Scalable Nonmonotonic Reasoning over RDF Data Using MapReduce

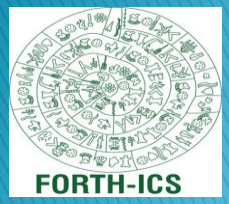
Ilias Tachmazidis[1,2], Grigoris Antoniou[1,2,3], Giorgos Flouris[2], *Spyros Kotoulas*[4]

[1] *University of Crete*

[2] *Foundation for Research and Technolony, Hellas (FORTH)*

[3] *University of Huddersfield*

[4] *Smarter Cities Technology Centre, IBM Research, Ireland*



Outline

- Motivation
- Background
  - Defeasible Logic
  - MapReduce Framework
- Multi-Argument Implementation over RDF
- Experimental Evaluation
- Future Work

Motivation (1/2)

- Linked Datasets
  - Huge
  - Doubtable quality data, difficult to predict consequences of inference
- Defeasible logic
  - Intuitive
    - is suitable for encoding commonsense knowledge and reasoning
    - avoids triviality of inference due to low-quality data
  - Low complexity
    - The consequences of a defeasible theory D can be computed in O(N) time, where N is the number of symbols in D

Motivation (2/2)

- State-of-the-art
  - Defeasible logic has been implemented for in-memory reasoning, however, it was not applicable for huge data sets
- Solution: scalability/parallelization using the MapReduce framework

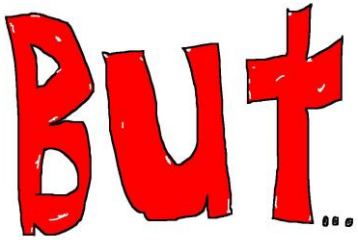
Defeasible Logic (1/2)

- Facts
  - e.g. $bird(eagle)$
- Strict Rules
  - e.g. $bird(X) \rightarrow animal(X)$
- Defeasible Rules
  - e.g. $bird(X) \Rightarrow flies(X)$

Defeasible Logic (2/2)

- Defeaters
  - e.g. $brokenWing(X) \rightsquigarrow \neg flies(X)$

- Priority Relation (acyclic relation on the set of rules)
  - e.g. $r: bird(X) \Rightarrow flies(X)$



$r': brokenWing(X) \Rightarrow \neg flies(X)$

$r' > r$

MapReduce Paradigm

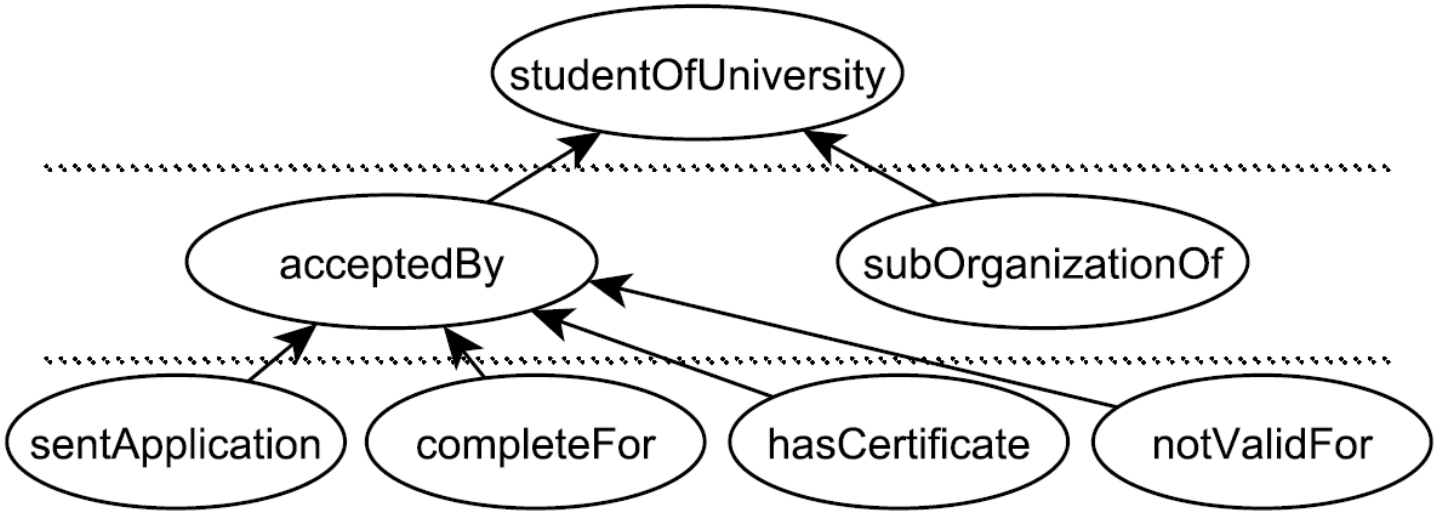
- Inspired by similar primitives in LISP and other functional languages
- Operates exclusively on <key, value> pairs
- Input and Output types of a MapReduce job:
  - Input: <k1, v1>
  - Map(k1,v1) → list(k2,v2)
  - Reduce(k2, list (v2)) → list(k3,v3)
  - Output: list(k3,v3)

