A Token-Based Access Control System for RDF Data in the Clouds

Arindam Khaled
Mohammad Farhan Husain
Latifur Khan
Kevin Hamlen
Bhavani Thuraisingham
Department of Computer Science
University of Texas at Dallas

Research Funded by AFOSR

CloudCom 2010

Outline

- Motivation and Background
  - Semantic Web
  - Security
  - Scalability
- Access control
- Proposed Architecture
- Results

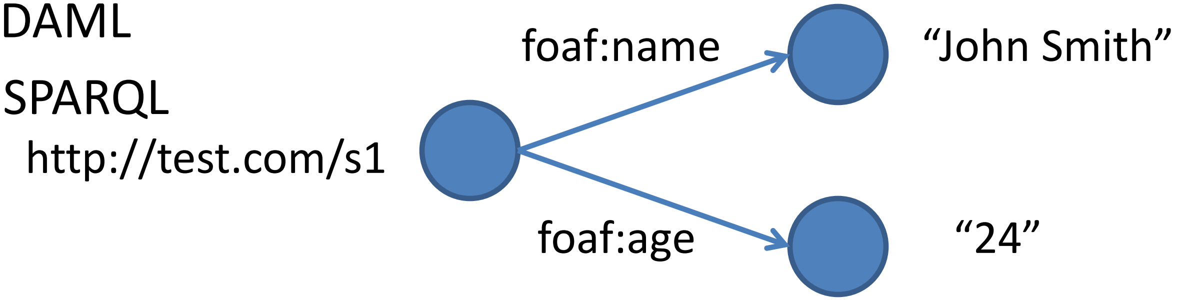
Motivation

- Semantic web is gaining immense popularity
- Resource Description Framework (RDF) is one of the ways to represent data in Semantic web.
- But most of the existing frameworks either lack scalability or don’t incorporate security.
- Our framework incorporates both of those.

Semantic Web Technologies

- Data in machine understandable format
- Infer new knowledge by ontology
- Allows relationships between web resources
- Standards
  - Data representation – RDF
    - Triples
      - Example:
  - Ontology – OWL, DAML
  - Query language - SPARQL

Subject	Predicate	Object
http://test.com/s1	foaf:name	"John Smith"
http://test.com/s1	foaf:age	"24"



Related Work

- Joseki [15], Kowari [17], 3store [10], and Sesame [5] are few RDF stores.
- Security is not addressed for these.
- In Jena [14, 20], efforts have been made to incorporate security.
- But Jena lacks scalability – often queries over large data become intractable [12, 13].

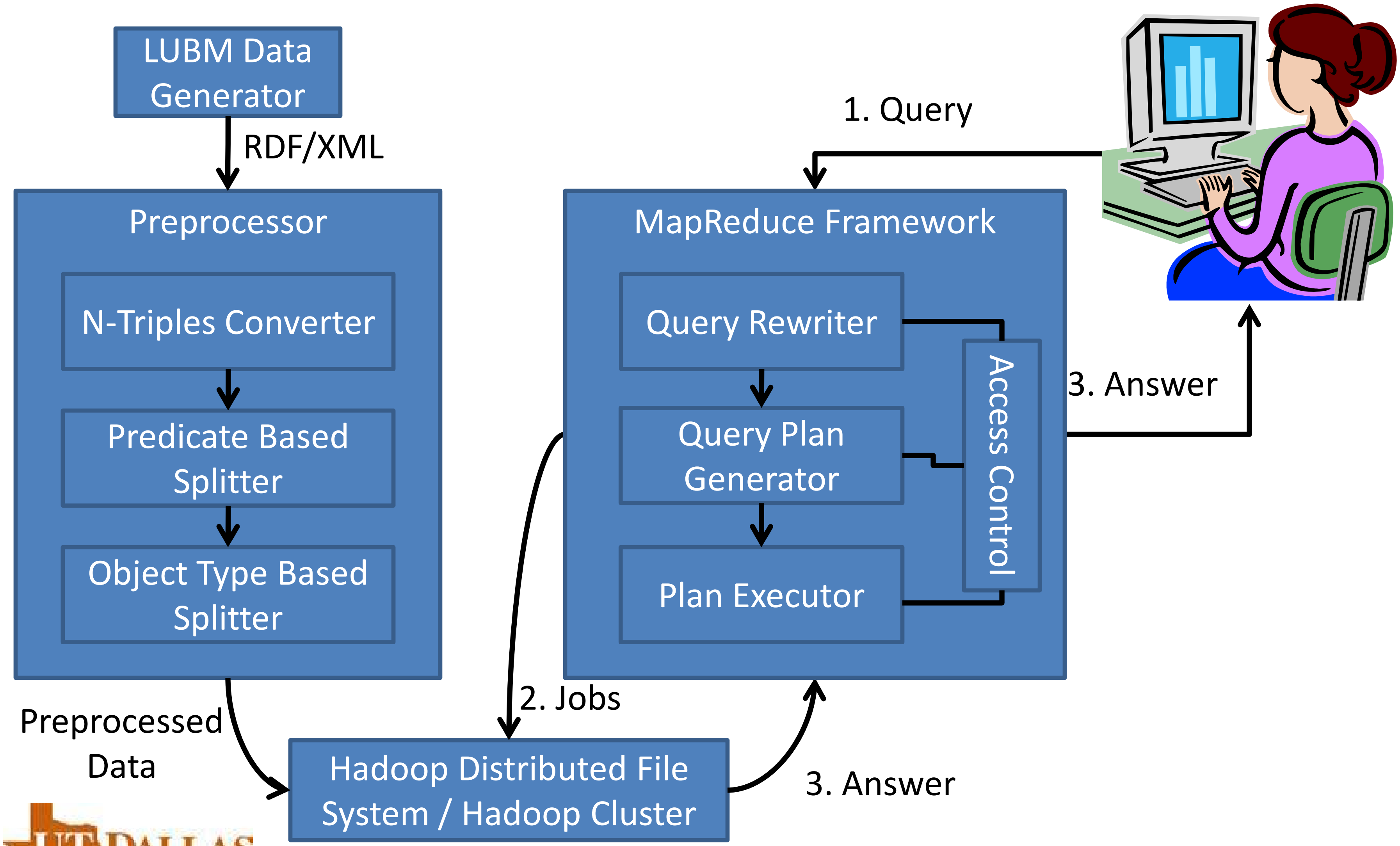
Cloud Computing Frameworks

- Proprietary
  - Amazon S3
  - Amazon EC2
  - Force.com
- Open source tool
  - Hadoop – Apache's open source implementation of Google's proprietary GFS file system
    - MapReduce – functional programming paradigm using key-value pairs

Cloud as RDF Stores

- Large RDF graphs can be efficiently stored and queried in the clouds [6, 12, 13, 18].
- These stores lack access control.
- We address this problem by generating tokens for specified access levels.
- Users are assigned these tokens based on their business requirements and restrictions.

