Scalable Ontology Reasoning in Spark Environment

김제민

명지대학교 방목기초교육대학

2015.08.21

Why Scalable Ontology Reasoning

An Ontology is an engineering artefact consisting of :

Shared vocabulary of terms

An explicit specification their intended meaning

Inference rules to represent domain-specific knowledge

They are set to play a key role in many applications:

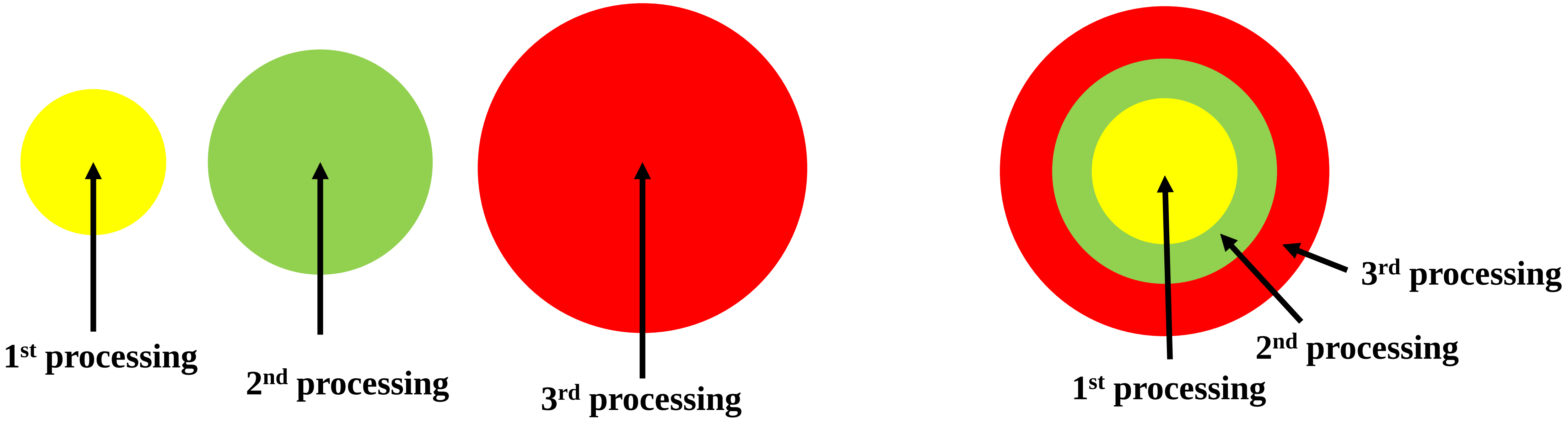
e-Science, Medicine, Databases, Semantic Web, Knowledge Service

These applications require a massive amount of knowledge

→ Need to reasoning approaches for large scale ontology

The Problem of General MapReduce Approach

MapReduce is a good approach for big data processing
- Acyclic data flow is a powerful abstraction
- Not efficient for applications that repeatedly reuse a working set of data
- Not efficient for join operation



Alternative Approach using In-memory of Cluster

Spark - in-memory cluster computing for iterative and interactive applications
- Loading data as Resilient Distributed Datasets into the memory in a cluster
- Executing MapReduce-based operations.

The cost of computer memory is continuously dropping
- Most computer systems have increasingly higher capacity memory

Distributed memories in cluster are efficiently utilized
- Reasoner can perform faster for scalable ontologies

Outline

- Ontology representation and reasoner
- Scalable Ontology Reasoning in Spark Environment
  - Rule execution order
  - Data layout and operations
  - Reasoning step
- Experimental evaluations

Ontology Representation

RDF - basic language that is used to present ontologies

- Originally design as a metadata data model using XML syntax

RDFS - an extension of the RDF

- Providing a data-modelling vocabulary for web resources
- some semantic rules to perform reasoning

OWL Horst - semantics that are weaker than the standard OWL

- The computation of closure has a low complexity
- Including more complex semantics than the RDFS

Ontology Representation

Languages to provide more complex semantics are required

- OWL - design to represent rich and complex knowledge
- OWL Full or DL-based scalable ontology reasoning has an undecidable problem

$\mathcal{S} \leftarrow$ intersection ($\sqcap$), union ($\sqcup$), existential ($\exists$), universal ($\forall$), complement ($\neg$), role transitive($\mathcal{R}^{+}$)

$\mathcal{H} \leftarrow$ role hierarchies, $\mathcal{I} \leftarrow$ inverse roles

$\mathcal{F} \leftarrow$ functional roles

Ontology Representation

RDFS

OWL Horst Lite

OWL Horst

	Antecedents	Consequent
R1:	p rdfs:domain x, $s\ p\ o$	$\Rightarrow$ s rdf:type x
R2:	p rdfs:range x, $s\ p\ o$	$\Rightarrow$ o rdf:type x
R3:	p rdfs:subPropertyOf q, q rdfs:subPropertyOf r	$\Rightarrow$ p rdfs:subPropertyOf r
R4:	$s\ p\ o$, p rdfs:subPropertyOf q	$\Rightarrow$ $s\ q\ o$
R5:	s rdf:type x, x rdfs:subClassOf y	$\Rightarrow$ s rdf:type y
R6:	x rdfs:subClassOf y, y rdfs:subClassof z	$\Rightarrow$ x rdfs:subClassOf z
O1:	p rdf:type owl:FunctionalProperty, $u\ p\ v$, $u\ p\ w$	$\Rightarrow$ v owl:sameAs w
O2:	p rdf:type owl:InverseFunctionalProperty, $v\ p\ u$, $w\ p\ u$	$\Rightarrow$ v owl:sameAs w
O3:	p rdf:type owl:SymmetricProperty, $v\ p\ u$	$\Rightarrow$ $u\ p\ v$
O4:	p rdf:type owl:TransitiveProperty, $u\ p\ w$, $w\ p\ v$	$\Rightarrow$ $u\ p\ v$
O5:	v owl:sameAs w	$\Rightarrow$ w owl:sameAs v
O6:	v owl:sameAs w, w owl:sameAs u	$\Rightarrow$ v owl:sameAs u
O7a:	p owl:inverseOf q, $v\ p\ w$	$\Rightarrow$ $w\ q\ v$
O7b:	p owl:inverseOf q, $v\ q\ w$	$\Rightarrow$ $w\ p\ v$
O8:	v rdf:type owl:Class, v owl:sameAs w	$\Rightarrow$ v rdfs:subClassOf w
O9:	p rdf:type owl:Property, p owl:sameAs q	$\Rightarrow$ p rdfs:subPropertyOf q
O10:	$u\ p\ v$, u owl:sameAs x, v owl:sameAs y	$\Rightarrow$ $x\ p\ y$
O11a:	v owl:equivalentClass w	$\Rightarrow$ v rdfs:subClassOf w
O11b:	v owl:equivalentClass w	$\Rightarrow$ w rdfs:subClassOf v
O11c:	v rdfs:subClassOf w, w rdfs:subClassOf v	$\Rightarrow$ v rdfs:equivalentClass w
O12a:	v owl:equivalentProperty w	$\Rightarrow$ v rdfs:subPropertyOf w
O12b:	v owl:equivalentProperty w	$\Rightarrow$ w rdfs:subPropertyOf v
O12c:	v rdfs:subPropertyOf w, w rdfs:subPropertyOf v	$\Rightarrow$ v rdfs:equivalentProperty w
O13a:	v owl:hasValue w, v owl:onProperty p, $u\ p\ w$	$\Rightarrow$ u rdf:type v
O13b:	v owl:hasValue w, v owl:onProperty p, u rdf:type v	$\Rightarrow$ $u\ p\ w$
O14:	v owl:someValuesFrom w, v owl:onProperty p, $u\ p\ x$, x rdf:type w	$\Rightarrow$ u rdf:type v
O15:	v owl:allValuesFrom u, v owl:onProperty p, w rdf:type v, $w\ p\ x$	$\Rightarrow$ x rdf:type u

