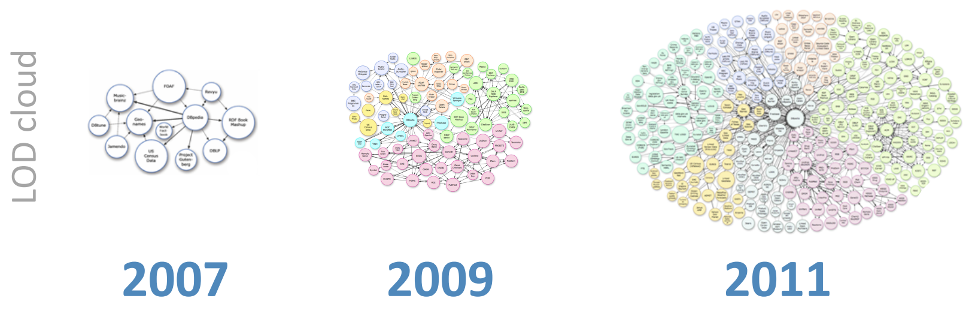
PigSPARQL

- A SPARQL Query Processing Baseline for Big Data -

Alexander Schätzle # • Martin Przyjaciel-Zablocki # • Thomas Hornung • Georg Lausen

1. Motivation

- RDF datasets grow continously in size
  → scalability of query processing?



- Single-place RDF stores limited in scale
- Reuse existing infrastructures and frameworks for distributed processing of Big Data
- Wide spread adoption of Hadoop MapReduce makes it an interesting candidate for distributed SPARQL processing

