Optimizations for Parallel, Rule-based Inference on RDF Data

2011 IBM Student Workshop for Frontiers of Cloud Computing

Jesse Weaver (@jrweave)
Tetherless World Constellation
Rensselaer Polytechnic Institute
Troy, NY, USA
1. Motivation: Data Integration
2. Background
   - Semantic Web: RDF and Ontology
   - Logic: “Rules” down to Datalog
3. Parallelism
   - Parallel Datalog Programs
   - Known Parallel Rule Sets in Semantic Web
   - Evaluations and Issues
4. Data Compression (time permitting)
   - Syntactic and Binary
   - Evaluation
5. Related Work
6. Conclusion
1. Motivation

• What does this have to do with cloud computing?

• RDF and related semantic web standards seek to solve the problem of meaningful, data integration.

• Undoubtedly, data integration is also a problem in cloud computing, particularly in forming a platform (PaaS) over multiple data sources.
2. Background

- Semantic Web
  - RDF
  - Ontology
- Logic
  - “Rules”
  - Datalog
It’s about a common view and understanding of data.

- Formalism
- Questions
- Sentences
- Alphabet
- Words

“Ontology”

Query

Data Model

Syntax

Names

User Interface & Applications

Trust

Proof

Unifying Logic

Ontology:

- OWL
- RIF

Rule:

RDFS

Data interchange:

RDF

XML

URI/IRI

Crypto
RDF Terms

**URIs**

<http://www.cs.rpi.edu/~weavej3/foaf.rdf#me>

**Blank nodes**

_:label

**Literals**

“plain literal”

“plain literal with language tag”@en-US

“true”^^<http://www.w3.org/2001/XMLSchema#boolean>

URIs often abbreviated with defined prefix.

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

“true”^^xsd:boolean
RDF Data Model

Triples

:me rdf:type foaf:Person .
:me foaf:name "Jesse Weaver" .
:me foaf:knows jim:jhendler .
jim:jhendler foaf:name "James Hendler" .

Can also view as a labeled, directed graph.
Ontology

• What is it?

“An explicit formal specification of how to represent the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them.”

• The primary purpose of an ontology is to attach explicit semantics to data.

• Ontologies can then be interrelated to facilitate data integration.
Ontology

• Potentially many ways to specify an ontology.

• Two predominate mechanisms used in the semantic web:
  – Description Logics
  – “Rules”

• Herein, we focus on “rules.”
2. Background

• Semantic Web
  – RDF
  – Ontology

• Logic
  – “Rules”
  – Datalog
Many kinds of rules. For example:
- First-Order Logic
- Logic Programming (Horn clauses)
- Production Rules (rules with actions)

Herein, we’re interested in Datalog, a subset of Logic Programming.
- Why Datalog? Amenable to parallelization.
  - No retraction of facts.
  - No negation.
  - Data-oriented.
  - Decidable, fixpoint semantics.
A Horn clause has the form
\[ L_0 :\text{\texttt{-}} L_1, \ldots, L_n \]
where each \( L_i \) is a literal of the form
\[ p_i(t_1, \ldots, t_{\text{arity}(p_i)}) \]
where \( p_i \) is a predicate and each \( t_j \) is a constant or variable.

A fact (clause with empty body) must not contain variables.

Variables in head of rule (clause, not a fact) must appear in body of the rule.
Let $r$ be a rule.

Let $r(F)$ denote the set of all inferences derivable from facts $F$ using only rule $r$, without adding any inferences back into $F$. Call $r(F)$ the \textit{application} of $r$ to $F$.

Let $F \cup r(F)$ be called a \textit{step} of $r$ on $F$. 
Given a Datalog program $P=R \cup F$, let

$$ap(P) = \bigcup_{r \in R} r(F)$$

$$st^1(P) = P \cup ap(P)$$

$$st^i(P) = st^1(st^{i-1}(P))$$

The closure of $P$ is

$$Cl(P) = st^m(P)$$

where $st^m(P) = st^{m+1}(P)$
3. Parallelism

- Parallel Datalog Programs
  - Parallel Closure Problem
  - Devising a Distribution Scheme
- Known Parallel Rules Sets in SemWeb
  - Minimal RDFS
  - Finite RDFS
  - OWL 2 RL Subset
- Evaluations and Issues
  - ISWC 2009
  - BTC 2009
Parallel Closure Problem

For a program $P$, is there a “good” set of programs

$$\{P_1, \ldots, P_n\}$$

such that

$$P = \bigcup_i P_i$$

$$\text{Cl}(P) = \bigcup_i \text{Cl}(P_i)$$

Say that such a set of programs closes $P$ in parallel.
“Good” Set of Programs

In an ideal world:

\[ \forall i,j \quad |P_i \cap P_j| = 0 \quad \text{(duplicate nothing)} \]
\[ \forall i \quad |P_i| = \frac{|P|}{n} \quad \text{(equal division)} \]

