Fast and Concurrent RDF Queries with *RDMA-based* Distributed Graph Exploration

JIAXIN SHI, YOUYANG YAO
RONG CHEN, HAIBO CHEN

Institute of Parallel and Distributed Systems (IPADS)
Shanghai Jiao Tong University

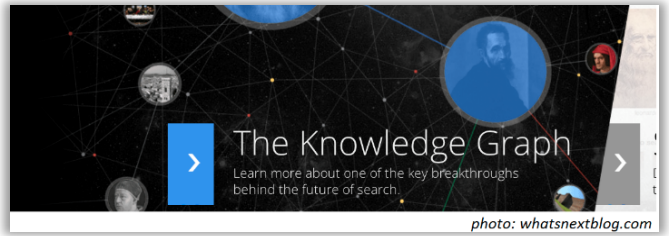
FEIFEI LI

School of Computing
University of Utah

http://ipads.se.sjtu.edu.cn/projects/wukong

Graphs are Everywhere

Online ***graph query*** plays a vital role for searching, mining and reasoning linked data



Jena

TAO

Unicorn

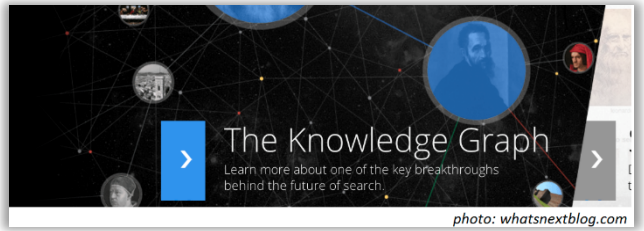
Graph Analytics vs. Graph Query

	Graph Analytics	Graph Query
Graph Model	Property Graph	Semantic (RDF) Graph
Working Set	A whole Graph	A small frac. of Graph
Processing	Batched & Iterative	Concurrent
Metrics	Latency	Latency & Throughput

RDF and SPARQL

Resource Description Framework (**RDF**)

- Representing linked data on the Web
- Public knowledge bases: DBpedia, PubChemRDF, Bio2RDF
- Google's knowledge graph



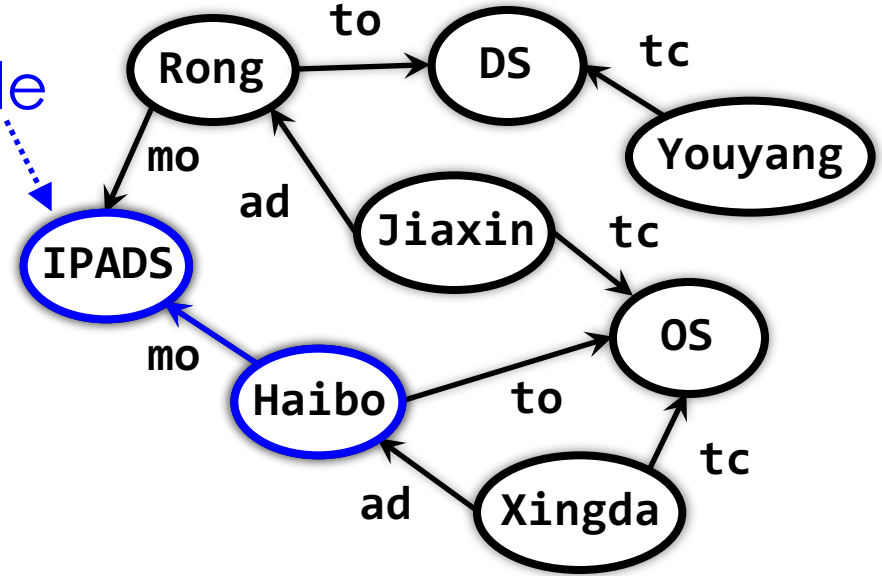
RDF and SPARQL

mo: MemberOf
ad: ADvisor
to: TeacherOf
tc: TakeCourse

RDF is a graph composed by a set of ⟨**S**ubject, **P**redicate, **O**bject⟩ triples

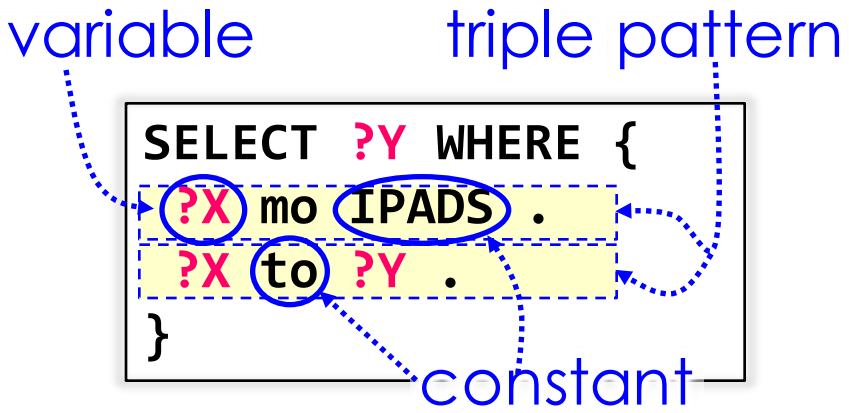
Haibo	mo	IPADS
Haibo	to	OS
Jiaxin	ad	RONG
Jiaxin	tc	OS
Rong	mo	IPADS
Rong	to	DS
Xingda	ad	Haibo
. . .		