System Architecture



LUBM Data Generator

RDF/XML

Preprocessor

N-Triples Converter

Predicate Based Splitter

Object Type Based Splitter

Preprocessed Data

1. Query

MapReduce Framework

Query Rewriter

Query Plan Generator

Plan Executor

Access Control

3. Answer

2. Jobs

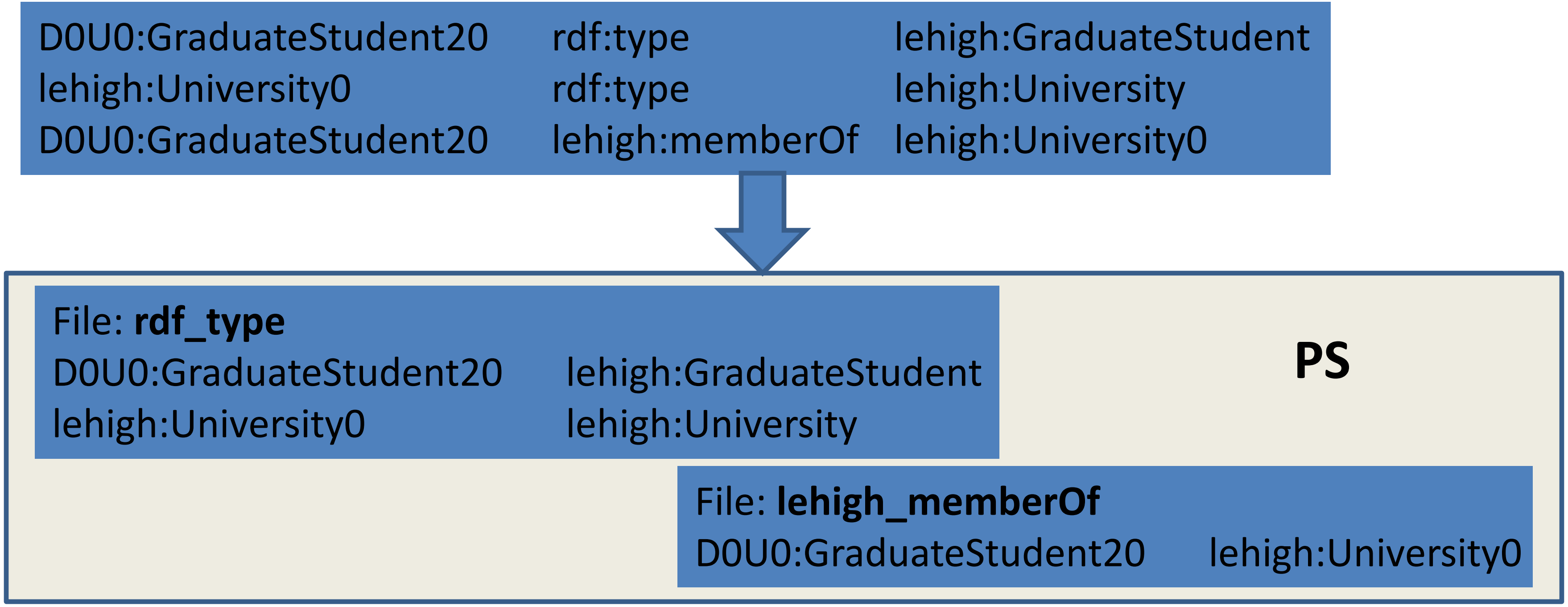
Hadoop Distributed File System / Hadoop Cluster

3. Answer

Storage Schema

- Data in N-Triples
- Using namespaces
  - Example:
    - http://utdallas.edu/res1 → utd:res1
- Predicate based Splits (PS)
  - Split data according to Predicates
- Predicate Object based Splits (POS)
  - Split further according to rdf:type of Objects

Example



D0U0:GraduateStudent20	rdf:type	lehigh:GraduateStudent
lehigh:University0	rdf:type	lehigh:University
D0U0:GraduateStudent20	lehigh:memberOf	lehigh:University0

File: **rdf_type**

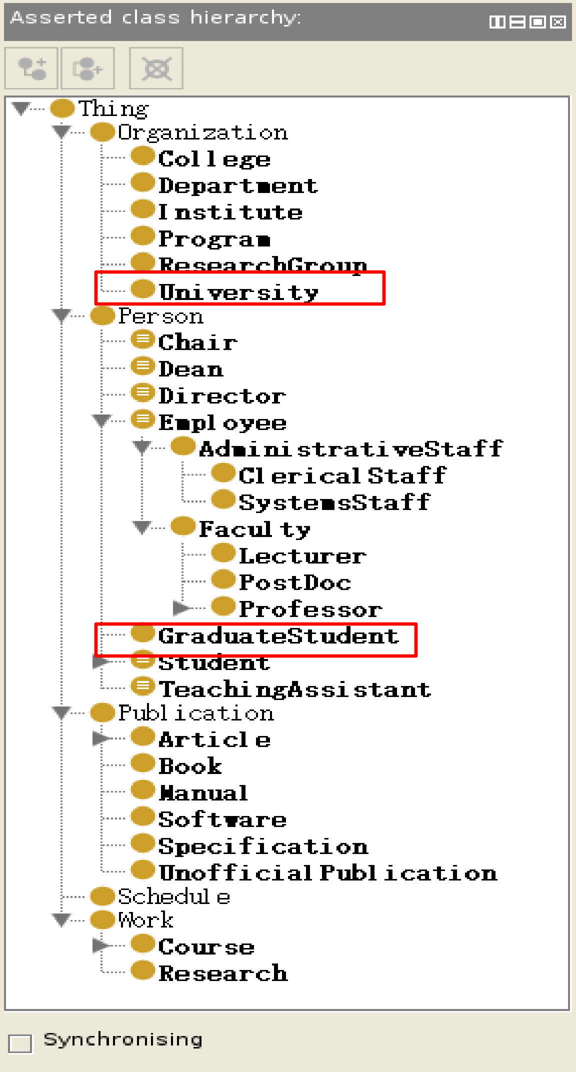
D0U0:GraduateStudent20	lehigh:GraduateStudent
lehigh:University0	lehigh:University