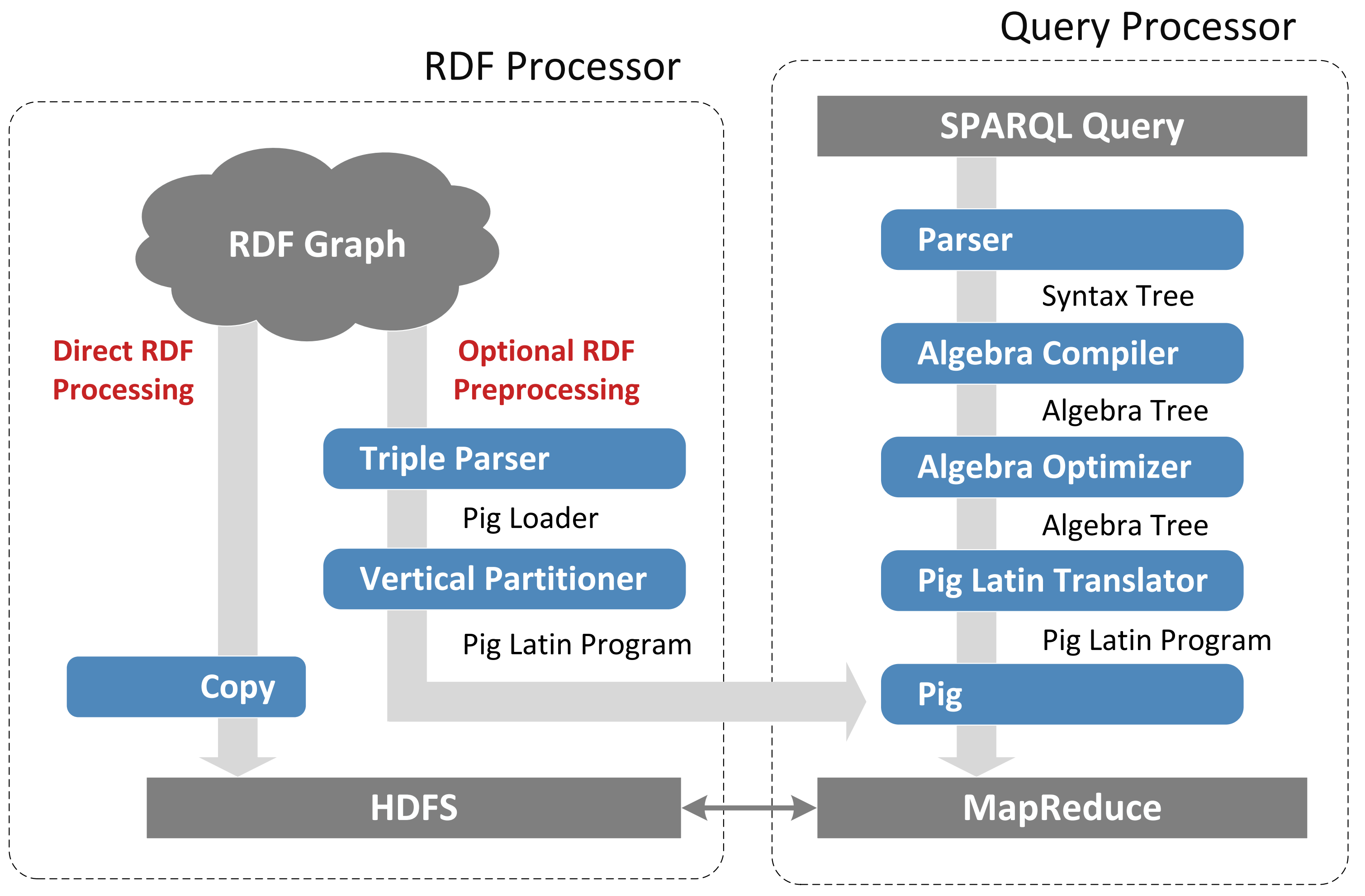
2. PigSPARQL

- SPARQL 1.0 engine on MapReduce for adhoc query processing of large RDF graphs
- Uses Pig (Latin), a data analysis platform on top of MapReduce, as intermediate layer between SPARQL and MapReduce
- Focus on rather costly queries involving many joins that cannot be executed in real-time at web-scale → offline processing
- Available for download *

Advantages of using Pig:

- Compatibility to future changes of Hadoop as they are covered by Pig
- Pig's processing framework is continuously optimized and enhanced with new features

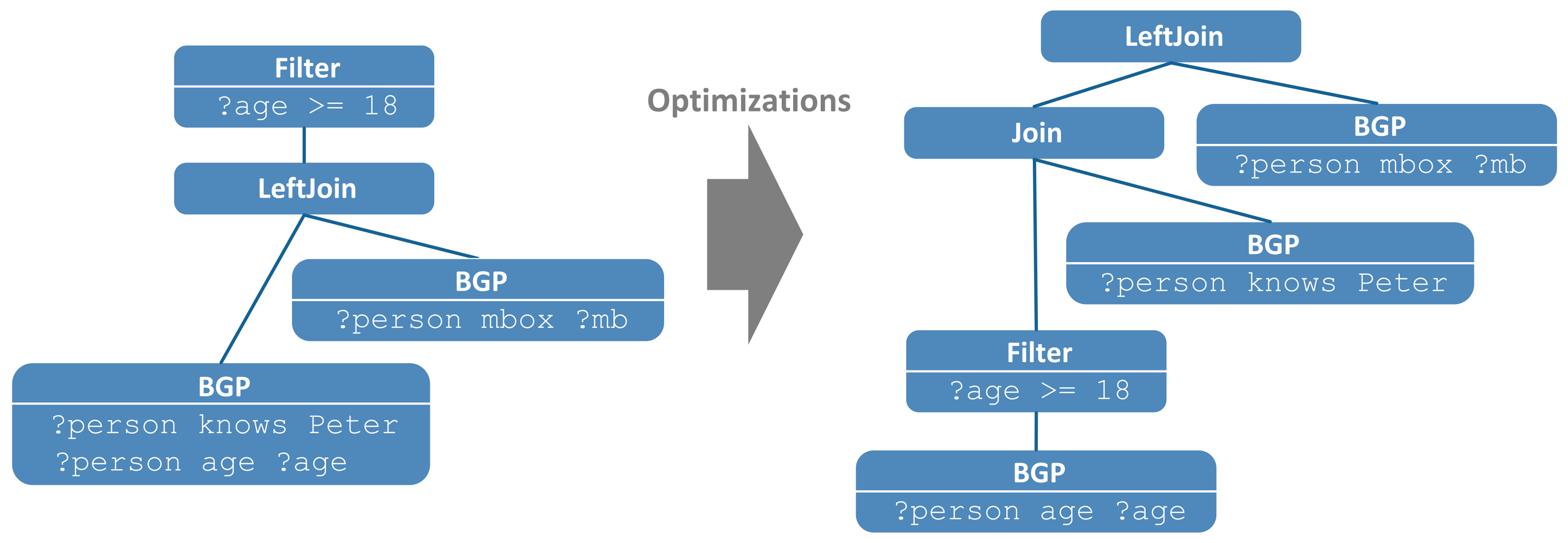
3. Architecture and Design



(1/3) SPARQL Query

```
SELECT *
WHERE {
    ?person knows Peter .
    ?person age ?age
    OPTIONAL {
       ?person mbox ?mb
  }
    FILTER (?age >= 18)
}
```

