Scalable Ontological Query Processing over Semantically Integrated Life Science Datasets using MapReduce

HyeongSik Kim and Kemafor Anyanwu
Semantic Computing Research Lab
{hkim22, kogan}@ncsu.edu
Life Science Data Processing

✓ Multiple heterogeneous (sub) datasets are loosely interconnected.
  ✓ Some are very big (e.g., Uniprot core: 6B triples.)

✓ Use rich, complex ontologies.
  ✓ Some Uniprot hierarchies up to 25 levels vs. 6 in DBpedia.
  ✓ BSBM - simple ontology, no hierarchies.

✓ A large #facts are “implicitly” represented!

✓ To gain the full benefit, ontological query support.
  ✓ inferencing

*Diagram: Joanne Luciano, Predictive Medicine; Drug discovery demo using RDF, Sideran Seamark and Oracle 10g
Graph Pattern

GP

?PrID

Type

protein

Amyloid beta A4 protein

interaction

Name

P05067

recommendedName

P05067

RecommendedName

P51693

interaction

Name

Amyloid beta A4 protein

isolatedFrom

Protein

isolatedFrom

tissues/80

Protein

classifiedWith

go/0048629

Protein

classifiedWith

Amyloid beta A4 protein

isoatedFrom

amyloid beta A4 protein
we need to transform a given query pattern into a union of alternative patterns (UCQs) derived from inferencing:

Rich ontologies produce many alternatives, UCQs of high width.
- queries with rdf:type Taxon and Protein
  \[ 12 \text{ (Protein)} \times 16 \text{ (Taxon)} = 192 \text{ alternative patterns}. \]
- Deep subclassOf/subProperty hierarchies
  \[ > \text{several hundred (}> 800) \text{ alternatives} \]
Challenge: Scalable Inferencing

- Inferencing Techniques
  - Forward-Chaining
  - Backward-Chaining
  - Query Rewritings (UCQs)

- ✓ Pre-materialize entailed facts.
  - ✓ Expensive.
    - ✓ BigOWLIM: 290 hours for 12B triples*
  - ✓ Orders-of-magnitude blowup.
  - ✓ Does not handle updates gracefully.

- ✓ Done at query time.
  - ✓ Difficult to scale up.

- ✓ Can be processed on DBs.
  - ✓ Theoretically
  - ✓ But for rich ontologies, e.g., life sciences, too complex for DBs.

* http://www.w3.org/wiki/LargeTripleStores
Life Sciences Datasets Are Also Big!!

- May not necessarily want to or be able to host all the data for your analysis.
  - E.g., do some integrated analysis of your bio data with some publically available chem. data.
  - Lease computational service from cloud

Cloud usage model is very different.

- Have to pay for time, long preprocessing may not be worth it
  - RDF-3x : 55 hours for 3B triples.*
- So you won't have indexes, statistics etc
- But accept longer query times (but rapidly improving!)

Data Processing on Hadoop

HDFS
Hadoop Distributed File System

Job Tracker

Slaves

Mapper1
map()

(k1, v1)

Disk

Mapper2
map()

(k1, v2)

(k2, v5)

Disk

MapperM
map()

(k1, v3)

(k2, v4)

Disk

Sort / Shuffle

Reducer1
reduce()

(k1, [v1, v2, v3])

(k1, val)

ReducerR
reduce()

(k2, [v4, v5])

(k2, val)

Output

The cost of 1 MR cycle
= \(\text{Map}_{\text{Read}} + (\text{Map}_{\text{Write}} + \text{MR}_{\text{Sort}} + \text{MR}_{\text{Tr}}) + \text{Reduce}_{\text{Write}}\)

Many Operations \(\rightarrow\) Many MR cycles \(\rightarrow\) High Execution Cost
Impact of Work

- Evaluate high-width UCQs on MapReduce.
  - Results in paper - ~ 7X performance improvement for very basic UCQs
  - Newer results order-of-magnitude performance increase (vs. Apache Hive)

- Offers a new tool (algebraic optimization) to add to the query optimization toolbox.
  - Applicable to real time query systems as well
One branch of UCQs

SELECT * WHERE {
    ?p type Protein .
    ?p organism ?o .
    ?o type Taxon .
    ?o commonName ?n .
    ?o subClassOf ?c .
}

UNION
{
    ...\( \sigma_{(P=type \land O=Protein)}(T) \) \( \sigma_{(P=organism)}(T) \)
}