PS

File: **lehigh_memberOf**

D0U0:GraduateStudent20	lehigh:University0

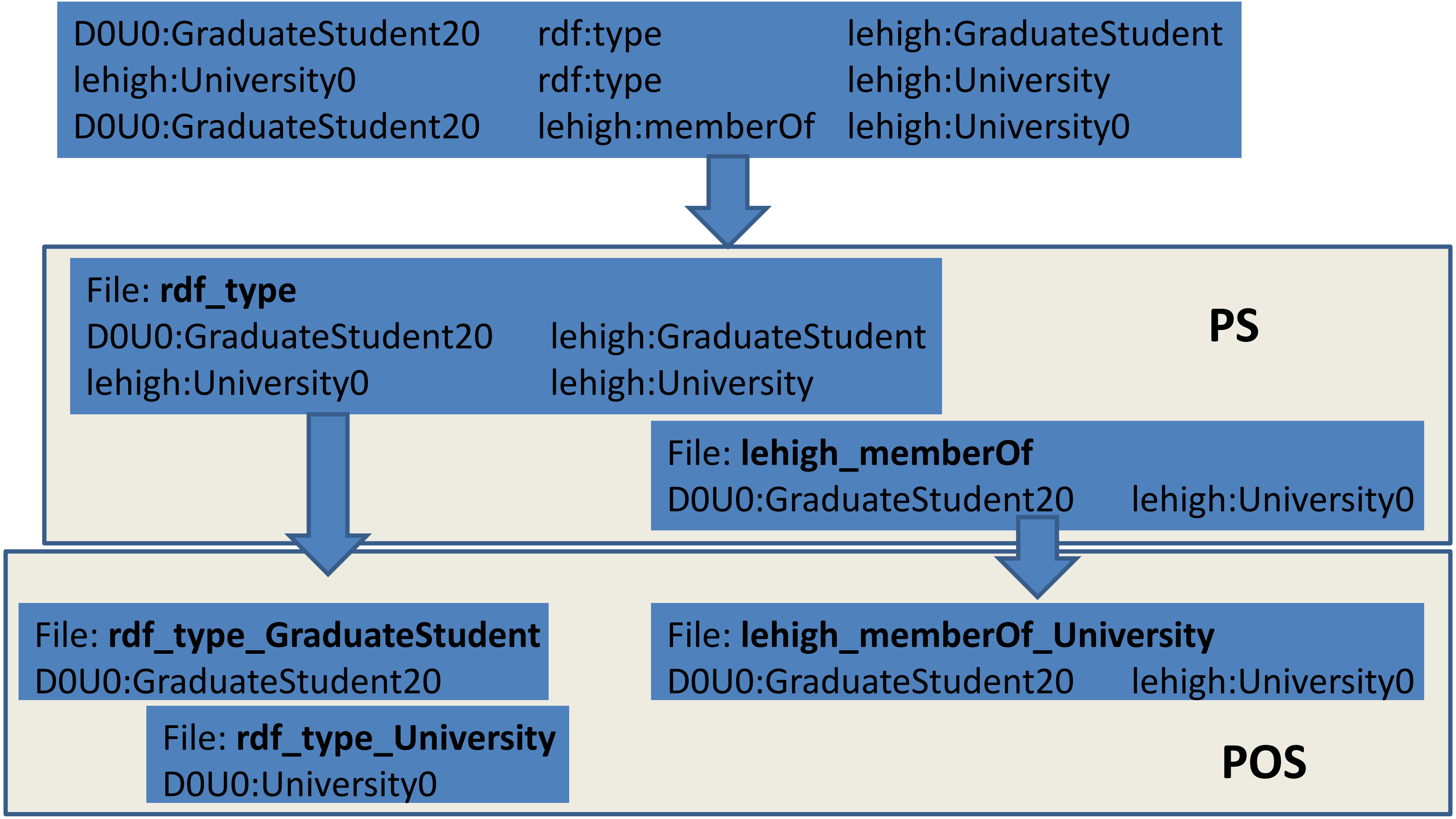
The Ontology



- Thing
  - Organization
    - College
    - Department
    - Institute
    - Program
    - ResearchGroup
    - University
  - Person
    - Chair
    - Dean
    - Director
    - Employee
      - AdministrativeStaff
        - ClericalStaff
        - SystemsStaff
      - Faculty
        - Lecturer
        - PostDoc
        - Professor
    - GraduateStudent
    - Student
    - TeachingAssistant
  - Publication
    - Article
    - Book
    - Manual
    - Software
    - Specification
    - UnofficialPublication
  - Schedule
  - Work
    - Course
    - Research

□ Synchronising

Example



D0U0:GraduateStudent20	rdf:type	lehigh:GraduateStudent
lehigh:University0	rdf:type	lehigh:University
D0U0:GraduateStudent20	lehigh:memberOf	lehigh:University0

PS

File: **rdf_type**

D0U0:GraduateStudent20	lehigh:GraduateStudent
lehigh:University0	lehigh:University

File: **lehigh_memberOf**

D0U0:GraduateStudent20	lehigh:University0

POS

File: **rdf_type_GraduateStudent**

D0U0:GraduateStudent20

File: **rdf_type_University**

D0U0:University0

File: **lehigh_memberOf_University**

D0U0:GraduateStudent20	lehigh:University0

Space Gain

- Example

Steps	Number of Files	Size (GB)	Space Gain
N-Triples	20020	24	--
Predicate Split (PS)	17	7.1	70.42%
Predicate Object Split (POS)	41	6.6	72.5%

