TriAD: A Distributed Shared-Nothing RDF Engine based on Asynchronous Message Passing

Sairam Gurajada[†], Stephan Seufert[†],
Iris Miliaraki[†], Martin Theobald[‡]

[†]Databases & Information Systems Group
Max-Planck Institute for Informatics
Saarbrücken, Germany

[‡] ADReM Research Group
University of Antwerp
Antwerp, Belgium

max planck institut
informatik

Universiteit
Antwerpen

Resource Description Framework (RDF)

RDF is a data model for representing information on the Web

Resource Description Framework (RDF)

RDF is a data model for representing information on the Web


Barack Obama

From Wikipedia, the free encyclopedia

"Obama" redirects here. For other uses, see Obama (disambiguation).

This article is about the 44th president of the United States. For his father, see Barack Obama, Sr.

Barack Hussein Obama II (/bəˈrɑːk huːˈseɪn oʊˈbɑːmə/; born August 4, 1961) is the 44th and current President of the United States, and the first African American to hold the office. Born in Honolulu, Hawaii, Obama is a graduate of Columbia University and Harvard Law School, where he served as president of the *Harvard Law Review*. He was a community organizer in Chicago before earning his law degree. He worked as a civil rights attorney and taught constitutional law at the University of Chicago Law School from 1992 to 2004. He served three terms representing the 13th District in the Illinois Senate from 1997 to 2004, running unsuccessfully for the United States House of Representatives in 2000.

In 2004, Obama received national attention during his campaign to represent Illinois in the United States Senate with his victory in the March Democratic Party primary, his keynote address at the Democratic National Convention in July, and his election to the Senate in November. He began his presidential campaign in 2007 and, after a close primary campaign against Hillary Rodham Clinton in 2008, he won sufficient delegates in the Democratic Party primaries to receive the presidential nomination. He then defeated Republican nominee John McCain in the general election, and was inaugurated as president on January 20, 2009. Nine months after his election, Obama was named the 2009 Nobel Peace Prize laureate.

During his first two years in office, Obama signed into law economic stimulus legislation in response to the Great Recession in the form of the American Recovery and Reinvestment Act of 2009 and the Tax Relief, Unemployment Insurance Reauthorization, and Job Creation Act of 2010. Other major domestic initiatives in his first term included the Patient Protection and Affordable Care Act, often referred to as "Obamacare"; the Dodd-Frank Wall Street

Barack Obama

Obama in December 2012

44th President of the United States

Incumbent

Assumed office
January 20, 2009

Vice President Joe Biden

Preceded by George W. Bush

United States Senator from Illinois

In office
January 3, 2005 – November 16, 2008

Preceded by Peter Fitzgerald

Succeeded by Roland Burris

Member of the Illinois Senate from the 13th District

In office
January 8, 1997 – November 4, 2004

Preceded by Alice Palmer

Succeeded by Kwame Raoul

Personal details

Born Barack Hussein Obama II
August 4, 1961 (age 52)
Honolulu, Hawaii, U.S.

Political party Democratic

Spouse(s) Michelle LaVaughn Robinson (m.

RDF triples obtained from IE

```
Barack_Obama isA President_of_USA.
Barack_Obama bornIn Honolulu .
Barack_Obama won Nobel_Peace_Prize .
Barack_Obama memberOf Democratic_Party .
```


Honolulu (/ˌhoʊnoʊˈluːluː/;[6][7] Hawaiian: *hono'lulu*) is the state capital and the most populous city in the U.S. state of Hawaii.[a][›] It is the county seat of the City and County of Honolulu. Situated on the island of Oahu, it is known worldwide as a major tourist destination; Honolulu is the main gateway to Hawaii and a major gateway into the United States of America. It is also a major hub for international business, military defense, as well as famously being host to a diverse variety of east-west and Pacific culture, cuisine, and traditions.

Honolulu is both the southernmost and westernmost major United States city. For statistical purposes, the U.S. Census Bureau recognizes the approximate area commonly referred to as "City of Honolulu" (not to be confused with the "City and County") as a census county division (CCD).[8] Honolulu is a major financial center of the islands and of the Pacific Ocean. The population of Honolulu CCD was 390,738 at the 2010 census,[9] while the population of the consolidated city and county was 953,207.

In the Hawaiian language, *Honolulu* means "sheltered bay" or

Honolulu
Hawaiian: ***hono'lulu***

State Capital

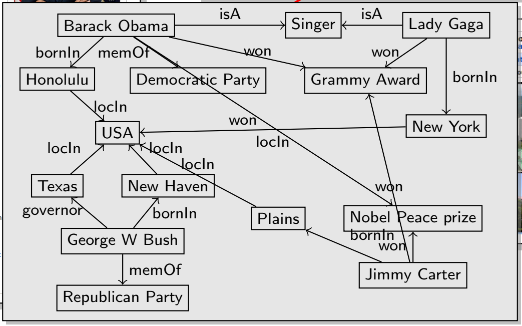
City and County of Honolulu

Resource Description Framework (RDF)

RDF is a data model for representing information on the Web

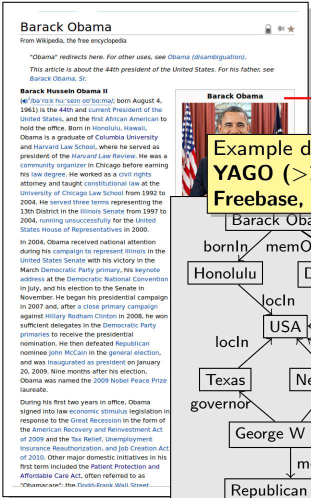
RDF triples obtained from IE

```
Barack_Obama isA President_of_USA.
Barack_Obama bornIn Honolulu .
Barack_Obama won Nobel_Peace_Prize .
Barack_Obama memberOf Democratic_Party .
```



Resource Description Framework (RDF)

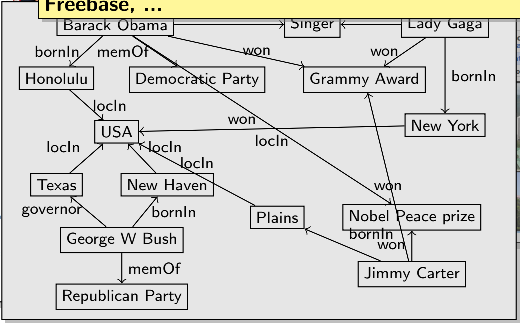
RDF is a data model for representing information on the Web