(2/3) Algebra Tree



(3/3) Pig Latin Program

```
knows = LOAD 'rdf/knows' USING rdfLoader() AS (s,o);
age   = LOAD 'rdf/age' USING rdfLoader() AS (s,o);
t1    = FILTER age BY o >= 18;
t1    = FOREACH t1 GENERATE s AS person, o AS age;
t2    = FILTER knows BY o == 'Peter';
t2    = FOREACH t2 GENERATE s AS person;
j1    = JOIN t1 BY person, t2 BY person;
j1    = FOREACH j1 GENERATE t1::person AS person,
                            t1::age AS age;
mbox  = LOAD 'rdf/mbox' USING rdf() AS (s,o);
t3    = FOREACH mbox GENERATE s AS person,o AS mb;
lj1   = JOIN j1 BY person LEFT OUTER, t3 BY person;
lj1   = FOREACH lj1 GENERATE j1::person AS person,
          j1::age AS age, t3::mb AS mb;
STORE lj1 INTO 'output' USING resultWriter();
```

SPARQL Query

- Support for all SPARQL 1.0 operators, not only BGPs
- Special cases like OPTIONAL with unsafe FILTER supported
- SPARQL 1.1 operators in development

Optimizations

- **SPARQL Algebra:**
  (1) Early filter execution
  (2) Reordering by selectivity
- **Translation:**
  (3) Early projection
  (4) Multi joins
- **Data model (optional):**
  (5) Vertical partitioning

Algebra Translation

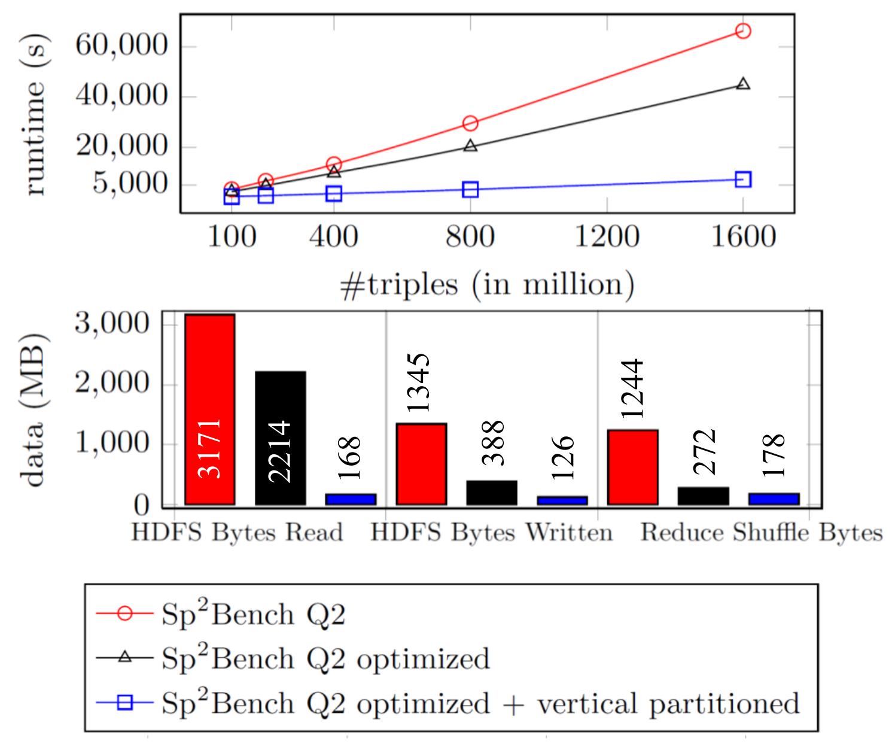
- Each SPARQL algebra operator is translated into a sequence of Pig Latin expressions
- BGPs are evaluated directly on the data using our loader UDF for RDF
- A BGP of n Triple Patterns needs n-1 JOINs in general (if multi joins are not applicable)
- OPTIONAL corresponds to a left outer join

Pig Latin Program

- Pig as intermediate abstraction layer between SPARQL and MapReduce
- Can be executed on any Hadoop cluster out-of-the box
- Automatically translated into a sequence of MapReduce jobs