Data size at various steps for LUBM1000

SPARQL Query

- SPARQL – **S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage
- Example

Subject	Predicate	Object
http://utdallas.edu/res1	foaf:name	“John Smith”
http://utdallas.edu/res1	foaf:age	“24”
http://utdallas.edu/res2	foaf:name	“John Doe”

Data

```
SELECT ?x ?y WHERE
{
  ?z foaf:name ?x
  ?z foaf:age   ?y
}
```

Query

?x	?y
“John Smith”	“24”

Result

SPAQL Query by MapReduce

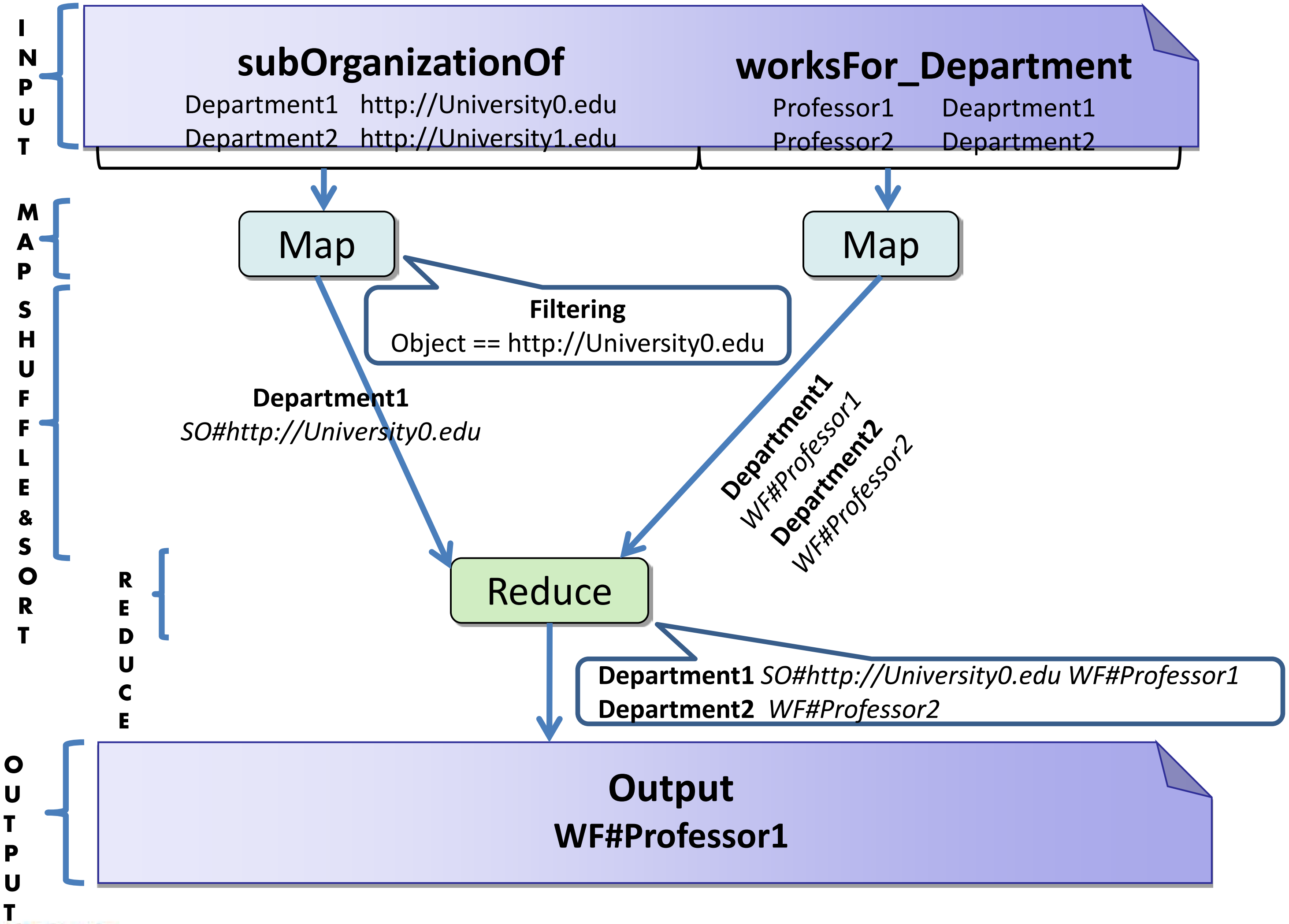
- Example query: select all who work for departments which are sub-organizations of http://University0.edu

  ```
  SELECT ?p WHERE
  {
    ?x   rdf:type  lehigh:Department
    ?p   lehigh:worksFor   ?x
    ?x   subOrganizationOf            http://University0.edu
  }
  ```

- Rewritten query

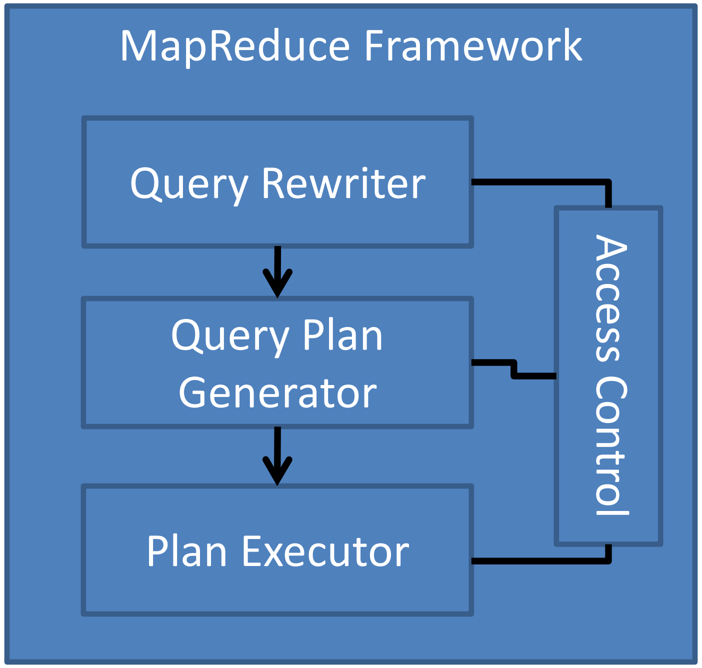
  ```
  SELECT ?p WHERE
  {
    ?p   lehigh:worksFor_Department          ?x
    ?x   subOrganizationOf            http://University0.edu
  }
  ```

Inside Hadoop MapReduce Job



Access Control in Our Architecture

Access control module is linked to all the components of MapReduce Framework



Motivation

- It’s important to keep the data safe from unwanted access.
- Encryption can be used, but it has no or small semantic value.
- By issuing and manipulating different levels of access control, the agent could access the data intended for him or make inferences.