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

Multi-Argument Implementation (1/2)

- Rule sets can be divided into two categories:
  - Stratified
  - Non-stratified
- Predicate Dependency Graph



Multi-Argument Implementation (2/2)

- Consider the following rule set:
  - r1: X *sentApplication* A, A *completeFor* D ⇒ X *acceptedBy* D.
  - r2: X *hasCerticate* C, C *notValidFor* D ⇒ X ¬*acceptedBy* D.
  - r3: X *acceptedBy* D, D *subOrganizationOf* U ⇒ X *studentOfUniversity* U.
  - r1 > r2.
- Both *acceptedBy* and ¬*acceptedBy* are represented by *acceptedBy*
- Superiority relation is not part of the graph

Reasoning overview

- Initial pass:
  - Transform facts into <fact, ($+\Delta$, $+\partial$)> pairs
- No reasoning needs to be performed for the lowest stratum (stratum 0)
- For each stratum from 1 to N
  - Pass1: Calculate fired rules
  - Pass2: Perform defeasible reasoning

Pass 1: Calculate Fired Rules (1 / 3)

INPUT
Literals in multiple files

File01

<John *sentApplication* App, (+Δ, +∂)>
<App *completeFor* Dep, (+Δ, +∂)>

File02

<John *hasCerticate* Cert, (+Δ, +∂)>
<Cert *notValidFor* Dep, (+Δ, +∂)>
<Dep *subOrganizationOf* Univ, (+Δ, +∂)>