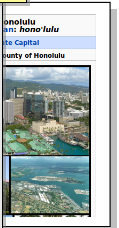


RDF triples obtained from IE

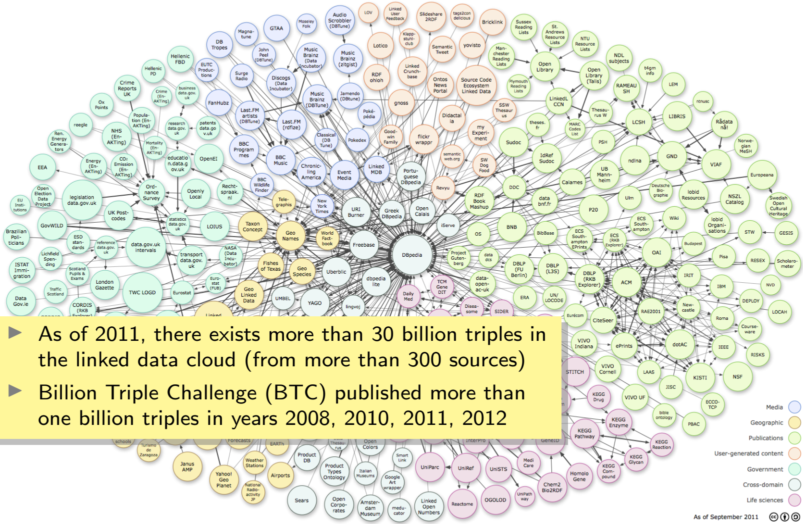
```
Barack_Obama isA President_of_USA.
Barack_Obama bornIn Honolulu .
Barack_Obama won Nobel_Peace_Prize .
ic_Party .
```

Example datasets:
YAGO (>120 M facts), DBpedia (>400 M facts), Freebase, ...





Scale of RDF Data Published – Linked Open Data Cloud



- As of 2011, there exists more than 30 billion triples in the linked data cloud (from more than 300 sources)
- Billion Triple Challenge (BTC) published more than one billion triples in years 2008, 2010, 2011, 2012

Source: http://linkeddata.org

Indexing and Querying RDF Data

- RDF triples are stored and indexed in a relational table (relational approach)
- SPARQL is the language suggested by W3C for querying RDF data
- SPARQL has many similarities with standard SQL
- `SELECT-PROJECT-JOIN` forms the main building blocks of SPARQL

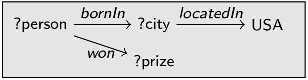
Indexing and Querying RDF Data

- RDF triples are stored and indexed in a relational table (relational approach)
- SPARQL is the language suggested by W3C for querying RDF data
- SPARQL has many similarities with standard SQL
- `SELECT-PROJECT-JOIN` forms the main building blocks of SPARQL

SPARQL Query:

Find persons who are born in USA and won a prize..

```
SELECT ?person, ?city, ?prize WHERE {
(R1) ?person <bornIn> ?city .
(R2) ?city <locatedIn> USA .
(R3) ?person <won> ?prize .  }
```



RDF Data:

Subject	Predicate	Object
Barack Obama	bornIn	Honolulu
Barack Obama	won	Nobel Peace Prize
Barack Obama	won	Grammy Award
Barack Obama	memberOf	Republican Party
Honolulu	locatedIn	United States
Barack Obama	isA	Singer
John F. Kennedy	bornIn	Brookline
John F. Kennedy	memberOf	Republican Party
John F. Kennedy	diedIn	Dallas
Dallas	locatedIn	United States
Brookline	locatedIn	United States
...	...	...

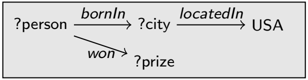
Indexing and Querying RDF Data

- RDF triples are stored and indexed in a relational table (relational approach)
- SPARQL is the language suggested by W3C for querying RDF data
- SPARQL has many similarities with standard SQL
- `SELECT-PROJECT-JOIN` forms the main building blocks of SPARQL

SPARQL Query:

Find persons who are born in USA and won a prize..

```
SELECT ?person, ?city, ?prize WHERE {
(R1) ?person <bornIn> ?city .
(R2) ?city <locatedIn> USA .
(R3) ?person <won> ?prize .  }
```



Results: (R1 $\bowtie$ R2 $\bowtie$ R3)

?person	?city	?prize
Barack_Obama	Honolulu	Peace_Nobel_Prize
Barack_Obama	Honolulu	Grammy_Award
...		

RDF Data:

Subject	Predicate	Object
Barack Obama	bornIn	Honolulu
Barack Obama	won	Nobel Peace Prize
Barack Obama	won	Grammy Award
Barack Obama	memberOf	Republican Party
Honolulu	locatedIn	United States
Barack Obama	isA	Singer
John F. Kennedy	bornIn	Brookline
John F. Kennedy	memberOf	Republican Party
John F. Kennedy	diedIn	Dallas
Dallas	locatedIn	United States
Brookline	locatedIn	United States
...	...	...

Current Approaches

Efficiency – thoroughly investigated in single-node setting

- crucial factors – *Join-order optimization*, *Join-ahead pruning*, *indexing layout*, *choice of operators*
- Eg. Jena, Sesame, HexaStore, MonetDB-RDF, SW-Store, RDF-3X, TripleBit, BitMat, gStore, ...

Current Approaches

Efficiency – thoroughly investigated in single-node setting

- crucial factors – *Join-order optimization*, *Join-ahead pruning*, *indexing layout*, *choice of operators*
- Eg. Jena, Sesame, HexaStore, MonetDB-RDF, SW-Store, RDF-3X, TripleBit, BitMat, gStore, ...

Scalability – recently a line of distributed systems has been proposed

- $\underbrace{\text{SHARD, H-RDF-3X}}_{\text{Relational-based}}$, $\underbrace{\text{Trinity.RDF}}_{\text{Graph-based}}$, ...

Current Approaches

Efficiency – thoroughly investigated in single-node setting

- crucial factors – *Join-order optimization*, *Join-ahead pruning*, *indexing layout*, *choice of operators*
- Eg. Jena, Sesame, HexaStore, MonetDB-RDF, SW-Store, RDF-3X, TripleBit, BitMat, gStore, ...

Scalability – recently a line of distributed systems has been proposed

- $\underbrace{\text{SHARD, H-RDF-3X}}_{\text{Relational-based}}$, $\underbrace{\text{Trinity.RDF}}_{\text{Graph-based}}$, ...

Relational-based (Joins) vs Graph-based (Exploration) [distributed setting]

- SPARQL 1.0 requires a row-oriented output $\longrightarrow$ joins are inevitable
- Relational approaches suffer from "large intermediate relations" (inaccurate/insufficient statistics resulting in poor query plans)

Current Approaches

Efficiency – thoroughly investigated in single-node setting

- crucial factors – *Join-order optimization*, *Join-ahead pruning*, *indexing layout*, *choice of operators*
- Eg. Jena, Sesame, HexaStore, MonetDB-RDF, SW-Store, RDF-3X, TripleBit, BitMat, gStore, ...

