A Parallel System for Efficiently Managing Large Graphs

Jeong-Hyon Hwang, Sean Spillane, Daniel Bokser,
Daniel Kemp, Jayadevan Vijayan, Jeremy Birnbaum

State University of New York - Albany
Graph Data Analytics

• Transportation

5:00 AM - 6:00 AM
Graph Data Analytics

- Transportation
  5:00 AM - 6:00 AM
  9.1 mi, 20 mins
Graph Data Analytics

- Transportation

5:00 AM - 6:00 AM

9.1 mi, 20 mins
Graph Data Analytics

- Transportation

5:00 AM - 6:00 AM

6:00 AM - 7:00 AM

9.1 mi, 20 mins

15 mi, 25 mins
Graph Data Analytics

- Transportation

5:00 AM - 6:00 AM

6:00 AM - 7:00 AM

9.1 mi, 20 mins

15 mi, 25 mins

Saturday, July 6, 2013
Graph Data Analytics

- Transportation

5:00 AM - 6:00 AM
9.1 mi, 20 mins

6:00 AM - 7:00 AM
15 mi, 25 mins

7:00 AM - 8:00 AM
20 mi, 30 mins
Graph Data Analytics

- Transportation

5:00 AM - 6:00 AM

6:00 AM - 7:00 AM

7:00 AM - 8:00 AM

9.1 mi, 20 mins

15 mi, 25 mins

20 mi, 30 mins
Graph Data Analytics

- Transportation

5:00 AM - 6:00 AM

6:00 AM - 7:00 AM

7:00 AM - 8:00 AM

A -> B: \( \frac{\text{STD(Driving Time)}}{\text{AVG(Driving Time)}} = \frac{\text{STD}(20, 25, 30)}{\text{AVG}(20, 25, 30)} = \frac{5}{25} = 0.2 \)
Graph Data Analytics

- **Transportation**
  - 5:00 AM - 6:00 AM
  - 6:00 AM - 7:00 AM
  - 7:00 AM - 8:00 AM

- **Social and Political Studies / Marketing / National Security**

- **Epidemic Simulations and Analysis**

\[ \frac{\text{STD(Driving Time)}}{\text{AVG(Driving Time)}} = \frac{\text{STD(20, 25, 30)}}{\text{AVG(20, 25, 30)}} = \frac{5}{25} = 0.2 \]
# Limitations of Existing Systems

<table>
<thead>
<tr>
<th>RDBMSs</th>
<th>inconvenient</th>
<th>inefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no SQL support for graphs</td>
<td>no built-in graph traversal capabilities</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Graph Systems</th>
<th>coding for each type of computation</th>
<th>only one graph at a time</th>
</tr>
</thead>
</table>
## Limitations of Existing Systems

<table>
<thead>
<tr>
<th></th>
<th>Inconvenient</th>
<th>Inefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RDBMSs</strong></td>
<td>no SQL support for graphs</td>
<td>no built-in graph traversal capabilities</td>
</tr>
<tr>
<td><strong>Graph Systems</strong></td>
<td>coding for each type of computation</td>
<td>only one graph at a time</td>
</tr>
</tbody>
</table>

*NEDB Day 2012*
Limitations of Existing Systems

Example: **shortest distance** in Pregel

```java
class ShortestPathVertex : public Vertex<int, int, int> {
    void Compute(MessageIterator* msgs) {
        int mindist = IsSource(vertex_id()) ? 0 : INF;
        for (; !msgs->Done(); msgs->Next())
            mindist = min(mindist, msgs->Value());
        if (mindist < GetValue()) {
            *mutableValue() = mindist;
            OutEdgeIterator iter = GetOutEdgeIterator();
            for (; !iter.Done(); iter.Next())
                sendMessageTo(iter.Target(),
                    mindist + iter.GetValue());
        }
        voteToHalt();
    }
}
```

```java
class MinIntCombiner : public Combiner<int> {
    virtual void Combine(MessageIterator* msgs) {
        int mindist = INF;
        for (; !msgs->Done(); msgs->Next())
            mindist = min(mindist, msgs->Value());
        Output("combined_source", mindist);
    }
}
```