Access Control Terminology

- **Access Tokens (AT):** Denoted by integer numbers allow agents to access security-relevant data.
- **Access Token Tuples (ATT):** Have the form <*AccessToken, Element, ElementType, ElementName*> where *Element* can be Subject, Object, or Predicate, and *ElementType* can be described as *URI* , *DataType*, *Literal* , *Model* (Subject), or *BlankNode*.

Six Access Control Levels

- **Predicate Data Access:** Defined for a particular predicate. An agent can access the predicate file. For example: An agent possessing ATT <*1, Predicate, isPaid,* _> can access the entire predicate file *isPaid.*
- **Predicate and Subject Data Access:** More restrictive than the previous one. Combining one of these Subject ATT's with a Predicate data access ATT having the same AT grants the agent access to a specific subject of a specific predicate. For example, having ATT's <*1, Predicate, isPaid,* _> and <*1, Subject, URI , MichaelScott*> permits an agent with AT 1 to access a subject with URI *MichaelScott* of predicate *isPaid*.

Access Control Levels (Cont.)

- **Predicate and Object:** This access level permits a principal to extract the names of subjects satisfying a particular predicate and object.
- **Subject Access:** One of the less restrictive access control levels. The subject can ne a *URI* , *DataType*, or *BlankNode*.
- **Object Access:** The object can be a *URI* , *DataType*, *Literal* , or *BlankNode*.

Access Control Levels (Cont.)

- **Subject Model Level Access:** This permits an agent to read all necessary predicate files to obtain all objects of a given subject. The ones which are URI objects obtained from the last step are treated as subjects to extract their respective predicates and objects. This iterative process continues until all objects finally become blank nodes or literals. Agents may generate models on a given subject.

Access Token Assignment

- Each agent contains an Access Token list (*AT*-list) which contains 0 or more *AT*s assigned to the agents along with their issuing timestamps.
- These timestamps are used to resolve conflicts (explained later).
- The set of triples accessible by an agent is the union of the result sets of the AT's in the agent's AT-*list*.

Conflict

- A conflict arises when the following three conditions occur:
  - An agent possesses two AT's 1 and 2,
  - the result set of AT 2 is a proper subset of AT 1, and
  - the timestamp of AT 1 is earlier than the timestamp of AT 2
- Later, more specific AT supersedes the former, so AT 1 is discarded from the AT-list to resolve the conflict.

