Computing the Stratified Semantics of Logic Programs over Big Data through Mass Parallelization

Ilias Tachmazidis, Grigoris Antoniou
University of Huddersfield, UK

University of
HUDDERSFIELD

Motivation: The Challenge of Big Data

- Big Data: Huge data set coming from
  - the Web, sensor networks and social media
- Applications: e.g. smart cities, intelligent environments, information extraction
- The challenge:
  - Scaling up to big data is not trivial
  - New approaches, new algorithms
- Opportunities for Knowledge Representation
  - Decision making
  - Decision support
  - Data cleaning
  - **Inferring high-level knowledge from low-level input**

Outline

- Motivation
- MapReduce paradigm
- Logic programming example
  - Joins
  - Anti-joins
- The power of stratification
- Experimental Results
- Future Directions

The Big Data Challenge for Reasoning

- Big Data poses significant computational challenges
  - Focus has to be not just complex knowledge structures, but their efficient processing in combination with huge amounts of data
- In particular for Knowledge Representation (KR), centralized in-memory solutions (the traditional KR approach) do not scale to the Big Data challenge:
  - Billions of facts result in over 20GB of data

Related Work

- Parallelization approaches:
  - Rule decomposition
  - Data decomposition
- Allows for efficient reasoning on large data sets
  - 100 billion triples
  - Datalog (e.g. Afrati & Ullmann)
  - RDF/S (e.g. Weaver & Hendler)
  - OWL dialects (e.g. Urbani et al.)

Novel Contribution

- All previous works addressed consistent sets of rules
- In practice, big data is messy and often inconsistent
- A type of non-classical reasoning, called nonmonotonic reasoning, supports reasoning
  - To deal with inconsistencies that arise naturally in the Web context
  - To deal with deficient (sensor) data
  - To reason with missing (incomplete) information
- **Apply MapReduce paradigm to nonmonotonic reasoning**

MapReduce Paradigm

- Inspired by similar primitives in LISP and other functional languages
- Operates exclusively on <key, value> pairs
- Input and Output types of a MapReduce job:
  - Input: <k1, v1>
  - Map(k1,v1) → list(k2,v2)
  - Reduce(k2, list (v2)) → list(k3,v3)
  - Output: list(k3,v3)

MapReduce Framework

- Provides an infrastructure that takes care of
  - distribution of data
  - management of fault tolerance
  - results collection
- For a specific problem
  - developer writes a few routines which are following the general interface

Negative Rule Calculation (1/5)

- Models both "join" and "anti–join" operations from database
- Example:

  <u>Facts:</u>

  parent(John, Alice), parent(John, Jill), sibling(Alice, Edward), sibling(Jill, Mary), female(Mary)

  <u>Rule:</u>

  son(X,Y) ← parent(Y,Z), sibling(Z,X), **not** female(X)

  Join

  parentOfSiblings(Y,X,Z)

Negative Rule Calculation (2/5)
"Join"

INPUT
Facts in multiple files

File01
____
parent(John, Alice)
parent(John, Jill)
sibling(Alice, Edward)

File02
____
sibling(Jill, Mary)
female(Mary)