MAP phase Input
<position in file, literal and knowledge>

<key, < John *sentApplication* App, (+Δ, +∂)>>

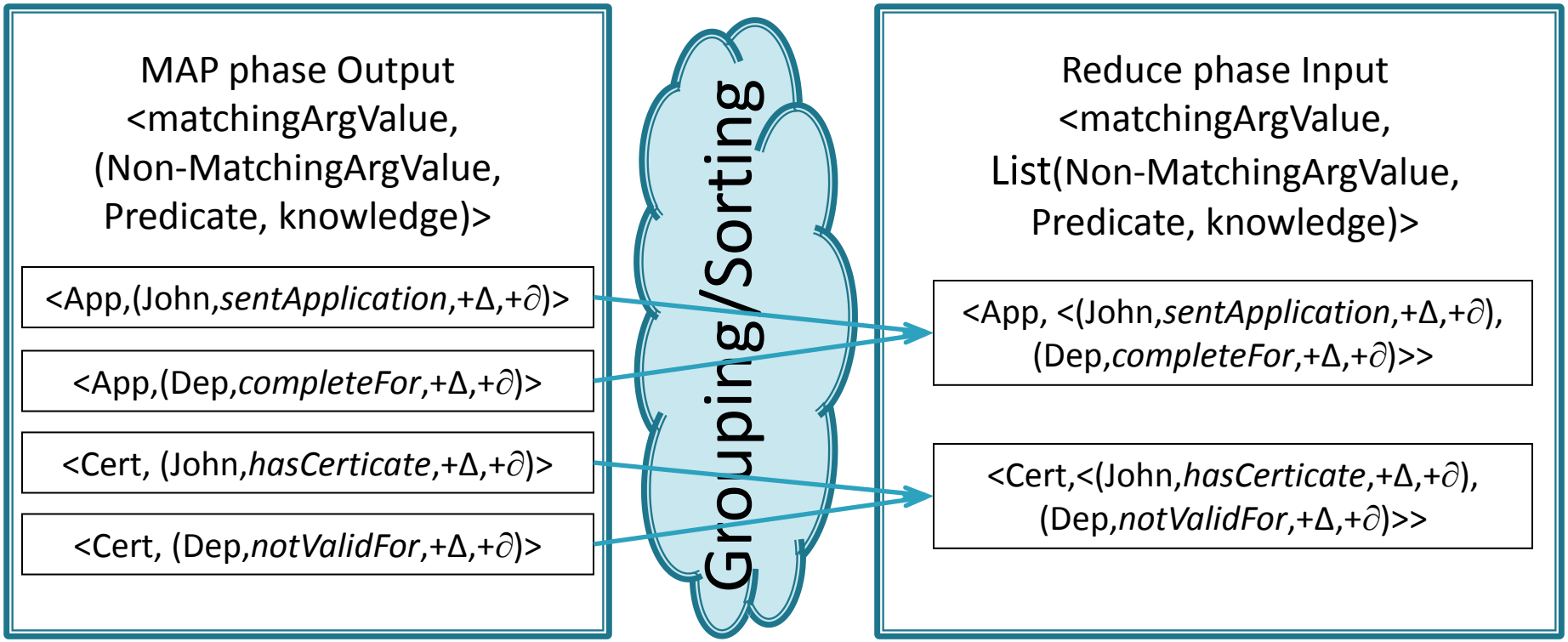
< key, < App *completeFor* Dep, (+Δ, +∂)>>

< key, < John *hasCerticate* Cert, (+Δ, +∂)>>

< key, < Cert *notValidFor* Dep, (+Δ, +∂)>>

~~< key, < Dep *subOrganizationOf* Univ, (+Δ, +∂)>>~~

Pass 1: Calculate Fired Rules (2/3)



Pass 1: Calculate Fired Rules (3/3)

Reduce phase Output
(Final Output)
<literal and knowledge>

<John *acceptedBy* Dep, (+∂, r1)>

<John *acceptedBy* Dep, (¬, +∂,r2)>

Pass 2: Perform defeasible reasoning (1/3)

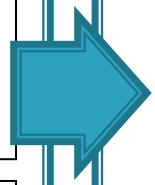
INPUT
Literals in multiple files

File01

<John *sentApplication* App, (+Δ, +∂)>
<App *completeFor* Dep, (+Δ, +∂)>
<John *hasCerticate* Cert, (+Δ, +∂)>
<Cert *notValidFor* Dep, (+Δ, +∂)>
<Dep *subOrganizationOf* Univ, (+Δ, +∂)>