➢ For each branch, graph pattern requires many operations (joins)
Challenge: UCQs Execution

- #branches is up to hundreds for complex queries.
- UCQs are sequentially executed across nodes.
  - Executing a MR job is expensive.

```
SELECT * WHERE {
  # branch #1
  { ?p type Protein .
    ?p organism ?o .
  }
  UNION
  # branch #2
  { ?o type Taxon .
    ?o commonName ?n .
    ?o subClassOf ?c .
  }
  UNION
  # branch #3
  { ?p annotation ?x .
    ?p organism ?o .
  }
  UNION
  ...
}
```

192 branches * 3 MR job + 1 = 577 MR jobs
Another Way of Looking at Things

- Queries = manipulations on “groups of triples”
  - *Triplegroups* become first class objects in data model
  - Focus on groups with same subject resource (star subgraphs)
  - Operations on triplegroups (Nested TripleGroup Algebra – NTGA)

**A SINGLE operation - GROUP BY on subject column**
- (irrespective of number of star patterns)!

---

**RDF Triple Relation (T)**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>P05067</td>
<td>type</td>
<td>Protein</td>
</tr>
<tr>
<td>P05067</td>
<td>organism</td>
<td>Homo ...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>9606</td>
<td>type</td>
<td>Taxon</td>
</tr>
<tr>
<td>9606</td>
<td>commonName</td>
<td>Species</td>
</tr>
<tr>
<td>9606</td>
<td>subClassOf</td>
<td>9605</td>
</tr>
<tr>
<td>9606</td>
<td>subClassOf</td>
<td>207598</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Groups of Triple or “TripleGroups”**

- \(tg_1\):
  - \(P05067, (\text{type, (organism, Protein) Homo...})\) \(\rightarrow SJ1\)

- \(tg_2\):
  - \(9606, (\text{type, (commonName, Human) (subClassOf, 9605) (subClassOf, 207598)})\) \(\rightarrow SJ2\)

**Graph Pattern Query (Q)**

- \(\text{SELECT * WHERE \{ ?p type Protein . \}}\)
- \(\text{?p organism ?o .}\)
- \(\text{?o type Taxon .}\)
- \(\text{?o commonName ?n .}\)
- \(\text{?o subClassOf ?c .}\)

---

**Computer Science**

NC State University
TripleGroups vs. Tuples

<table>
<thead>
<tr>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>P05067</td>
<td>type</td>
<td>Protein</td>
</tr>
<tr>
<td>P05067</td>
<td>organism</td>
<td>Homo ...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>9606</td>
<td>type</td>
<td>Taxon</td>
</tr>
<tr>
<td>9606</td>
<td>commonName</td>
<td>Species</td>
</tr>
<tr>
<td>9606</td>
<td>subClassOf</td>
<td>9605</td>
</tr>
<tr>
<td>9606</td>
<td>subClassOf</td>
<td>207598</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Using NTGA

- Fewer operations
- Smaller footprint of intermediate results!

Relational join representation (type \( \bowtie \) commonName \( \bowtie \) subClassOf)

- (9606, type, Taxon, commonName, Human, subClassOf, 9605)
- (9606, type, Taxon, commonName, Human, subClassOf, 207598)

\[\cong \text{(content-equivalent)}\]
Computing with Triplegroups
Star-join = Filter on Triplegroups

➢ Group By (TG_GroupBy) does not enforce structural constraints.
  ✓ TG_GroupFilter ($\gamma^\sigma$) – select triplegroups that satisfy at least one of the star pattern

Groups of Triple or “TripleGroups”

\[
tg_2 = \{9606, \{(type, \text{Taxon})\}, \{(commonName, \text{Human})\}, \{(subClassOf, 9605)\}, \{(subClassOf, 207598)\}\}
\]

Graph Pattern Query (Q)

\[
\text{SELECT} * \text{WHERE} \{
\begin{align*}
\text{SJ1} & \{ \text{?p type Protein} \}, \\
\text{SJ2} & \{ \text{?o type Taxon} \}, \\
\text{SJ2} & \{ \text{?o commonName ?n} \}, \\
\text{SJ2} & \{ \text{?o subClassOf ?c} \}
\end{align*}
\}
\]

\[
\text{TG}\_\text{GroupFilter}(S_{J1} \text{ OR } S_{J2})(TG)
\]

➢ Other useful operators – TG_Join (“join” between triplegroups)
NTGA Logical-to-MR Plan for a Single Graph Pattern