MAP phase Input
Key: position in file (ignored)
Value: fact

<key, parent(John, Alice) >

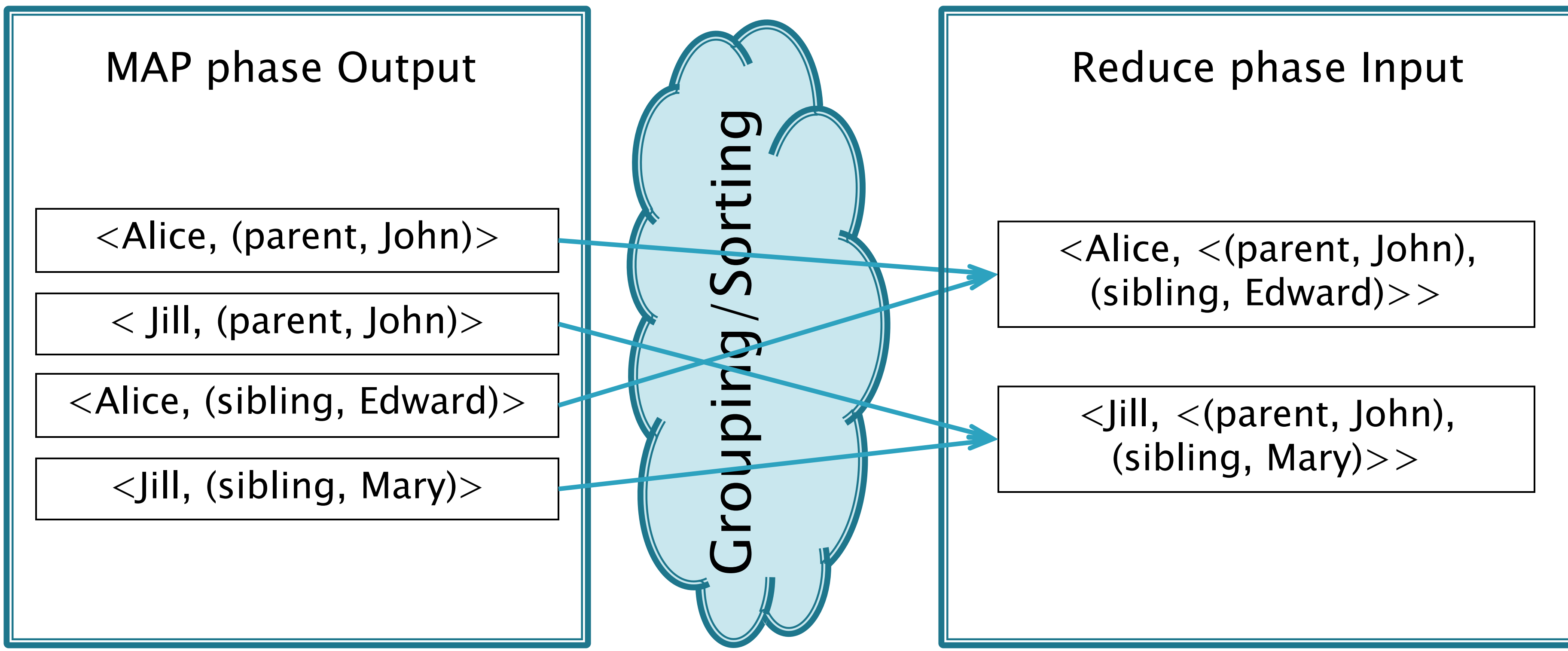
<key, parent(John, Jill)>

< key, sibling(Alice, Edward)>

<key, sibling(Jill, Mary)>

~~<key, female(Mary)>~~

Negative Rule Calculation (3/5): "Join"



Negative Rule Calculation (4/5)
"Join"

Reduce phase Input	Reduce phase Output **Output:** new conclusion
<Alice, <(parent, John), (sibling, Edward)>>	parentOfSiblings(John, Edward, Alice)
<Jill, <(parent, John), (sibling, Mary)>>	parentOfSiblings(John, Mary, Jill)

Negative Rule Calculation (5/5)

- Rule:

son(X,Y) ← parent(Y,Z), sibling(Z,X), **not** female(X)

Join

parentOfSiblings(Y,X,Z)

female(Mary)

parentOfSiblings(John, Edward, Alice)