Ontology Reasoner

Ontology Reasoner	Development Company/Univ	
FaCT, FaCT++	U. of Manchester	OWL, tableaux algorithm, single machine
Pellet	U. of Maryland, MIND Lab.	OWL, tableaux algorithm, Ontology debugging, single machine
Hermit	U. of Oxford	OWL2, hypertableau algorithm, single machine
AllegroGraph	Franz Inc	RDF++, rule-based reasoning, single machine
OWLIM	ontotext	OWL-Horst, rule-based reasoning, single machine
WebPIE	U. of Amsterdam	OWL-Horst, mapreduce, cluster

Tbox & Abox reasoning

TBox – the terminology, i.e., the vocabulary of an application domain
= concepts (set of individuals) + roles
TBox reasoning - organizing the concepts of a terminology into a hierarchy according to their generality

ABox – assertions about named individuals in terms of this vocabulary
ABox Reasoning - entailing that a particular individual is an instance of a given concept description

Tbox & Abox reasoning

TBox

SubclassOf	SubpropertyOf
Equivalent Classes	Equivalent Properties

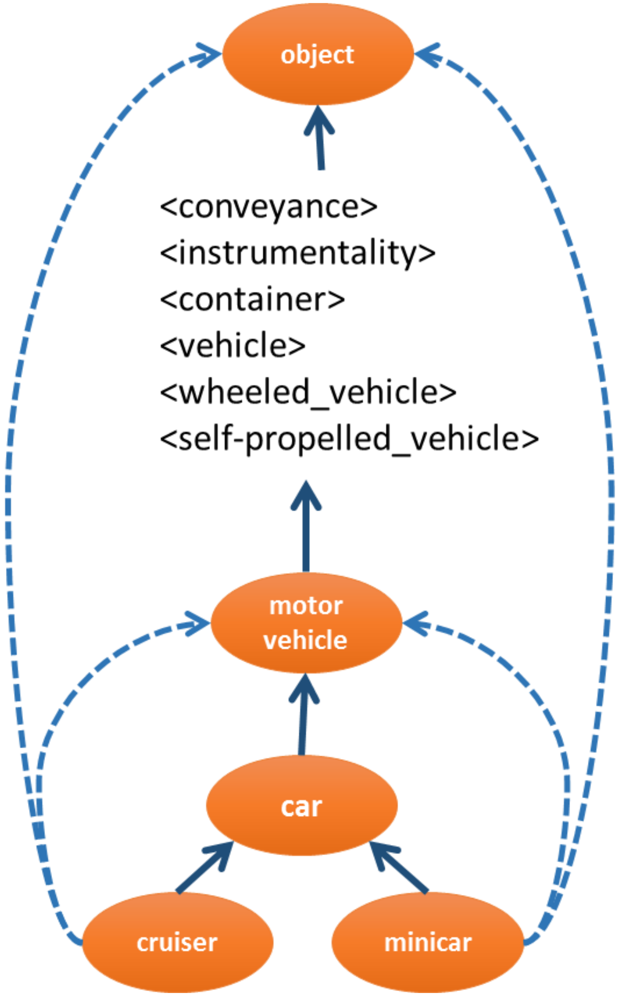
ABox

Property Inheritance	Object Property Domain	Object Property Range
Type Inheritance	Instance Equivalent	Property Functional & InvFunctional
Property Symmetric	Property Transitive	Property Inverse
Object SomeValuesFrom	Object AllValuesFrom	Object HasValue
Type IntersectionOf	Type UnionOf	

Ontology Reasoning

Triples
<cruiser_03026980> <subClassOf> <car_02853224> .

Reasoned Triples
<cruiser_03026980>
<subClassOf> <motor_vehicle_03649150> .
<subClassOf> <self-propelled_vehicle_04011657> .
<subClassOf> <wheeled_vehicle_04398733> .
<subClassOf> <vehicle_04348422> .
<subClassOf> <container_02982528> .
<subClassOf> <instrumentality_03443493> .
<subClassOf> <conveyance_02988377> .
<subClassOf> <artifact_00019244> .
<subClassOf> <entity_00001740> .
<subClassOf> <object_00016236> .



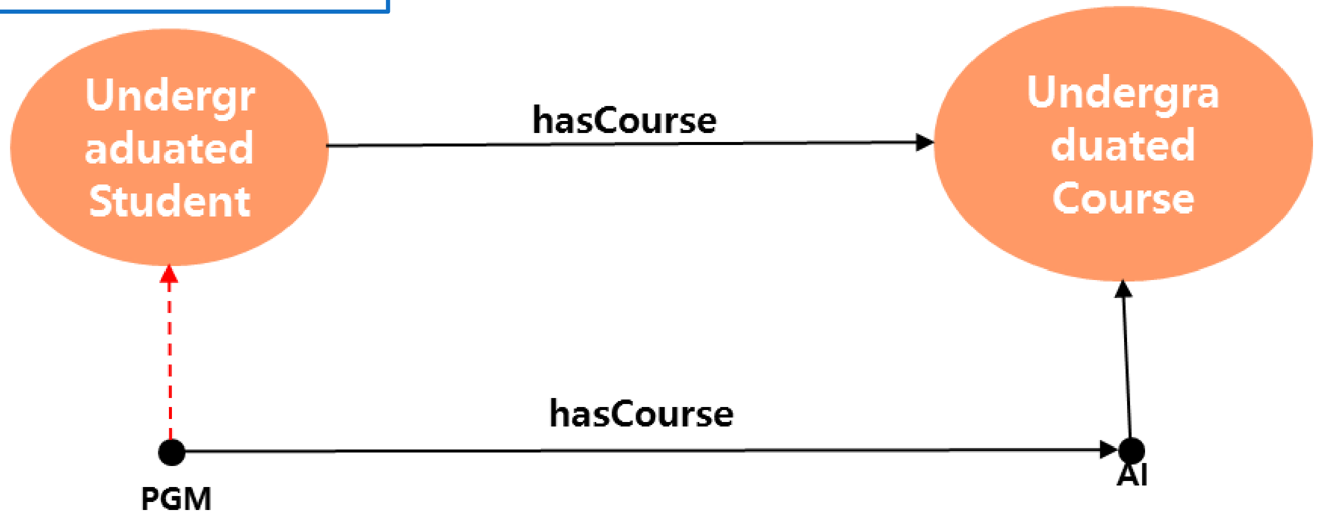
Ontology Reasoning

Triples

UndergraduatedStudent	someValuesFrom	UndergraduatedCourse
UndergraduatedStudent	onProperty	hasCourse
PGM	hasCourse	AI
AI	type	UndergraduatedCourse