triple

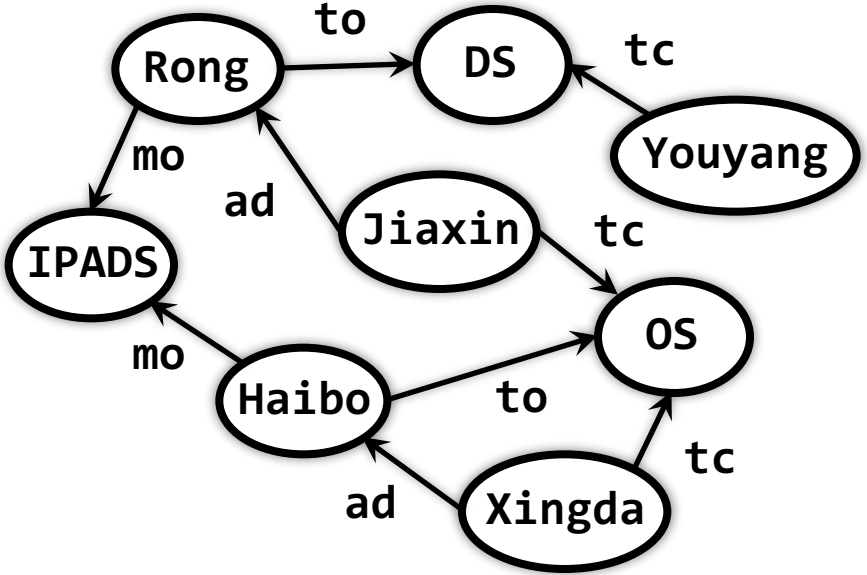


RDF and SPARQL

SPARQL is standard query language for RDF



Courses (?Y) taught by
Teachers (?X) from IPADS



RDF and SPARQL

SPARQL is standard query language for RDF

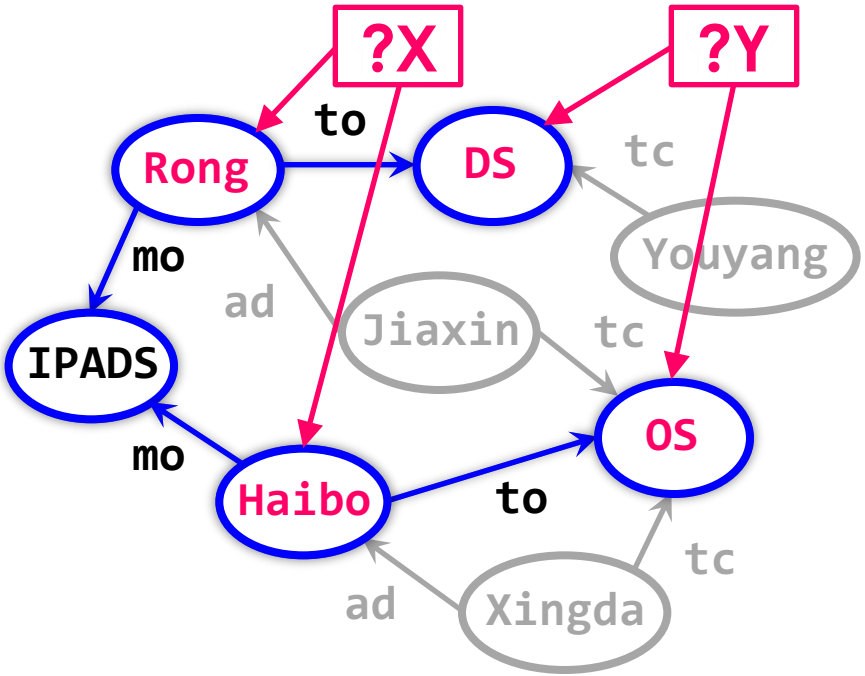
variable

triple pattern

```
SELECT ?Y WHERE {
  ?X mo IPADS .
  ?X to ?Y .
}
```

constant

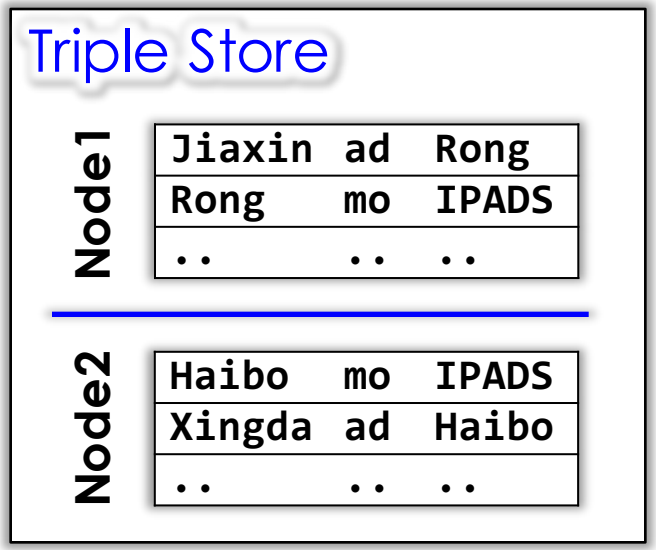
Courses (?Y) taught by
Teachers (?X) from IPADS



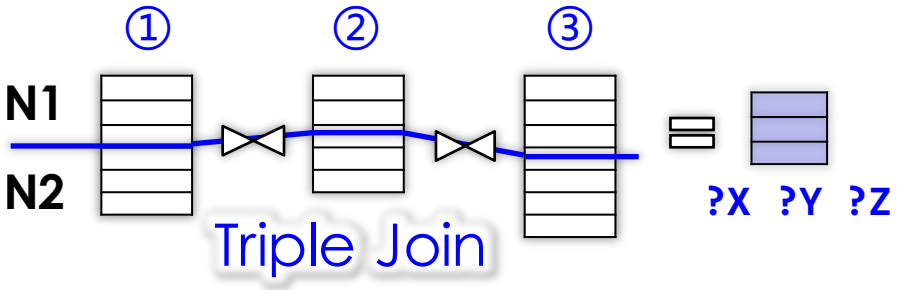
Existing Solutions

Triple Store and Triple Join

- Store RDF data as a set of triples in RDBMS



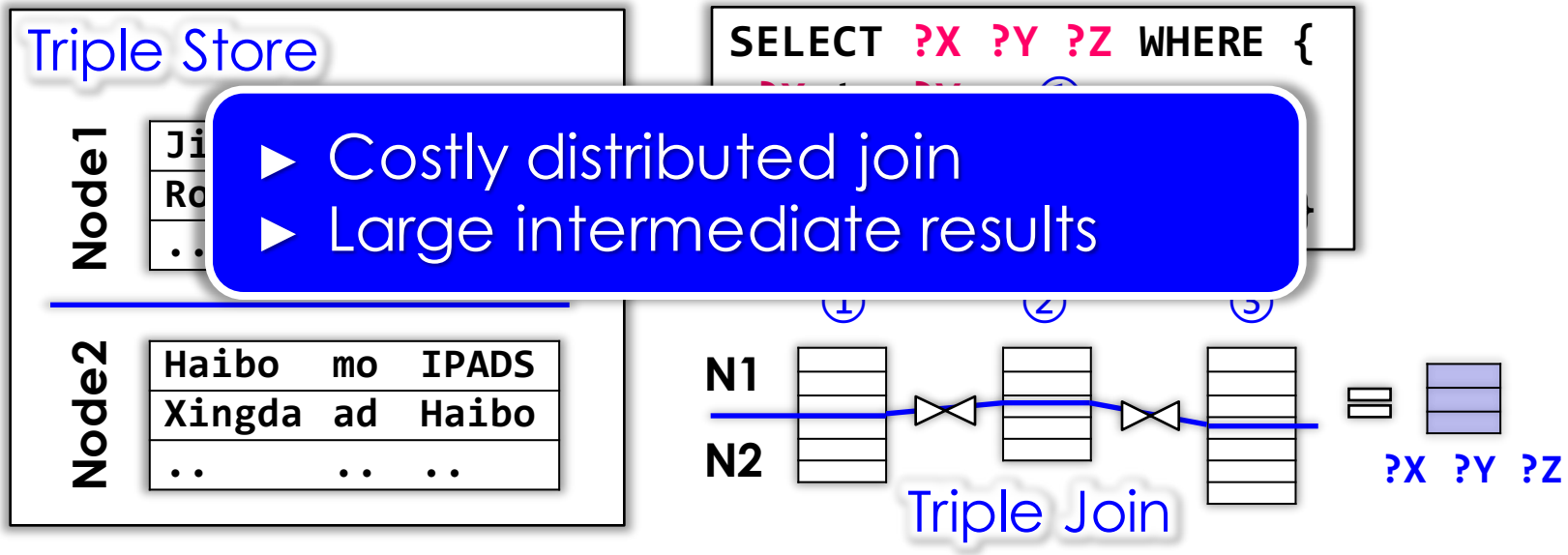

```
SELECT ?X ?Y ?Z WHERE {
  ?X to ?Y . ①
  ?Z tc ?Y . ②
  ?Z ad ?X . ③          }
```



