Fast and Concurrent RDF Queries using RDMA-assisted GPU Graph Exploration

Siyuan Wang, Chang Lou, Rong Chen, Haibo Chen

Institute of Parallel and Distributed Systems (IPADS)
Shanghai Jiao Tong University
http://ipads.se.sjtu.edu.cn
Graphs are Everywhere

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

---

The Knowledge Graph

Learn more about one of the key breakthroughs behind the future of search.

photo: whatsnextblog.com

---

TAO

Unicorn
RDF and SPARQL

Resource Description Framework (**RDF**)

- Representing linked data on the Web
- Public knowledge bases: DBpedia, Wikidata, PubChemRDF
- 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>Rong</td>
<td>to</td>
<td>DS</td>
</tr>
<tr>
<td>Rong</td>
<td>mo</td>
<td>IPADS</td>
</tr>
<tr>
<td>Siyuan</td>
<td>ad</td>
<td>Rong</td>
</tr>
<tr>
<td>Siyuan</td>
<td>tc</td>
<td>OS</td>
</tr>
<tr>
<td>Haibo</td>
<td>to</td>
<td>OS</td>
</tr>
<tr>
<td>Haibo</td>
<td>mo</td>
<td>IPADS</td>
</tr>
<tr>
<td>Jiaxin</td>
<td>ad</td>
<td>Haibo</td>
</tr>
</tbody>
</table>

...
SPARQL is standard query language for RDF

Professor (?X) advises (ad) student (?Z) who also takes (tc) a course (?Y) taught by (tc) the professor
Queries are Heterogeneous

Heavy Query ($Q_H$)
- Start from a set of vertices
- Explore a large part of graph

Light Query ($Q_L$)

```sparql
SELECT ?X ?Y ?Z WHERE {
  ?X teacherof ?Y .
}
```
Queries are Heterogeneous

Heavy Query ($Q_H$)
- Start from a set of vertices
- Explore a large part of graph

Light Query ($Q_L$)

SELECT ?X ?Y ?Z WHERE {
  ?X teacherof ?Y .
}
Queries are Heterogeneous

Heavy Query \((Q_H)\)
- Start from a set of vertices
- Explore a large part of graph

Light Query \((Q_L)\)

```
SELECT ?X ?Y ?Z WHERE {
  ?X teacherof ?Y .
}
```
Queries are Heterogeneous

**Heavy Query (Q_H)**
- Start from a set of vertices
- Explore a large part of graph

**Light Query (Q_L)**

```
SELECT ?X ?Y ?Z WHERE {
  ?X teacherof ?Y .
}
```
Queries are Heterogeneous

**Heavy Query** ($Q_H$)
- Start from a set of vertices
- Explore a large part of graph

**Light Query** ($Q_L$)
- Start from a given vertex
- Explore a small part of graph

```sql
SELECT ?X WHERE {
  ?X advisor Rong .
  ?X takecourse OS .
}
```
Queries are Heterogeneous

**Heavy Query (Q_H)**
- Start from a *set* of vertices
- Explore a *large* part of graph

**Light Query (Q_L)**
- Start from a *given* vertex
- Explore a *small* part of graph

```sql
SELECT ?X WHERE {
  ?X advisor Rong .
  ?X takecourse OS .
}
```
Queries are Heterogeneous

Heavy Query (Q_H)
- Start from a set of vertices
- Explore a large part of the graph

Light Query (Q_L)
- Start from a given vertex
- Explore a small part of the graph

Incompetent to handle heavy queries efficiently

Q7*
390 ms

Q5*
0.13 ms

* Wukong on a 10-server cluster for LUBM-10240 dataset
Concurrent Workload

Throughput vs. Latency with logarithmic scale.

Throughput (queries/sec) vs. Median Latency (msec) graph showing better performance.

