Scalable Ontology Reasoning in Spark Environment

김제민

명지대학교 방목기초교육대학

2015.08.21
Why Scalable Ontology Reasoning

An **Ontology** is an engineering artefact consisting of:
- Shared vocabulary of terms
- An explicit specification their intended meaning
- Inference rules to represent domain-specific knowledge

They are set to play a **key role** in many applications:
- e-Science, Medicine, Databases, Semantic Web, Knowledge Service

These applications require a massive amount of knowledge
→ Need to **reasoning approaches for large scale ontology**
The Problem of General MapReduce Approach

MapReduce is a good approach for big data processing
Acyclic data flow is a powerful abstraction
Not efficient for applications that repeatedly reuse a working set of data
Not efficient for join operation
Alternative Approach using In-memory of Cluster

Spark - in-memory cluster computing for iterative and interactive applications
Loading data as Resilient Distributed Datasets into the memory in a cluster
Executing MapReduce-based operations.

The cost of computer memory is continuously dropping
Most computer systems have increasingly higher capacity memory

Distributed memories in cluster are efficiently utilized
Reasoner can perform faster for scalable ontologies
Outline

• Ontology representation and reasoner

• Scalable Ontology Reasoning in Spark Environment
  • Rule execution order
  • Data layout and operations
  • Reasoning step

• Experimental evaluations
Ontology Representation

RDF - basic language that is used to present ontologies
Originally design as a metadata data model using XML syntax

RDFS - an extension of the RDF
Providing a data-modelling vocabulary for web resources
some semantic rules to perform reasoning

OWL Horst - semantics that are weaker than the standard OWL
The computation of closure has a low complexity
Including more complex semantics than the RDFS
Ontology Representation

Languages to provide more complex semantics are required
OWL - design to represent rich and complex knowledge
OWL Full or DL-based scalable ontology reasoning has an undecidable problem

\[ S \leftarrow \text{intersection } (\cap), \text{union } (\cup), \text{existential } (\exists), \text{universal } (\forall), \text{complement } (\neg), \text{role transitive } (R^+) \]
\[ \mathcal{H} \leftarrow \text{role hierarchies}, \quad \mathcal{I} \leftarrow \text{inverse roles} \]
\[ \mathcal{F} \leftarrow \text{functional roles} \]
## Ontology Representation