Scalability – recently a line of distributed systems has been proposed

- $\underbrace{\text{SHARD, H-RDF-3X}}_{\text{Relational-based}}$, $\underbrace{\text{Trinity.RDF}}_{\text{Graph-based}}$, ...

Relational-based (Joins) vs Graph-based (Exploration) [distributed setting]

- SPARQL 1.0 requires a row-oriented output $\longrightarrow$ joins are inevitable
- Relational approaches suffer from "large intermediate relations" (inaccurate/insufficient statistics resulting in poor query plans)
- Graph-based systems use **distributed graph exploration** followed by relational joins
- Effective when graph exploration prunes a lot of bindings (in case of selective queries)

Current Approaches

Efficiency – thoroughly investigated in single-node setting

- crucial factors – *Join-order optimization*, *Join-ahead pruning*, *indexing layout*, *choice of operators*
- Eg. Jena, Sesame, HexaStore, MonetDB-RDF, SW-Store, RDF-3X, TripleBit, BitMat, gStore, ...

Scalability – recently a line of distributed systems has been proposed

- $\underbrace{\text{SHARD, H-RDF-3X}}_{\text{Relational-based}}$, $\underbrace{\text{Trinity.RDF}}_{\text{Graph-based}}$, ...

Relational-based (Joins) vs Graph-based (Exploration) [distributed setting]

- SPARQL 1.0 requires a row-oriented output $\longrightarrow$ joins are inevitable
- Relational approaches suffer from "large intermediate relations" (inaccurate/insufficient statistics resulting in poor query plans)
- Graph-based systems use **distributed graph exploration** followed by relational joins
- Effective when graph exploration prunes a lot of bindings (in case of selective queries)

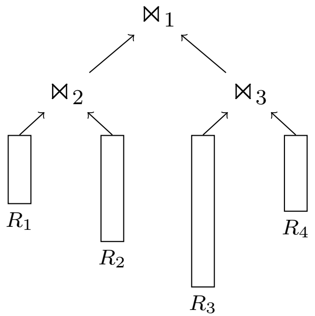
Problems with existing distributed relational-based RDF engines

- **1. Synchronous processing of joins**
- **2. Dangling triples** – occur in intermediate relations but not in final results

1. Synchronous Processing of Joins

Consider a query with four relations R_1, R_2, R_3, R_4 with join order:

$$(R_1 \bowtie_2 R_2) \bowtie_1 (R_3 \bowtie_3 R_4)$$

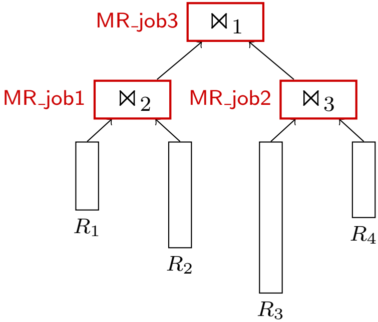


1. Synchronous Processing of Joins

Consider a query with four relations R_1, R_2, R_3, R_4 with join order:

$$(R_1 \bowtie_2 R_2) \bowtie_1 (R_3 \bowtie_3 R_4)$$

Synchronous case 1 :
MR_job1 $\rightarrow$ MR_job2 $\rightarrow$ MR_job3

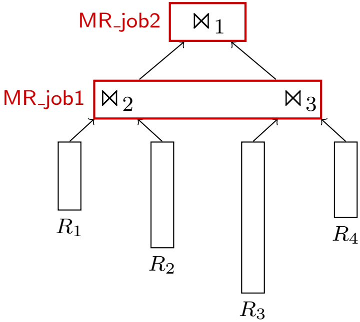


1. Synchronous Processing of Joins

Consider a query with four relations R_1, R_2, R_3, R_4 with join order:

$$(R_1 \bowtie_2 R_2) \bowtie_1 (R_3 \bowtie_3 R_4)$$

Synchronous case 1 :
MR_job1 $\rightarrow$ MR_job2 $\rightarrow$ MR_job3
Synchronous case 2 :
MR_job1 $\rightarrow$ MR_job2

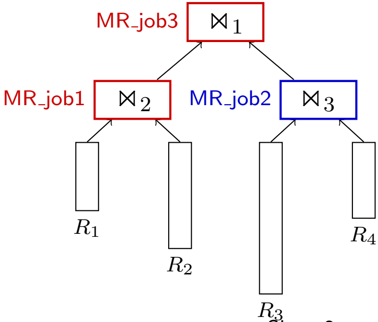


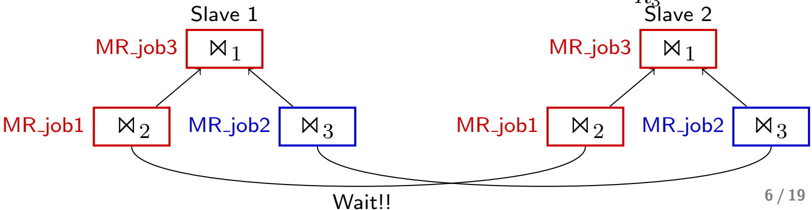
1. Synchronous Processing of Joins

Consider a query with four relations R_1, R_2, R_3, R_4 with join order:

$$(R_1 \bowtie_2 R_2) \bowtie_1 (R_3 \bowtie_3 R_4)$$

Synchronous case 1 :
MR_job1 $\rightarrow$ MR_job2 $\rightarrow$ MR_job3
Synchronous case 2 :
MR_job1 $\rightarrow$ MR_job2
Synchronous case 3 :
(MR_job1 || MR_job2) $\rightarrow$ MR_job3



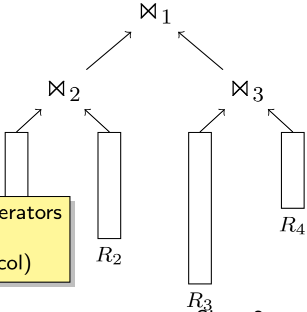


1. Synchronous Processing of Joins

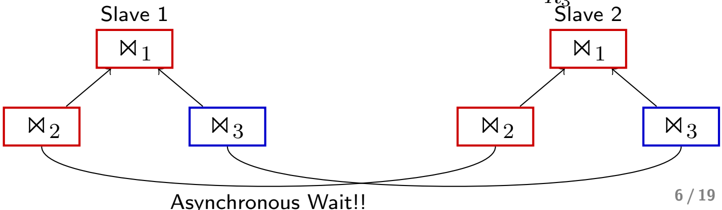
Consider a query with four relations R_1, R_2, R_3, R_4 with join order:

$$(R_1 \bowtie_2 R_2) \bowtie_1 (R_3 \bowtie_3 R_4)$$

Synchronous case 1 :
MR_job1 $\rightarrow$ MR_job2 $\rightarrow$ MR_job3
Synchronous case 2 :
MR_job1 $\rightarrow$ MR_job2
Synchronous case 3 :
(MR_job1 || MR_job2) $\rightarrow$ MR_job3