- Better performance at the lower end of the throughput scale.
Concurrent Workload

Throughput

Latency

logarithmic scale

Thpt: 398K q/s
Lat: 0.10 ms

PURE light query workload
Concurrent Workload

**HYBRID** light & heavy query workload

**Poor performance when facing hybrid workload**

- **Light query workload**
  - Throughput (Thpt): 398K q/s
  - Latency (Lat): 0.10 ms

- **Heavy query workload**
  - Throughput (Thpt): 10 q/s
  - Latency (Lat): 8,600 ms
  - Throughput (Thpt): 10 q/s
  - Latency (Lat): 100 ms

- **Logarithmic scale**

---

**15**
Advanced Hardware

Heterogeneity

- **GPU** has many cores and high memory bandwidth

Fast Communications

- **RDMA** enables direct data transfer between machines
General Idea

Heterogeneous Workload

Light Query

Heterogeneous Hardware

Heavy Query

CPU

GPU
System Overview

Wukong+G: a distributed graph query system that can leverage CPU/GPU to handle hybrid workload

1. GPU-enable graph exploration
2. GPU-friendly RDF store
3. Heterogeneous RDMA Communication

Performance improvement

- Latency: 2.3X - 9.0X speedup for heavy query
- Throughput: 345K queries/sec in hybrid workloads
Wukong Architecture

Proxy / Client

Lib str id String Server str id Lib

Networking w/ RDMA

SPARQL queries

Engine Store CPU Server 0
Engine Store CPU Server 1
Engine Store CPU Server N

Shared Local Graph

Worker Worker Worker

CPU CPU CPU

Client
Wukong+G Architecture
Query Execution on CPU

SELECT ?X ?Y ?Z WHERE {
  ?X teacherof ?Y .
}
Select ?X ?Y ?Z WHERE {
?X teacherof ?Y .
?Z takecourse ?Y.
}
Query Execution on CPU

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

Work Thread

Metadata

Graph Store

CPU DRAM

CPU

Intermediate Result

Query
TP-1
TP-2
...

Key

Value

Store

History Table

?X ?Y ?Z
Query Execution on CPU

**Observation:** all of traversal paths can be explored independently
Query Execution on CPU

```
SELECT ?X ?Y ?Z WHERE {
  ?X teacherof ?Y .
}
```
Query Execution on GPU

```
SELECT ?X ?Y ?Z WHERE {
  ?X teacherof ?Y .
}
```
Query Execution on GPU

```
SELECT ?X ?Y ?Z WHERE {
  ?X teacherof ?Y .
}
```
Challenges

1. Small GPU memory
2. Limited PCIe bw.
3. Long comm. path
Smart data prefetching

GPU-friendly key/value store

Heterogeneous RDMA comm.

Evaluation
Smart Data Prefetching

**Entire RDF graph**

Time

X Out-of-memory

**OB1:** a query only touches a part of RDF data

**OB2:** the predicate of TP is commonly known

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

Case study: Q7 on LUBM - 2560

Granularity | Footprint | Transfer | Data
---|---|---|---
Entire graph | 16.3GB | Failed
Smart Data Prefetching

Case study: Q7 on LUBM-2560

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

<table>
<thead>
<tr>
<th>Case study: Q7 on LUBM-2560</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT ?X ?Y ?Z WHERE {</td>
</tr>
<tr>
<td>?X teacherof ?Y .</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

OB3: TPs of a query will be executed in sequence.

<table>
<thead>
<tr>
<th>Entire RDF graph</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Out-of-memory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Per-query</th>
<th>load time</th>
<th>compute time</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OB3: TPs of a query will be executed in sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per-query</td>
</tr>
<tr>
<td>5.6GB</td>
</tr>
<tr>
<td>5.6GB</td>
</tr>
</tbody>
</table>
Smart Data Prefetching

**Case study**: Q7 on LUBM-2560

<table>
<thead>
<tr>
<th>Granularity</th>
<th>Memory Footprint</th>
<th>Data Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entire graph</td>
<td>16.3GB</td>
<td>Failed</td>
</tr>
<tr>
<td>Per-query</td>
<td>5.6GB</td>
<td>5.6GB</td>
</tr>
<tr>
<td>Per-pattern</td>
<td><strong>2.9GB</strong></td>
<td>5.6GB</td>
</tr>
</tbody>
</table>