### Antecedents

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td><code>p rdfs:domain x, s p o</code></td>
</tr>
<tr>
<td>R2</td>
<td><code>p rdfs:range x, s p o</code></td>
</tr>
<tr>
<td>R3</td>
<td><code>p rdfs:subPropertyOf q, q rdfs:subPropertyOf r</code></td>
</tr>
<tr>
<td>R4</td>
<td><code>s p o, p rdfs:subPropertyOf q</code></td>
</tr>
<tr>
<td>R5</td>
<td><code>s rdf:type x, x rdfs:subClassOf y</code></td>
</tr>
<tr>
<td>R6</td>
<td><code>x rdfs:subClassOf y, y rdfs:subClassOf z</code></td>
</tr>
<tr>
<td>O1</td>
<td><code>p rdf:type owl:FunctionalProperty, u p v, u p w</code></td>
</tr>
<tr>
<td>O2</td>
<td><code>p rdf:type owl:InverseFunctionalProperty, v p u, w p u</code></td>
</tr>
<tr>
<td>O3</td>
<td><code>p rdf:type owl:SymmetricProperty, v p w</code></td>
</tr>
<tr>
<td>O4</td>
<td><code>p rdf:type owl:TransitiveProperty, u p w, w p v</code></td>
</tr>
<tr>
<td>O5</td>
<td><code>v owl:sameAs w</code></td>
</tr>
<tr>
<td>O6</td>
<td><code>v owl:sameAs w, w owl:sameAs u</code></td>
</tr>
<tr>
<td>O7a</td>
<td><code>p owl:inverseOf q, v p w</code></td>
</tr>
<tr>
<td>O7b</td>
<td><code>p owl:inverseOf q, v q w</code></td>
</tr>
<tr>
<td>O8</td>
<td><code>v rdf:type owl:Class, v owl:sameAs w</code></td>
</tr>
<tr>
<td>O9</td>
<td><code>p rdf:type owl:Property, p owl:sameAs q</code></td>
</tr>
<tr>
<td>O10</td>
<td><code>u p v, u owl:sameAs x, v owl:sameAs y</code></td>
</tr>
<tr>
<td>O11a</td>
<td><code>v owl:equivalentClass w</code></td>
</tr>
<tr>
<td>O11b</td>
<td><code>v owl:equivalentClass w</code></td>
</tr>
<tr>
<td>O11c</td>
<td><code>v rdfs:subClassOf w, w rdfs:subClassOf v</code></td>
</tr>
<tr>
<td>O12a</td>
<td><code>v owl:equivalentProperty w</code></td>
</tr>
<tr>
<td>O12b</td>
<td><code>v owl:equivalentProperty w</code></td>
</tr>
<tr>
<td>O12c</td>
<td><code>v rdfs:subPropertyOf w, w rdfs:subPropertyOf v</code></td>
</tr>
<tr>
<td>O13a</td>
<td><code>v owl:hasValue w, v owl:Property p, w p w</code></td>
</tr>
<tr>
<td>O13b</td>
<td><code>v owl:hasValue w, v owl:Property p, r rdf:type v</code></td>
</tr>
<tr>
<td>O14</td>
<td><code>u owl:someValuesFrom w, v owl:Property p, u p x, x rdfs:domain w</code></td>
</tr>
<tr>
<td>O15</td>
<td><code>v owl:allValuesFrom w, v owl:Property p, w rdf:type v, w p x</code></td>
</tr>
</tbody>
</table>

### Consequent

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3</td>
<td><code>p rdfs:subPropertyOf q, q rdfs:subPropertyOf r</code></td>
</tr>
<tr>
<td>R4</td>
<td><code>s q o</code></td>
</tr>
<tr>
<td>R5</td>
<td><code>s rdf:type y</code></td>
</tr>
<tr>
<td>R6</td>
<td><code>x rdfs:subClassOf z</code></td>
</tr>
<tr>
<td>O1</td>
<td><code>v owl:sameAs w</code></td>
</tr>
<tr>
<td>O2</td>
<td><code>v owl:sameAs w</code></td>
</tr>
<tr>
<td>O3</td>
<td><code>u p v</code></td>
</tr>
<tr>
<td>O4</td>
<td><code>u p v</code></td>
</tr>
<tr>
<td>O5</td>
<td><code>w owl:sameAs v</code></td>
</tr>
<tr>
<td>O6</td>
<td><code>v owl:sameAs u</code></td>
</tr>
<tr>
<td>O7a</td>
<td><code>w q v</code></td>
</tr>
<tr>
<td>O7b</td>
<td><code>w p v</code></td>
</tr>
<tr>
<td>O8</td>
<td><code>v rdfs:subClassOf w</code></td>
</tr>
<tr>
<td>O9</td>
<td><code>p rdfs:subPropertyOf q</code></td>
</tr>
<tr>
<td>O10</td>
<td><code>x p y</code></td>
</tr>
<tr>
<td>O11a</td>
<td><code>v rdfs:subClassOf w</code></td>
</tr>
<tr>
<td>O11b</td>
<td><code>w rdfs:subClassOf v</code></td>
</tr>
<tr>
<td>O11c</td>
<td><code>v rdfs:subClassOf w</code></td>
</tr>
<tr>
<td>O12a</td>
<td><code>v rdfs:subPropertyOf w</code></td>
</tr>
<tr>
<td>O12b</td>
<td><code>w rdfs:subPropertyOf v</code></td>
</tr>
<tr>
<td>O12c</td>
<td><code>v rdfs:subPropertyOf w</code></td>
</tr>
<tr>
<td>O13a</td>
<td><code>u rdf:type v</code></td>
</tr>
<tr>
<td>O13b</td>
<td><code>u p w</code></td>
</tr>
<tr>
<td>O14</td>
<td><code>u rdf:type v</code></td>
</tr>
<tr>
<td>O15</td>
<td><code>x rdf:type u</code></td>
</tr>
</tbody>
</table>

