RDFpro
an Extensible Tool for Building Stream-Oriented RDF Processing Pipelines

Riva del Garda, 19 October 2014
Marco Rospocher¹, Marco Amadori², Michele Mostarda², Francesco Corcoglioniti

¹ Data and Knowledge Management Unit, FBK-Irst, http://dkm.fbk.eu/
² Web of Data Unit, FBK-Irst http://wod.fbk.eu/
http://fracor.bitbucket.org/rdfpro
The problem

perform simple RDF processing tasks
- filtering and transformation (quad-level)
- basic inference (RDFS)
- dataset merging → deduplication, owl:sameAs smushing
- simple statistics extraction (VOID+)
- ...

on large datasets
- LOD-sized: 100M+ triples
- quads, not just triples

on a single commodity machine
- no cluster / distributed computing
- no triplestore or other data index
The solution

\( \text{RDF} \text{pro} \)

pro = processor (and not 'professional'!)

~ Java command line tool ~
~ embeddable Java library ~
~ public domain code ~

http://fracor.bitbucket.org/rdfpro/
RDFpro ingredients

① streaming

realized via the RDF processor abstraction

\[
\text{input stream} \xrightarrow[@P]{} \text{output stream}
\]

invocation syntax: rdfpro @P args

pro:
- natural model for many tasks
- \(O(n)\) time complexity \(\rightarrow\) fast, also due to sequential data access
- \(O(1)\) space complexity (usually) \(\rightarrow\) copes with arbitrarily large datasets

cons:
- restrictive model!
RDF pro ingredients

① streaming

② sorting

realized via external sorting (sort utility)

allows tasks not doable with pure streaming
- duplicate removal
- set operations (quad union, intersection, diff.)
- VOID statistics extraction
- ...

@stats

<x a void:Dataset>
<x void:entities 3>

...
RDFpro ingredients

① streaming
② sorting
③ pipelining

① sequence composition

\[
\text{rdfpro } \@P_1 \text{ args}_1 \ldots \@P_N \text{ args}_N
\]

② parallel composition

\[
\text{rdfpro } \{ \@P_1 \text{ args}_1, \ldots, \@P_N \text{ args}_N \} f
\]

pro:
- reduced I/O costs (less temporary files)
- reduced execution time (parallelism)
**RDFpro ingredients**

1. **streaming**
2. **sorting**
3. **pipelining**
4. **multi-threading**

### Parallelism Types

1. **inter-processor parallelism**
   - multiple processors run in parallel

2. **intra-processor parallelism**
   - `handleStatement()` called concurrently

3. **I/O parallelism**
   - multiple files read/written in parallel
   - single files split in chunks processed in parallel (line-oriented RDF formats only)

![Diagram showing parallel processing](image)
Putting all together, you can ...

move data around
- @read / @write files
- @download from / @upload to SPARQL endpoints

transform data
- general purpose data @transform using Groovy
- @infer the RDFS closure
- @smush data, replacing URI aliases with canonical URIs
- extract @tbox and VOID @stats

compose these tasks freely
- also via set operations
A simple use case

integrate:
- Freebase (2014/07/10 dump, 2623 MQuads)
- GeoNames (2013/08/27 dump 125 MQuads)
- DBpedia EN, ES, IT, NL (subset of ver. 3.9, 271 MQuads)

performing:
- filtering (remove redundant quads & quads in unwanted languages)
- smushing (based on owl:sameAs links in DBpedia)
- inference (excluding <X rdf:type rdfs:Resource> stuff)
- statistics extraction (VOID with class & property partitions)

using:
- a small workstation (I7 860, 16 GB ram, 500 GB 7200 rpm hd)
- RDFpro + parallel sort + pigz + pbzip2
A simple use case

tasks performed individually - 5h 16m total

1 pass 0.57 MQ/s 1h 27m
1 pass 1.36 MQ/s ~9m
2 passes 0.31 MQ/s ~41m
1 pass 0.22 MQ/s ~1h
1 pass + sort 0.38 MQ/s ~1h 14m
1 pass + sort 0.36 MQ/s ~44m