Existing Solutions

Triple Store and Triple Join

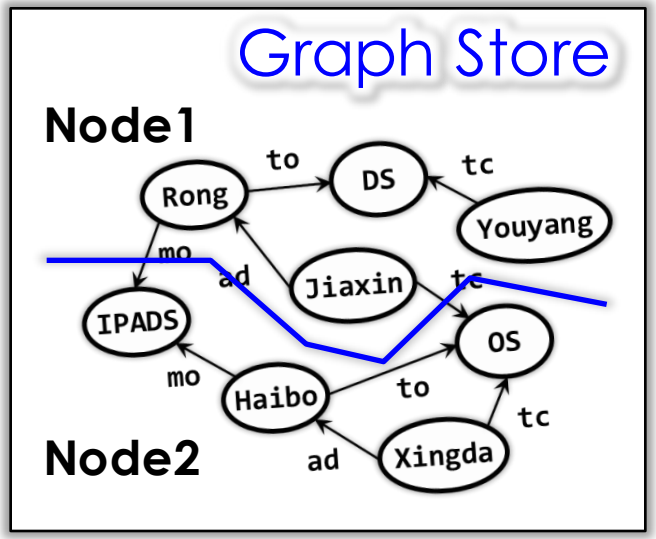
- Store RDF data as a set of triples in RDBMS



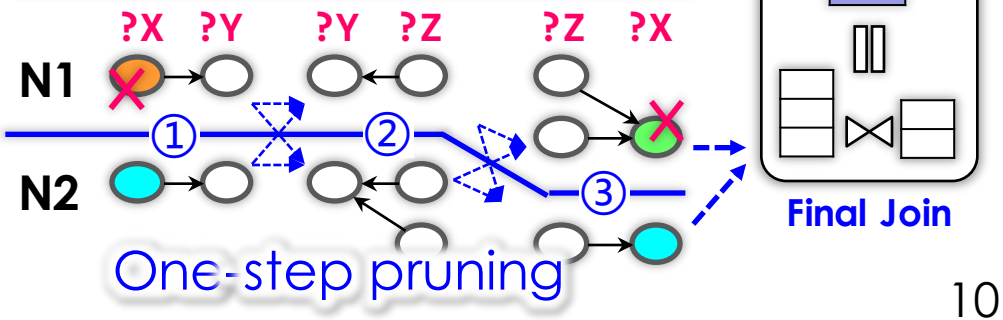
Existing Solutions

Graph Store and Graph Exploration

- Store RDF data in a *native* graph model



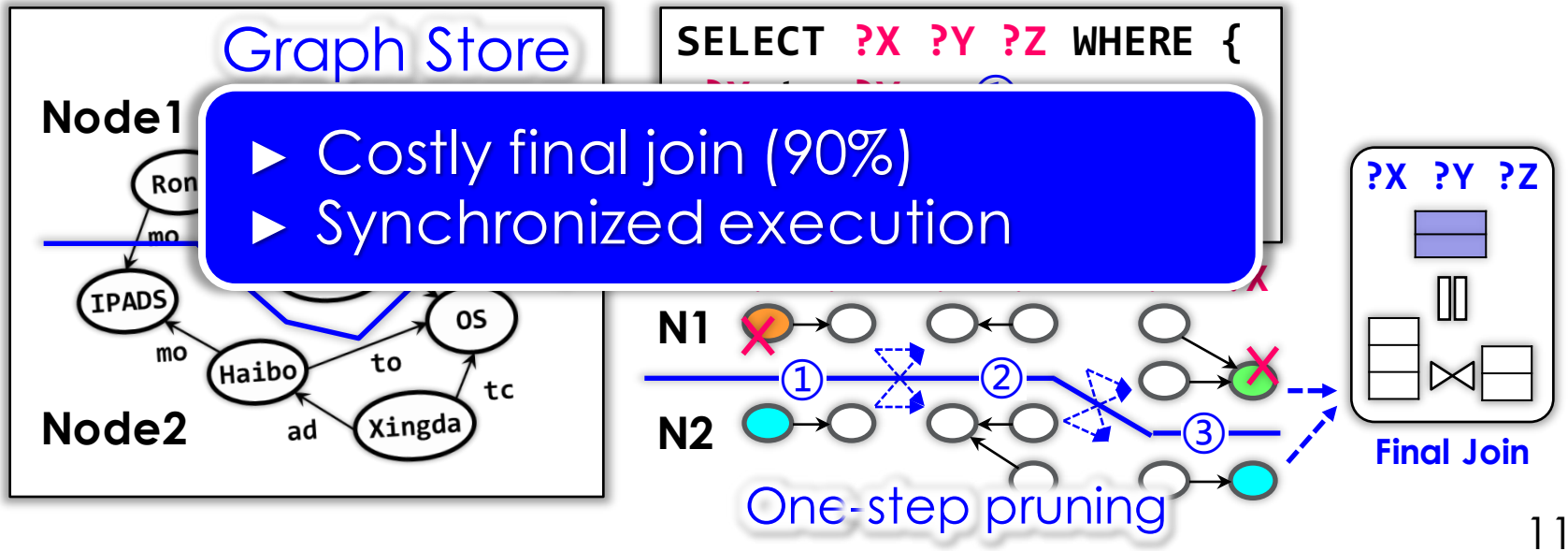

```
SELECT ?X ?Y ?Z WHERE {
  ?X to ?Y . ①
  ?Z tc ?Y . ②
  ?Z ad ?X . ③          }
```