---

**RDFS**

**OWL Horst Lite**

**OWL Horst**
## Ontology Reasoner

<table>
<thead>
<tr>
<th>Ontology Reasoner</th>
<th>Development Company/Univ</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FaCT, FaCT++</td>
<td>U. of Manchester</td>
<td>OWL, tableaux algorithm, single machine</td>
</tr>
<tr>
<td>Pellet</td>
<td>U. of Maryland, MIND Lab.</td>
<td>OWL, tableaux algorithm, Ontology debugging, single machine</td>
</tr>
<tr>
<td>Hermit</td>
<td>U. of Oxford</td>
<td>OWL2, hypertableau algorithm, single machine</td>
</tr>
<tr>
<td>AllegroGraph</td>
<td>Franz Inc</td>
<td>RDF++, rule-based reasoning, single machine</td>
</tr>
<tr>
<td>OWLIM</td>
<td>ontotext</td>
<td>OWL-Horst, rule-based reasoning, single machine</td>
</tr>
<tr>
<td>WebPIE</td>
<td>U. of Amsterdam</td>
<td>OWL-Horst, mapreduce, cluster</td>
</tr>
</tbody>
</table>
Tbox & Abox reasoning

**TBox** – the terminology, i.e., the vocabulary of an application domain
= concepts (set of individuals) + roles

**TBox reasoning** - organizing the concepts of a terminology into a hierarchy according to their generality

**ABox** – assertions about named individuals in terms of this vocabulary

**ABox Reasoning** - entailing that a particular individual is an instance of a given concept description
Tbox & Abox reasoning

TBox

- SubclassOf
- SubpropertyOf
- Equivalent Classes
- Equivalent Properties

ABox

- Property Inheritance
- Object Property Domain
- Object Property Range
- Type Inheritance
- Instance Equivalent
- Property Functional & InvFunctional
- Property Symmetric
- Property Transitive
- Property Inverse
- Object SomeValuesFrom
- Object AllValuesFrom
- Object HasValue
- Type IntersectionOf
- Type UnionOf
Ontology Reasoning

Triples
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{car}_02853224 .\]

Reasoned Triples
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{motor_vehicle}_03649150 .\]
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{self-propelled_vehicle}_04011657 .\]
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{wheeled_vehicle}_04398733 .\]
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{vehicle}_04348422 .\]
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{container}_02982528 .\]
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{instrumentality}_03443493 .\]
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{conveyance}_02988377 .\]
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{artifact}_00019244 .\]
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{entity}_00001740 .\]
\[\text{cruiser}_03026980 \quad \text{subClassOf} \quad \text{object}_00016236 .\]
Ontology Reasoning

Triples
UndergraduatedStudent someValuesFrom UndergraduatedCourse
UndergraduatedStudent onProperty hasCourse
PGM hasCourse AI
AI type UndergraduatedCourse

Reasoned Triples
PGM type UndergraduatedStudent
Scalable ontology reasoning approach

- **Approaches of Distributed file**
  - MapReduce approach
    - Building reasoner using MapReduce algorithm
  - SQL approach
    - Hive, Pig
      - Optimized interpreter – automatically generating MapReduce jobs

- **Approaches of Distributed in-memory**
  - SQL approach
    - Impala
  - Spark framework
    - In-memory cluster computing
SPARK: Cluster Computing Platform

Spark - in-memory cluster computing for iterative and interactive applications run programs up to 100x faster than Hadoop MapReduce in memory

Key concepts - Write programs in terms of transformations on distributed datasets

Resilient Distributed Datasets
- Collections of objects spread across a cluster, stored in RAM or on Disk
  Built through parallel transformations

Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)