File02

<John *acceptedBy* Dep, (+∂, r1)>
<John *acceptedBy* Dep, (¬, +∂, r2)>



MAP phase Input
<position in file, literal and knowledge>

~~<key, < John *sentApplication* App, (+Δ, +∂)>>~~

~~< key, < App *completeFor* Dep, (+Δ, +∂)>>~~

~~< key, < John *hasCerticate* Cert, (+Δ, +∂)>>~~

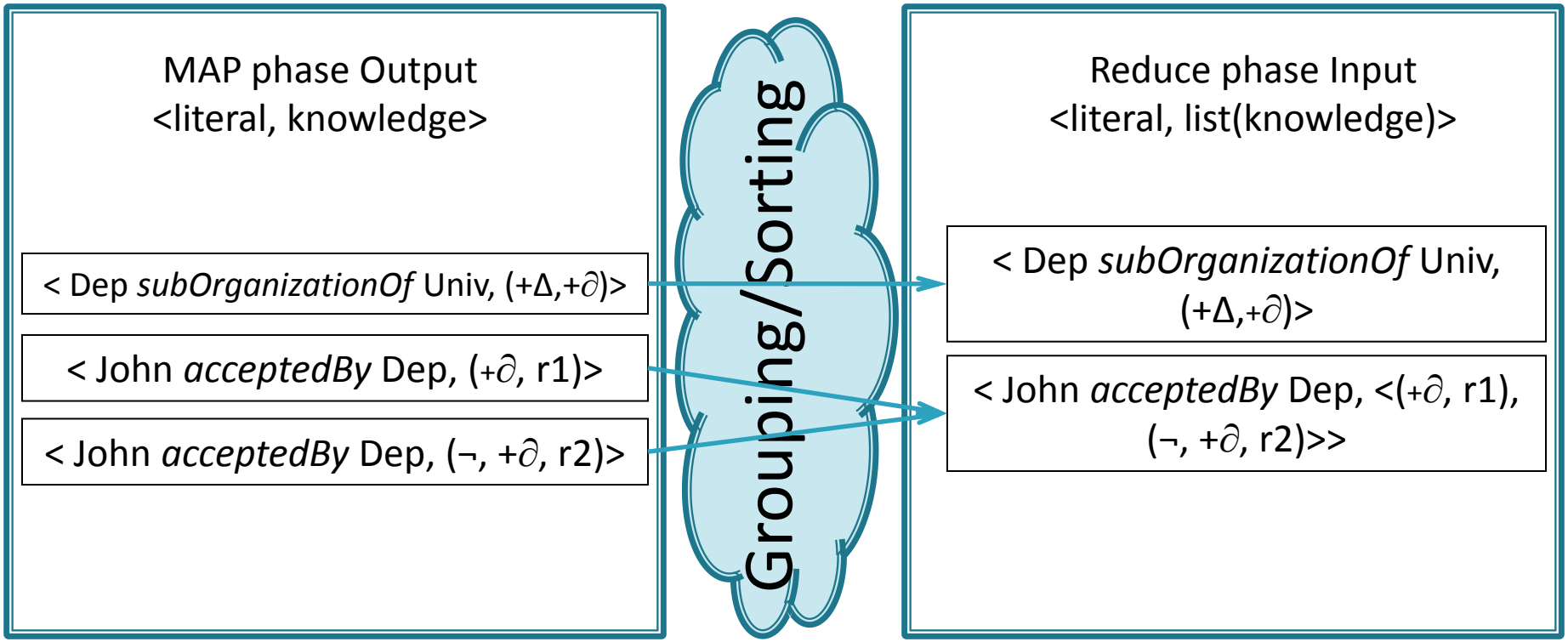
~~< key, < Cert *notValidFor* Dep, (+Δ, +∂)>>~~

<key, < Dep *subOrganizationOf* Univ, (+Δ,+∂)>>

< key, < John *acceptedBy* Dep, (+∂, r1)>>

< key, < John *acceptedBy* Dep, (¬, +∂, r2)>>

Pass 2: Perform defeasible reasoning (2/3)



Pass 2: Perform defeasible reasoning (3/3)