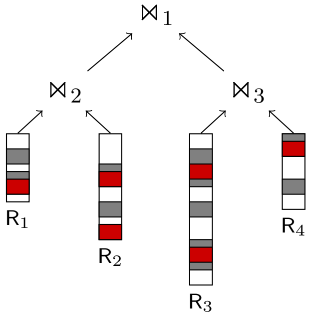
Our approach: Minimize dependancies among join operators and only synchronize when needed!
(using asynchronous communication (MPICH2) protocol)



2. Pruning Dangling Triples

Consider a query with four relations R_1, R_2, R_3, R_4 with join order:

$$(R_1 \bowtie_2 R_2) \bowtie_1 (R_3 \bowtie_3 R_4)$$



[1] RDF-3X: a RISC-style Engine for RDF, VLDBJ 2010

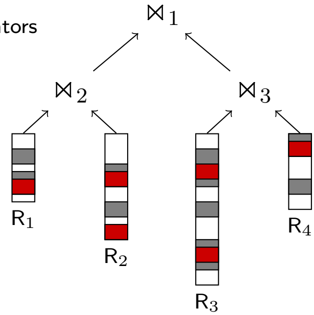
2. Pruning Dangling Triples

Consider a query with four relations R_1, R_2, R_3, R_4 with join order:

$$(R_1 \ltimes_2 R_2) \ltimes_1 (R_3 \ltimes_3 R_4)$$

Sideways Information Passing (SIP) of RDF-3X[1]

- Powerful runtime pruning technique
- Shares information across join operators
- Requires synchronization



[1] RDF-3X: a RISC-style Engine for RDF, VLDBJ 2010

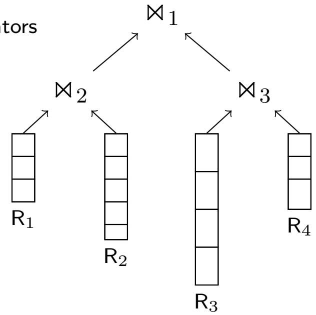
2. Pruning Dangling Triples

Consider a query with four relations R_1, R_2, R_3, R_4 with join order:

$$(R_1 \bowtie_2 R_2) \bowtie_1 (R_3 \bowtie_3 R_4)$$

Sideways Information Passing (SIP) of RDF-3X[1]

- Powerful runtime pruning technique
- Shares information across join operators
- Requires synchronization



Our approach: Join-ahead pruning

1. Pre-partition the triples (into groups)

[1] RDF-3X: a RISC-style Engine for RDF, VLDBJ 2010

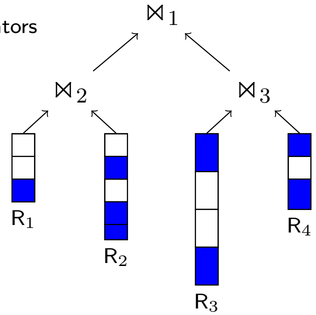
2. Pruning Dangling Triples

Consider a query with four relations R_1, R_2, R_3, R_4 with join order:

$$(R_1 \bowtie_2 R_2) \bowtie_1 (R_3 \bowtie_3 R_4)$$

Sideways Information Passing (SIP) of RDF-3X[1]

- Powerful runtime pruning technique
- Shares information across join operators
- Requires synchronization



Our approach: Join-ahead pruning

1. Pre-partition the triples (into groups)
2. Query over groups to find the ones which are relevant to the query

[1] RDF-3X: a RISC-style Engine for RDF, VLDBJ 2010

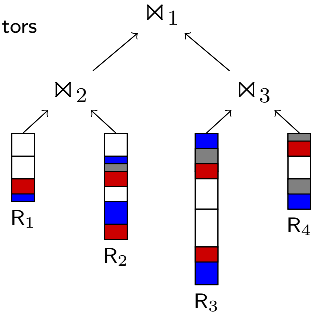
2. Pruning Dangling Triples

Consider a query with four relations R_1, R_2, R_3, R_4 with join order:

$$(R_1 \bowtie_2 R_2) \bowtie_1 (R_3 \bowtie_3 R_4)$$

Sideways Information Passing (SIP) of RDF-3X[1]

- Powerful runtime pruning technique
- Shares information across join operators
- Requires synchronization

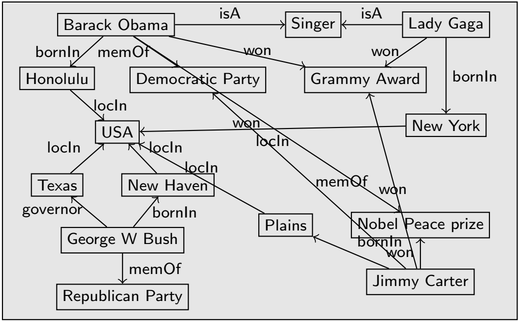


Our approach: Join-ahead pruning

1. Pre-partition the triples (into groups)
2. Query over groups to find the ones which are relevant to the query
3. Scan & Join only triples from the relevant groups

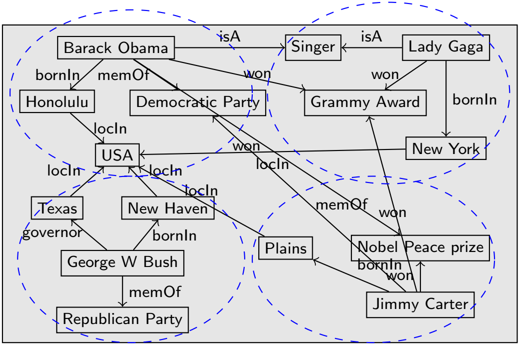
[1] RDF-3X: a RISC-style Engine for RDF, VLDBJ 2010

2. Join-ahead Pruning via Graph Summarization



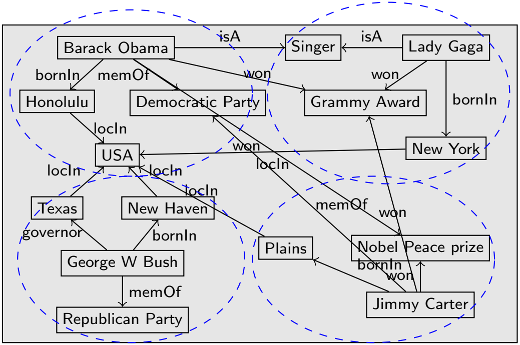
2. Join-ahead Pruning via Graph Summarization

Locality-based grouping (using METIS)