Reasoned Triples

PGM	type	UndergraduatedStudent



Scalable ontology reasoning approach

- **Approaches of Distributed file**
  - MapReduce approach
    - Building reasoner using MapReduce algorithm
  - SQL approach
    - Hive, Pig
      - Optimized interpreter – automatically generating MapReduce jobs
- **Approaches of Distributed in-memory**
  - SQL approach
    - Impala
  - Spark framework
    - In-memory cluster computing

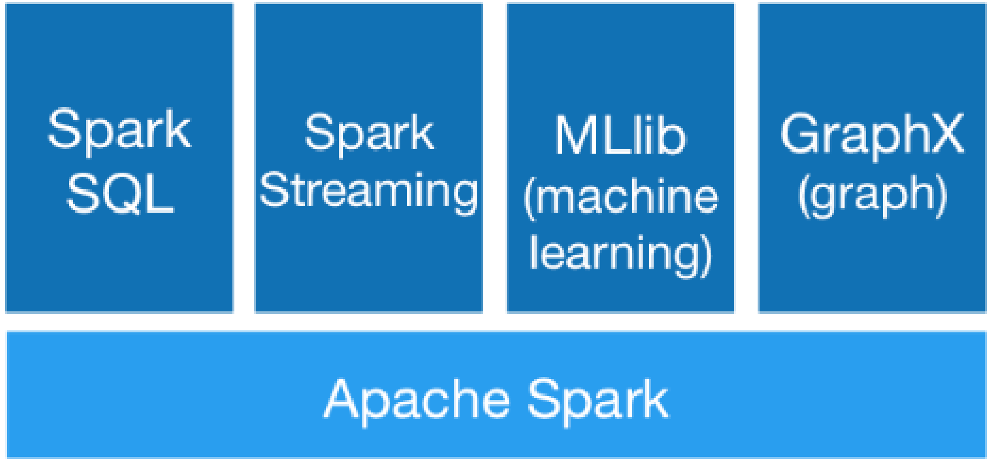
SPARK : Cluster Computing Platform

Spark - in-memory cluster computing for iterative and interactive applications
run programs up to 100x faster than Hadoop MapReduce in memory

Key concepts - Write programs in terms of transformations on distributed datasets

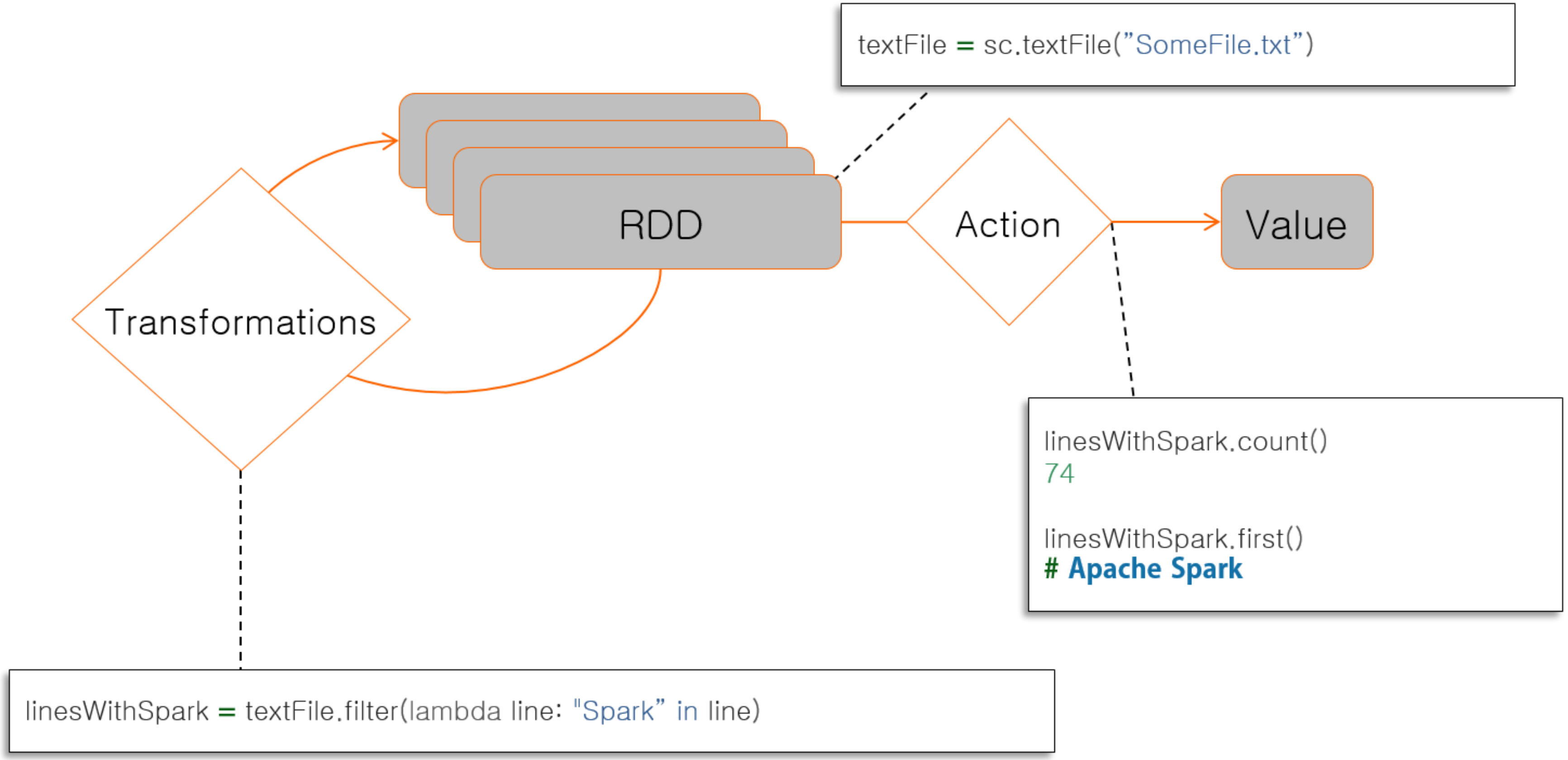
Resilient Distributed Datasets

- Collections of objects spread across a cluster, stored in RAM or on Disk
  Built through parallel transformations