Reduce phase Output
(Final Output)
<Conclusions after reasoning>

No output

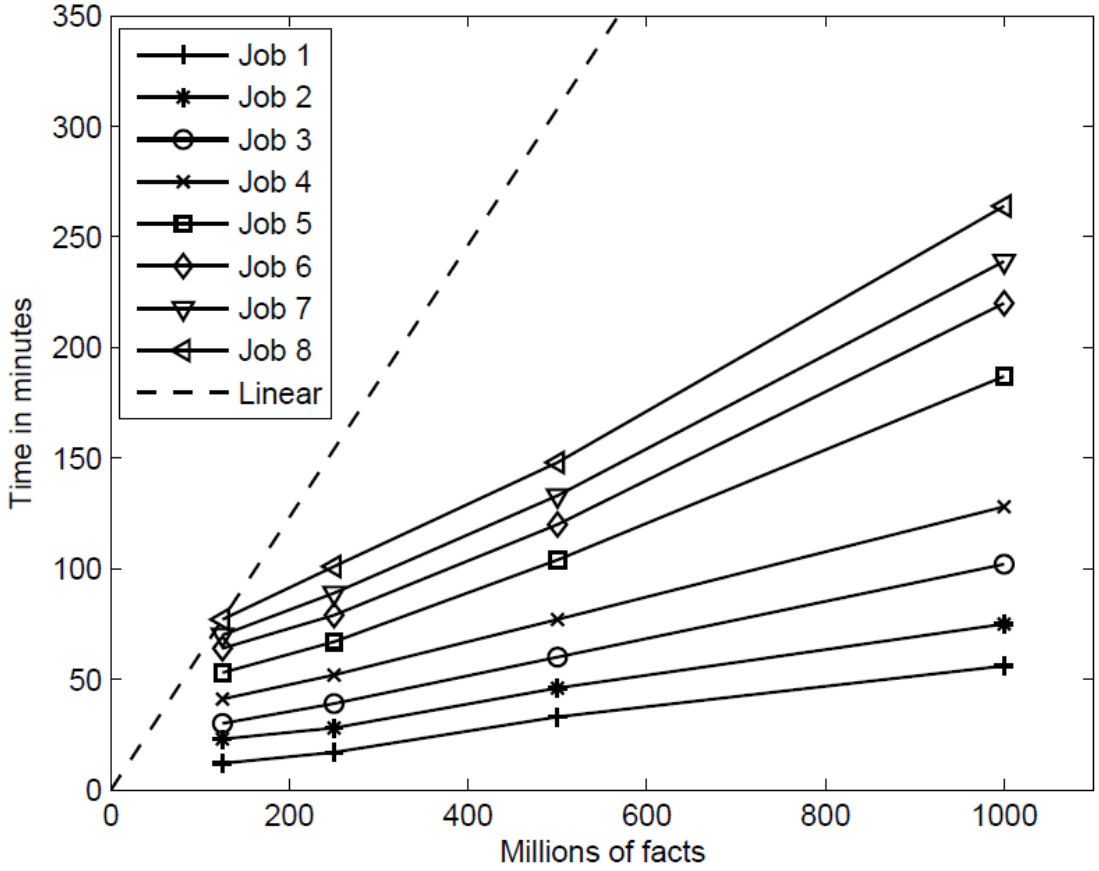
< John *acceptedBy* Dep, $(+\partial)$>

Experimental Setting