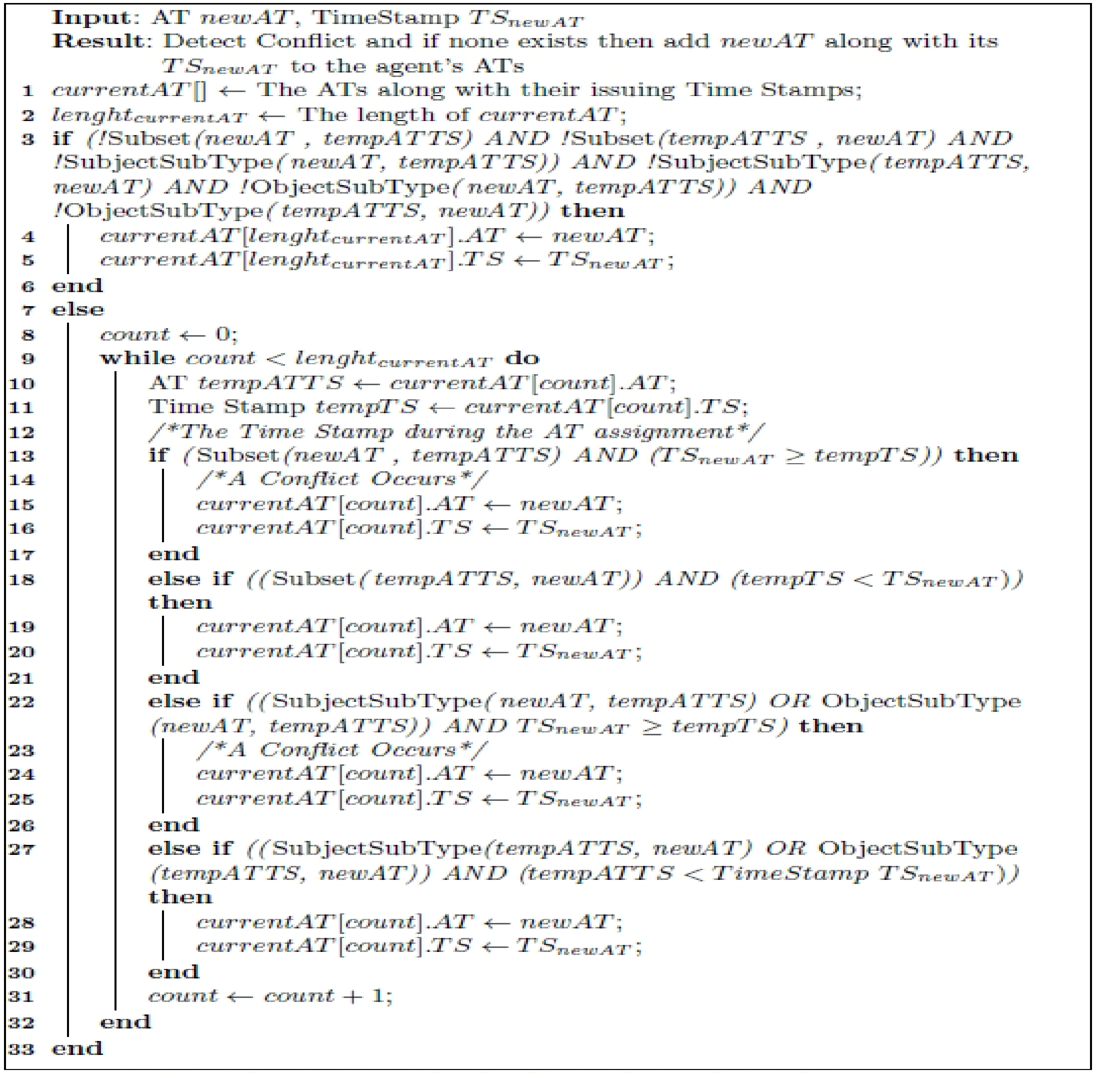
Conflict Type

- **Subset Conflict**: It occurs when AT 2 (later issued) is a conjunction of ATT's that refine AT 1. For example, AT 1 is defined by <*1, Subject, URI, Sam*> and AT 2 is defined by <*2, Subject, URI, Sam*> and <*2, Predicate, HasAccounts, _*> ATT's. If AT 2 is issued to the possessor of AT 1 at a later time, then a conflict will occur and AT 1 will be discarded from the agent's AT-list.

Conflict Type

- **Subtype conflict:** Subtype conflicts occur when the ATT's in AT 2 involve data types that are subtypes of those in AT 1. The data types can be those of subjects, objects or both.

Conflict Resolution Algorithm



```
Input: AT $newAT$, TimeStamp $TS_{newAT}$
Result: Detect Conflict and if none exists then add $newAT$ along with its
        $TS_{newAT}$ to the agent's ATs
$currentAT[]$ ← The ATs along with their issuing Time Stamps;
$lenght_{currentAT}$ ← The length of $currentAT$;
if (!Subset($newAT$ , $tempATTS$) AND !Subset($tempATTS$ , $newAT$) AND
!SubjectSubType($newAT$, $tempATTS$)) AND !SubjectSubType($tempATTS$,
$newAT$) AND !ObjectSubType($newAT$, $tempATTS$)) AND
!ObjectSubType($tempATTS$, $newAT$)) then
    $currentAT[lenght_{currentAT}].AT \leftarrow newAT$;
    $currentAT[lenght_{currentAT}].TS \leftarrow TS_{newAT}$;
end
else
    $count \leftarrow 0$;
    while $count < lenght_{currentAT}$ do
        AT $tempATTS \leftarrow currentAT[count].AT$;
        Time Stamp $tempTS \leftarrow currentAT[count].TS$;
        /*The Time Stamp during the AT assignment*/
        if (Subset($newAT$ , $tempATTS$) AND ($TS_{newAT} \geq tempTS$)) then
            /*A Conflict Occurs*/
            $currentAT[count].AT \leftarrow newAT$;
            $currentAT[count].TS \leftarrow TS_{newAT}$;
        end
        else if ((Subset($tempATTS$, $newAT$)) AND ($tempTS < TS_{newAT}$))
        then
            $currentAT[count].AT \leftarrow newAT$;
            $currentAT[count].TS \leftarrow TS_{newAT}$;
        end
        else if ((SubjectSubType($newAT$, $tempATTS$) OR ObjectSubType
        ($newAT$, $tempATTS$)) AND $TS_{newAT} \geq tempTS$) then
            /*A Conflict Occurs*/
            $currentAT[count].AT \leftarrow newAT$;
            $currentAT[count].TS \leftarrow TS_{newAT}$;
        end
        else if ((SubjectSubType($tempATTS$, $newAT$) OR ObjectSubType
        ($tempATTS$, $newAT$)) AND ($tempATTS < TimeStamp\ TS_{newAT}$))
        then
            $currentAT[count].AT \leftarrow newAT$;
            $currentAT[count].TS \leftarrow TS_{newAT}$;
        end
        $count \leftarrow count + 1$;
    end
end
```

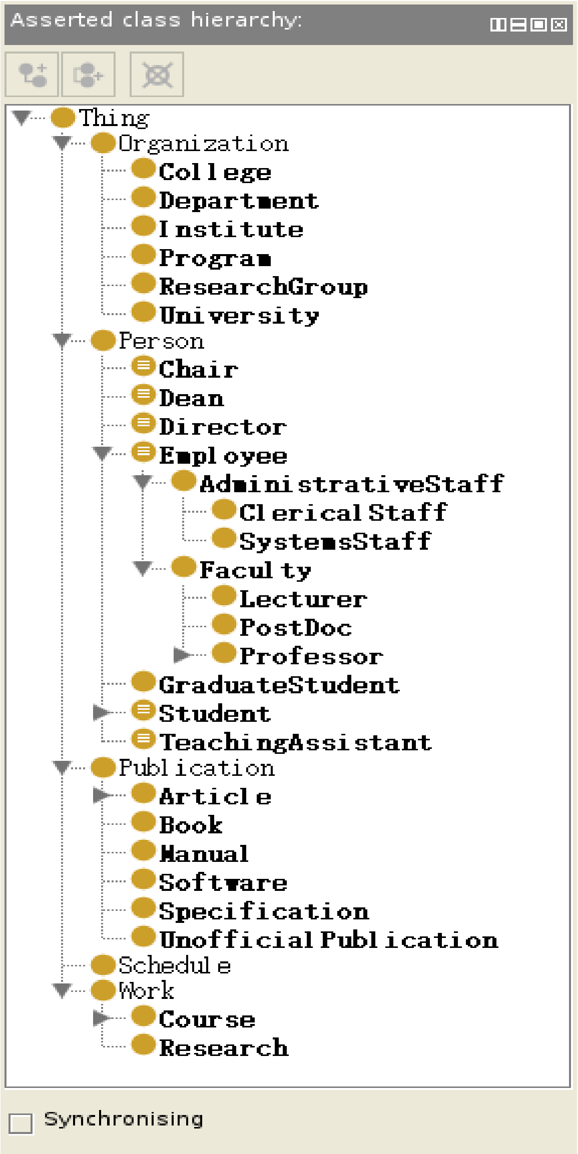
Algorithm 1: The Conflict Detection and Resolution.

