Acacia-RDF: An X10-based Scalable Distributed RDF Graph Database Engine

Miyuru Dayarathna(1,2,6), Yasima Dewmini(2), Isuru Herath(2), Sameera Nandasiri(2), Gayan Mettananda(2), Sanath Jayasena(2), Toyotaro Suzumura (3,4,5,6)

1. WSO2, Inc.
2. University of Moratuwa
3. IBM T.J. Watson Research Center
4. Barcelona Supercomputing Center
5. The University of Tokyo
6. JST CREST

2016 9th International Conference on Cloud Computing (CLOUD)
Introduction

- Data in the form of linked/graph data have become prominent in recent computing applications.

Network of protein interactions in yeast (Roger Guimerà at al, 2006)

DBPedia Ontology Graph (Jisc infoNet)
Resource Description Framework (RDF) is a standard model for meta-data interchange on the web.

Number of database systems have been developed in recent years by both academia and industry to support the need of managing and mining large linked data sets.

www.xml.com/pub/a/2003/02/05/brownsauce.html
Research Problem

1. Current systems are less efficient and less scalable in storing RDF data sets in secondary storage.

   Facebook had 1.09 billion daily active users on average for March 2016 *
   Web graph had 3.5 billion vertices and 129 billion edges in Feb 2015 **

2. How to create disk-based distributed storage for graphs?

3. How to run both RDF specific queries and also conduct graph property exploration on the same RDF data set?

Presentation Outline

• Introduction
• **Research Problem and Proposed Solution**
• Related Work
• Acacia-RDF System Design
• Implementation
  • RDF Data Partitioner and Native Store
  • SPARQL Query Processor
  • Fault Tolerant Execution
• Evaluation
• Conclusions
Proposed Solution and Contributions

- Acacia-RDF: distributed RDF graph database server based on X10
- RDF graphs in Acacia-RDF are partitioned and distributed across the X10 places.

https://github.com/miyurud/acacia
Contributions

- Distributed RDF graph database engine
- X10-based SPARQL Executor
- X10-based Fault Tolerance
- Evaluation
What is RDF?

Resource Description Framework (RDF) is a standard model for data interchange on the web.

Some example sizes of RDF data sets is given in the below table.

<table>
<thead>
<tr>
<th>Name</th>
<th># of Triplets</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBpedia 3.9 English</td>
<td>474M</td>
<td>2.4GB</td>
</tr>
<tr>
<td>Wikidata</td>
<td>367 M</td>
<td>2.1GB</td>
</tr>
<tr>
<td>Freebase</td>
<td>770 M</td>
<td>2.6 GB</td>
</tr>
<tr>
<td>LUBM 200K</td>
<td>48+ Billion</td>
<td></td>
</tr>
<tr>
<td>LUBM 4400K</td>
<td>1.08 Trillion</td>
<td></td>
</tr>
</tbody>
</table>

What is RDF? (Contd.)

Data in RDF are represented as a set of triples, each with a subject, a predicate, and an object.

A collection of such triples is called an RDF graph.

An RDF graph can be visualized as a node and directed-arc diagram where each triple is represented as a node-arc-node link.
RDF Example

<?xml version="1.0" encoding="UTF-8" ?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ub="http://lubm.org#">

</rdf:Description>

<ub:University rdf:about="http://www.University0.edu">
  <ub:name>University0</ub:name>
</ub:University>

<ub:Department rdf:about="http://www.Department0.University0.edu">
  <ub:name>Department0</ub:name>
</ub:Department>

<ub:subOrganizationOf>
  <ub:University rdf:about="http://www.University0.edu"/>
</ub:subOrganizationOf>

</rdf:RDF>

Given above is a sample RDF data set from LUBM 1 Data. There are three predicates as indicated by the arrows.
What is X10?