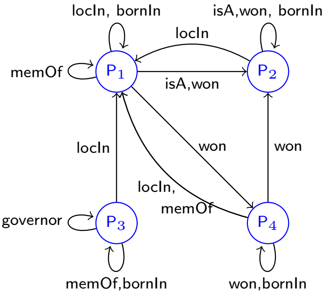


2. Join-ahead Pruning via Graph Summarization

Locality-based grouping (using METIS)

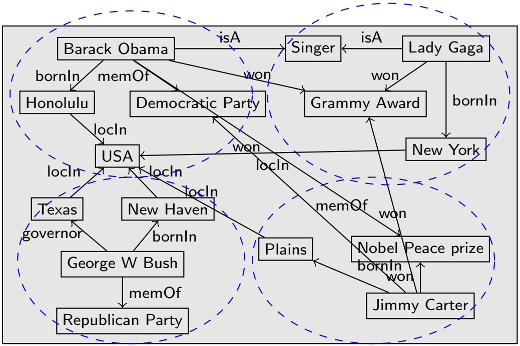


Summary Graph

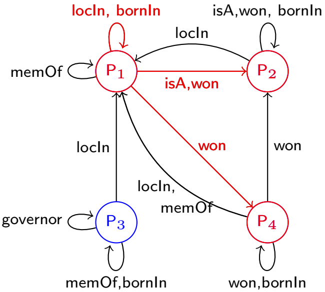


2. Join-ahead Pruning via Graph Summarization

Locality-based grouping (using METIS)



Summary Graph



SPARQL Query:

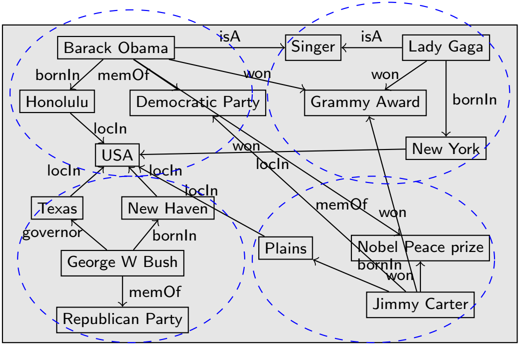
```
SELECT ?city, ?prize WHERE {
Barack Obama <bornIn> ?city .
?city <locatedIn> USA .
Barack Obama <won> ?prize .  }
```

Supernode Bindings:

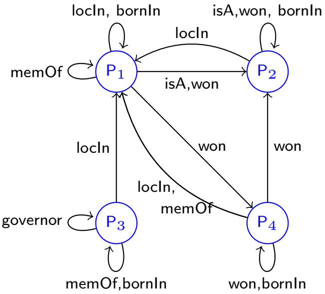
?city : P_1
?prize : P_2, P_4

2. Join-ahead Pruning via Graph Summarization

Locality-based grouping (using METIS)



Summary Graph



SPARQL Query:

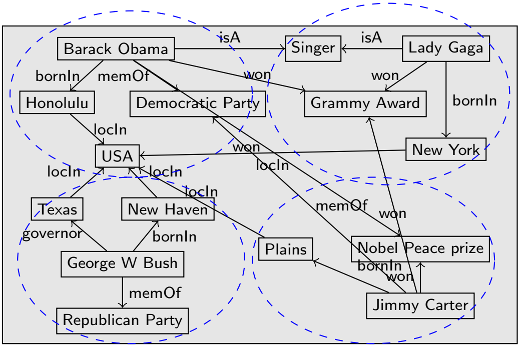
```
SELECT ?city, ?prize WHERE {
Barack Obama <bornIn> ?city .
?city <locatedIn> USA .
Barack Obama <governor> ?city2 .  }
```

Supernode Bindings: !!! Empty Result

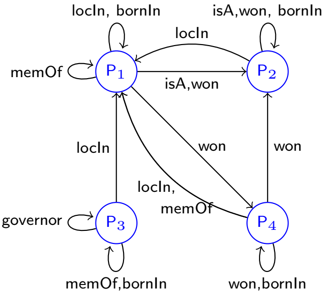
```
?city    :  --
?prize   :  --
```

2. Join-ahead Pruning via Graph Summarization

Locality-based grouping (using METIS)



Summary Graph



SPARQL Query:

```
SELECT ?city, ?prize WHERE {
Barack Obama <bornIn> ?city .
?city <locatedIn> USA .
Barack Obama <governor> ?city2 .   }
```

Supernode Bindings: !!! Empty Result

```
?city    :  --
?prize   :  --
```

False postives may occur but no false negatives!!!

TriAD Distributed RDF Store

TriAD (for "Triple Asynchronous and Distributed") – a distributed RDF Store

TriAD Distributed RDF Store

TriAD (for "Triple Asynchronous and Distributed") – a distributed RDF Store

Main-memory backed six permutation distributed indexes
(with locality-aware sharding and encoded summary information)

TriAD Distributed RDF Store

TriAD (for "Triple Asynchronous and Distributed") – a distributed RDF Store

Asynchronous Processing of Joins
via MPICH2 communication protocol

Main-memory backed six permutation distributed indexes
(with locality-aware sharding and encoded summary information)

TriAD Distributed RDF Store

TriAD (for "Triple Asynchronous and Distributed") – a distributed RDF Store

Join-ahead pruning via Graph Summarization	Asynchronous Processing of Joins via MPICH2 communication protocol

Main-memory backed six permutation distributed indexes
(with locality-aware sharding and encoded summary information)

TriAD Distributed RDF Store

TriAD (for "Triple Asynchronous and Distributed") – a distributed RDF Store

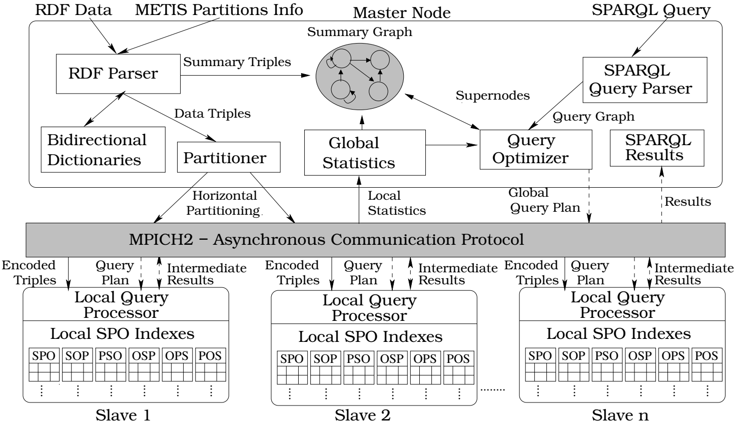
Two-stage Query Optimization
(distributed, multi-threading, and join-ahead pruning into account)