Experiment

- Dataset and queries
- Cluster description
- Comparison with Jena In-Memory, SDB and BigOWLIM frameworks
- Experiments with number of Reducers
- Algorithm runtimes: Greedy vs. Exhaustive
- Some query results

Dataset And Queries

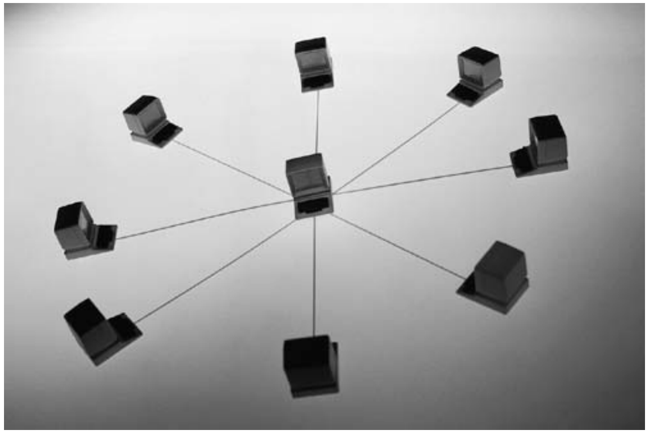
- LUBM
  - Dataset generator
  - 14 benchmark queries
  - Generates data of some imaginary universities
  - Used for query execution performance comparison by many researches



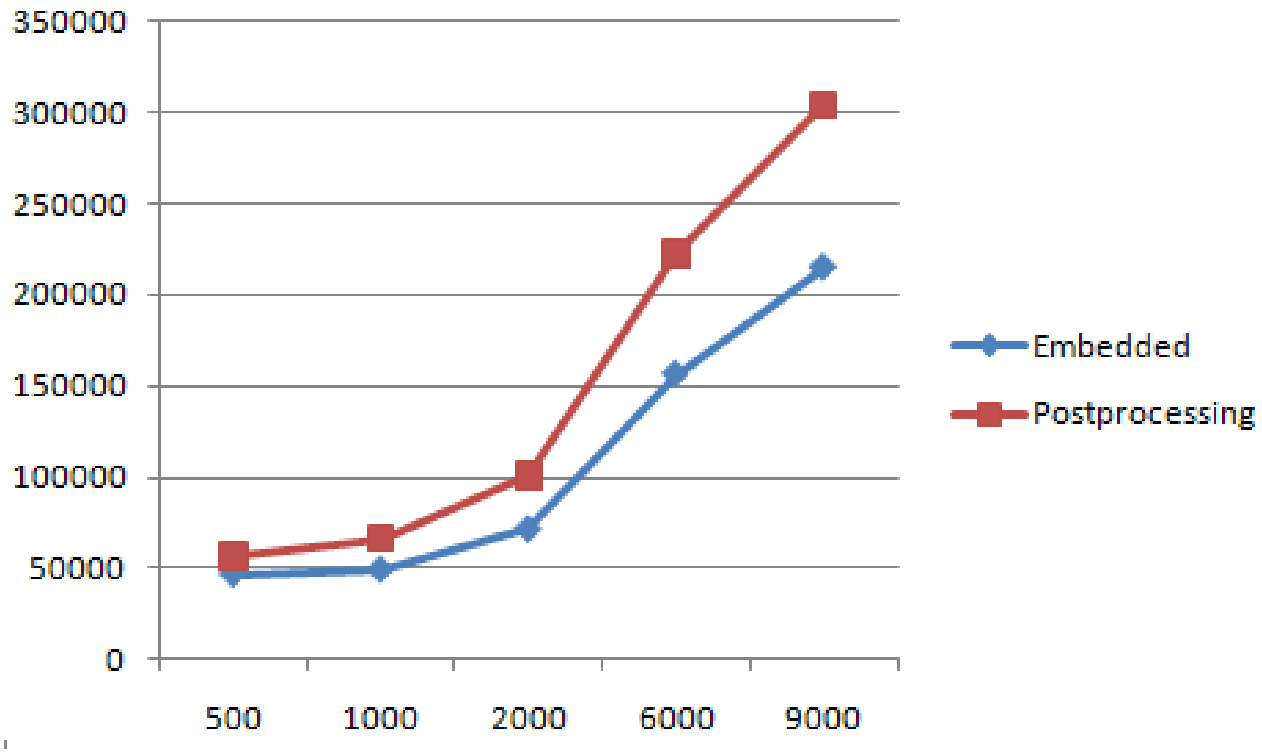
- Thing
  - Organization
    - College
    - Department
    - Institute
    - Program
    - ResearchGroup
    - University
  - Person
    - Chair
    - Dean
    - Director
    - Employee
      - AdministrativeStaff
        - ClericalStaff
        - SystemsStaff
      - Faculty
        - Lecturer
        - PostDoc
        - Professor
    - GraduateStudent
    - Student
    - TeachingAssistant
  - Publication
    - Article
    - Book
    - Manual
    - Software
    - Specification
    - UnofficialPublication
  - Schedule
  - Work
    - Course
    - Research

Our Clusters

- 10 node cluster in SAIAL lab
  - 4 GB main memory
  - Intel Pentium IV 3.0 GHz processor
  - 640 GB hard drive
- OpenCirrus HP labs test bed