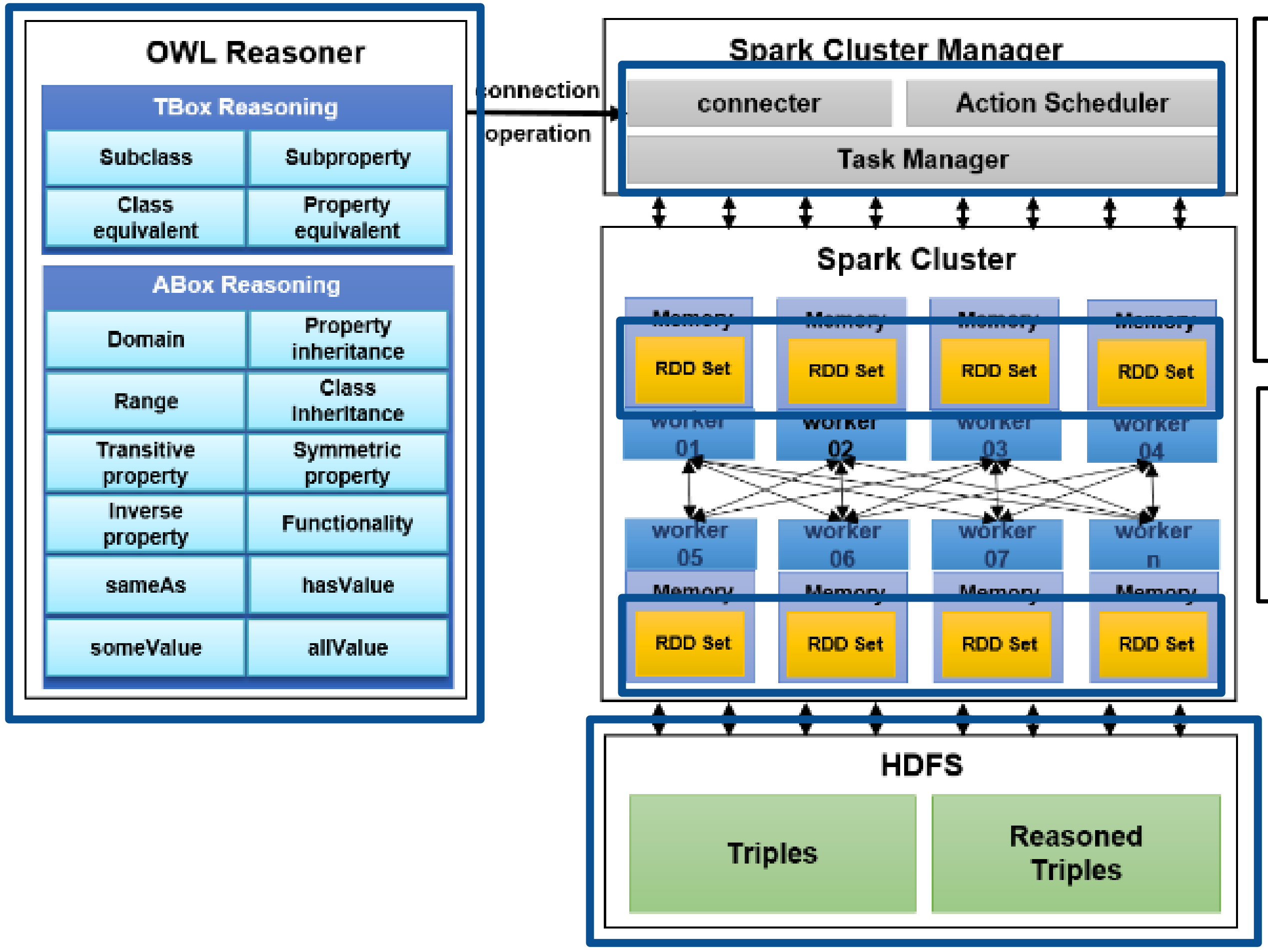


SPARK : Cluster Computing Platform

Working with RDD



SPARK based OWL Horst Reasoner



The OWL reasoner –
- A driver that performs reasoning on Spark
- Containing operations for TBox and ABox reasoning
- Defining distributed data sets (ontologies) on a cluster
- Appling operations to Spark

1. Connecting to the cluster through a Spark connecter
2. Converting ontology triples in HDFS to triple-RDDs
3. Loading the RDD triples to memories in the cluster

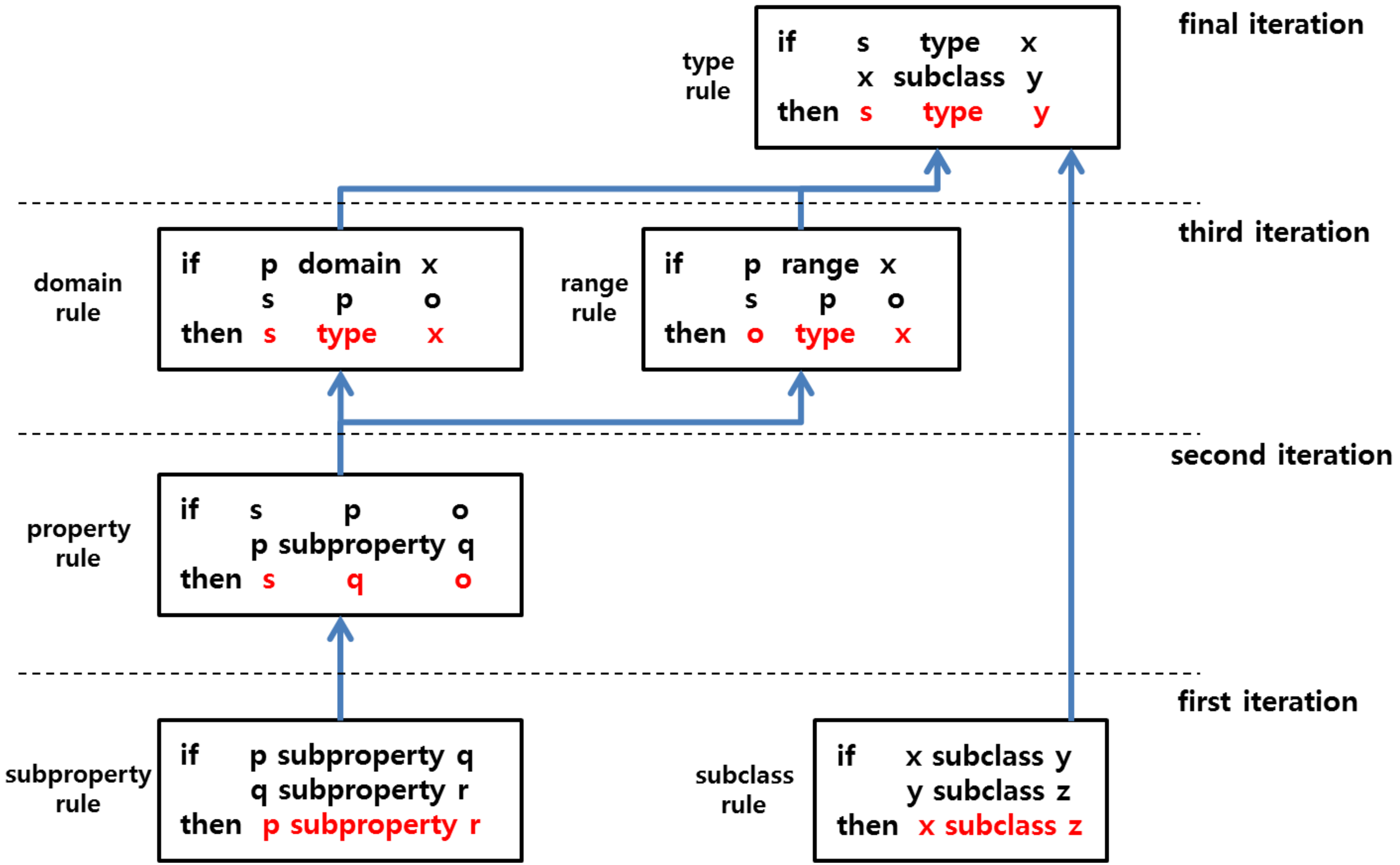
The task manager assigns jobs to each worker through the action scheduler.
Each worker performs reasoning independently.

Scalable Ontology Reasoning in Spark Environment

There are 3 key points for scalable ontology reasoning in Spark environment

- The execution order of reasoning rules that takes into account interdependencies and semantic relations.
- The data structure and operations for rule-based reasoning.
- The reasoning algorithms for executing each rule on a distributed memory system.