~~parentOfSiblings(John, Mary, Jill)~~

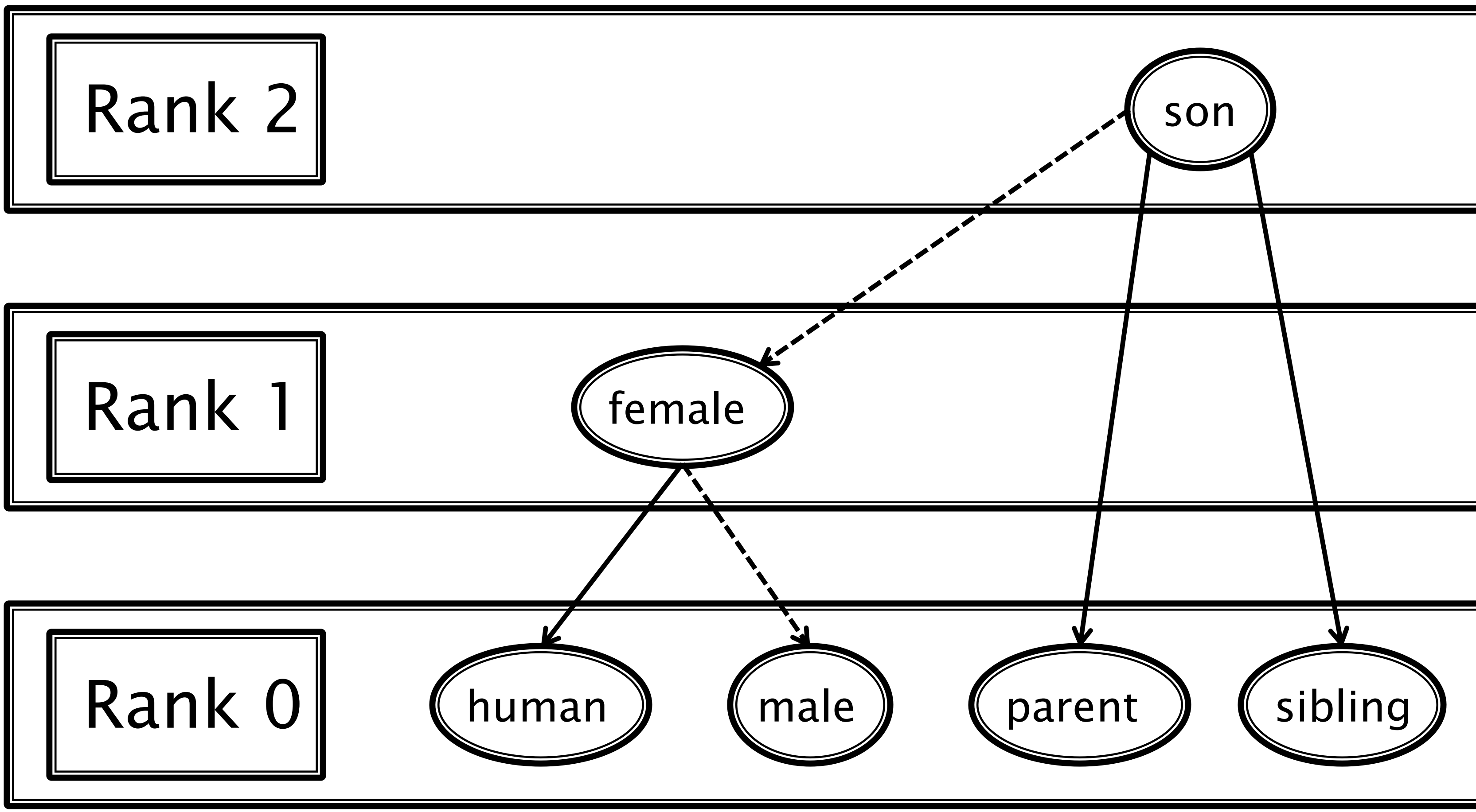
Anti-join

son(Edward, John)

Stratified Semantics (1/2)

- son(X,Y) ← parent(Y,Z), sibling(Z,X), **not** female(X)
- female(x) ← human(x), **not** male(x)

Stratified Semantics (2/2)