Existing Solutions

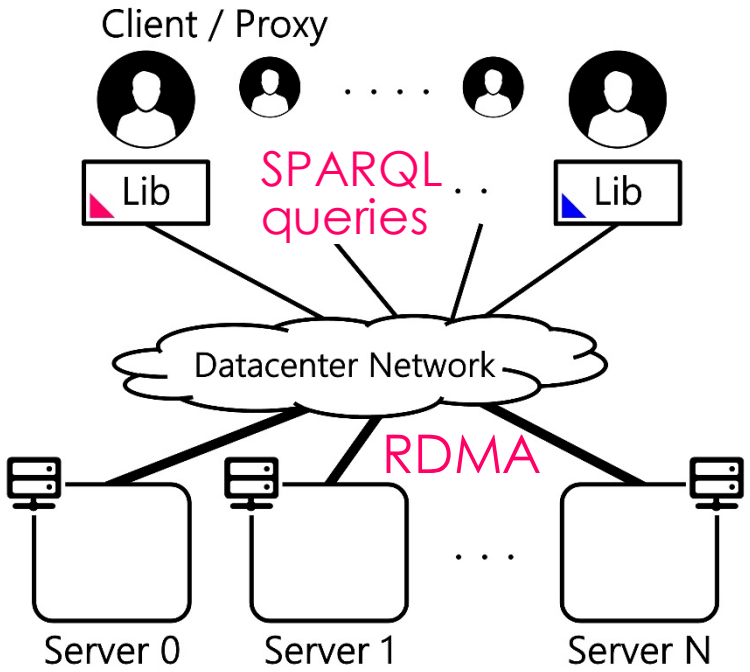
Graph Store and Graph Exploration

- Store RDF data in a *native* graph model



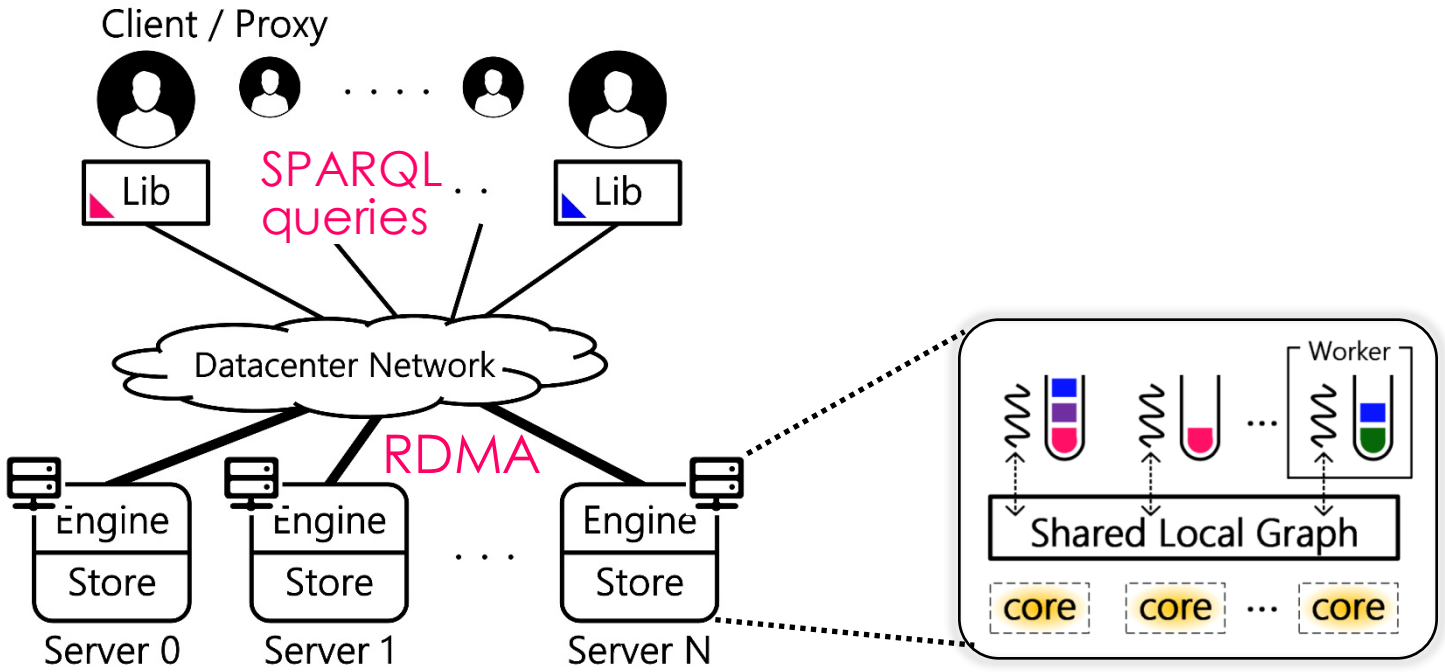
System Overview

Wukong: A distributed in-memory RDF store



System Overview

Wukong : A distributed in-memory RDF store



System Overview

Wukong : A distributed in-memory RDF store

- *RDMA-friendly* graph model
- RDMA-based join-free graph exploration
- Concurrent query processing
- Results vs. state-of-the-art (TriAD/Trinity.RDF)
  - Latency: 11.9X – 28.1X reduction
  - Throughput: 269K queries/sec (up to 740X improvement)

Agenda

Graph-based Model & Store

Query Processing Engine

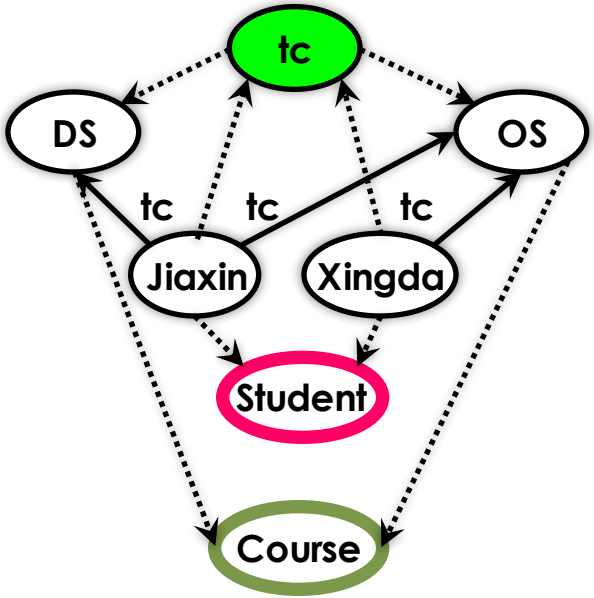
Evaluation

Graph Model and Indexes

```
SELECT ?X WHERE {
  Jiaxin tc ?X .
}
```

Easy to start
from a constant

Predicate index



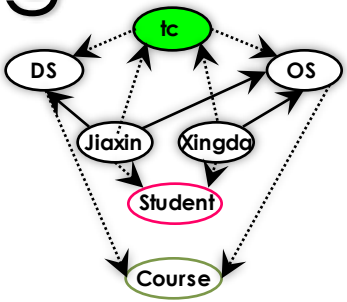
Type index

```
SELECT ?X ?Y
WHERE {
  ?X tc ?Y .
  ?X type Student .
}
```

Hard to query
w/o indexing

Differentiated Graph Partitioning

```
SELECT ?X WHERE {
  Jiaxin tc ?X .
}
```



- Start from normal vertex
- Exploit locality

```
SELECT ?X ?Y
WHERE {
 ?X tc ?Y .
 ?X type Student .
}
```

- Start from index vertex
- Exploit parallelism

Differentiated Graph Partitioning

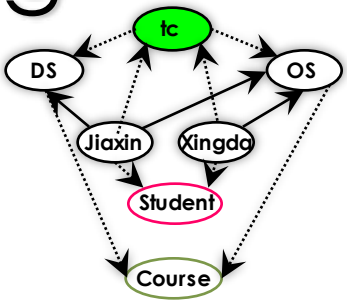
```
SELECT ?X WHERE {
  Jiaxin tc ?X .
}
```

