RDFox:
A Highly-Scalable RDF Store

Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks
University of Oxford

Zhe Wu, Jay Banerjee
Oracle Corporation

ISWC 2015
Bethlehem, Pennsylvania

Motivation

- Semantic Web applications commonly represent data using the Resource Description Framework (RDF)
- Such applications rely on *RDF stores* to:
  - efficiently handle large amounts of RDF data
  - handle data from heterogeneous sources
  - manage and reason with background knowledge
  - answer queries w.r.t. *both* data and knowledge

Motivation

Hardware trends open new possibilities for highly-scalable main-memory RDF stores

- Main memory
  - personal computers: 4GB-16GB
  - small servers: hundreds of gigabytes
  - higher-end servers: terabytes
- Processing units
  - personal computers: 2-4 cores
  - small servers: tens of cores
  - higher-end servers: thousands of cores

RDFox Overview

- A cross-platform, centralised, main-memory RDF store
- Supports parallel import and efficient storage of large amounts of RDF data
- Supports highly-scalable materialisation-based reasoning;
  - incremental reasoning;
  - native handling of owl:sameAs
- Provides efficient query answering (tree decomposition)
- Provides versatile modes of use (C, Java and Python APIs, SPARQL endpoint)
- Developed at the University of Oxford and available under an academic licence at http://rdfox.org

Storage Scheme

- RDFox uses a flexible and highly-efficient storage scheme that supports highly-scalable parallel updates
  - it stores RDF data in a triple table
  - it has a configurable indexing scheme for efficient pattern matching (default [s,p,o], [s,p], [p], [o,p])
  - it supports ‘mostly’ lock-free updates
  - it is crucial for scalable parallel importation and parallel reasoning

Datalog Reasoning

- RDFox supports reasoning with datalog ontologies
- RDFox incorporates a materialisation-based datalog engine that implements state-of-the-art reasoning algorithms
- Datalog is a rule-based language that can capture OWL 2 RL and SWRL rules
  - each OWL 2 RL ontology can be encoded as datalog rules
  - one can also use the fixed datalog program that corresponds to the rules in the OWL 2 RL specification
  - SWRL rules are a syntactic variant of datalog rules

Datalog Materialisation

- Precompute the consequences of the input RDF data and datalog rules
- Queries are answered over the materialised consequences
  - no need for reasoning during query evaluation
- RDFox uses a shared-memory parallel algorithm **[Motik et al., AAAI 2014]**
  - a ‘triple-at-a-time’ variant of the semi-naive algorithm
  - no repetition of work (every rule instance is considered at most once)
  - even partitioning of reasoning into small subtasks (one per triple)
  - allows high scalability
  - no need of explicit load balancing

Incremental Reasoning

- Semantic Web data changes continuously
- RDFox supports incremental reasoning for efficient updating of datalog materialisations **[Motik et al., AAAI 2015]**
  - uses the **FBF** algorithm, which efficiently identifies the triples that need to be deleted and added
  - requires no extra information collected during the initial materialisation (no counts, no dependencies, no proofs)
  - improves the well-known DRed algorithm known from the database community by performing **exact** deletions

Native Equality Reasoning

- owl:SameAs is used to assert equalities between resources
- Equality reasoning can affect performance: increased memory consumption and reasoning times
- RDFox uses rewriting **[Motik et al., AAAI 2015]**
  - assigns a common representative to equal individuals
  - handles correctly rewritten constants in rules
- RDFox implements the **first** incremental algorithm for equality reasoning with rewriting **[Motik et al., IJCAI 2015]**

Evaluation

- RDFox on mid-range servers (previous work)
  - stores 1.5 G triples in 52 GB of RAM
  - 14x speedup on using 16 physical cores
  - efficient incremental reasoning for small and medium sized updates (with and without native equality reasoning)
- We evaluate RDFox on an **Oracle SPARC T5-8**
  - 4 TB of RAM
  - 8 SPARC V9 processors at 3.6 GHz
  - 128 physical threads and 1024 virtual threads