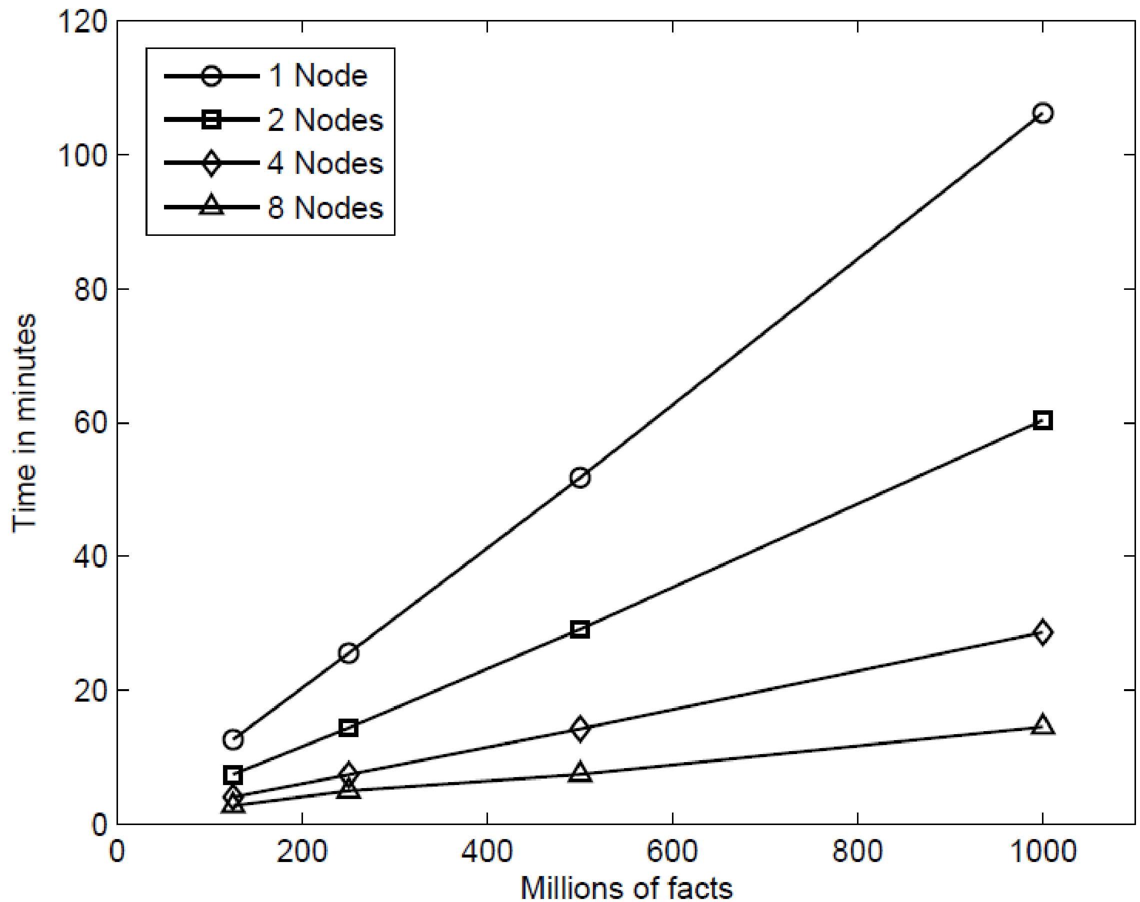


Experimental Setup

- Measure scalability in terms of
  - Number of nodes (computers in the cluster) maximum parallelization possible
  - Number of facts
  - Number of rules
- Used a synthetic dataset
  - up to 1 billion facts
  - up to 128 rules

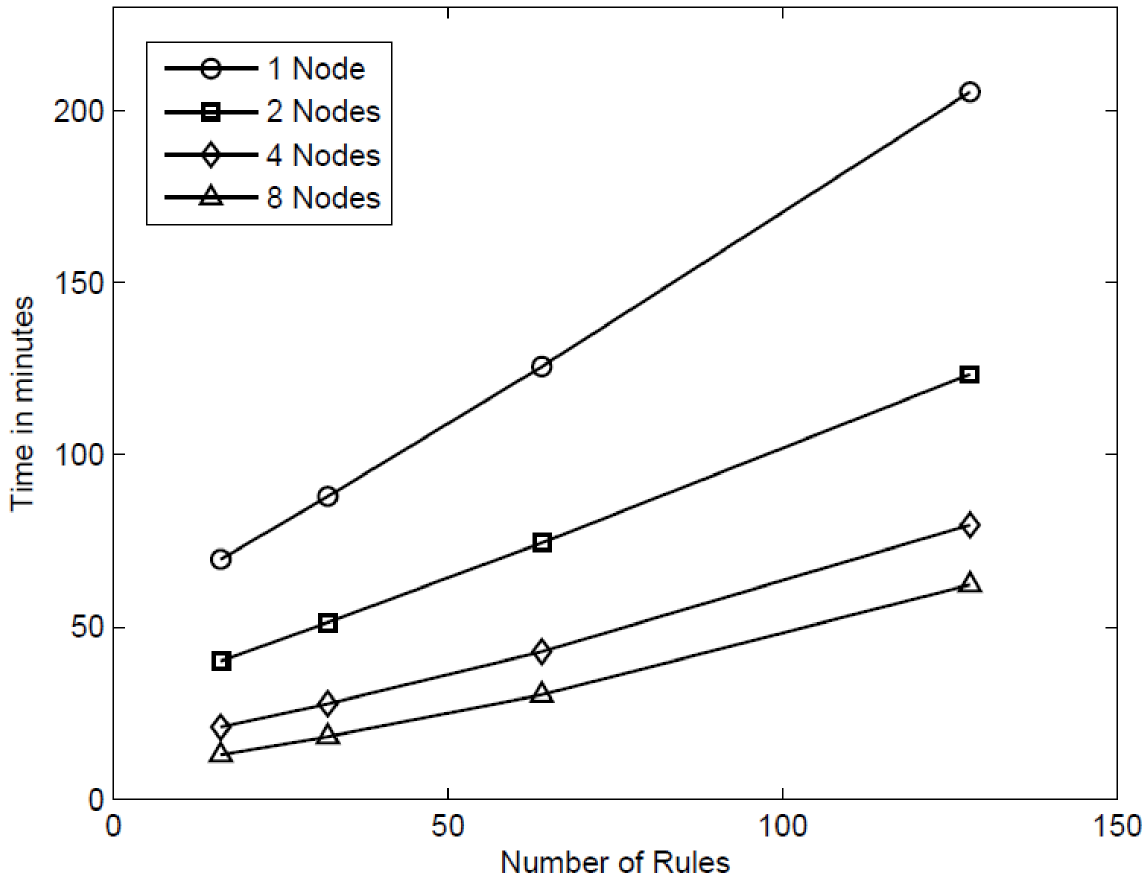
Experimental Results (1/2)

parallelization factor of 8: linear performance



Experimental Results (2/2)

parallelization factor of 8: linear performance up to 64 rules



Summary

- Computed the stratified semantics over Big Data
- Ran experiments for various
  - data sizes
  - rule sizes
- Demonstrated that reasoning can scale well up to 1 billion facts

Future Work

- Beyond stratification
  - What happens is we do not have this nice structure
  - Solve the problem by allowing dependency cycles
- Beyond MapReduce
  - We will study more complex NMR approaches, including ontology evolution/repair and Answer-Set Programming
  - We believe that MapReduce is not well placed to support this kind of approaches: they are probably not “embarrassingly parallel”

Thank You!

Experimental Results