**Graph Pattern Query (Q)**

```sparql
SELECT * WHERE {
  ?p type Protein .
  ?p organism ?o .
  ?o type Taxon .
  ?o commonName ?n .
  ?o subclassOf ?c .
}
```

**MR Job 1**

- `TG_LoadFilter((p = type ∧ o = Taxon) ∨ ...)(T)`
  (Load triples + property-based filtering)
- `TG_GroupFilter(SJ1 OR SJ2)`
  (Structure-based Filtering)
- `TG_GroupBy(S)`
  (Group triples based on Subject column)

**MR Job 2**

- `TG_Join(SJ1, SJ2)`
  (Join between TripleGroups)

- **The max. # of MR jobs**: $n$ in NTGA vs. $(2n-1)$ in relational-style interpretation. ($n = \#$ of star subpatterns)
- **Shorter execution workflows**: a smaller amount of disk/network I/Os ➔ faster query execution
UCQs Execution in NTGA

✓ Include all star patterns as disjuncts to TG_GroupFilter.

SELECT * WHERE {}

SJ1

?p type Protein.
?p organism ?o.

SJ2

?o type Taxon.
?o commonName ?n.
?o subClassOf ?c.

SJ3

?p annotation ?x.
?p organism ?o.

SJ4

?o type Taxon.
?o commonName ?n.
?o subClassOf ?c

MR Job 1

TG_GroupFilter(sj1 OR sj2 OR sj3 OR sj4 OR...)

TG_GroupBy(s)

TG_LoadFilter((p = type ∧ o = Protein) ∨ ...) (T)
UCQs Execution in NTGA (Cont.d)

✓ Need to join (connect) stars in each UNION branch.
✓ Optimize by query rewriting using special “union-pushdown”.
  ✓ Significantly reduce MR cycles.
Architecture of RAPID+

1. SPARQL parser

2. Rule-based Rewriter

3. Jena

4. Schema-aware Rewriter

5-1. Pig Latin Plan Generator

5-2. NTGA Plan Generator

6. Logical-to-Physical Plan Translator

7. MapReduce Job Compiler

8. Hadoop Cluster
Preliminary Evaluation – UniProt (~ 7 Billion Triples), Chem2Bio2RDF (400M Triples)

- 80-node cluster of very basic nodes (Dual Xeon (2.33GHz), 4G RAM)
- Very basic UNION queries: max. width of union is 2 (UQ3)
- NTGA vs. Relational Union Queries (Hive (Union)) vs. Relational MQO – LOJ/Optional (Hive (Optional))

Preliminary Evaluation – Non-Union Queries (Hive vs. NTGA)

- NTGA often outperform relational ones.
  - when a query has multiple star patterns (UQ6-UQ8)
  - when star patterns are denser (UQ6-UQ9).
- Hive failed to execute UQ6 due to data skew issues at reduce phase.
Recent Evaluation Result

- Setup: (the same as before)
- Data: LUBM (20k univ, ~2.2B triples)
- Pre-processing
  - precompute triplegroups, replace URI with Qname, etc.,
  - NTGA: ~48 mins vs 48 hrs (RDF-3X)

- Query: LUBM Q5
  - UCQ for Q5: 48 union branch, 48 stars, 96 joins + 1 union.
  - Result:
    - Hive: 9561 sec
    - Pig: 4951 sec
    - NTGA: 400 sec
Acknowledgements

(NSF-IIS-1218277)
Backup: UCQs Execution in NTGA

- Need to join (connect) stars in each UNION branch.
- Traverses the query graph to find common star structures/join variables across union branches, and group using unions.
- Extend TG_Join to accept more than one star patterns as a left/right operand using primitives similar to a logical OR (V).

```
SELECT * WHERE {{
  ?p type Protein .
  ?p organism ?o .
}
SJ1
UNION
{
  ?p annotation ?x .
  ?p organism ?o .
}
SJ2
UNION
{
  ?o type Taxon .
  ?o commonName ?n .
  ?o subClassOf ?c .
}
SJ3
UNION
{
  ?o type Taxon .
  ?o commonName ?n .
  ?o subClassOf ?c .
}
SJ4

TG_Join(SJ1, SJ2)  TG_Join(SJ3, SJ4)

∵ SJ2 = SJ4

TG_Join(TG_Union(SJ1, SJ3), SJ4)

TG_Join(SJ1 \lor SJ2, SJ3)
```