Join-ahead pruning
via Graph Summarization

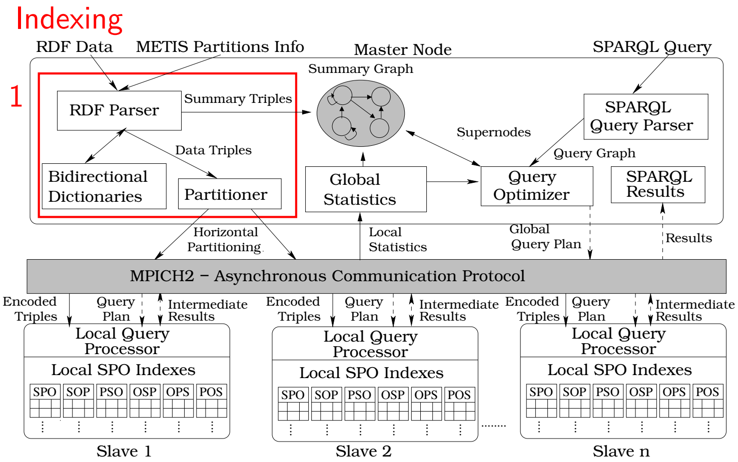
Asynchronous Processing of Joins
via MPICH2 communication protocol

Main-memory backed six permutation distributed indexes
(with locality-aware sharding and encoded summary information)

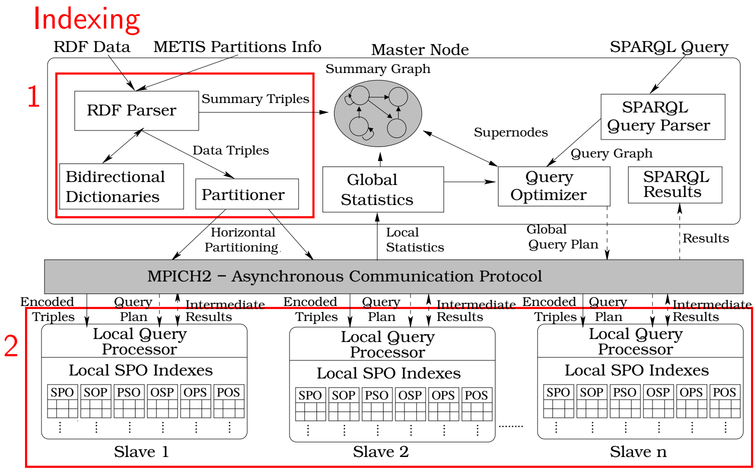
TriAD Architecture



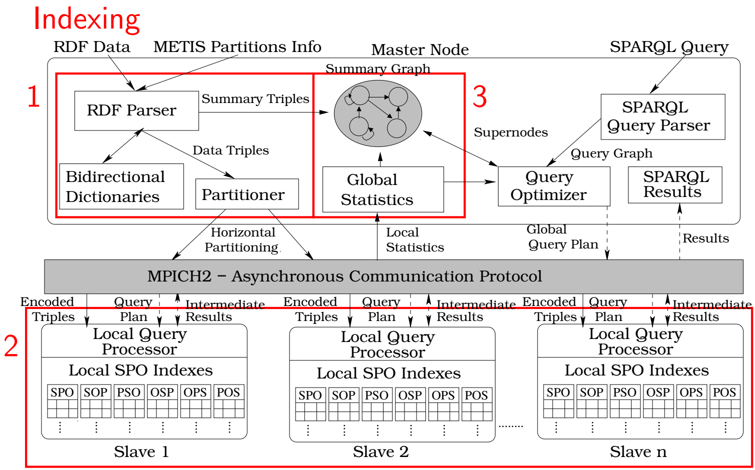
TriAD Architecture



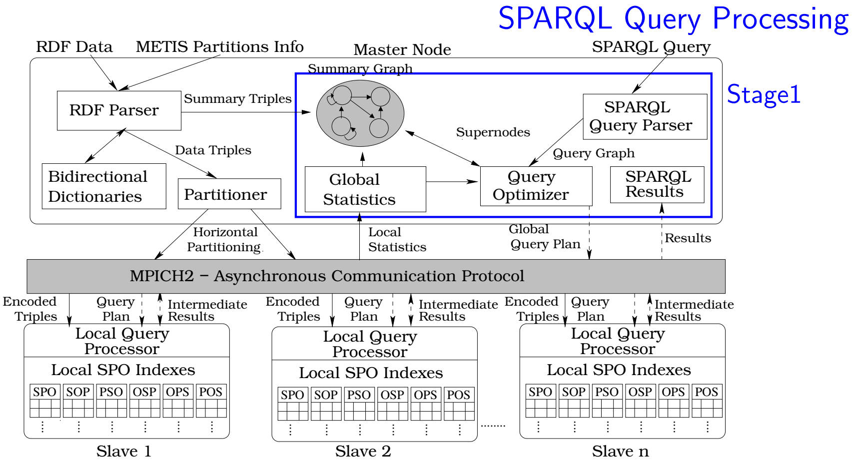
TriAD Architecture



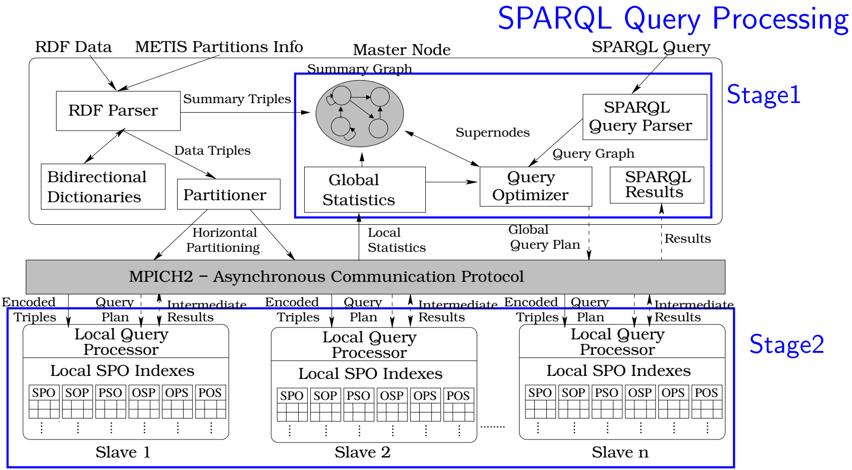
TriAD Architecture



TriAD Architecture



TriAD Architecture



Indexing

RDF graph partitioning

- using a locality-based non-overlapping partitioning algorithm
- i.e Each s (or o) of RDF triple $\langle s, p, o \rangle$ mapped to one supernode P_s (or P_o)

Indexing

RDF graph partitioning

- using a locality-based non-overlapping partitioning algorithm
- i.e Each s (or o) of RDF triple $\langle s, p, o \rangle$ mapped to one supernode P_s (or P_o)