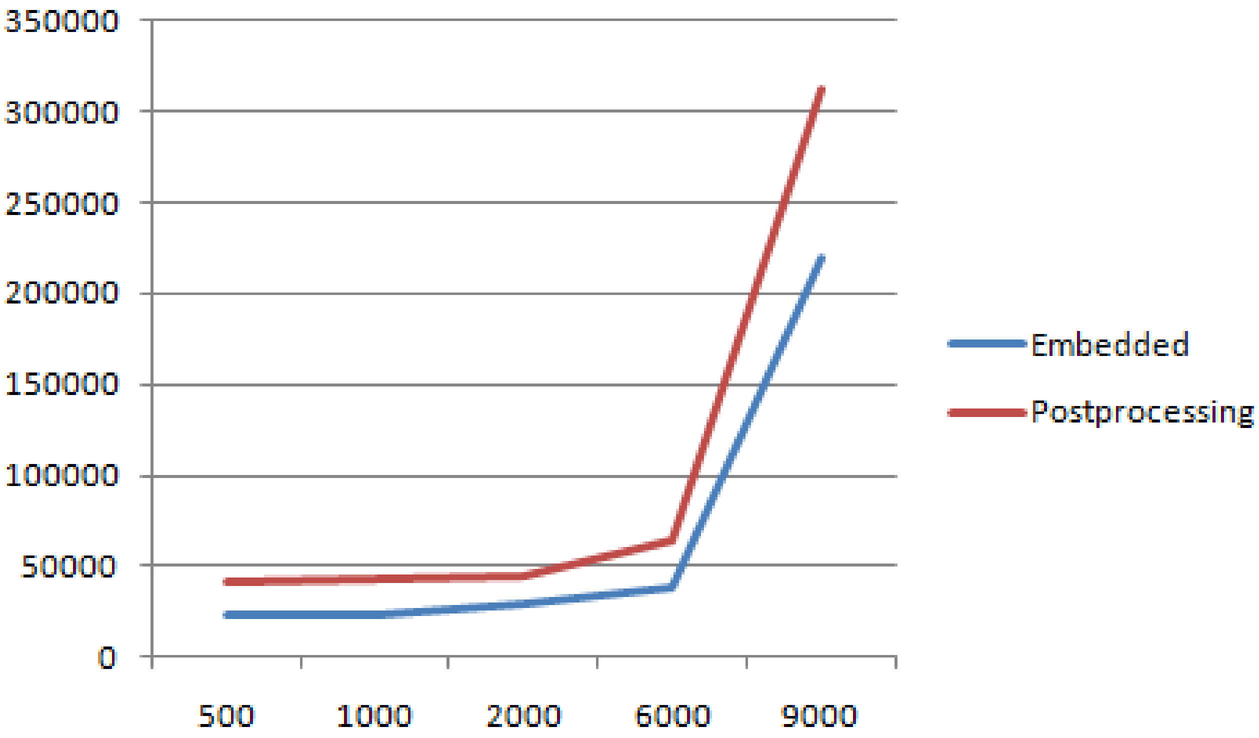


Results



Scenario 1: “takesCourse”
A list of sensitive courses cannot
be viewed by a normal user for any student

Results



Scenario 2: “displayTeachers”
A normal user is allowed to view information about the lecturers only

Future Works

- Build a generic system that incorporates tokens and resolve policy conflicts.
- Implement Subject Model Level Access that recursively extracts objects of subjects and treats these objects as subjects as long as these objects are URIs. An agent with proper access level can construct a model on that subject.

References

- [1] Apache. Hadoop. http://hadoop.apache.org/.
- [2] D. Beckett. RDF/XML syntax specification (revised). Technical report, W3C, February 2004.
- [3] T. Berners-Lee. Semantic web road map. http://www.w3.org/DesignIssues/Semantic.html, 1998.
- [4] L. Bouganim, F. D. Ngoc, and P. Pucheral. Client based access control management for XML documents. In Proc. 20´emes Journ´ees Bases de Donn´ees Avanc´ees (BDA),pages 65–89, Montpellier, France, October 2004.

References

- [5] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF. In Proc. 1st International Semantic Web Conference (ISWC), pages 54–68, Sardinia, Italy, June 2002.
- [6] H. Choi, J. Son, Y. Cho, M. K. Sung, and Y. D. Chung. SPIDER: a system for scalable, parallel / distributed evaluation of large-scale RDF data. In Proc. 18th ACM Conference on Information and Knowledge Management (CIKM), pages 2087–2088, Hong Kong, China, November 2009.
- [7] J. Grant and D. Beckett. RDF test cases. Technical report, W3C, February 2004.
- [8] Y. Guo, Z. Pan, and J. Heflin. An evaluation of knowledge base systems for large OWL datasets. In In Proc. 3rd International Semantic Web Conference (ISWC), pages 274–288, Hiroshima, Japan, November 2004.
- [9] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics, 3(2–3):158–182, 2005.

References

- [10] S. Harris and N. Shadbolt. SPARQL query processing with conventional relational database systems. In Proc. Web Information Systems Engineering (WISE) International Workshop on Scalable Semantic Web Knowledge Base Systems
- (SSWS), pages 235–244, New York, New York, November 2005.
- [11] L. E. Holmquist, J. Redstr¨om, and P. Ljungstrand. Token based access to digital information. In Proc. 1st International Symposium on Handheld and Ubiquitous Computing (HUC), pages 234–245, Karlsruhe, Germany, September 1999.
- [12] M. F. Husain, P. Doshi, L. Khan, and B. M. Thuraisingham. Storage and retrieval of large RDF graph using Hadoop and MapReduce. In Proc. 1st International Conference on Cloud Computing (CloudCom), pages 680–686, Beijing, China, December 2009.

References

- [13] M. F. Husain, L. Khan, M. Kantarcioglu, and B. Thuraisingham. Data intensive query processing for large RDF graphs using cloud computing tools. In Proc. IEEE 3rd International Conference on Cloud Computing (CLOUD), pages 1–10, Miami, Florida, July 2010.
- [14] A. Jain and C. Farkas. Secure resource description framework: an access control model. In Proc. 11th ACM Symposium on Access Control Models and Technologies (SACMAT), pages 121–129, Lake Tahoe, California, June 2006.
- [15] Joseki. http://www.joseki.org.

References

- [16] J. Kim, K. Jung, and S. Park. An introduction to authorization conflict problem in RDF access control. In Proc. 12th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES), pages 583– 592, Zagreg, Croatia, September 2008.
- [17] Kowari. http://kowari.sourceforge.net.
- [18] P. Mika and G. Tummarello. Web semantics in the clouds. IEEE Intelligent Systems, 23(5):82–87, 2008.
- [19] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. Technical report, W3C, January 2008.
- [20] P. Reddivari, T. Finin, and A. Joshi. Policy based access control for an RDF store. In Proc. Policy Management for the Web Workshop, 2005.