Apache Spark
Working with RDD

```python
linesWithSpark = textFile.filter(lambda line: "Spark" in line)
linesWithSpark.count()
# Apache Spark
```

```python
textFile = sc.textFile("SomeFile.txt")
```
SPARK based OWL Horst Reasoner

The OWL reasoner –
A driver that performs reasoning on Spark
Containing operations for TBox and ABox reasoning
Defining distributed data sets (ontologies) on a cluster
Applying operations to Spark

1. Connecting to the cluster through a Spark connector
2. Converting ontology triples in HDFS to triple-RDDs
3. Loading the RDD triples to memories in the cluster

The task manager assigns jobs to each worker through the action scheduler.
Each worker performs reasoning independently.
Scalable Ontology Reasoning in Spark Environment

There are 3 key points for scalable ontology reasoning in Spark environment

- The execution order of reasoning rules that takes into account interdependencies and semantic relations.
- The data structure and operations for rule-based reasoning.
- The reasoning algorithms for executing each rule on a distributed memory system.
Rule Dependency of RDFS
Rule Dependency of OWL Horst

Tbox Reasoning

Inferred condition $c \in T$, $c \notin A$
\[ \therefore \text{ first execution} \]

1st Abox Reasoning

Inferred condition $c \in T$, $c \in A$
Result $r \in A$
\[ \therefore \text{ second execution} \]

2nd Abox Reasoning

Inferred condition, $c \in A$
Result $r \in \text{sameAs}$
\[ \therefore \text{ final execution} \]
Rule Dependency of OWL Horst

Tbox Reasoning

1st Abox Reasoning

2nd Abox Reasoning

1st cycle

2nd cycle
N-triple - represents all RDF statements as <subject>, <predicate> and <object> (SPO) statements.

The basic data structure is pair RDDs, composed of key-value pair. converting triples to an RDD that is suitable for executes operations for each reasoning rule.
Data Layout

The eight types of pair RDDs that are classified by features of patterns in OWL Horst rules

<table>
<thead>
<tr>
<th>Form</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type1_pair</td>
<td>&lt;key, value&gt;</td>
</tr>
<tr>
<td>Type2_pair</td>
<td>&lt;key, composed value&gt;</td>
</tr>
<tr>
<td>Type3_pair</td>
<td>&lt;key, &lt;first value, second value&gt;&gt;</td>
</tr>
<tr>
<td>Type4_pair</td>
<td>&lt;composed key, value&gt;</td>
</tr>
<tr>
<td>Type5_pair</td>
<td>&lt;composed key, composed value&gt;</td>
</tr>
<tr>
<td>Type6_pair</td>
<td>&lt;composed key, &lt;composed first value, composed second value&gt;&gt;</td>
</tr>
<tr>
<td>Type7_pair</td>
<td>&lt;key, &lt;first value, composed second value&gt;&gt;</td>
</tr>
<tr>
<td>Type8_pair</td>
<td>&lt;key, &lt;composed second value, second value&gt;&gt;</td>
</tr>
</tbody>
</table>
Data Layout

Subclass RDD    Type1_pair <s, o>

Father    Parents
Parents    Ancestor

SPO RDD    Type2_pair <p, s+t+o>

hasFather    John+t+Park

Domain RDD    Type1_pair <s, o>

hasFather    Father

Type RDD    Type1_pair <o, s>

Person    John

Transitive SPO RDD    Type5_pair <p+t+s, p+t+o>

hasDes+t+kim    hasDes+t+DG
Operations for Ontology Reasoning

Operations are divided into rule-execution and RDD-reformatting categories

**Join** - When there are multiple values for the same key in one of the inputs, the resulting pair RDD will also have multiple entries for the same key.