- X10 is a new programming language.
- Managed X10 was used to develop the Acacia-RDF
  - Interoperability with Java libraries
- Language constructs
  - async, GlobalReference
- Built-in fault tolerance mechanism

Kiyokuni Kawachiya (2014), Writing Fault-Tolerant Applications Using Resilient X10, X10 Workshop 2014
Related Work

• Distributed Graph Databases
  • G* [17]
  • System G [35]
  • GoFFish [21]
  • L.-Y. Ho et al. [13]

**They do not support RDF data storage and processing**

• Distributed RDF Processing Engines
  • DREAM [12] - Does not partition RDF data
  • TriAD [11] - shared-nothing main memory-based architecture
  • Trinity.RDF [28]
  • g-Store [30]

**They only store RDF data**

---


Related Work (Contd.)

- **Graph Partitioning**
  - METIS [15] - We use METIS in Acacia-RDF

- **X10 for Large Graph Data Processing**
  - ScaleGraph [5]
  - XGDBench [6][7]

*Acacia-RDF is the first such attempt in use of X10 language for storage and processing of RDF data.*

---


Overview of Acacia-RDF

Diagram:
- Front-end Client
- Front-end Protocol
- Master
  - Data Loader
  - CSR Converter
  - Vertex Processor
  - Partitioner
- Central Store
- RDF Partitioner
- MetaDB Interface
- SPARQL Executor
- Hybrid Scheduler
- Fault Tolerance
- Back-end
  - Manager
- Back-end Protocol
- Worker
  - File Transfer Service
  - Graph Algorithms
  - Native Store
- Worker Protocol

Introduction > Research Problem > Related Work > System Design > Implementation > Evaluation > Conclusion

29/06/2016 2016 9th International Conference on Cloud Computing (CLOUD) Acacia-RDF
RDF Partitioner and Native Store

Data Extractor → Data File → Jena RDF Parser → Partitioner

Data File → Statements

RDF Partitioner

Data Store
- Vertex Store
- Relationship Store
- Attribute Store

Native Store
Acacia-RDF's Distributed Data Storage

Note: Central store keeps intersecting subgraphs
SPARQL Query Processor

● Executes SPARQL queries on the partitioned RDF data

● ANTLR [1] based SPARQL grammar is used for parsing SPARQL queries

● User specified SPARQL query is transferred to each worker by the Acacia Manager

● Workers run the SPARQL query they received in parallel and returns back the result to Master

● Central stores are contacted as and when required

User specified SPARQL query is sent to each worker by the AcaciaManager.
Workers run the SPARQL query on the partitioned data sets and the results are aggregated at the master.
Pseudo codes for SPARQL query execution is given in the Algorithms 1, 2, and 3.
Fault Tolerant Execution

- We divide the fault tolerance model of Acacia-RDF into two categories based on the location where the fault occurs.
  - Place 0 is alive and some other place is dead.
  - Please 0 is dead and other places are alive.
- We have implemented the replication distribution algorithm and the first scenario of fault tolerance when a place other than Place 0 is dead.
Evaluation

- Three types of experiments were conducted.
- Used LUBM (Lehigh University Benchmark)
- Acacia-RDF was setup in a cluster of 4 computers (Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz, 4 cores, 16GB RAM, 1TB HDD)

- Each running Ubuntu Linux, X10 2.5.2 and JDK 1.7

<table>
<thead>
<tr>
<th>ID</th>
<th>Data set name</th>
<th>Vertices</th>
<th>Edges</th>
<th>File size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_1$</td>
<td>LUBM-5</td>
<td>0.10M</td>
<td>0.83M</td>
<td>51.9MB</td>
</tr>
<tr>
<td>$G_2$</td>
<td>LUBM-10</td>
<td>0.21M</td>
<td>1.70M</td>
<td>105.9MB</td>
</tr>
<tr>
<td>$G_3$</td>
<td>LUBM-20</td>
<td>0.44M</td>
<td>3.59M</td>
<td>224.8MB</td>
</tr>
<tr>
<td>$G_4$</td>
<td>LUBM-40</td>
<td>0.86M</td>
<td>7.10M</td>
<td>445MB</td>
</tr>
<tr>
<td>$G_5$</td>
<td>LUBM-80</td>
<td>1.7M</td>
<td>14.33M</td>
<td>862MB</td>
</tr>
<tr>
<td>$G_6$</td>
<td>LUBM-160</td>
<td>3.6M</td>
<td>28.50M</td>
<td>1.7GB</td>
</tr>
</tbody>
</table>
Evaluation (1) – Performance Comparison with Neo4j