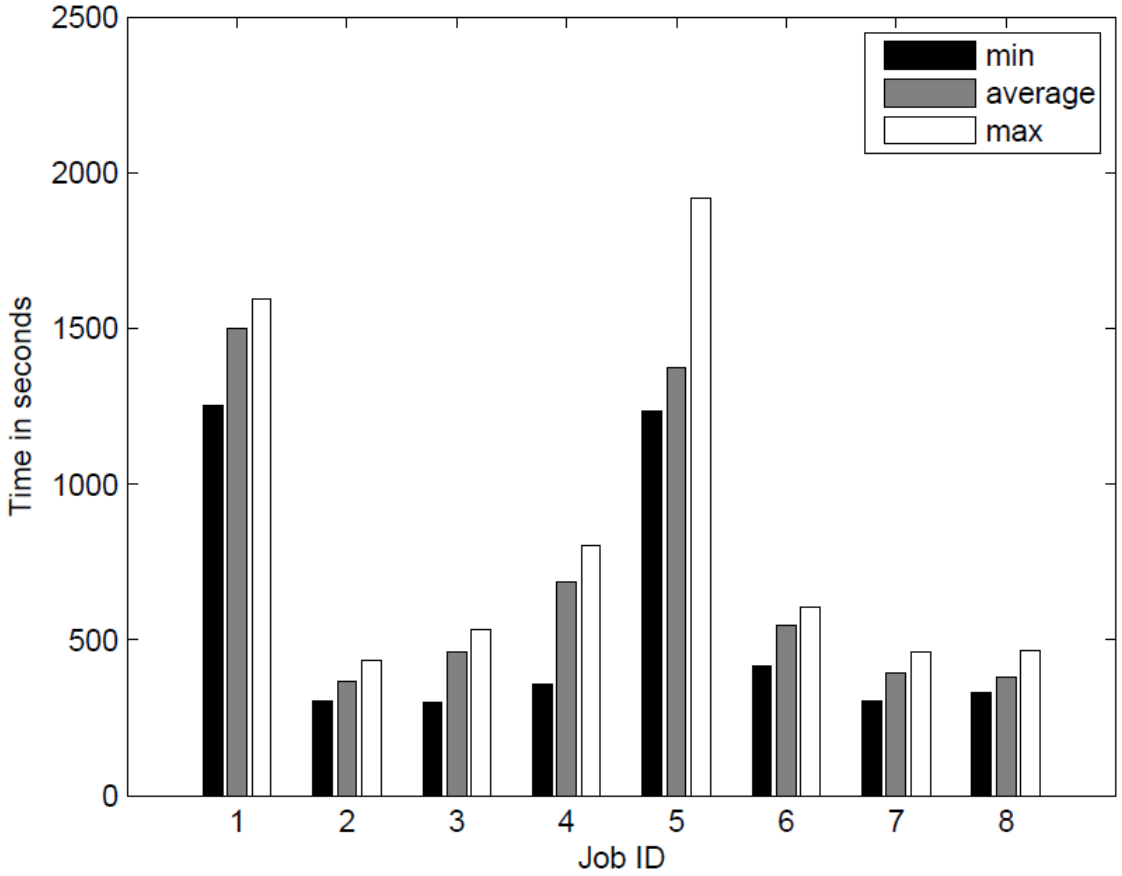
- LUBM (up to 1B)
- Custom defeasible ruleset

- IBM Hadoop Cluster v1.3 (Apache Hadoop 0.20.2)
- 40-core server
- XIV storage SAN