Triple encoding

- Each triple $\langle s, p, o \rangle$ is encoded into two triples: data triple, summary triple
- **Dictionary encoding:** Each entity is assigned a globally unique id is computed by concatenating (supernode id) P_s and a (local id) id_s
  Eg. $\langle$`Barack_Obama`, `won`, `Nobel_Prize`$\rangle$
  Data triple: $\langle 1||1, 6, 4||3 \rangle$ Summary triple $\langle 1, 6, 4 \rangle$

Indexing

RDF graph partitioning

- using a locality-based non-overlapping partitioning algorithm
- i.e Each s (or o) of RDF triple $\langle s, p, o \rangle$ mapped to one supernode P_s (or P_o)

Triple encoding

- Each triple $\langle s, p, o \rangle$ is encoded into two triples: data triple, summary triple
- **Dictionary encoding:** Each entity is assigned a globally unique id is computed by concatenating (supernode id) P_s and a (local id) id_s
  Eg. $\langle$`Barack_Obama`, `won`, `Nobel_Prize`$\rangle$
  Data triple: $\langle 1||1, 6, 4||3 \rangle$ Summary triple $\langle 1, 6, 4 \rangle$

Locality-aware sharding and distributed indexing of data triples

- Each data triple is hash partitioned onto atmost two slaves and indexed in six permutations (in total)
  Eg. triple $\langle 1||1, 6, 4||3 \rangle$ is hashed on to slaves 1 **mod** n and 4 **mod** n (for n-slaves)

Indexing

RDF graph partitioning

- using a locality-based non-overlapping partitioning algorithm
- i.e Each s (or o) of RDF triple $\langle s, p, o\rangle$ mapped to one supernode P_s (or P_o)

Triple encoding

- Each triple $\langle s, p, o\rangle$ is encoded into two triples: data triple, summary triple
- **Dictionary encoding:** Each entity is assigned a globally unique id is computed by concatenating (supernode id) P_s and a (local id) id_s
  Eg. $\langle$`Barack_Obama`, `won`, `Nobel_Prize`$\rangle$
  Data triple: $\langle 1||1, 6, 4||3\rangle$ Summary triple $\langle 1, 6, 4\rangle$

Locality-aware sharding and distributed indexing of data triples

- Each data triple is hash partitioned onto atmost two slaves and indexed in six permutations (in total)
  Eg. triple $\langle 1||1, 6, 4||3\rangle$ is hashed on to slaves 1 **mod** n and 4 **mod** n (for n-slaves)

Summary graph index

- Summary triples are indexed in two-permutation adjacency list for efficient graph exploration

Query Processing

In TriAD, query processing is performed in two stages

Stage 1: Summary graph query processing (join-ahead pruning)

- Performed using **graph exploration** at master node to generate supernode bindings

Query Processing

In TriAD, query processing is performed in two stages

Stage 1: Summary graph query processing (join-ahead pruning)

- Performed using **graph exploration** at master node to generate supernode bindings

Stage 2: Data graph query processing

- Done using **relational joins** in a distributed setup
- Inspired by the RISC style processing, we employ three physical operators:
  Distributed Index Scan (DIS): Invokes a parallel scan over a permutation list that is sharded across all slaves
  Distributed Merge Join (DMJ): If both input (or intermediate) relations are sorted according to the join key(s) in the query plan
  Distributed Hash Join (DHJ): If the input (or intermediate) relations are not sorted according to their join key(s)

Global Statistics & Query Optimization

Precomputed statistics

- Computed in parallel at slaves and sent to master node
- Statistics:
  Individual cardinalities: s, p, o of SPO triples
  Pair cardinalities: (s, o), (s, p), (p, o)
  Join selectivities of predicate pairs (p_1, p_2)
- Similar statistics are computed over summary graph

For a given query with patterns $Q\ \{R_1, R_2, ..R_n\}$

Stage 1: Exploration Optimization

We compute the exploration plan by using a bottom-up dynamic programming and the following cost model:

$$Cost(R_1, \ldots, R_n) \propto Card(R_1) + \sum_{i=2}^{n} \Big(Card(R_i) \prod_{j=1}^{i} Sel(R_i, R_j) \Big) \qquad (1)$$

Global Statistics & Query Optimization

Precomputed statistics

- Computed in parallel at slaves and sent to master node
- Statistics:
  Individual cardinalities: s, p, o of SPO triples
  Pair cardinalities: (s, o), (s, p), (p, o)
  Join selectivities of predicate pairs (p_1, p_2)
- Similar statistics are computed over summary graph

For a given query with patterns $Q\ \{R_1, R_2, ..R_n\}$

Stage 1: Exploration Optimization

We compute the exploration plan by using a bottom-up dynamic programming and the following cost model:

$$Cost(R_1, \ldots, R_n) \propto Card(R_1) + \sum_{i=2}^{n} \Big(Card(R_i) \prod_{j=1}^{i} Sel(R_i, R_j) \Big) \qquad (1)$$

Re-estimating cardinalities of a relations R_i

New cardinalities are re-estimated from the Stage 1 supernode bindings

- $Card(R_i) := \frac{|C_s^Q|}{|C_s|} \cdot \frac{|C_o^Q|}{|C_o|} \cdot Card(R_i)$ (on the fly computation)
  where $|C_s|, |C_o|$ are the supernode bindings of R_i
  and $|C_s^Q|, |C_o^Q|$ are the supernode bindings of Q ($R_i \in Q$)

Stage 2: Global plan optimization

A global plan is computed for stage 2 using **re-estimated cardinalities** and the following cost model

$$Cost(Q) := \begin{cases} Cost(R_i^k) & \text{if } R_i \text{ denotes a DIS over permutation } k; \\ max(Cost(Q^{left}), Cost(Q^{right})) & \\ +Cost(Q^{left} \bowtie^{op} Q^{right}) & \\ +Cost(Q^{left} \rightleftharpoons^{op} Q^{right}) & \text{otherwise.} \end{cases}$$

The cost model captures the multi-threaded and distributed execution framework

Re-estimating cardinalities of a relations R_i

New cardinalities are re-estimated from the Stage 1 supernode bindings

- $Card(R_i) := \frac{|C_s^Q|}{|C_s|} \cdot \frac{|C_o^Q|}{|C_o|} \cdot Card(R_i)$ (on the fly computation)
  where $|C_s|, |C_o|$ are the supernode bindings of R_i
  and $|C_s^Q|, |C_o^Q|$ are the supernode bindings of Q ($R_i \in Q$)

Stage 2: Global plan optimization

A global plan is computed for stage 2 using **re-estimated cardinalities** and the following cost model

$$Cost(Q) := \begin{cases} Cost(R_i^k) & \text{if } R_i \text{ denotes a DIS over permutation } k; \\ max(Cost(Q^{left}), Cost(Q^{right})) & \\ +Cost(Q^{left} \bowtie^{op} Q^{right}) & \\ +Cost(Q^{left} \rightleftharpoons^{op} Q^{right}) & \text{otherwise.} \end{cases}$$

The cost model captures the multi-threaded and distributed execution framework

Querying Optimization & Processing - Example

SPARQL Query:

Find persons who are born in USA and won a prize..