Non-caching setup

Ran Q1 and Q3 LUBM queries on both the systems
Evaluation (1) – Performance Comparison with Neo4j

Caching setup

Acacia-RDF Setup

Host 0
Place 0
CPU 0

Host 1
Place 1
CPU 1

Host 2
Place 2
CPU 2

Host 3
Place 3
CPU 3

Results were cached at the Master

Ran Q1 and Q3 LUBM queries on both the systems

Neo4j Setup

Host 0
CPU 0

Caching enabled

29/06/2016
Evaluation – LUBM Queries

Q1 - It queries about just one class and one property and does not assume any hierarchy information or inference.

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX ub: <http://acacia.org#> SELECT ?X WHERE {?X ub:takesCourse <http://www.Department0.University0.edu/GraduateCourse0> }

Q3 - This query is similar to Query 1 but class Publication has a wide hierarchy.

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX ub: <http://acacia.org#> SELECT ?X WHERE { ?X rdf:type ub:Publication . ?X ub:publicationAuthor <http://www.Department0.University0.edu/AssistantProfessor0> }
Evaluation – Performance Comparison with Neo4j – Contd.

Elapsed Time for LUBM Query 1

Number of universities

- Ne04j-Q1-With Caching
- Acacia-Q1
- Acacia-Q1-With Caching
- Ne04j-Q1

(a)
Evaluation – Performance Comparison with Neo4j – Contd.

Elapsed Time for LUBM Query 1

Acacia-RDF outperforms Neo4j
Evaluation – Performance Comparison with Neo4j – Contd.

Elapsed Time for LUBM Query 3

Number of universities

(b)
Evaluation – Performance Comparison with Neo4j – Results

- Acacia-RDF outperforms Neo4j in certain scenarios
  - Both systems operate without caching mechanism’s help for graphs of size less than LUBM 20
- Communication between the master and the worker increases with increasing LUBM scale
Evaluation (2) – Scalability of LUBM Queries

2 places

4 places
Evaluation (2) – Scalability of LUBM Queries

- Host 0
  - Place 0
  - Place 4
  - CPU 0

- Host 1
  - Place 1
  - Place 5
  - CPU 1

- Host 2
  - Place 2
  - Place 6
  - CPU 2

- Host 3
  - Place 3
  - Place 7
  - CPU 3

8 places

... 8 places...

- Host 3
  - Place 3
  - Place 7
  - Place 11
  - Place 15
  - CPU 4

16 places
Evaluation (2) – Scalability of LUBM Queries

Elapsed Time of SPARQL queries with varying X10 Places

Considerable performance gain when adding more X10 places. However, it is not linear speedup.
No significant performance gain with adding more places.
Evaluation (3) – Fault Tolerance Mechanism of Acacia-RDF

- LUBM-80 (862MB) data set was used for this experiment.
-Observed system characteristics when the system run Q2 on 16 places and suddenly one place crashes.
  - Elapsed time of faulty execution was 51.2 s
  - Elapsed time with 15 live places was 53.4
Conclusion

- Acacia-RDF is a distributed RDF graph database engine.
- Acacia-RDF stores its data in secondary storage.
- Less than 10 seconds elapsed times on 16 Places for running the first and the third queries of the LUBM 160 data set of 1.7GB.
- Acacia-RDF is one of the first such implementation of a scalable RDF data store and a SPARQL processor completely developed in a PGAS language.
Future Work

- Acacia-RDF in hybrid clouds
- Improve fault tolerance mechanism for Acacia-RDF
- Implement multiple other graph algorithms
- Time Evolving graphs
- Detailed performance comparison with other graph database servers
Thanks!