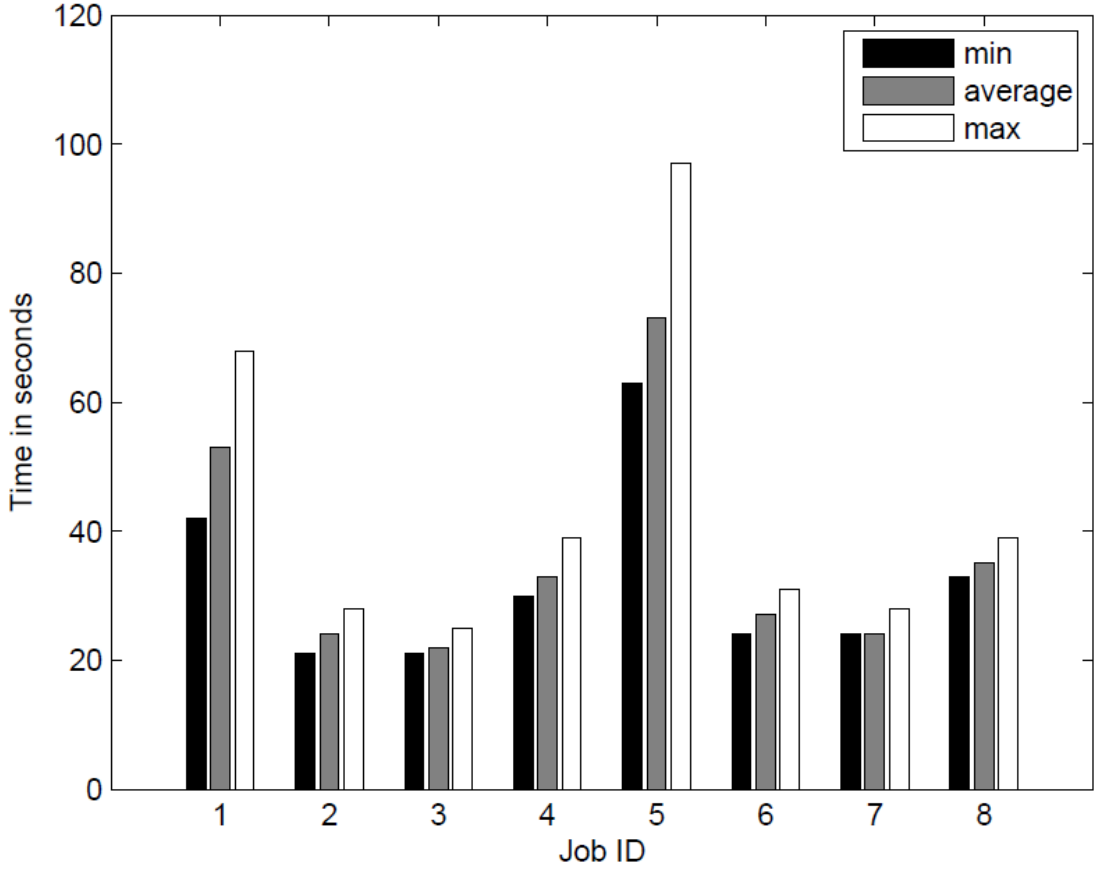
Multi-Argument over RDF Runtime



Multi-Argument over RDF Reduce Time for 40 tasks



Multi-Argument over RDF Reduce Time for 400 tasks



Future Work (1/2)

Challenges of Non-Stratified Rule Sets

- An efficient mechanism is need for $-\Delta$ and $-\partial$
  - all the available information for the literal must be processed by a single node causing:
    - main memory insufficiency
    - skewed load balancing
- Storing conclusions for $+/-\Delta$ and $+/-\partial$ is not feasible
  - Consider the cartesian product of X, Y, Z for X *Predicate1* Y, Y *Predicate2* Z.

Future Work (2/2)

- Run extensive experiments to test the efficiency of multi-argument defeasible logic
- Applications on real datasets, with low-quality data
- More complex knowledge representation methods such as:
  - Answer-Set programming
  - Ontology evolution, diagnosis and repair
- AI Planning

Thanks!

Backup (ruleset)

r1: X rdf:type FullProfessor $\rightarrow$ X rdf:type Professor.
r2: X rdf:type AssociateProfessor $\rightarrow$ X rdf:type Professor.
r3: X rdf:type AssistantProfessor $\rightarrow$ X rdf:type Professor.
r4: P publicationAuthor X, P publicationAuthor Y $\rightarrow$ X commonPublication Y.
r5: X teacherOf C, Y takesCourse C $\rightarrow$ X teaches Y.
r6: X teachingAssistantOf C, Y takesCourse C $\rightarrow$ X teaches Y.
r7: X commonPublication Y $\rightarrow$ X commonResearchInterests Y.
r8: X hasAdvisor Z, Y hasAdvisor Z $\rightarrow$ X commonResearchInterests Y.
r9: X hasResearchInterest Z, Y hasResearchInterest Z $\rightarrow$ X commonResearchInterests Y.
r10: X hasAdvisor Y $\Rightarrow$ X canRequestRecommendationLetter Y.
r11: Y teaches X $\Rightarrow$ X canRequestRecommendationLetter Y.
r12: Y teaches X, Y rdf:type PostgraduateStudent $\Rightarrow$ X $\neg$canRequestRecommendationLetter Y.
r13: X rdf:type Professor, X worksFor D, D subOrganizationOf U $\Rightarrow$ X canBecomeDean U.
r14: X rdf:type Professor, X headOf D, D subOrganizationOf U $\Rightarrow$ X $\neg$canBecomeDean U.
r15: X worksFor D $\Rightarrow$ X canBecomeHeadOf D.
r16: X worksFor D, Z headOf D, X commonResearchInterests Z $\Rightarrow$ X $\neg$canBecomeHeadOf D.
r17: Y teaches X $\Rightarrow$ X suggestAdvisor Y.
r18: Y teaches X, X hasAdvisor Z $\rightsquigarrow$ X $\neg$suggestAdvisor Y.
r12 $>$ r11, r14 $>$ r13, r16 $>$ r15, r18 $>$ r17.