Rule Dependency of RDFS



Rule Dependency of OWL Horst

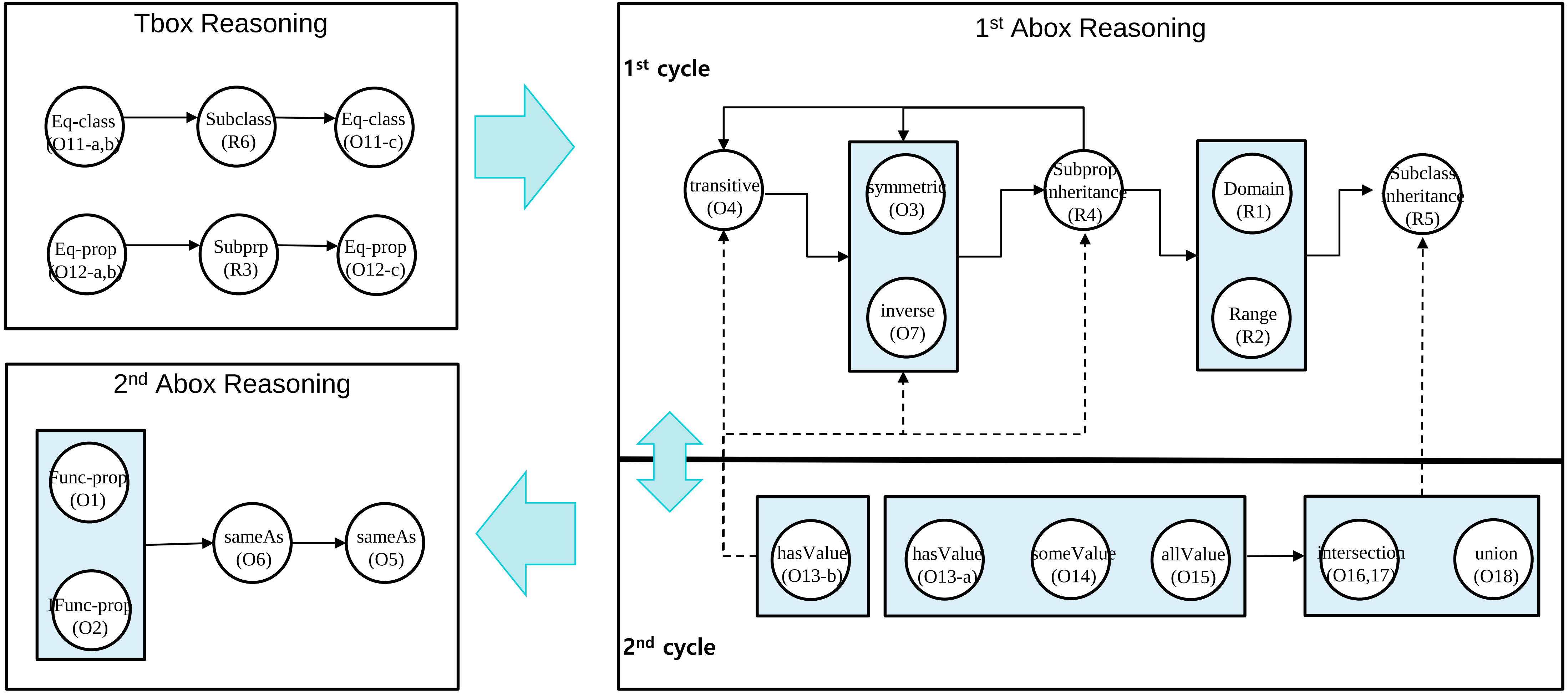
Tbox Reasoning → 1^{st} Abox Reasoning → 2^{nd} Abox Reasoning

Inferred condition $c \in \mathcal{T}, c \notin \mathcal{A}$
∴ first execution

Inferred condition $c \in \mathcal{T}, c \in \mathcal{A}$
Result $r \in \mathcal{A}$
∴ second execution

Inferred condition, $c \in \mathcal{A}$
Result $r \in \mathit{sameAs}$
∴ final execution

Rule Dependency of OWL Horst



Data Layout

N-triple - represents all RDF statements as
<subject>, <predicate> and <object> (SPO) statements.

The basic data structure is pair RDDs, composed of key-value pair.
converting triples to an RDD that is suitable for executes operations for each reasoning rule

```
<owl:Class rdf:about="Test#Teacher"/>
<owl:Class rdf:about="Test#Professor">
    <rdfs:subClassOf rdf:resource="Test#Teacher"/>
</owl:Class>
<owl:Class rdf:about="Test#Subject"/>
<owl:ObjectProperty rdf:about="Test#hasSubject">
    <rdfs:range rdf:resource="Test#Subject"/>
    <rdfs:domain rdf:resource="Test#Teacher"/>
</owl:ObjectProperty>
```

```
<Teacher> <type> <Class>.
<Professor> <type> <Class>.
<Subject> <type> <Class>.
<hasSubject> <type> <ObjectProperty>.
<Professor> <subclassOf> <Teacher>.
<hasSubject> <domain> <Teacher>.
<hasSubject> <range> <Subject>.
```

Subclass RDD

Father	Parents

Domain RDD

hasSubject	Teacher

Range RDD

hasSubject	Subject

Data Layout

The eight types of pair RDDs that are classified by features of patterns in OWL Horst rules

Form	Structure
Type1_pair	<key, value>
Type2_pair	<key, composed value>
Type3_pair	<key, <first value, second value>>
Type4_pair	<composed key, value>
Type5_pair	<composed key, composed value>
Type6_pair	<composed key, <composed first value, composed second value>>
Type7_pair	<key, <first value, composed second value>>
Type8_pair	<key, <composed second value, second value>>

Data Layout

<Father>	<rdfs:subclassOf>	<Parents>
<John>	<hasFather>	<Park>
<Parents>	<rdfs:subclassOf>	<Ancestor>
<hasFather>	<rdfs:domain>	<Father>
<John>	<rdf:type>	<Person>
<hasDes>	<rdf:type>	<owl:Trans>
<kim>	<hasDes>	<DG>

Subclass RDD **Type1_pair <s, o>**

Father	Parents
Parents	Ancestor

SPO RDD **Type2_pair <p, s+\t+o>**

hasFather	John+\t+Park

Domain RDD **Type1_pair <s, o>**

hasFather	Father

Type RDD **Type1_pair <o, s>**

Person	John

Transitive SPO RDD **Type5_pair <p+\t+s, p+\t+o>**

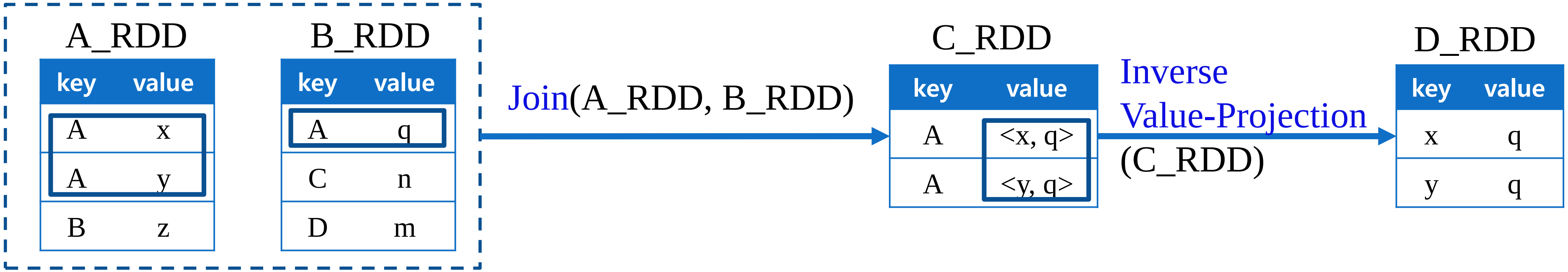
hasDes+\t+kim	hasDes+\t+DG

Operations for Ontology Reasoning

Operations are divided into rule-execution and RDD-reformatting categories

Join - When there are multiple values for the same key in one of the inputs, the resulting pair RDD will also have multiple entries for the same key