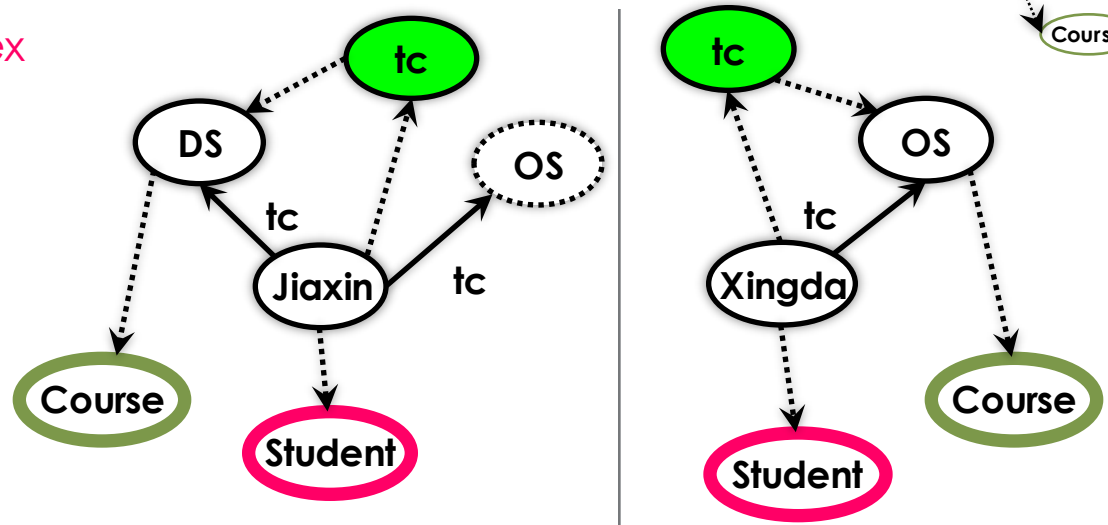
- Start from normal vertex
- Exploit locality

```
SELECT ?X ?Y
WHERE {
 ?X tc ?Y .
 ?X type Student .
}
```

- Start from index vertex
- Exploit parallelism

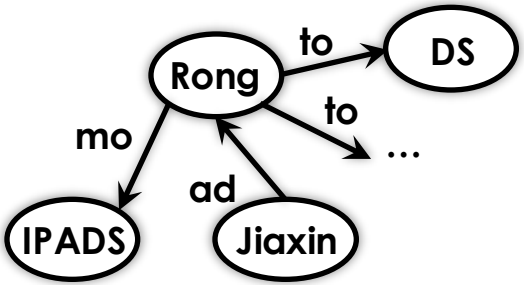
Normal vertex : Distributed
Index vertex : Partitioned

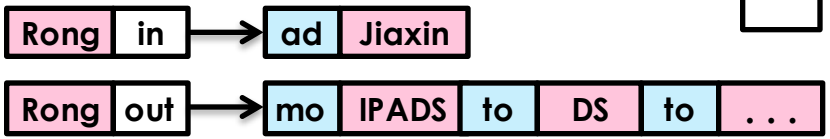


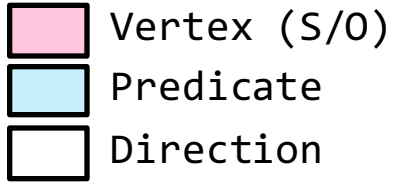


Inspired by PoweLyra [Eurosys'15]

Predicate-based KV Store





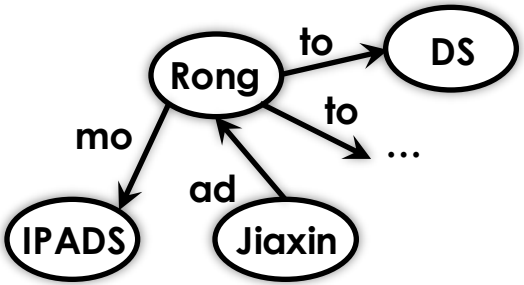



```
SELECT ?X WHERE {
 Rong to ?X .
}
```

constant

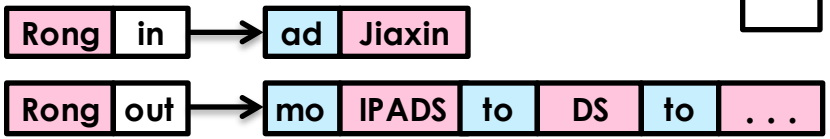
- Inefficient lookup
- Unnecessary data transfer

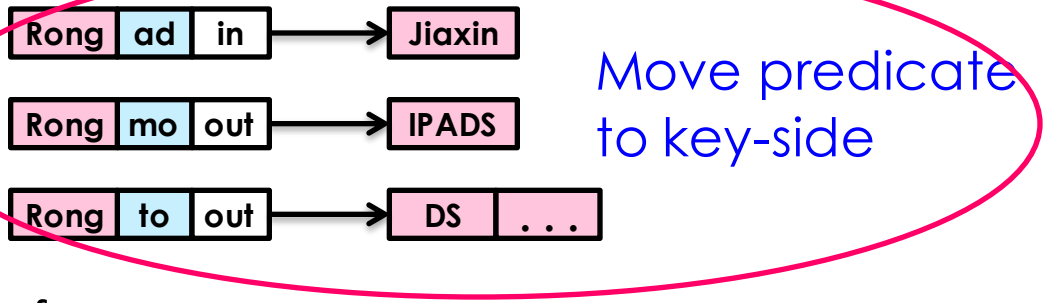
Predicate-based KV Store




```
SELECT ?X WHERE {
 Rong to ?X .
}
```