**Inverse Value-Projection** - This operation switches the position of the first element and the second element in tuples of all values, and make a new pair RDD using both elements.
# Operations for Ontology Reasoning

## The rule-execution category

<table>
<thead>
<tr>
<th>Operation</th>
<th>RDD format before execution</th>
<th>RDD format after execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Join</td>
<td>A = {&lt;a,b&gt;, &lt;b,c&gt;, &lt;c,d&gt;}</td>
<td>B = {&lt;b,a&gt;, &lt;c,b&gt;, &lt;d,c&gt;}</td>
</tr>
<tr>
<td>Swap</td>
<td>{&lt;a,b&gt;, &lt;b,c&gt;, &lt;c,d&gt;}</td>
<td>{&lt;b,a&gt;, &lt;c,b&gt;, &lt;d,c&gt;}</td>
</tr>
</tbody>
</table>
## Operations for Ontology Reasoning

### The RDD-reformatting category

<table>
<thead>
<tr>
<th>Operation</th>
<th>RDD format before execution</th>
<th>RDD format after execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value-Projection</td>
<td>{&lt;b, &lt;c, a&gt;&gt;, &lt;c, &lt;d, b&gt;&gt;}</td>
<td>{&lt;c, a&gt;, &lt;d, b&gt;}</td>
</tr>
<tr>
<td>First-element-Projection</td>
<td>{&lt;rc, &lt;ia, &lt;c, ib&gt;&gt;&gt;}</td>
<td>{&lt;c, ib&gt;}</td>
</tr>
<tr>
<td>Inverse Value-Projection</td>
<td>{&lt;b, &lt;c, a&gt;&gt;, &lt;c, &lt;d, b&gt;&gt;}</td>
<td>{&lt;a, c&gt;, &lt;b, d&gt;}</td>
</tr>
<tr>
<td>Equivalent Value-Projection</td>
<td>{&lt;a, &lt;b, b&gt;&gt;, &lt;b, &lt;a, a&gt;&gt;, &lt;c, &lt;x, y&gt;&gt;}</td>
<td>{&lt;a, b&gt;, &lt;b, a&gt;}</td>
</tr>
<tr>
<td>Key-value replacement</td>
<td>{&lt;p+'t'+s, p+'t'+o&gt;}</td>
<td>{&lt;p, s+'t'+o&gt;}</td>
</tr>
<tr>
<td>Key-value transform</td>
<td>\text{A_RDD} = {&lt;p, a+'t'+b&gt;}</td>
<td>\text{inverseMap} = [[p,q]]</td>
</tr>
<tr>
<td>Subject-key selection</td>
<td>{&lt;p, &lt;a+'t'+b, c&gt;&gt;}</td>
<td>{&lt;c, a&gt;}</td>
</tr>
<tr>
<td>Object-key selection</td>
<td>{p, &lt;a+'t'+b, c&gt;}</td>
<td>{&lt;c,b&gt;}</td>
</tr>
<tr>
<td>Composed value to single key</td>
<td>{&lt;c, &lt;p, i&gt;&gt;}</td>
<td>{&lt;p,c&gt;}</td>
</tr>
<tr>
<td>Composed value transform</td>
<td>{&lt;c, &lt;ib, &lt;p, ia&gt;&gt;&gt;}</td>
<td>{&lt;p, ib+'t'+ia&gt;}</td>
</tr>
<tr>
<td>recomposed value to tuple</td>
<td>{&lt;p, &lt;c+'t'+rc, ia+'t'+ib&gt;&gt;}</td>
<td>{&lt;rc, &lt;c, ia&gt;&gt;}</td>
</tr>
</tbody>
</table>
Reasoning Cases

A subclass B
B subclass C
C subclass D

A subclass C
A subclass D
B subclass D

Key value
A B
B C
C D

Swap

Key value
B A
C B
D C

Union

Key value
B <A, D>
C <D, B>

Inverse Value-Projection

Key value
A D
B D
Reasoning Cases

\[ A \equiv \exists R.C, \ inst2 \in C, \ R(inst1, inst2) \]

- **onPropRDD**
  - key | value
  - anon1 | R
  - ... | ...

- **someRDD**
  - key | value
  - anon1 | C
  - ... | ...

- **inferredTypeRDD**
  - key | value
  - anon1 | inst1
  - ... | ...