Inverse Value-Projection - This operation switches the position of the first element and the second element in tuples of all values, and make a new pair RDD using both elements



Operations for Ontology Reasoning

The rule-execution category

Operation	RDD format before execution	RDD format after execution
Join	A = {<a,b>, <b,c>, <c,d>} B = {<b,a>, <c,b>, <d,c>}	{<b,<c,a>>, <c,<d,b>>}
Swap	{<a,b>, <b,c>, <c,d>}	{<b,a>, <c,b>, <d,c>}

Operations for Ontology Reasoning

The RDD-reformatting category

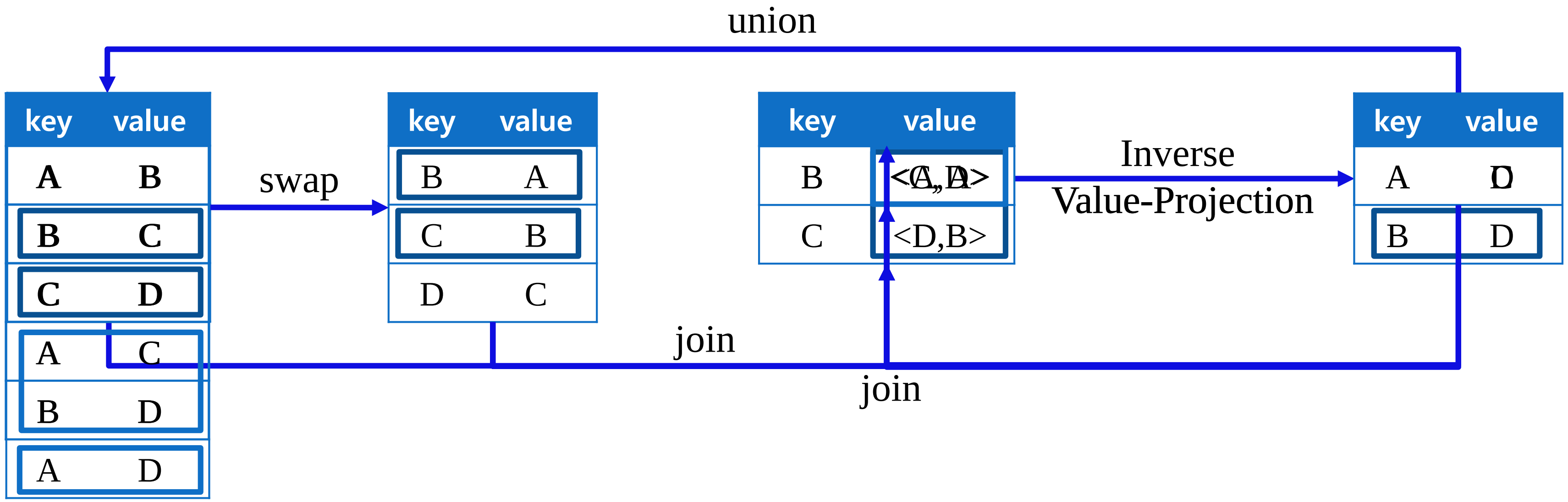
Operation	RDD format before execution	RDD format after execution
Value-Projection	{<b,<c,a>>, <c,<d,b>>}	{<c,a>, <d,b>}
First-element-Projection	{<rc,<ia,<c, ib>>>}	{<c, ib>}
Inverse Value-Projection	{<b,<c,a>>, <c,<d,b>>}	{<a,c>, <b,d>}
Equivalent Value-Projection	{<a,<b,b>>, <b,<a,a>>, <c,<x,y>>}	{<a, b>, <b,a>}
Key-value replacement	{<p+’\t’+s, p+’\t’+o>}	{<p, s+’\t’+o>}
Key-value transform	A_RDD = {<p, a+’\t’+b>} inverseMap = [[p,q]]	{<q, b+’\t’+a>}
Subject-key selection	{<p,<a+’\t’+b, c>>}	{<c, a>}
Object-key selection	{p,<a+’\t’+b, c>}	{<c,b>}
Composed value to single key	{<c, <p, i>>}	{<p,c>}
Composed value transform	{<c, <ib, <p, ia>>>}	{<p, ib+’\t’+ia>}
recomposed value to tuple	{<p, <c+’\t’+rc, ia+’\t’+ib>>}	{<rc, <c, ia>>}

