum processor time of 348.1 seconds. Interestingly, all processors except for one finish in less than 120 seconds. The RDFS closure of the BTC2009 dataset (with the entailment exceptions from Section 2.2 and the added ontologies) contains 1,620,437,279 (unique) triples. The original BTC2009 dataset contains 898,966,813 (unique) triples, so 721,470,466 inferences were materialized.

## 3.2 Reducing the Dataset

Given our use case, we are interested in extracting just information about people to produce a reduced dataset. Using the previously discussed parallel query answering system, we can easily extract all people with full names and email addresses with the following basic graph pattern query:

```
PREFIX up: <http://www.cs.rpi.edu/~weavej3/btc2009#>
SELECT ?p ?e ?n WHERE {
    ?p a up:Person ;
       up:email ?e ;
       up:full_name ?n .
}
```

Although we use the SPARQL SELECT syntax to describe the basic graph pattern in this example, we actually write the results to disk as RDF, emulating

[7] http://www.rpi.edu/research/ccni/

[8] http://www-03.ibm.com/systems/clusters/software/gpfs/index.html

a SPARQL CONSTRUCT query for the triples that are matched by the basic graph pattern.

We ran this query on the CCNI's IBM Blue Gene/L supercomputer using 8,192 nodes, each of which has two 700-MHz PowerPC 440 processors and 1,024MB of system memory. We utilize three of the Blue Gene/L's specialized hardware networks: a 175MBps 3-dimensional torus for point-to-point communication, a 350MBps global-collective network, and a global barrier network. Loading all 1.6 billion triples took 62 seconds (representing a rate of almost 26 million triples per second) while executing the query and writing the results to disk took 878 seconds for a total running time of roughly 16 minutes. We note that query execution time depends heavily on the selectivity of the triple patterns that comprise the query.

### 3.3 Analysis of Reduced Dataset

After extracting data from the full dataset, the resulting RDF file consists of 784,783 triples, a much more manageable dataset size for use in further analysis without requiring a large cluster or supercomputer. Stored in the N-Triples serialization, this data occupies 113MB on disk. Using the BitMat system[3], the same data can be stored in an 8MB compressed index along with a 25MB uncompressed dictionary file mapping node values to integer IDs. We performed the conversion of triples to BitMat and all queries on the BitMat on an IBM Thinkpad T60 laptop with a 2GHz Intel T2500 Core Duo Processor having 1GB of RAM running Ubuntu 9.04. The conversion from triples to BitMat took roughly 25 seconds.

We choose two queries to run against the reduced dataset. Suppose we want to look up a person's email address by name. Using Libby Miller as an example, we formulate the following query:

```
PREFIX up: <http://www.cs.rpi.edu/~weavej3/btc2009#>
SELECT ?s ?email
{
    ?s up:full_name "Libby Miller" ;
       up:email ?email .
}
```

This query returns 96 unique results with 3 unique email addresses in 0.101 seconds.

Suppose we want to look up URIs for persons by email address. Considering up:email to be an inverse functional property, such a lookup indicates that some URIs and/or blank nodes represent the same person. Again, using Libby Miller as an example, we formulate the following query:

```
PREFIX up: <http://www.cs.rpi.edu/~weavej3/btc2009#>
SELECT ?uri {
    ?uri up:email <mailto:libby.miller@bristol.ac.uk> .
}
```

This query returns 205 unique results in 0.030 seconds.

## 4 Conclusion

In this paper we have shown that the strength of high performance clusters can be leveraged to provide rapid RDF inferencing and data extraction from large datasets. Using a combination of Opteron blade cluster and Blue Gene/L, we are able to load, materialize inferences, and extract a specific subset from the Billion Triples Challenge dataset in roughly 20 minutes. To our knowledge no other system is able to provide similar performance.

We have also shown that use of a compressed triplestore such as BitMat allows storing the extracted dataset very efficiently while preserving the ability to directly query the data. Query answering with BitMat provides competative performance with memory requirements that make it suitable for use not just on personal computers, but even low powered embedded devices.

In the future, we hope to extend our system in several ways. First, we intend to integrate the inferencing and the query execution into a single system. This would allow query evaluation without requiring the preprocessing step of inference materialization, and reduce the overhead cost of writing the materialized data to disk. We also intend to extend the supported query expressivity from basic graph patterns to support operations such as optional graph patterns and filtering, providing the user with greater latitude in choosing what data subset to extract. Finally, we hope to continue developing BitMat, increasing scalability to allow analysis of much larger data subsets.

## References

1. Weaver, J., Hendler, J.A.: Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples. In: Proceedings of the 8th International Semantic Web Conference. (2009)
2. Weaver, J., Williams, G.T.: Scalable RDF query processing on clusters and supercomputers. In: Proceedings of the 5th International Workshop on Scalable Semantic Web Knowledge Base Systems. (2009)
3. Atre, M., Hendler, J.: BitMat: A Main Memory Bit-matrix of RDF Triples. In: Proceedings of the 5th International Workshop on Scalable Semantic Web Knowledge Base Systems. (2009)