- **onPropSPORDD**
  - key | value
  - R | inst1+"t"+inst2
  - ... | ...

- **onPropTypeRDD**
  - key | value
  - C+"t"+inst2 | <anon1, inst1>
Reasoning step

- Read triples
  - Make scheme array
    - Functional prop
    - Inverse Functional Prop
    - Transitive prop
    - Symmetric prop
    - onProperty
  - Make scheme map
    - Inverse prop
  - Broadcast scheme data
  - Divide Rdds

- Execute class hierarchy rule
  - Execute property hierarchy rule
  - Execute transitive rule
    - Execute symmetric rule
    - Execute inverse rule
  - Execute property inherence rule
  - Execute domain, range rule
  - Execute allValue rule
  - Execute someValue rule
  - Execute hasValue (O16) rule
  - Execute hasValue (O17) rule
  - Execute type inherence rule
Experimental Evaluation - RDFS

- Test data: LUBM
- Reasoning level of Ontology: RDFS
- Test goal: Reasoning Speed
- Test Environment: 1 master node (8 Cores with 12GByte Ram), 7 worker nodes (8 Cores with 98GByte Ram)

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Reasoning Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hive</td>
</tr>
<tr>
<td></td>
<td>Time (Min)</td>
</tr>
<tr>
<td>Ontology</td>
<td>Number of Triples (Million)</td>
</tr>
<tr>
<td>LUBM1000</td>
<td>137</td>
</tr>
<tr>
<td>LUBM2000</td>
<td>275</td>
</tr>
<tr>
<td>LUBM3000</td>
<td>413</td>
</tr>
</tbody>
</table>
Experimental Evaluation – OWL-horst, \textit{SHIF}

The environment of experiment –
- A cluster with \textit{one master} and \textit{four workers}
- 2.6 GHz processor (8 cores) with \textit{64 GB} of main memory
- 2 TB of hard disk

\textbf{WebPIE} - representative ontology reasoner based on MapReduce in Hadoop.

\textbf{LUBM} - formal benchmark dataset for evaluating ontology inferences and search.
Experimental Evaluation – OWL-horst

Time(sec) Reasoning Time

- Our Approach
- WebPIE(RDFS Reasoning)
- WebPIE(OWL Horst Reasoning)

<table>
<thead>
<tr>
<th>Number of triple</th>
<th>Time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>113M</td>
<td>Over 20,000</td>
</tr>
<tr>
<td>275M</td>
<td>Over 20,000</td>
</tr>
<tr>
<td>413M</td>
<td>Over 20,000</td>
</tr>
<tr>
<td>550M</td>
<td>Over 20,000</td>
</tr>
<tr>
<td>715M</td>
<td>Over 20,000</td>
</tr>
<tr>
<td>860M</td>
<td>Over 20,000</td>
</tr>
<tr>
<td>1.15B</td>
<td>Over 20,000</td>
</tr>
</tbody>
</table>

- Not enough memory
Experimental Evaluation - $SHIF$

Time(sec) vs Reasoning Time

- **Our Approach**
- **WebPIE(RDFS Reasoning)**
- **WebPIE(OWL Horst Reasoning)**

<table>
<thead>
<tr>
<th>Number of triple</th>
<th>113M</th>
<th>275M</th>
<th>413M</th>
<th>550M</th>
<th>715M</th>
<th>860M</th>
<th>1.15B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sec)</td>
<td>951</td>
<td>2195</td>
<td>2199</td>
<td>3997</td>
<td>3507</td>
<td>5843</td>
<td>6561</td>
</tr>
<tr>
<td>Reasoning Time</td>
<td>Over 20,000</td>
<td>Over 20,000</td>
<td>Over 20,000</td>
<td>Over 20,000</td>
<td>Over 20,000</td>
<td>Over 20,000</td>
<td>Over 20,000</td>
</tr>
</tbody>
</table>

- Not enough memory

[(Experimental Evaluation - SHIF)](image)
Thank You!

kimjemins@hotmail.com