But we’ll settle for:

\[ \forall i,j \quad |P_i \cap P_j| \ll |P| \quad \text{(duplicate a little)} \]
\[ \forall i \quad |P_i| \approx \frac{|P|}{n} \quad \text{ (“even” division)} \]
In the presence of large data (like in semantic web), it is often the case that there are far fewer rules than facts, so it’s okay to replicate all the rules.

\[ |R| \ll |F| \leq |P| \]

The hard part: how do we distribute the facts? There’s lots of them, and they’re very diverse. Few assumptions can be made, if any.
Parallel Joins

- To satisfy the rule bodies, joins are performed.
- When it comes to parallel joins, there are two ends of the spectrum.

\[ S \bowtie T \]

- Hash joins assign facts to processes by hashing on join variables. Thus communication may be required after each step for hash distribution of inferences.

\[ \bigcup_i S_i \bowtie T_i \]

\[ \bigcup_i S \bowtie T_i \]
Replication Joins

- With replication joins, as long as one side is replicated, it doesn’t matter how the other side is distributed. (hint)
- If we can make sure that the results of the join are input only into the non-replicated side of any other join, then no communication will be required between steps.

\[ S \bowtie [ \bigcup_i T \bowtie U_i ] \]

- We must make sure this condition is satisfied in two places:
  - Rule Application: querying facts to satisfy rule bodies.
  - Between Steps: inference placement.
Example Rules

\[ I(\text{?s}, \text{?p}, \text{?o}) :- \\
I(\text{?q}, \text{rdfs:subPropertyOf}, \text{?p}), \\
I(\text{?s}, \text{?q}, \text{?o}) . \]

\[ I(\text{?q}, \text{rdfs:subPropertyOf}, \text{?p}) :- \\
I(\text{?x}, \text{rdfs:subPropertyOf}, \text{?p}), \\
I(\text{?q}, \text{rdfs:subPropertyOf}, \text{?x}) . \]

\[ I(\text{?s}, \text{?p}, \text{?o}) :- \\
T(\text{?s}, \text{?p}, \text{?o}). \]
Example Replication

For each rule, say the rule has \( k \) subqueries in its body. For at least \( k-1 \) subqueries, the associated facts should be replicated. Choose wisely.

\[
I(?s, ?p, ?o) :-
I(?q, rdfs:subPropertyOf, ?p),
I(?s, ?q, ?o) .
\]

\[
I(?q, rdfs:subPropertyOf, ?p) :-
I(?x, rdfs:subPropertyOf, ?p),
I(?q, rdfs:subPropertyOf, ?x) .
\]
Inference Placement

Inferences coming from replication joins (in which one side is *not* known to be replicated) must not need replication.

(What if \( ?p = \text{rdfs:subPropertyOf} \))

\[
\begin{align*}
I(?s, ?p, ?o) & :- \\
I(?q, \text{rdfs:subPropertyOf}, ?p), \\
I(?s, ?q, ?o).
\end{align*}
\]

\[
\begin{align*}
I(?q, \text{rdfs:subPropertyOf}, ?p) & :- \\
I(?x, \text{rdfs:subPropertyOf}, ?p), \\
I(?q, \text{rdfs:subPropertyOf}, ?x).
\end{align*}
\]
Possible Solutions

• Restrict the rule.

\[
I(?s, ?p, ?o) :-
I(?q, rdfs:subPropertyOf, ?p),
?p != rdfs:subPropertyOf,
I(?s, ?q, ?o) .
\]

• Refine distribution scheme.

If known a priori:
\[
T(A, rdfs:subPropertyOf, rdfs:subPropertyOf)
\]
Also replicate facts of the form:
\[
T(?s, A, ?o)
\]

• Admit conditional completeness.

Completeness is guaranteed only when there are no facts of the form:
\[
T(?q, rdfs:subPropertyOf, rdfs:subPropertyOf)
\]
3. Parallelism

• Parallel Datalog Programs
  – Parallel Closure Problem
  – Devising a Distribution Scheme

• Known Parallel Rules Set in SemWeb
  – Minimal RDFS
  – Finite RDFS
  – OWL 2 RL Subset

• Evaluations and Issues
  – ISWC 2009
  – BTC 2009
## Minimal RDFS

- Replicate triples with predicate `rdfs:domain`, `rdfs:range`, `rdfs:subClassOf`, or `rdfs:subPropertyOf`.
- Complete only if no triple of the form $T(?p, rdfs:subPropertyOf, A)$, where $A$ is one of the URIs mentioned above.