Reasoning Cases

A subclass B
B subclass C
C subclass D

reasoning

A subclass C
A subclass D
B subclass D



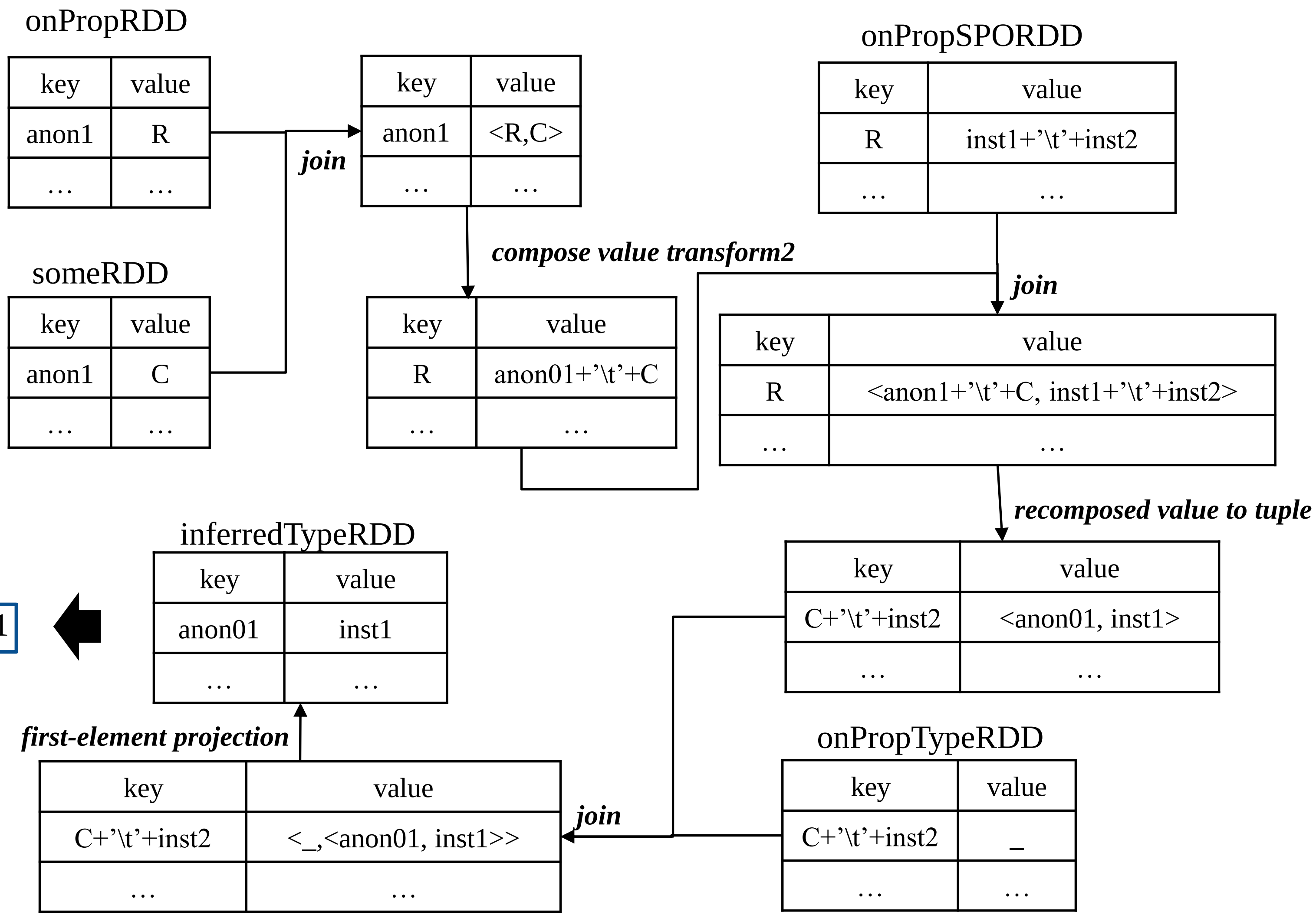
Reasoning Cases

A ≡ ∃R.C,   inst2 ∈ C,   R(inst1,inst2)

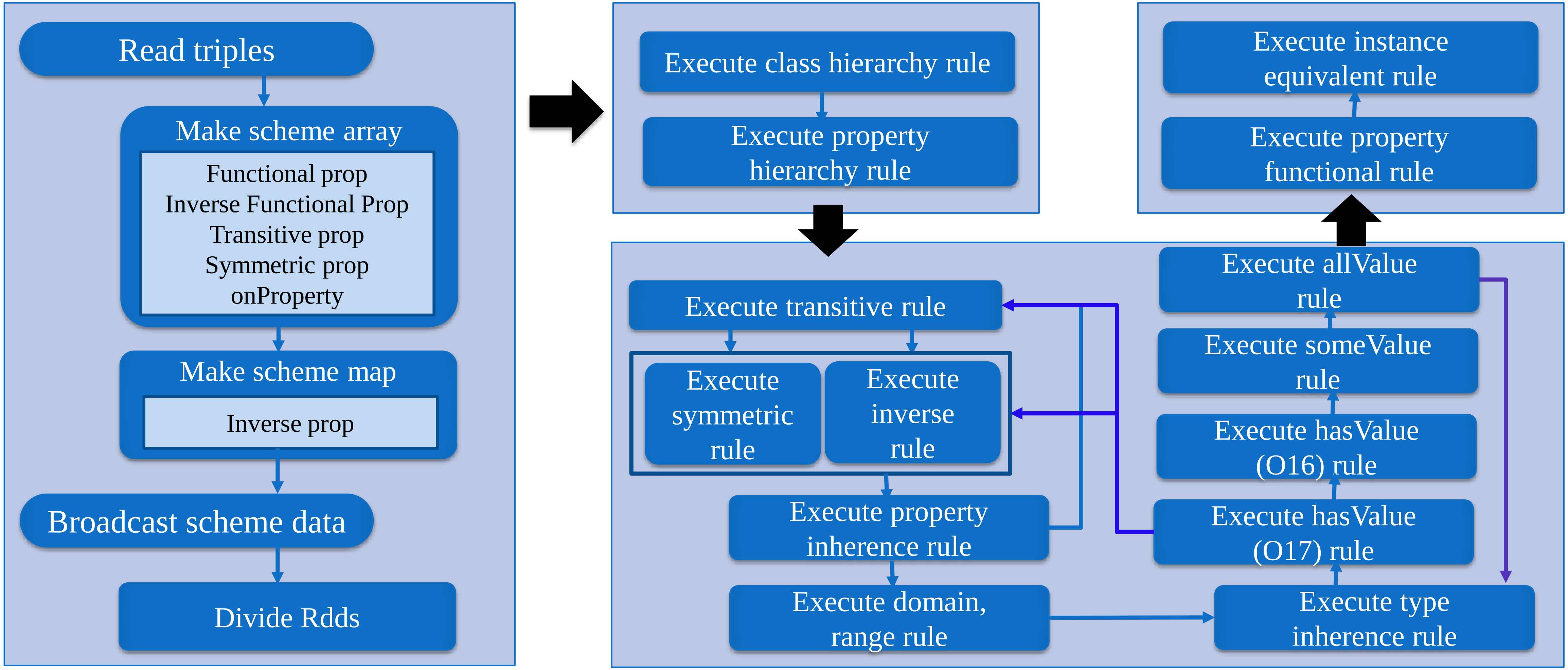
A	owl:equvalentClass	anon01
anon01	owl:someValuesFrom	C
anon01	owl:onProperty	R
inst1	R	inst2
inst2	rdfs:type	C

inst1	rdfs:type	anon01

inst1	rdfs:type	anon01



Reasoning step



Experimental Evaluation - RDFS

- Test data : LUBM
- Reasoning level of Ontology : RDFS
- Test goal : Reasoning Speed
- Test Environment : 1 master node(8 Cores with 12GByte Ram),
  7 worker nodes(8 Cores with 98GByte Ram)

Test Case				**Reasoning Results**			
				Hive	Impala	Scala-Spark	Java-Spark
Ontology	Number of Triples (Million)	Size (GB)	Number of Inferred Triples (Million)	Time(Min) **Reasoning time (No elimination time of duplication)**			Time (Min) **reasoning time + elimination time of Duplication**
LUBM1000	137	17	51	6.58	1.34	0.6	4.1
LUBM2000	275	36	102	7.71	1.97	1.1	10.5
LUBM3000	413	54	153	10.1	3.31	2.1	19.08