1-2 aggregated:
1 pass, 0.56 MQ/s, 1h 29m

3-6 aggregated:
2 passes, 0.09 MQ/s, 2h 16m

aggregated tasks - 3h 46m total (-28%)
A simple use case

### individual tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Input size</th>
<th>Output size</th>
<th>Throughput</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[MQuad]</td>
<td>[GB]</td>
<td>[MQuad]</td>
<td>[GB]</td>
</tr>
<tr>
<td>1. Filtering</td>
<td>3019.89</td>
<td>29.31</td>
<td>750.78</td>
<td>9.68</td>
</tr>
<tr>
<td>2. TBox extraction</td>
<td>750.78</td>
<td>9.68</td>
<td>0.15</td>
<td>0.01</td>
</tr>
<tr>
<td>3. Smushing</td>
<td>750.78</td>
<td>9.68</td>
<td>780.86</td>
<td>10.33</td>
</tr>
<tr>
<td>4. Inference</td>
<td>781.01</td>
<td>10.34</td>
<td>1693.59</td>
<td>15.56</td>
</tr>
<tr>
<td>5. Deduplication</td>
<td>1693.59</td>
<td>15.56</td>
<td>954.91</td>
<td>7.77</td>
</tr>
<tr>
<td>6. Statistics</td>
<td>954.91</td>
<td>7.77</td>
<td>0.32</td>
<td>0.01</td>
</tr>
<tr>
<td>whole processing</td>
<td>3019.89</td>
<td>29.31</td>
<td>955.23</td>
<td>7.78</td>
</tr>
</tbody>
</table>

### aggregated tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Input size</th>
<th>Output size</th>
<th>Throughput</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[MQuad]</td>
<td>[GB]</td>
<td>[MQuad]</td>
<td>[GB]</td>
</tr>
<tr>
<td>1-2 aggregated</td>
<td>3019.89</td>
<td>29.31</td>
<td>750.92</td>
<td>9.69</td>
</tr>
<tr>
<td>3-6 aggregated</td>
<td>750.92</td>
<td>9.69</td>
<td>955.23</td>
<td>7.78</td>
</tr>
<tr>
<td>whole processing</td>
<td>3019.89</td>
<td>29.31</td>
<td>955.23</td>
<td>7.78</td>
</tr>
</tbody>
</table>
RDFpro cookbook

① download

http://fracor.bitbucket.org/rdfpro (or Google for it!)

RDFpro
An Extensible Tool for Building Stream-Oriented RDF Processing Pipelines

About
RDFpro (RDF Processor) is a public domain, Java command line tool and library for RDF processing. RDFpro offers a suite of stream-oriented, highly optimized RDF processors for common tasks that can be assembled in complex pipelines to efficiently process RDF data in one or more passes. RDFpro originated from the need of a tool supporting typical Linked Data integration tasks, involving dataset sizes up to few billions triples.

Features
- RDF quad (triple + graph) filtering and replacement
- RDFS inference with selectable rules
- owl:sameAs smushing
- TBox and VOID statistics extraction
- RDF deduplication, intersection and difference
- data upload/download via SPARQL endpoints
- data read/write in multiple (compressed) formats (rdf, rj, jsonld, nt, nq, trix, trig, tql, ttl, n3, brf)
- command line tool + core, tql, jsonld libraries
- based on Sesame.
- public domain software (Creative Commons CC0)

News
- 2014-08-04 Version 0.2 has been released
- 2014-07-24 Version 0.1 has been released

Back to top
RDFpro cookbook

① download

② install

check requirements:
- Java 1.7+ (Oracle, OpenJDK, whatever)
- gzip, bzip2, sort utilities available on PATH

extract the download tarball:

$ tar tf rdfpro-0.3.tar.gz

check that everything works:

$ cd rdfpro
$ ./rdfpro -v
RDF Processor Tool (RDFpro) 0.3
Java 64 bit (Oracle Corporation) 1.7.0_67
This is free software released into the public domain

suggestions:
- add rdfpro directory to PATH
- install and configure pigz and pbzip2 (see web site)
RDFpro cookbook

① download
② install
③ try it out!

let's get and process some data from Dbpedia:

```
$ ./rdfpro \
>    @read http://dbpedia.org/resource/Riva_del_Garda \
>    http://it.dbpedia.org/resource/Riva_del_Garda \
>    @smush \
>    @infer http://downloads.dbpedia.org/3.9/dbpedia_3.9.owl.bz2 \
>    @transform “emitIf(t == rdf:type)” \
>    @unique \
>    @write riva_del_garda.ttl.gz
```
That's all:
enjoy cooking triples with RDF<sub>pro</sub> and...
happy eating !!

for any question about the menu RDF<sub>pro</sub>, contact Francesco Corcoglioniti <corcoglio@fbk.eu>