SELECT ?X ?Y ?Z WHERE {
  ?X teacherof ?Y .
}
Smart Data Prefetching

Case study: Q7 on LUBM-2560

<table>
<thead>
<tr>
<th>Granularity</th>
<th>Memory Footprint</th>
<th>Data Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entire graph</td>
<td>16.3GB</td>
<td>Failed</td>
</tr>
<tr>
<td>Per-query</td>
<td>5.6GB</td>
<td>5.6GB</td>
</tr>
<tr>
<td>Per-pattern</td>
<td>2.9GB</td>
<td>5.6GB</td>
</tr>
<tr>
<td>Per-block</td>
<td>2.9GB</td>
<td>0.7GB*</td>
</tr>
</tbody>
</table>

* evaluated on 6GB GPU memory
GPU-enable query processing

GPU-friendly key/value store

Heterogeneous RDMA comm.

Evaluation
1. Efficient query processing on CPU

2. Provide possibility to prefetching triples with a certain predicate

Triples with the same predicate are sprinkled all over the store.
GPU-friendly RDF Store

- Predicate-based grouping

$\text{SEG\_OFFSET}(\text{pred}, \text{dir})$

$+ \text{Hash}(vtx)$

$\% \text{SEG\_SZ}(\text{pred}, \text{dir})$
GPU-friendly RDF Store

- Predicate-based grouping

1. **Segment**: prefetching in batch
2. **Hashing**: lookup efficiency
3. **Occupancy rate of segment**:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logan to out</td>
<td>OS DS</td>
</tr>
<tr>
<td>Erik to out</td>
<td></td>
</tr>
<tr>
<td>Pidx to in</td>
<td>Logan Erik</td>
</tr>
<tr>
<td>DS tc in</td>
<td>Kurt Bobby</td>
</tr>
<tr>
<td>OS tc in</td>
<td></td>
</tr>
<tr>
<td>Marie ad out</td>
<td>Logan</td>
</tr>
<tr>
<td>Bobby ad out</td>
<td></td>
</tr>
</tbody>
</table>

**Prefetching Cost**

**vs.**

**Lookup Overhead**

**Tradeoff**

**Segment**
GPU-friendly RDF Store

- Predicate-based grouping
- Fine-grained swapping

Block

RDF Cache (GPU)

RDF Store (CPU)

Key

Logan to out
Erik to out
Pidx to in
DS tc in
Erik to out

Value

Logan, Erik
OS, Kurt

Segment

Ⅱ

N x Blocks

Prefetching

N x Blocks

Logan, Erik
Kurt, Bobby

OS, DS

Logan

Logan

Marie ad out
Bobby ad out

38
GPU-friendly RDF Store

- Predicate-based grouping
- Fine-grained swapping
- Pairwise caching
- Look-ahead replacement (see paper)

RDF Store (CPU)

Key
- Logan to out
- Erik to out
- Pidx to in
- DS tc in
- OS tc in
- Marie ad out
- Bobby ad out

Value
- OS
- DS
- Logan
- Erik
- Kurt
- Bobby
- Logan
Agenda

- GPU-enable query processing
- GPU-friendly key/value store
- Heterogeneous RDMA comm.
- Evaluation
Heterogeneous RDMA Communication

Metadata: *Query*

Data: *History Table*
Heterogeneous RDMA Communication

Metadata: *Query*

Data: *History Table*
Heterogeneous RDMA Communication

(Native) RDMA

Metadata: Query
- ② CPU → CPU (RDMA)

Data: History Table
- ① GPU → CPU (PCIe)
- ② CPU → CPU (RDMA)
- ③ CPU → GPU (PCIe)
- ④ GPU → CPU (PCIe)
- ⑤ CPU → CPU (RDMA)
Heterogeneous RDMA Communication

RDMA with **GPUDirect**

**Metadata:** *Query*
- ② CPU \(\rightarrow\) CPU (RDMA)

**Data:** *History Table*
- ① GPU \(\rightarrow\) CPU (PCIe)
- ② CPU \(\rightarrow\) CPU (RDMA)
- ③ CPU \(\rightarrow\) GPU (PCIe)
- ④ GPU \(\rightarrow\) CPU (PCIe)
- ⑤ CPU \(\rightarrow\) CPU (RDMA)
Heterogeneous RDMA Communication

