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

Jiaxin Shi, Youyang Yao, Rong Chen, Haibo Chen

Feifei Li

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

School of Computing
University of Utah

http://ipads.se.sjtu.edu.cn/projects/wukong
Online **graph query** plays a vital role for searching, mining and reasoning linked data.
## Graph Analytics vs. Graph Query

<table>
<thead>
<tr>
<th></th>
<th>Graph Analytics</th>
<th>Graph Query</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph Model</strong></td>
<td>Property Graph</td>
<td>Semantic (RDF) Graph</td>
</tr>
<tr>
<td><strong>Working Set</strong></td>
<td>A whole Graph</td>
<td>A small frac. of Graph</td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Batched &amp; Iterative</td>
<td>Concurrent</td>
</tr>
<tr>
<td><strong>Metrics</strong></td>
<td>Latency</td>
<td>Latency &amp; Throughput</td>
</tr>
</tbody>
</table>
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

RDF is a graph composed by a set of \(\langle \text{Subject}, \text{Predicate}, \text{Object} \rangle\) triples

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haibo</td>
<td>mo</td>
<td>IPADS</td>
</tr>
<tr>
<td>Haibo</td>
<td>to</td>
<td>OS</td>
</tr>
<tr>
<td>Jiaxin</td>
<td>ad</td>
<td>RONG</td>
</tr>
<tr>
<td>Jiaxin</td>
<td>tc</td>
<td>OS</td>
</tr>
<tr>
<td>Rong</td>
<td>mo</td>
<td>IPADS</td>
</tr>
<tr>
<td>Rong</td>
<td>to</td>
<td>DS</td>
</tr>
<tr>
<td>Xingda</td>
<td>ad</td>
<td>Haibo</td>
</tr>
</tbody>
</table>

\(\text{mo: MemberOf}\)
\(\text{ad: ADvisor}\)
\(\text{to: TeacherOf}\)
\(\text{tc: TakeCourse}\)
SPARQL is a standard query language for RDF.

Courses (?Y) taught by Teachers (?X) from IPADS.
SPARQL is standard query language for RDF

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

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

<table>
<thead>
<tr>
<th>Node1</th>
<th>Node2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jiaxin ad Rong</td>
<td>Haibo mo IPADS</td>
</tr>
<tr>
<td>Rong mo IPADS</td>
<td>Xingda ad Haibo</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

① ② ③

Triple Join

N1 ▸ N2

=?X ?Y ?Z
Existing Solutions

Triple Store and Triple Join

- Store RDF data as a set of triples in RDBMS

**Triple Store**

- Costly distributed join
- Large intermediate results

Existing Solutions

Graph Store and Graph Exploration

- Store RDF data in a native graph model

**Graph Store**

```
Rong to DS to Youyang
IPADS mo ad Jiaxin te QSM
Haibo to Xingda
```


**One-step pruning**

```
N1

N2
```

**Final Join**
Existing Solutions

Graph Store and Graph Exploration

- Store RDF data in a *native* graph model

- Costly final join (90%)
- Synchronized execution
System Overview

Wukong: A distributed in-memory RDF store
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

Hard to query w/o indexing

Predicate index

Type index

SELECT ?X ?Y
WHERE {
  ?X tc ?Y .
  ?X type Student .
}
Differentiated Graph Partitioning

- Start from normal vertex
- Exploit locality

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

- Start from index vertex
- Exploit parallelism

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

- Inefficient lookup
- Unnecessary data transfer

SELECT ?X WHERE {
  Rong to ?X .
}

constant
Predicate-based KV Store

- Inefficient lookup
- Unnecessary data transfer

SELECT ?X WHERE {
  Rong to ?X .
} 

Move predicate to key-side

Finer-grained vertex decomposition
Agenda

Graph-based Model & Store

Query Processing Engine

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

```
SELECT ?X ?Y ?Z WHERE {
  ?X to ?Y .
}
```
Observation

One-step pruning

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

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

One-step pruning

Costly final join (90%)

Synchronized execution

Final Join

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

e.g. 8B/1.56μs vs. 2KB/2.25μs

Graph exploration w/ full-history pruning
Full-History Pruning

Parallel execution on predicate index

```sparql
SELECT ?X ?Y ?Z WHERE {
  ?X to ?Y .
}
```
Full-History Pruning

SELECT ?X ?Y ?Z WHERE {
  ?X to ?Y .
}

- H:Rong
- H:Rong DS
- H:Rong DS Youyang
Full-History Pruning

```
SELECT ?X ?Y ?Z WHERE {
  ?X to ?Y .
}
```
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)
- Send sub-query by RDMA WRITE
- Async exploration w/ full-History
  Exploit parallelism

In-place (migrate data)
- Fetch data by RDMA READ
- Bypass remote CPU & OS
  Exploit low latency

Dynamic switch at runtime
Other Designs of Wukong

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

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Triples</th>
<th>#Subjects</th>
<th>#Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM-10240</td>
<td>1,410M</td>
<td>222M</td>
<td>165M</td>
</tr>
<tr>
<td>WSDTS</td>
<td>109M</td>
<td>5.2M</td>
<td>9.8M</td>
</tr>
<tr>
<td>DBPSB</td>
<td>15M</td>
<td>0.3M</td>
<td>5.2M</td>
</tr>
<tr>
<td>YAGO2</td>
<td>190M</td>
<td>10.5M</td>
<td>54.0M</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>LUBM</th>
<th>BASE</th>
<th>+RDMA</th>
<th>+FHP</th>
<th>+IDX</th>
<th>+PBS</th>
<th>+DYN</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>9,766</td>
<td>9,705</td>
<td>888</td>
<td>853</td>
<td>814</td>
<td>516</td>
</tr>
<tr>
<td>L2</td>
<td>2,272</td>
<td>2,161</td>
<td>1,559</td>
<td>84</td>
<td>79</td>
<td>78</td>
</tr>
<tr>
<td>L3</td>
<td>421</td>
<td>404</td>
<td>404</td>
<td>205</td>
<td>203</td>
<td>203</td>
</tr>
<tr>
<td>L4</td>
<td>1.49</td>
<td>0.79</td>
<td>0.78</td>
<td>0.78</td>
<td>0.56</td>
<td>0.41</td>
</tr>
<tr>
<td>L5</td>
<td>1.00</td>
<td>0.39</td>
<td>0.39</td>
<td>0.39</td>
<td>0.31</td>
<td>0.17</td>
</tr>
<tr>
<td>L6</td>
<td>3.84</td>
<td>1.40</td>
<td>1.37</td>
<td>1.37</td>
<td>1.17</td>
<td>0.89</td>
</tr>
<tr>
<td>L7</td>
<td>2,176</td>
<td>2,041</td>
<td>657</td>
<td>494</td>
<td>466</td>
<td>464</td>
</tr>
<tr>
<td>Geo. M</td>
<td>102.3</td>
<td>69.1</td>
<td>39.6</td>
<td>22.6</td>
<td>19.9</td>
<td>15.7</td>
</tr>
</tbody>
</table>
Mixed workload: 6 classes of small queries\(^1\)

Throughput (query/sec):

- Wukong
- TriAD

Throughput: 269K queries/sec

Latency (msec):

- 50\(^{th}\): 0.80 ms
- 99\(^{th}\): 5.90 ms

\(^1\) 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](http://ipads.se.sjtu.edu.cn/projects/wukong)
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.

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