---

NEDB Day 2012

---

Graph Systems: coding for each type of computation

only one graph at a time
## Limitations of Existing Systems

<table>
<thead>
<tr>
<th></th>
<th>Inconvenient</th>
<th>Inefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RDBMSs</strong></td>
<td>no SQL support for graphs</td>
<td>no built-in graph traversal capabilities</td>
</tr>
<tr>
<td><strong>Graph Systems</strong></td>
<td>coding for each type of computation</td>
<td>only one graph at a time</td>
</tr>
</tbody>
</table>
The G* Project

• Goal
  - convenient and efficient queries on graphs using a server cluster

• Key Features
  - succinct graph query language
  - distributed, deduplicate storage of graphs
  - parallel query processing that shares computations across graphs
The G* Project

• Goal
  - convenient and efficient queries on graphs using a server cluster

• Key Features
  - succinct graph query language
  - distributed, deduplicate storage of graphs
  - parallel query processing that shares computations across graphs
The G* Project

• **Goal**
  - convenient and efficient queries on graphs using a server cluster

• **Key Features**
  - succinct graph query language
  - distributed, deduplicate storage of graphs
  - parallel query processing that shares computations across graphs
The G* Project

• Goal
  - convenient and efficient queries on graphs using a server cluster

• Key Features
  - succinct graph query language
  - distributed, deduplicate storage of graphs
  - parallel query processing that shares computations across graphs

NEDB Day 2012
The G* Project

• Goal
  - convenient and efficient queries on graphs using a server cluster

• Key Features
  - succinct graph query language
  - distributed, deduplicate storage of graphs
  - parallel query processing that shares computations across graphs
The G* Project

• Goal
  - convenient and efficient queries on graphs using a server cluster

• Key Features
  - succinct graph query language
  - distributed, deduplicate storage of graphs
  - parallel query processing that shares computations across graphs
The G* Project

• Goal
  - convenient and efficient queries on graphs using a server cluster

• Key Features
  - succinct graph query language
  - distributed, deduplicate storage of graphs
  - parallel query processing that shares computations across graphs
The G* Project

- **Goal**
  - convenient and efficient queries on graphs using a server cluster

- **Key Features**
  - succinct graph query language
  - distributed, deduplicate storage of graphs
  - parallel query processing that shares computations across graphs
Distribution of Graph Data
Distribution of Graph Data
Distribution of Graph Data

G

S1

S2

S3

......

NEDB Day 2012

Saturday, July 6, 2013
Distribution of Graph Data

\[ G_1 \]

\[ S_1 \]

\[ S_2 \]

\[ S_3 \]

\[ \ldots \]
Distribution of Graph Data

G₁

S₁

S₂

S₃

......
Distribution of Graph Data

G_1

G_2

S_1

S_2

S_3

......
Distribution of Graph Data

G_1

G_2

G_1 ∩ G_2

G_1 - G_2

G_2 - G_1

S_1

S_2

S_3

......
Distribution of Graph Data

\[ G_1 \]
\[
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c} \\
\text{d}
\end{array}
\]

\[ G_2 \]
\[
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c} \\
\text{e}
\end{array}
\]

\[ G_1 \cap G_2 \]
\[
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{d}
\end{array}
\]

\[ G_1 - G_2 \]
\[
\begin{array}{c}
\text{c}
\end{array}
\]

\[ G_2 - G_1 \]
\[
\begin{array}{c}
\text{c} \\
\text{e}
\end{array}
\]

\[ S_1 \]

\[ S_2 \]

\[ S_3 \]

\[ \ldots \]

Saturday, July 6, 2013
Distribution of Graph Data

G

G

G

S

S

S

G

G

G

NEDB Day 2012
Distribution of Graph Data
Storage of Graphs (on server)
Storage of Graphs (on server)

G

Memory

Disk

attributes of vertex c

attributes of vertex d
Storage of Graphs (on server)

G_1

{G_1}

Memory

Disk

attributes of vertex c

attributes of vertex d

Saturday, July 6, 2013
Storage of Graphs (on server)

- $G_1 - G_2$
- $G_1 \cap G_2$
- $G_2 - G_1$

- Memory
- Disk

- Attributes of vertex c
- Attributes of vertex d
Storage of Graphs (on server)