<table>
<thead>
<tr>
<th>PROPERTY DOMAIN</th>
<th>PROPERTY RANGE</th>
</tr>
</thead>
</table>
| $T(?u, rdf:type, ?x) :-$ | $T(?v, rdf:type, ?x) :-$

<table>
<thead>
<tr>
<th>SUBCLASS TRANSITIVITY</th>
<th>CLASS INHERITANCE</th>
</tr>
</thead>
</table>
| $T(?u, rdfs:subClassOf, ?x) :-$ | $T(?u, rdf:type, ?x) :-$

<table>
<thead>
<tr>
<th>SUBPROPERTY TRANSITIVITY</th>
<th>SUBPROPERTY INHERITANCE</th>
</tr>
</thead>
</table>
| $T(?u, rdfs:subPropertyOf, ?x) :-$ | $T(?u, ?b, ?y) :-$
Finite RDFS

– Replicate triples if:
  - Predicate is rdfs:domain, rdfs:range, rdfs:subClassOf, or rdfs:subPropertyOf.
  - Predicate is rdf:type and object is rdfs:Class, rdfs:ContainerMembershipProperty, or rdfs:Datatype.

– Complete if, aside from axiomatic facts:
  - rdfs:domain, rdfs:range, rdfs:subClassOf, and rdfs:subPropertyOf appear only in predicate position.
  - rdfs:Class, rdfs:ContainerMembershipProperty, and rdfs:Datatype appear only in object position of triples with predicate rdf:type.
OWL 2 RL Subset

- **Unrestricted use of:**
  - owl:IrreflexiveProperty

- **Slightly restricted use of:**
  - rdfs:domain
  - rdfs:range
  - rdfs:subClassOf
  - rdfs:subPropertyOf
  - owl:IrreflexiveProperty
  - owl:SymmetricProperty
  - owl:equivalentProperty
  - owl:equivalentClass
  - owl:inverseOf
  - owl:hasValue
  - owl:onProperty

- **Significantly restricted use of:**
  - owl:someValuesFrom
  - owl:allValuesFrom

- **Modifications to list structure allow for restricted use of:**
  - owl:AllDifferent
  - owl:intersectionOf
  - owl:unionOf
  - owl:oneOf
3. Parallelism

- Parallel Datalog Programs
  - Parallel Closure Problem
  - Devising a Distribution Scheme
- Known Parallel Rules Sets in SemWeb
  - Minimal RDFS
  - Finite RDFS
  - OWL 2 RL Subset

- Evaluations and Issues
  - ISWC 2009
  - BTC 2009
ISWC 2009 Evaluation

The diagram shows the relationship between the number of processes and the time taken. Each line represents a different dataset, with the legend indicating the specific dataset size (e.g., L10k/1024, L10k/512, etc.). As the number of processes increases, the time taken decreases, indicating improved performance with parallel processing.
Potential Problem
BTC 2009

[Graph showing the fraction of inferences produced over time (seconds)]

- Vertical axis: Fraction of Inferences Produced
- Horizontal axis: Time (seconds)
4. Compression

- **Syntax Compression**
  - N-Triples and Turtle
  - Sterno, for syntactic compression

- **Binary Compression**
  - LZO compression
  - Parallelizing LZO decompression

- **Evaluation**
  - Compression
  - Read times
N-Triples and Turtle

@prefix mine: <file:///foaf.rdf#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
mine:me a foaf:Person ;
    foaf:nick "André" ;
    foaf:age 40 .
("line1\n	line2 "quoted string\" ") a rdf:List .
[] a <http://www.w3.org/2002/07/owl#Thing> .
# What a contrived triple.

<file:///foaf.rdf#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<file:///foaf.rdf#me> <http://xmlns.com/foaf/0.1/nick> "André" .
<file:///foaf.rdf#me> <http://xmlns.com/foaf/0.1/age> "40"^^<http://www.w3.org/2001/XMLSchema#integer> .
_:list <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/1999/02/22-rdf-syntax-ns#List> .
_:list <http://www.w3.org/1999/02/22-rdf-syntax-ns#first> "line1\n	line2 "quoted string\" ".
_:list <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest> <http://www.w3.org/1999/02/22-rdf-syntax-ns#nil> .
_:contrived <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Thing> .
# What a contrived triple.
Use chunking with N-triples.

```
<file:///foaf.rdf#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<file:///foaf.rdf#me> <http://xmlns.com/foaf/0.1/nick> "Andr\u00E9" .
<file:///foaf.rdf#me> <http://xmlns.com/foaf/0.1/age> "40"^^<http://www.w3.org/2001/XMLSchema#integer> .
_:list <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/1999/02/22-rdf-syntax-ns#List> .
_:list <http://www.w3.org/1999/02/22-rdf-syntax-ns#first> "line1
	line2 "quoted string" " .
_:list <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest> <http://www.w3.org/1999/02/22-rdf-syntax-ns#nil> .
_:contrived <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Thing> .
# What a contrived triple.
```
Prefix Declarations

When is it beneficial to declare?