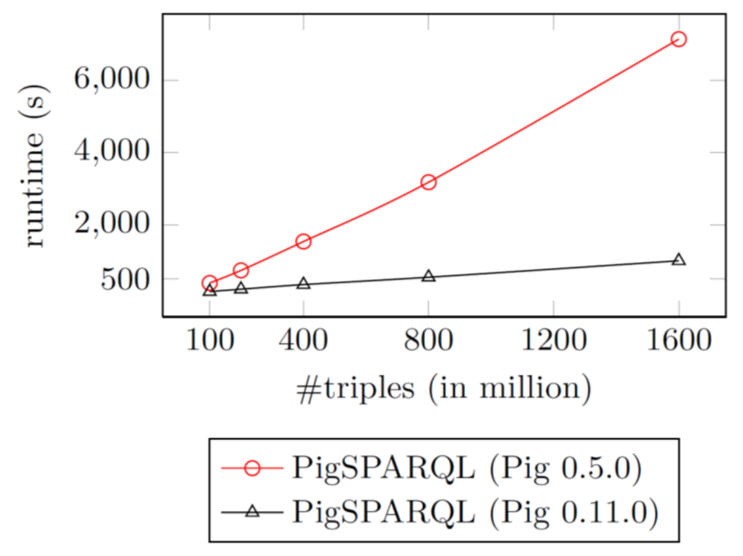
4. Experiments

PigSPARQL Optimizations



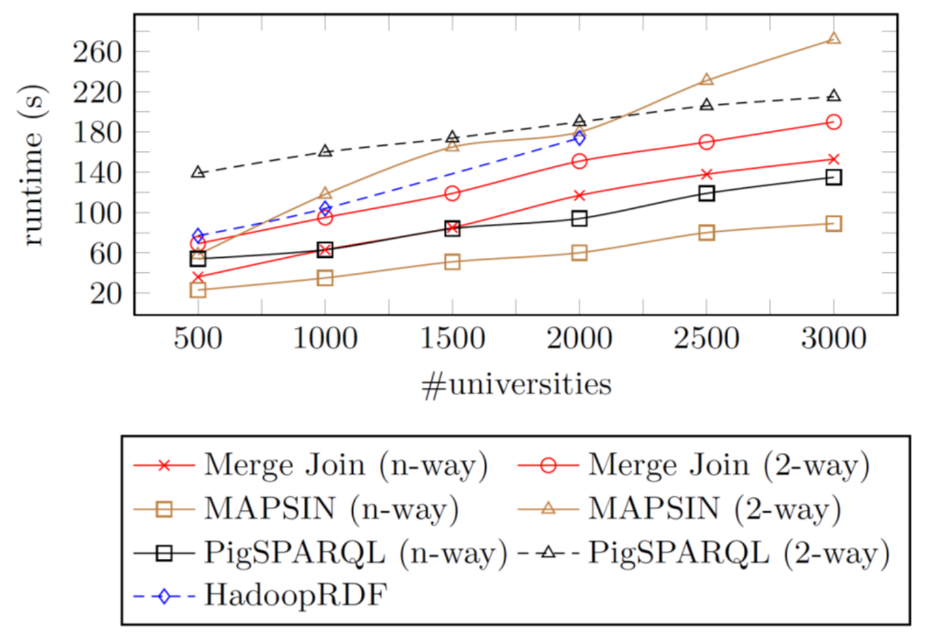
- Multi joins and vertical partitioning reduces overall query execution time by nearly 90%

Switching version of Pig



- Pig 0.5.0 → Pig 0.11.0
- No code changes in PigSPARQL
- Query execution times improved by up to one order of magnitude
- Speedup increases with datasize

Comparison with other approaches



- Competitive performance while scaling smoothly with increasing dataset size
- Vertical partitioning done in less than 14 min, other systems need up to several hours for preprocessing (1.6 billion triples)

Experimental results

- Linear scaling of query execution time with respect to datasize
- Optimizations reduce I/O and query time
- PigSPARQL runs without any tricky configurations → evaluation done in one day
- Competitive performance for offline queries

Pig as intermediate layer:

(1) Significant performance improvements by Pig version update without changing a single line of code
(2) Reliable and stable since Pig is widely-used and maintained by Yahoo! Research

5. Related Research & Download

- PigSPARQL: Mapping SPARQL to Pig Latin
  SWIM 2011, in conjunction with SIGMOD 2011. Athens (Greece)
- Cascading Map-Side Joins over HBase for Scalable Join Processing
  SSWS+HPCSW 2012, in conjunction with ISWC 2012. Boston (USA)
- Map-Side Merge Joins for Scalable SPARQL BGP Processing
  IEEE CloudCom 2013, Bristol (UK)
- Large-Scale RDF Processing with MapReduce
  Book Chapter in: Data Processing Techniques in the Era of Big Data, 2014

* http://dbis.informatik.uni-freiburg.de/PigSPARQL

6. Summary

Conclusion

- PigSPARQL, an implemented translation from SPARQL to Pig Latin
- Pig translates Pig Latin into MapReduce jobs and executes them in parallel on Hadoop
- It's an easy to use and competitive baseline for the comparison of MapReduce based SPARQL processing
- With the support of SPARQL 1.0, it already exceeds functionalities of most existing research prototypes

Future Work

- Support for SPARQL 1.1 operators
- Integration of investigated join techniques

Institute of Computer Science, University of Freiburg, Germany
{schaetzle, zablocki, hornungt, lausen}
@informatik.uni-freiburg.de

# The work of this author was partially funded by
Deutsche Forschungsgemeinschaft,
grant LA 598/7-1.