**RDMA with GPUDirect**

Metadata: *Query*

1. CPU → CPU (RDMA)

Data: *History Table*

1. GPU → CPU (PCIe)
2. GPU → CPU (RDMA+G)
3. CPU → GPU (PCIe)
4. GPU → CPU (PCIe)
5. CPU → CPU (RDMA)
Heterogeneous RDMA Communication

RDMA with GPUDirect

Metadata: *Query*

1. CPU $\rightarrow$ CPU (RDMA)

Data: *History Table*

1. GPU $\rightarrow$ CPU (PCIe)
2. GPU $\rightarrow$ CPU (RDMA+G)
3. CPU $\rightarrow$ GPU (PCIe)
4. GPU $\rightarrow$ CPU (PCIe)
5. GPU $\rightarrow$ CPU (RDMA+G)
GPU-enable query processing

GPU-friendly key/value store

Heterogeneous RDMA comm.

Evaluation
Evaluation

Baseline: state-of-the-art systems
- Wukong, TriAD (distributed triple store)

Platforms: 10 servers on a rack-scale 5-node cluster
- RDMA: Mellanox 56Gbps IB NIC, 40Gbps IB Switch
- Two servers run on a single machine
- Each server: 12-core Intel Xeon, 128GB DRAM, NVIDIA Tesla K40m (2880 cores, 12GB DRAM)

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#T</th>
<th>#S</th>
<th>#O</th>
<th>#P</th>
<th>Size†</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM-2560</td>
<td>352 M</td>
<td>55 M</td>
<td>41 M</td>
<td>17</td>
<td>58GB</td>
</tr>
<tr>
<td>LUBM-10240</td>
<td>1,410 M</td>
<td>222 M</td>
<td>165 M</td>
<td>17</td>
<td>230GB</td>
</tr>
<tr>
<td>DBPSB</td>
<td>15 M</td>
<td>0.3 M</td>
<td>5.2 M</td>
<td>14,128</td>
<td>2.8GB</td>
</tr>
<tr>
<td>YAGO2</td>
<td>190 M</td>
<td>10.5 M</td>
<td>54.0 M</td>
<td>99</td>
<td>13GB</td>
</tr>
</tbody>
</table>
Single Query Latency (msec)

Heavy queries (Q1-Q3, Q7)
- Start from a set of vertices
- Touch a large part of graph
- Speedup: **2.3X~9X** vs. Wukong

Light queries (Q4-Q6)
- Start from a given vertex
- Touch a small part of graph
- Negligible slowdown

<table>
<thead>
<tr>
<th>LUBM-2560</th>
<th>Wukong</th>
<th>Wukong+G</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q1 (3.6GB)</td>
<td>992</td>
<td>165</td>
</tr>
<tr>
<td>Q2 (2.4GB)</td>
<td>138</td>
<td>31</td>
</tr>
<tr>
<td>Q3 (3.6GB)</td>
<td>340</td>
<td>63</td>
</tr>
<tr>
<td>Q7 (5.6GB)</td>
<td>828</td>
<td>100</td>
</tr>
<tr>
<td>Geo. M</td>
<td>443</td>
<td>75</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LUBM-10240</th>
<th>Wukong</th>
<th>Wukong+G</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q1 (14.25GB)</td>
<td>480</td>
<td>211</td>
</tr>
<tr>
<td>Q2 (9.74GB)</td>
<td>66</td>
<td>12</td>
</tr>
<tr>
<td>Q3 (14.25GB)</td>
<td>171</td>
<td>19</td>
</tr>
<tr>
<td>Q7 (22.58GB)</td>
<td>390</td>
<td>100</td>
</tr>
<tr>
<td>Geo. M</td>
<td>215</td>
<td>47</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q4</td>
<td>0.13</td>
<td>0.16</td>
</tr>
<tr>
<td>Q5</td>
<td>0.09</td>
<td>0.11</td>
</tr>
<tr>
<td>Q6</td>
<td>0.49</td>
<td>0.51</td>
</tr>
<tr>
<td>Geo. M</td>
<td>0.18</td>
<td>0.21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10-server cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>single server</td>
</tr>
</tbody>
</table>
Performance of Hybrid Workloads