@prefix X: <U>.

Suppose the namespace URI $U$ occurs $m$ times. Then, the above prefix declaration is beneficial only when:

$$|@prefix X: <U> .\n| + m^*|X:| < m^*|<U>|$$

Simplifying, we derive:

$$\frac{(15 + |U| + |X|)(1 + |U| - |X|)}{1 + |U| - |X|} < m$$

As an example, let $U$=http://xmlns.com/foaf/0.1/ and $X$=foaf. Then, $m$ must be greater than $45/23= \approx 1.957$. In other words, unless it occurs at least 2 times, there is no gain in compression by declaring “foaf” as a prefix for the FOAF namespace URI.
4. Compression

- Syntax Compression
  - N-Triples and Turtle
  - Sterno, for syntactic compression

- Binary Compression
  - LZO compression
  - Parallelizing LZO decompression

- Evaluation
  - Compression
  - Read times
LZO Compression

LZO is a block-level compression which allows for parallel compression.

![Diagram of LZO compression]
Indexed LZO Decompression

Indexing LZO-compressed documents allows for parallel decompression.

LZO is designed for fast decompression.
4. Compression

• Syntax Compression
  – N-Triples and Turtle
  – Sterno, for syntactic compression

• Binary Compression
  – LZO compression
  – Parallelizing LZO decompression

• Evaluation
  – Compression
  – Read times
### Document Sizes

#### Table

<table>
<thead>
<tr>
<th>File Type</th>
<th>No LZO</th>
<th>LZO</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT</td>
<td>100%</td>
<td>9.2%</td>
</tr>
<tr>
<td>ST[0]</td>
<td>94%</td>
<td>9.0%</td>
</tr>
<tr>
<td>ST[10]</td>
<td>53%</td>
<td>8.1%</td>
</tr>
<tr>
<td>ST[1K]</td>
<td>50%</td>
<td>7.9%</td>
</tr>
<tr>
<td>ST[100K]</td>
<td>43%</td>
<td>7.7%</td>
</tr>
<tr>
<td>ST[N]</td>
<td>42%</td>
<td>7.7%</td>
</tr>
</tbody>
</table>

#### Bar Chart

- **No LZO**
- **LZO**

![Bar Chart](image-url)
Namespace URIs

2,354,435 namespace URIs occur exactly 1 time:
http://www.last.fm/user/zzzzz7/#
http://www.bibsonomy.org/uri/bibtex/e0b97fa77c4aee155bcc3ec0e646de2d/

4,535 namespace URIs occur exactly 100 times:
http://www.uniprot.org/uniprot/Q22XT6.rdf#
http://www.uniprot.org/uniprot/Q20V25.rdf#
http://www.uniprot.org/uniprot/Q20UH5.rdf#

368 namespace URIs occur exactly 12,085 times:

The OWL namespace URI is the only namespace URI occurring exactly 21,087,397 times.
The RDF namespace URI is the only namespace URI occurring exactly 37,847,462 times.
http://openean.kaufkauf.net/id/ is the single most commonly occurring namespace URI, 969,900,279 times
Namespace URIs

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31,124</td>
<td>26,802,133</td>
<td>452,557,723</td>
<td>5,148,200,489</td>
</tr>
</tbody>
</table>

# Occurrences

# Namespace URIs
Times (with LZO)

- NT
- ST[0]
- ST[10]
- ST[1K]
- ST[100K]
- ST[N]

Time (min)

# Processes

8  16  32  64

100.0
10.0
1.0
5. Related Work

- **Weaver and Hendler. Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples. ISWC 2009.**
- Urbani et al. Scalable Distributed Reasoning using MapReduce. ISWC 2009.
- Urbani et al. OWL reasoning with WebPIE: calculating the closure of 100 billion triples. ESWC 2010.
- Goodman and Mizell. Scalable In-memory RDFS Closure on Billions of Triples. SSWS 2010.
- Hogan et al. SAOR: Template Rule Optimisations for Distributed Reasoning over 1 Billion Linked Data Triples. ISWC 2010.
6. Conclusion

- A little semantics combined with a lot of data goes a long way in meaningful data integration.
- Presented a way to determine whether a rule set is parallelizable with respect to a (simple) partitioning scheme.
  - And if not, some measures to take.
- There exist “useful” rule sets for which closure can be produced in parallel.
- Evaluated performance gains for two different compression schemes on raw RDF data.
Thank you!

Questions?