constant





Move predicate to key-side

Vertex (S/O)
Predicate
Direction

- Inefficient lookup
- Unnecessary data transfer

Finer-grained vertex decomposition

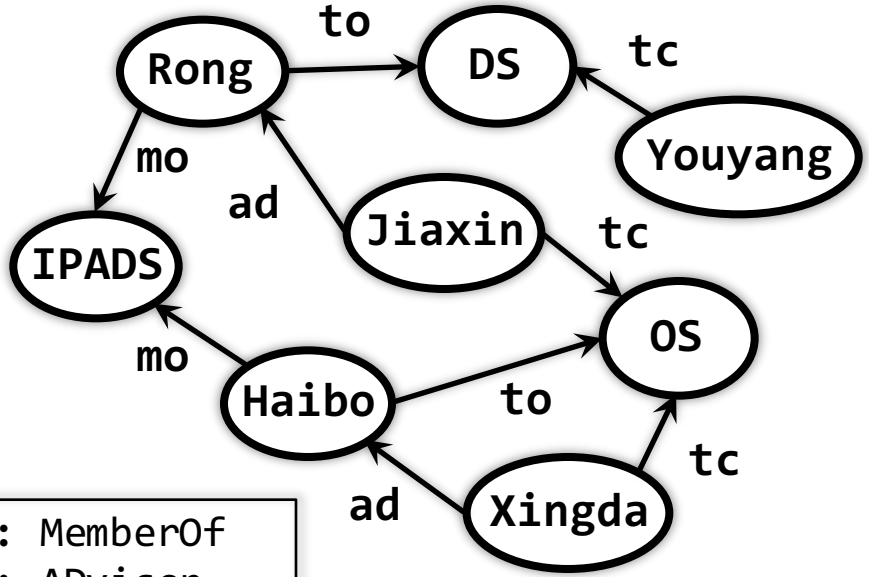
Agenda

Graph-based Model & Store

Query Processing Engine

Evaluation

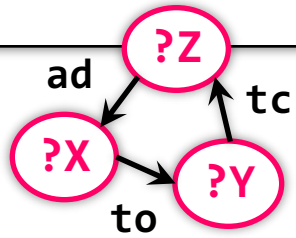
Query Processing



mo: MemberOf
ad: ADvisor
to: TeacherOf
tc: TakeCourse

The teacher advises the student who also takes a course taught by the teacher

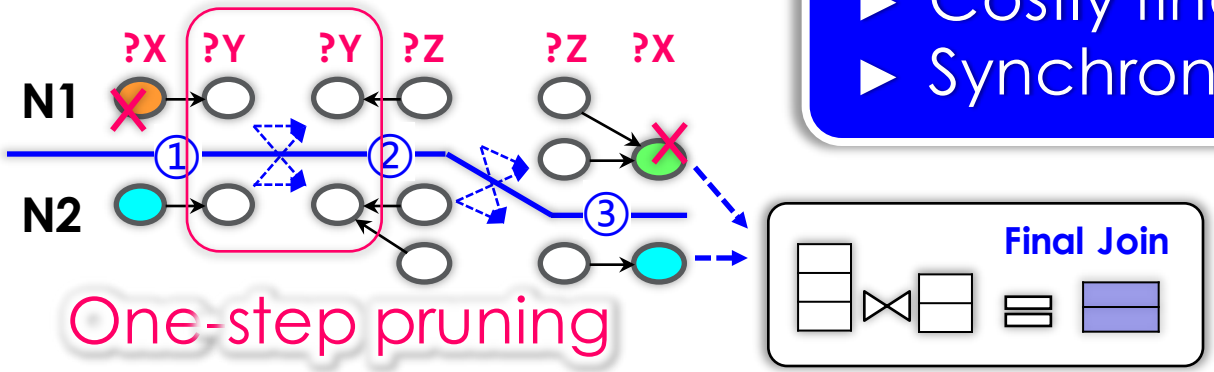
```
SELECT ?X ?Y ?Z WHERE {
   ?X to ?Y .
   ?Z tc ?Y .
   ?Z ad ?X .
}
```



Observation

```
SELECT ?X ?Y ?Z WHERE {
                      }
```

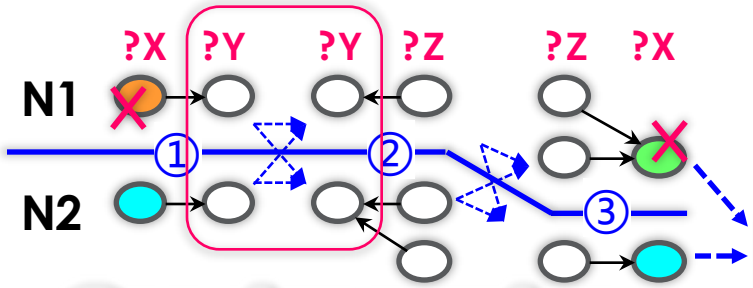
- Costly final join (**90%**)
- Synchronized execution



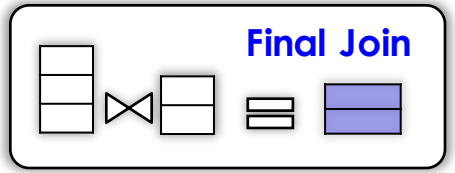
Observation

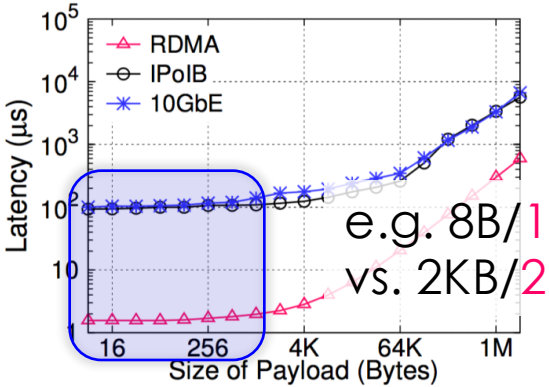
```
SELECT ?X ?Y ?Z WHERE {
                      }
```

- Costly final join (**90%**)
- Synchronized execution



One-step pruning





e.g. 8B/1.56µs
vs. 2KB/2.25µs

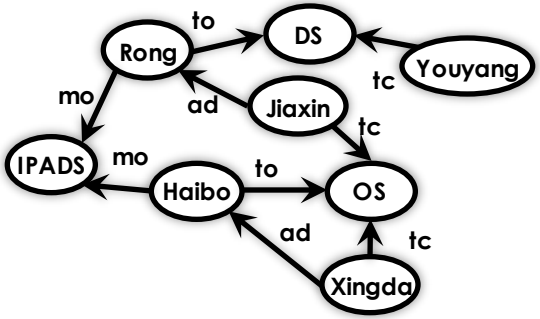
the latency of RDMA is relatively insensitive to payload sizes (~2K)

Graph exploration w/
full-history pruning

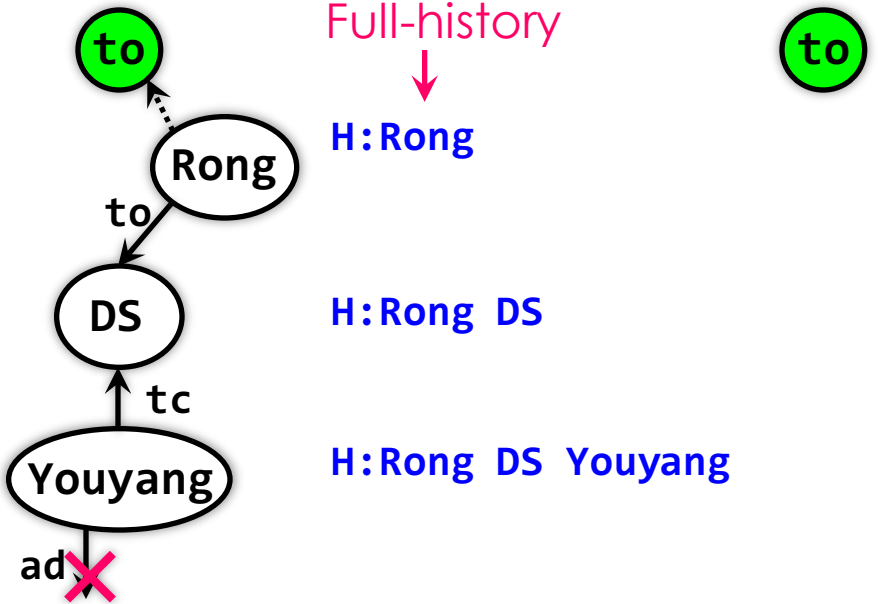
Full-History Pruning


```
SELECT ?X ?Y ?Z WHERE {
  ?X to ?Y .
  ?Z tc ?Y .
  ?Z ad ?X .                }
```