Parallelisation Scalability

	LUBM-50k		Claros		DBpedia	
	Time(s)	Speedup	Time(s)	Speedup	Time(s)	Speedup
1	27000		10,000		31,000	
64	727	**37x**	375	**27x**	1,200	**26x**
128	387	**70x**	226	**44x**	698	**45x**
256	—	—	226	**44x**	684	**46x**
512	—	—	154	**65x**	432	**72x**
1024	—	—	125	**80x**	359	**87x**
Max Rate	**6.1M t/s**		**4.2M t/s**		**4.0M t/s**	
Triples	6.7G —> 9.3G		19M —> 539M		113M —> 1.5G	

Parallelisation Scalability

	LUBM-9k		Claros		DBpedia	
	Time(s)	Speedup	Time(s)	Speedup	Time(s)	Speedup
1	1600		10,000		31,000	
64	50	32x	375	27x	1,200	26x
128	28	58x	226	44x	698	45x
256	17	97x	226	44x	684	46x
512	8	190x	154	65x	432	72x
1024	8	**213x**	125	**80x**	359	**87x**
Max Rate	**60M t/s**		**4.2M t/s**		**4.0M t/s**	
Triples	6.7G —> 9.3G		19M —> 539M		113M —> 1.5G	

Data Scalability on LUBM: RDFox with 1024 threads

	Input	Mat.	Time (s)
LUBM 20k	3.1G	4.2G	42
LUBM 40k	5.5G	7.5G	85
LUBM 60k	8.0G	10.9G	118
LUBM 80k	11.0G	15.0G	179
LUBM 100k	13.5G	18.4G	228
LUBM 120k	15.9G	**21.7G**	**251**

Conclusion

RDFox provides a unique combination of versatility, rich functionality, high performance and scalability:

- a highly efficient and flexible storage scheme
- state of the art datalog reasoning algorithms
- versatile modes of access

Suitable for data-intensive applications requiring expressive and highly scalable reasoning

- storage of up to **21G** triples
- reasoning speeds of up to **60M t/s**
- speedups of up to **213** times using 1024 threads

Outlook

- Improving our query answering algorithms
- Extending our support to full SPARQL 1.1
- Adding support for named graphs
- Reasoning with aggregation and non-monotonic negation
- Distributing storage, querying and reasoning in RDFox

Thank you!

References

Motik, B., Nenov, Y., Piro, R., Horrocks, I., Olteanu, D.: Parallel materialisation of datalog programs in centralised, main-memory RDF systems. AAAI 2014

Motik, B., Nenov, Y., Piro, R., Horrocks, I.: Handling owl:sameAs via rewriting. AAAI 2015

Motik, B., Nenov, Y., Piro, R.E.F., Horrocks, I.: Incremental update of datalog materialisation: The Backward/Forward algorithm. AAAI 2015

Motik, B., Nenov, Y., Piro, R.E.F., Horrocks, I.: Combining rewriting and incremental materialisation maintenance for datalog programs with equality. IJCAI 2015

RDFox: Use Cases

- **Kaiser Permanente**—a US health care consortium
  - uses RDFox to analyse patient data records
- **E´lectricite´ de France (EDF)**—a French electric utility company
  - uses RDFox to manage and analyse information about their electricity distribution network
- **Statoil ASA**—a Norwegian multinational oil and gas company
  - uses RDFox as part of a large-scale Semantic Web application that facilitates the integration and analysis of oil production and geological survey data

Evaluation: Memory

	LUBM-50k		Claros		DBpedia	
	Triples	B/t	Triples	B/t	Triples	B/t
After Import	6.7G	**124**	19M	**80**	113M	**58**
After Materialisation	9.3G	**101**	539M	**37**	1.5G	**39**

RDFox Architecture