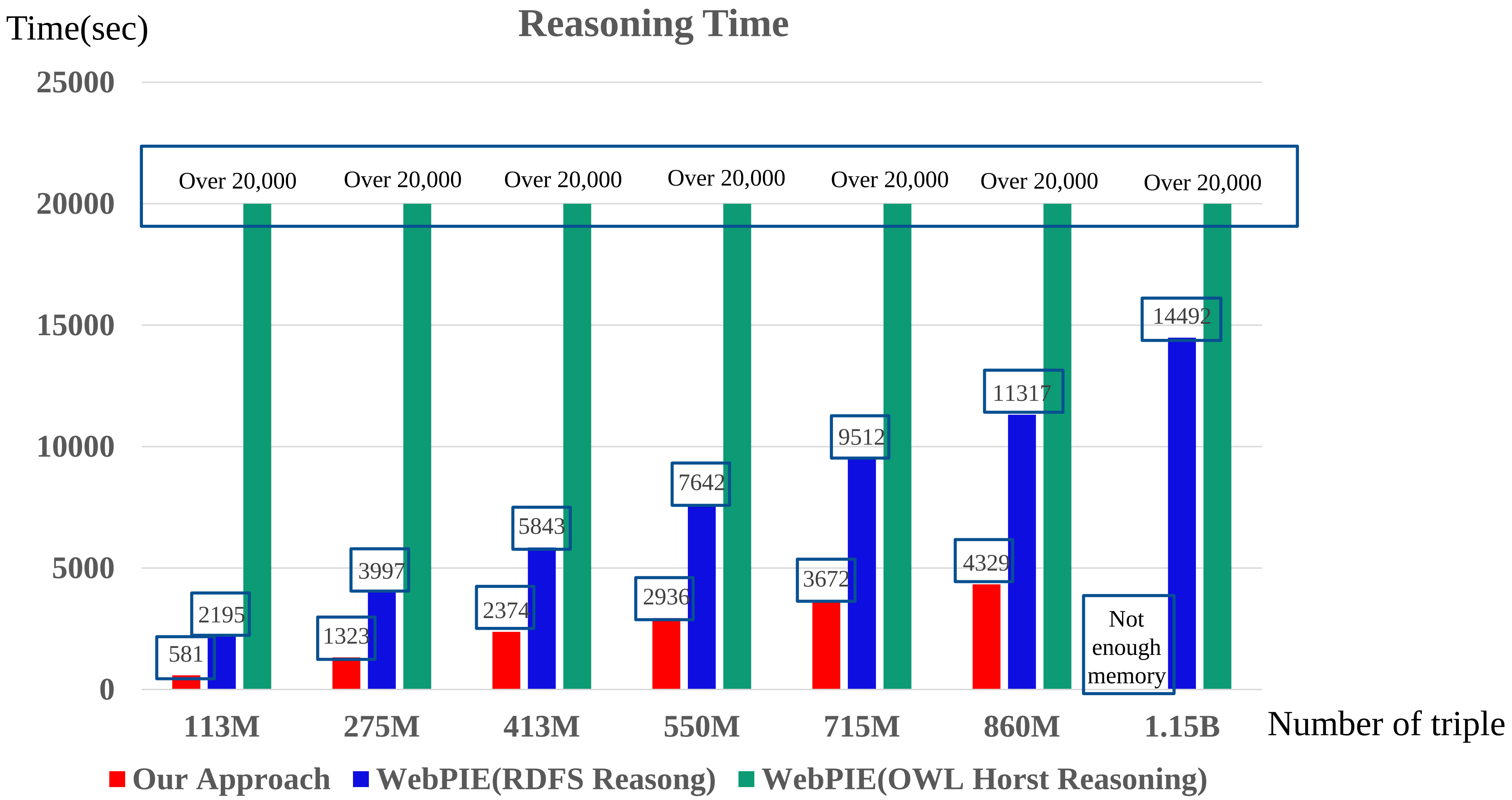
Experimental Evaluation – OWL-horst, $\mathcal{SHIF}$

The environment of experiment –
- A cluster with one master and four workers
- 2.6 GHz processor (8 cores) with 64 GB of main memory
- 2 TB of hard disk

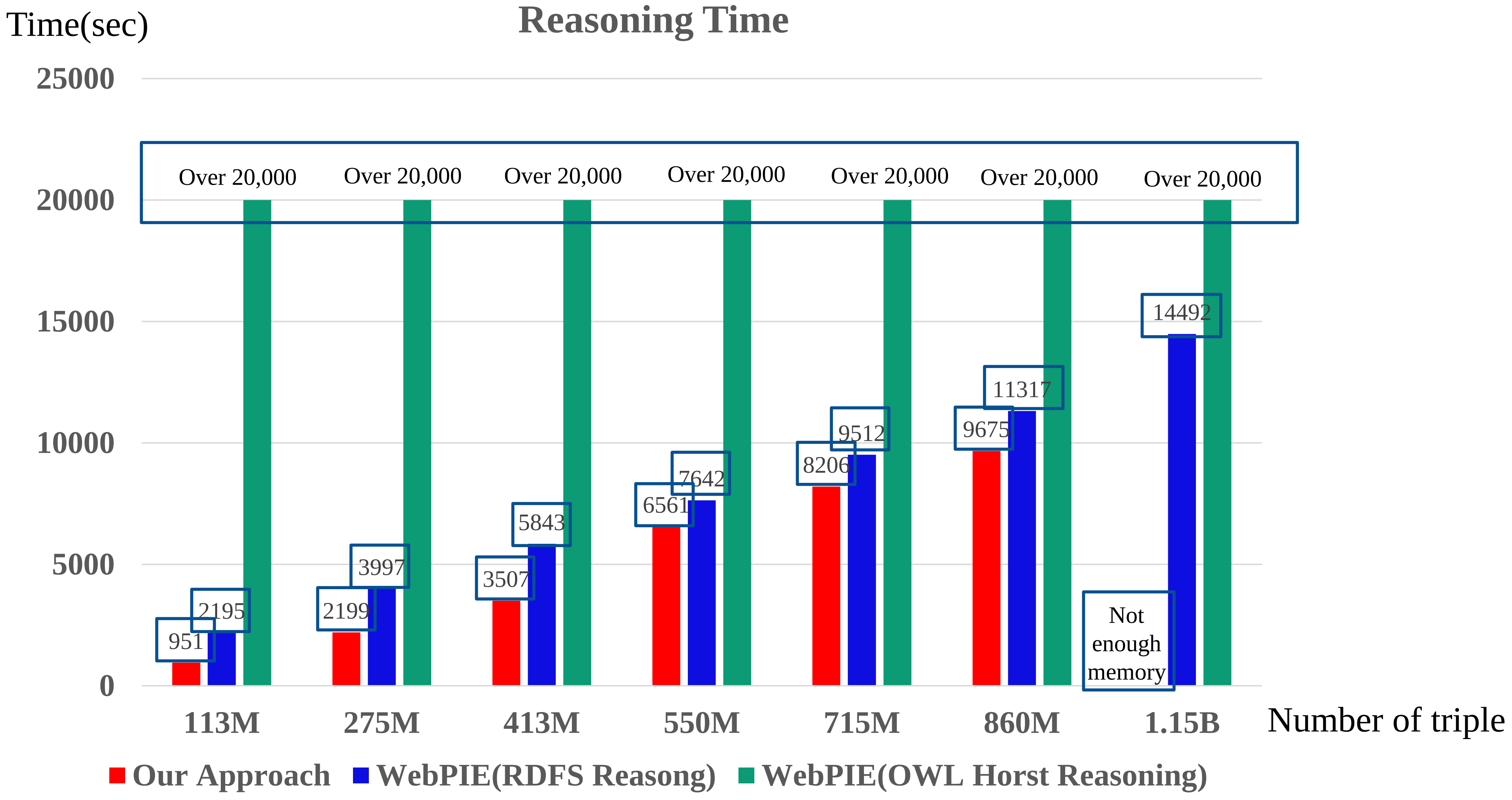
WebPIE - representative ontology reasoner based on MapReduce in Hadoop.

LUBM - formal benchmark dataset for evaluating ontology inferences and search.

Experimental Evaluation – OWL-horst



Experimental Evaluation - *SHIF*



Thank You!

kimjemins@hotmail.com