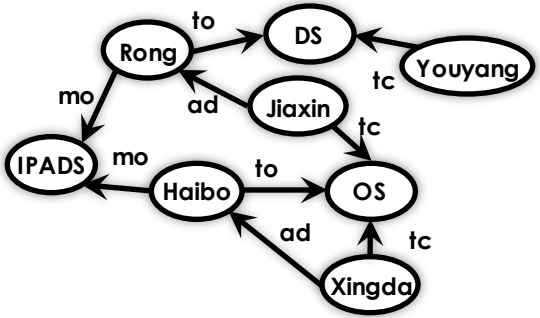
Parallel execution on *predicate* index



Full-History Pruning

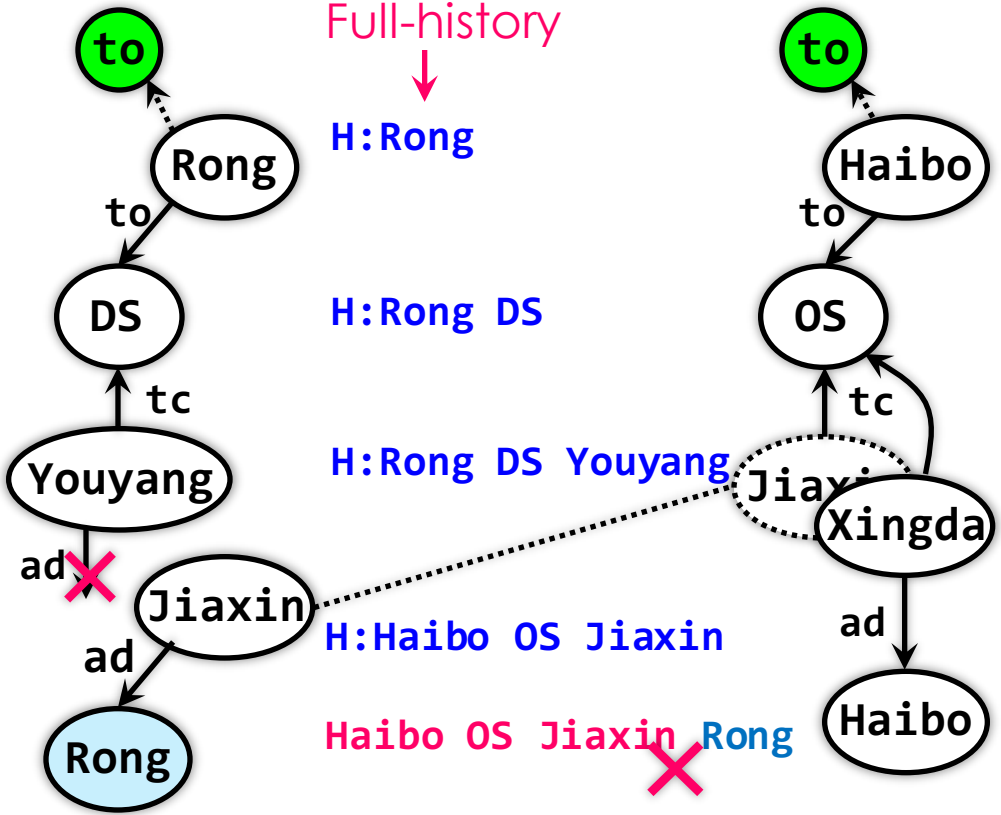


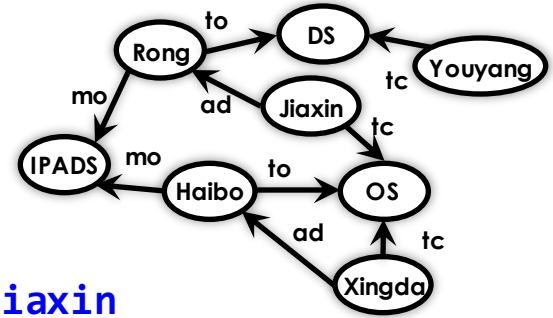

```
SELECT ?X ?Y ?Z WHERE {
 ?X to ?Y .
 ?Z tc ?Y .
 ?Z ad ?X .          }
```



Full-History Pruning

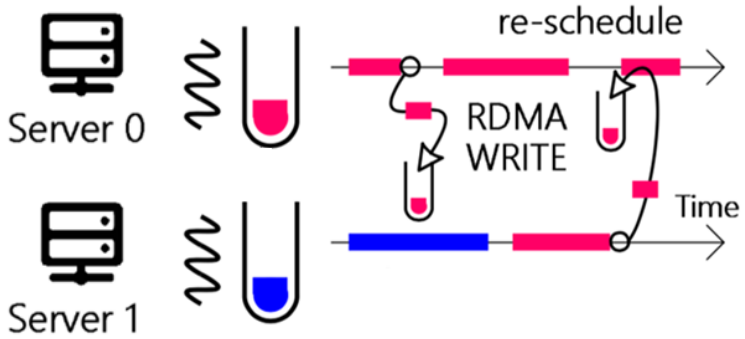
```
SELECT ?X ?Y ?Z WHERE {
 ?X to ?Y .
 ?Z tc ?Y .
 ?Z ad ?X .          }
```





Migrate Execution or Data

Fork-join
(migrate exec)



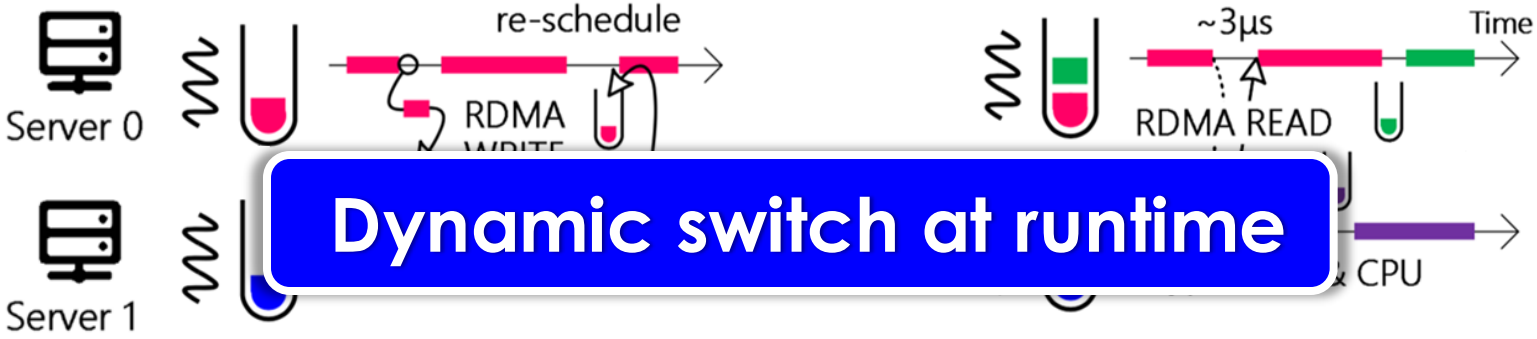
- Send sub-query by RDMA WRITE
- *Async* exploration w/ full-History

Exploit parallelism

Migrate Execution or Data

Fork-join
(migrate exec)

In-place
(migrate data)



- Send sub-query by RDMA WRITE
- *Async* exploration w/ full-History

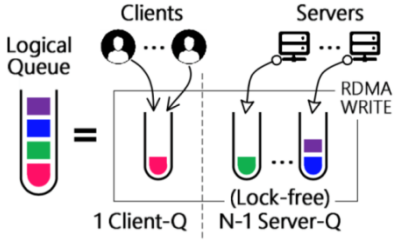
Exploit parallelism

- Fetch data by RDMA READ
- Bypass remote CPU & OS

Exploit low latency

Other Designs of Wukong

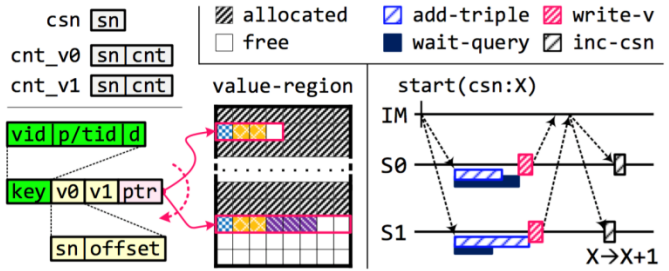
- Logical task queues
- Multi-threading large-query
- Latency-centric work stealing
- Support Evolving graph




```
                                    [tid+next)%N]
3   q = NULL
4   s.lock()
5   if (s.cur == tid //reentry
6    || s.end < now)
7     s.cur = tid;
8     s.end = now + T
9     next++
10    q = s.dequeue()
11  s.unlock()
12  return q
```

```
  SELF()
13  s = state[tid]
14  s.lock()
15  s.cur = tid
16  s.end = now + T
17  next = 1
18  q = s.dequeue()
19  s.unlock()
20  return q

  NEXT_QUERY()
21  if (q = OBLIGER())
22    return q
23  return SELF()
```