SELECT ?person, ?city, ?prize WHERE {

R_1 ?person <bornIn> ?city .

R_2 ?city <locatedIn> USA .

R_3 ?person <won> ?prize .

R_4 ?prize <hasName> ?name . }

Querying Optimization & Processing - Example

SPARQL Query:

Find persons who are born in USA and won a prize..

SELECT ?person, ?city, ?prize WHERE {

R_1 ?person <bornIn> ?city .

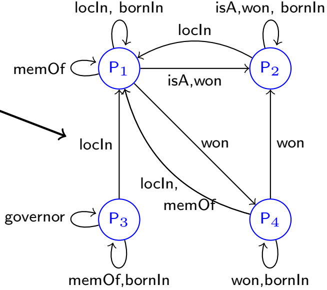
R_2 ?city <locatedIn> USA .

R_3 ?person <won> ?prize .

R_4 ?prize <hasName> ?name . }

Stage 1 (Optimization)

Summary Graph



Exploration

Supernode Bindings:

?person	:	P1, P2, P4
?city	:	P1, P2, P4
?prize	:	P2, P4

Querying Optimization & Processing - Example

SPARQL Query:

Find persons who are born in USA and won a prize..

SELECT ?person, ?city, ?prize WHERE {

R_1 ?person <bornIn> ?city .

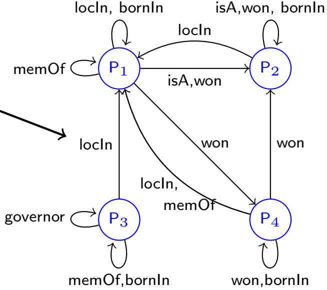
R_2 ?city <locatedIn> USA .

R_3 ?person <won> ?prize .

R_4 ?prize <hasName> ?name . }

Stage 1 (Optimization)

Summary Graph



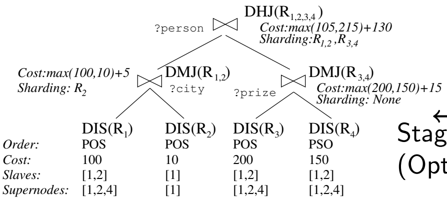
Exploration

Supernode Bindings:

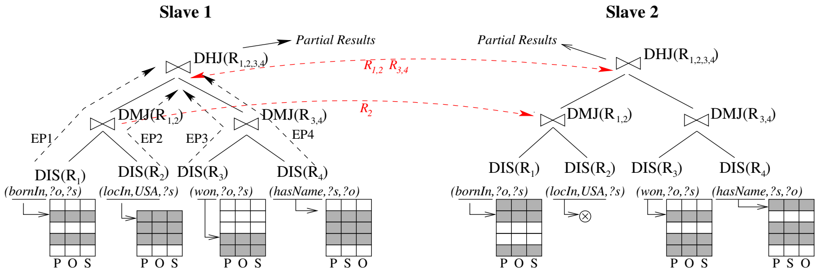
?person	:	P1, P2, P4
?city	:	P1, P2, P4
?prize	:	P2, P4

Stage 2 (Optimization)

Global Plan



Query Optimization & Processing – Example (2)



Stage 2: Distributed query execution

Evaluation

We compared performance of TriAD with the following state-of-the-art systems

- Centralized systems – RDF-3X, MonetDB-RDF, BitMat
- Distributed systems – H-RDF-3X, Trinity.RDF, 4store, SHARD, Spark

Datasets:

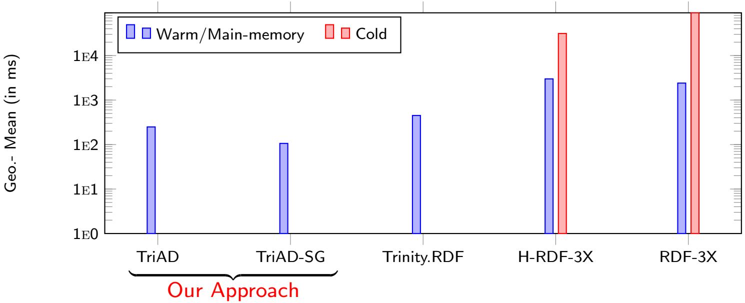
- (Synthetic) LUBM 160 – 28 Million triples, 16GB raw data
- (Synthetic) LUBM 10240 – 1.8 Billion triples, 730GB raw data
- (Real world) BTC 2012 – 1.4 Billion triples, 231 GB raw data
- (Synthetic) WSDTS – 109 Million triples, 15 GB raw data
- Benchmark queries for LUBM, BTC, & WSDTS datasets

System Setup:

- TriAD, TriAD-SG is implemented in C++
- Cluster setup: 12-nodes, 48GB RAM, 2 quad-core CPUs of 2.4GHz (HT enabled)

Evaluation - Large datasets

LUBM 10240 Dataset – 1.8 Billion triples (Query Performance in milli seconds)



Queries	Characteristics	TriAD	TriAD-SG	Trinity.RDF	H-RDF-3X		RDF-3X	
Q1	Selective (6 joins)	7,631	**2,146**	12,648	2.3E6	1.7E5	1.9E6	1.8E6
Q2	Non-Selective(1 join)	**1,663**	2,025	6,018	5.3E5	4,095	2.4E5	1.8E5
Q4	Selective (5 joins)	2.1	**1.3**	5	166	1	243	3
Q7	Selective (6 joins)	**14,895**	16,863	31,214	2.3E6	2.1E5	6.5E5	46,262

Summary

TriAD is a fast distributed RDF engine built on top of **asynchronous communication layer** and **multi-threaded execution** framework

- efficient **distributed and parallel** join executions
- **join-ahead pruning technique via graph summarization** helps in pruning dangling triples and making query processing efficient
- **distributed- and join-ahead pruning aware** query optimizer
- so far reported fastest runtimes over three benchmark datasets: LUBM, BTC, WSDTS

Summary

TriAD is a fast distributed RDF engine built on top of **asynchronous communication layer** and **multi-threaded execution** framework

- efficient **distributed and parallel** join executions
- **join-ahead pruning technique via graph summarization** helps in pruning dangling triples and making query processing efficient
- **distributed- and join-ahead pruning aware** query optimizer
- so far reported fastest runtimes over three benchmark datasets: LUBM, BTC, WSDTS

Questions & Thank You!!