**WKD** (default)
Heavy/Light: ALL of CPUs (10)

**WKI** (Isolation)
Heavy: HALF of CPUs (5)
Light: HALF of CPUs (5)

**WKG (+G)**
Heavy: CPU (1) + GPUs (1)
Light: REST of CPU (9)

Workload: 6 classes of light queries + 4 classes of heavy queries
Performance of Hybrid Workloads

**Workload**: 6 classes of light queries + 4 classes of heavy queries

- **WKD** (default)
  - Heavy/Light: ALL of CPUs (10)

- **WKI** (Isolation)
  - Heavy: HALF of CPUs (5)
  - Light: HALF of CPUs (5)

- **WKG (+G)**
  - Heavy: CPU (1) + GPUs (1)
  - Light: REST of CPU (9)
Performance of Hybrid Workloads

Workload: 6 classes of light queries + 4 classes of heavy queries

- **WKD** (default)
  - Heavy/Light: ALL of CPUs (10)

- **WKI** (Isolation)
  - Heavy: HALF of CPUs (5)
  - Light: HALF of CPUs (5)

- **WKG** (+G)
  - Heavy: CPU (1) + GPUs (1)
  - Light: REST of CPU (9)
Conclusion

Hardware heterogeneity opens opportunities for hybrid workloads on graph data

Wukong+G: a distributed RDF query system supports heterogeneous CPU/GPU processing for hybrid queries on graph data

Outperform prior state-of-the-art systems by more than one order of magnitude when facing hybrid workloads

Website: http://ipads.se.sjtu.edu.cn/projects/wukong
GitHub: https://github.com/SJTU-IPADS/wukong
Wukong+G

GitHub: https://github.com/SJTU-IPADS/wukong

Institute of Parallel and Distributed Systems
Shanghai Jiao Tong University
Distinguish Heavy & Light Queries

Query plan optimizer

- **Query plan**: the order of triple patterns
- Using a **cost-model** to estimate the execution time of different plans for a given query
- For SPARQL query, cost model is roughly based on #paths may be explored

Wukong+G uses a **user-defined threshold for #paths** to distinguish heavy and light queries
GPU Memory Size Limitation

Too large predicate segment
1. Load one part of segment to GPU memory
2. Do traversal work
Repeat 1 and 2

Too large intermediate results
1. Load one part of history table to GPU memory
2. Do traversal work
Repeat 1 and 2
Multi-GPUs Support

- Run a separate server for each GPU card and several co-located CPU cores (usually a socket)
- All servers comply with the same communication mechanism via GPUDirect RDMA operations
# 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>
# Factor Analysis of Improvement

- Single Server w/ 3GB GPU memory
- LUBM-2560

<table>
<thead>
<tr>
<th>LUBM-2560</th>
<th>Per-query</th>
<th>Per-pattern</th>
<th>Per-block</th>
<th>Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 (3.6GB)</td>
<td>x</td>
<td>743</td>
<td>313</td>
<td>295</td>
</tr>
<tr>
<td>Q2 (2.4GB)</td>
<td>284</td>
<td>283</td>
<td>32</td>
<td>31</td>
</tr>
<tr>
<td>Q3 (3.6GB)</td>
<td>x</td>
<td>309</td>
<td>62</td>
<td>63</td>
</tr>
<tr>
<td>Q7 (5.6GB)</td>
<td>x</td>
<td>893</td>
<td>622</td>
<td>610</td>
</tr>
</tbody>
</table>
RDF Cache of GPU

- LUBM-2560
- 10GB GPU Memory

Fig. 11: The latency with the increase of GPU memory.

Fig. 12: The CDF of latency for mixed heavy workload.