Agenda

Graph-based Model & Store

Query Processing Engine

Evaluation

Evaluation

Baseline: state-of-the-art systems

- Centralized: RDF-3X, BitMat
- Distributed: TriAD, Trinity.RDF, SHARD

Platforms: a rack-scale 6-machine cluster

- Each: two 10-cores Intel Xeon, 64GB DRAM, Mellanox 56Gbps InfiniBand NIC w/ RDMA[1]

Benchmarks

- Synthetic: LUBM, WSDTS
- Real-life: DBPSB, YAGO2

Dataset	#Triples	#Subjects	#Objects
LUBM-10240	1,410M	222M	165M
WSDTS	109M	5.2M	9.8M
DBPSB	15M	0.3M	5.2M
YAGO2	190M	10.5M	54.0M

[1] All machines run Ubuntu 14.04 with Mellanox OFED v3.0-2.0.1 stack.

Single Query Latency (msec)

Group I (L1-3,7): large queries

- Start from index vertex
- Touch a large subset of graph
- Speedup: 4.1X - 21.7X

Group II (L4-6): small queries

- Start from normal vertex
- Touch a small subset of graph
- Speedup: 8.4X – 70.6X

LUBM 10240	Wukong	TriAD	TriAD-SG (200K)	Trinity .RDF	SHARD
L1	516	2,110	1,422	12,648	19.7E6
L2	78	512	695	6,081	4.4E6
L3	203	1,252	1,225	8,735	12.9E6
L4	0.41	3.4	3.9	5	10.6E6
L5	0.17	3.1	4.5	4	4.2E6
L6	0.89	63	4.6	9	8.7E6
L7	464	10,055	11,572	31,214	12.0E6
Geo. M	16	190	141	450	9.1E6

Outperform state-of-the-art systems *(Geometric Mean)*

- vs. Trinity.RDF: **28.1X**
- vs. TriAD: **11.9X**

Factor Analysis of Improvement (msec)

BASE
- Graph-exploration
- One-step pruning
- Comm. w/ TCP/IP

+RDMA
- Comm. w/ RDMA

+FHP
- Full-history pruning

+IDX
- Index vertex
- Diff. partitioning

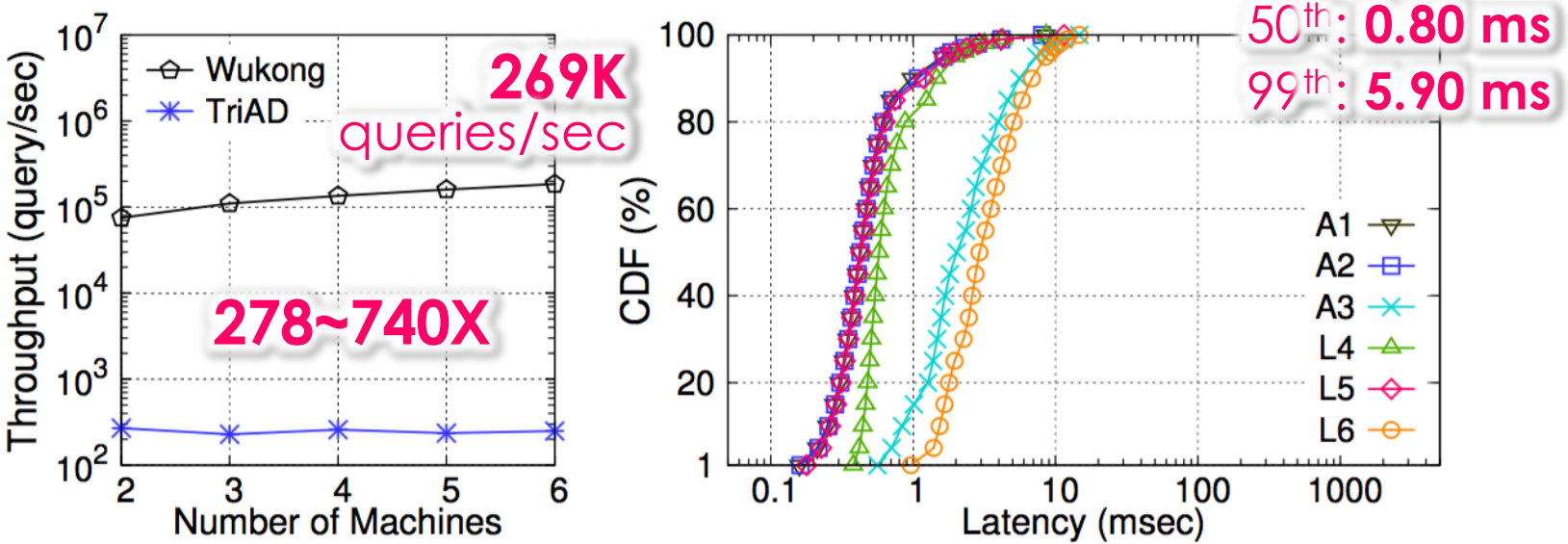
+PBS
- Predicate-base fine-grained Store

+DYN
- In-place execution
- Dynamic switching

LUBM	BASE	+RDMA	+FHP	+IDX	+PBS	+DYN
L1	9,766	9,705	888	853	814	516
L2	2,272	2,161	1,559	84	79	78
L3	421	404	404	205	203	203
L4	1.49	0.79	0.78	0.78	0.56	0.41
L5	1.00	0.39	0.39	0.39	0.31	0.17
L6	3.84	1.40	1.37	1.37	1.17	0.89
L7	2,176	2,041	657	494	466	464
Geo. M	102.3	69.1	39.6	22.6	19.9	15.7

Throughput of Mixed Workloads

Mixed workload: 6 classes of small queries[1]



[1] The templates of 6 classes of queries are based on group (II) queries (L4, L5 and L6) and three additional queries from official website (A1, A2 and A3).

Conclusion

New hardware technologies open opportunities

Wukong: a distributed in-memory RDF store that leverages RDMA-based graph exploration to support fast and concurrent RDF queries

Achieving orders-of-magnitude lower latency & higher throughput than prior state-of-the-art systems

http://ipads.se.sjtu.edu.cn/projects/wukong

Thanks

Wukong, short for Sun Wukong, who is known as the Monkey King and is a main character in the Chinese classical novel "Journey to the West". Since Wukong is known for his extremely fast speed (21,675 kilometers in one somersault) and the ability to fork himself to do massive multi-tasking, we term our system as Wukong.

Questions

http://ipads.se.sjtu.edu.cn/projects/wukong