- $G_1 - G_2$
- $G_1 \cap G_2$
- $G_2 - G_1$

- Attributes of vertex c
- Attributes of vertex d
- Attributes of edge to e
- Attributes of vertex e

Memory

Disk
Storage of Graphs (on server)

- $G_1 - G_2$
- $G_1 \cap G_2$
- $G_2 - G_1$

**Memory**
- $\{G_1\}$
  - $c$
  - $d$
- $\{G_1, G_2\}$
- $\{G_2\}$
  - $c$
  - $e$

**Disk**
- [1]: attributes of vertex c
- [2]: attributes of vertex d
- [3]: attributes of edge to e
- [4]: attributes of vertex e
Storage of Graphs (on server)

G₁-G₂-G₃  (G₁∩G₂)-G₃  (G₂∩G₃)-G₁  G₃-G₁-G₂

{G₁}  {G₁,G₂}  {G₂,G₃}

Memory

Disk

[1] attributes of vertex c
[2] attributes of vertex d
[3] attributes of edge to e
[4] attributes of vertex e
Storage of Graphs (on server)

- $G_1$ - $G_2$ - $G_3$
- $(G_1 \cap G_2)$ - $G_3$
- $(G_2 \cap G_3)$ - $G_1$
- $G_3$ - $G_1$ - $G_2$

Memory:
- $\{G_1\}$: attributes of vertex c
- $\{G_1, G_2\}$: attributes of vertex d
- $\{G_2, G_3\}$: attributes of vertex e

Disk:
- $[1]$ attributes of vertex c
- $[2]$ attributes of vertex d
- $[3]$ attributes of edge to e
- $[4]$ attributes of vertex e
- $[5]$ attributes of edge to f
- $[6]$ attributes of vertex f
Storage of Graphs (on server)

- $G_1 - G_2 - G_3$
- $(G_1 \cap G_2) - G_3$
- $(G_2 \cap G_3) - G_1$
- $G_3 - G_1 - G_2$

**Memory**

- $\{G_1\}$
  - c
  - [1]
- $\{G_1, G_2\}$
  - d
  - [2]
- $\{G_2, G_3\}$
  - e
  - [3]
  - f
  - [4]
- $\{G_3\}$
  - d
  - [5]
  - f
  - [6]

**Disk**

- [1] attributes of vertex c
- [2] attributes of vertex d
- [3] attributes of edge to e
- [4] attributes of vertex e
- [5] attributes of edge to f
- [6] attributes of vertex f
Storage of Graphs (on server)

Attributes of vertex c
Attributes of vertex d
Attributes of edge to e
Attributes of vertex e
Attributes of edge to f
Attributes of vertex f

Memory

Disk
Example: average degree of each graph in \{G_1, G_2\}

Graph Query Execution
Example: average degree of each graph in \{G_1, G_2\}

**Graph Query Execution**

![Diagram](image)

- **average**
- **union**
- **count, sum**
- **degree**
- **vertex**

- **S_1**
- **S_2**
- **S_3**

*Saturday, July 6, 2013*
Example: average degree of each graph in \{G_1,G_2\}

**Graph Query Execution**

![Graph Query Execution Diagram]
Example: average degree of each graph in \( \{G_1, G_2\} \)

Graph Query Execution

\[
\begin{align*}
\text{average} & \quad \text{union} \\
\text{count, sum} & \quad (a, 2, \{G_1, G_2\}) \\
\text{degree} & \quad (a, \Diamond, \{G_1, G_2\}) \\
\text{vertex} & \\
\{G_1, G_2, G_3\} & \quad \{G_1, G_2, G_3\} & \quad \{G_1\} & \quad \{G_1, G_2\} & \quad \{G_2, G_3\} & \quad \{G_3\} & \quad \ldots
\end{align*}
\]
Example: average degree of each graph in \( \{G_1, G_2\} \)

Graph Query Execution

- average
- union
- \((1,2,\{G_1, G_2\})\)
- count, sum
- \((a,2,\{G_1, G_2\})\)
- degree
- \((a, \diamond, \{G_1, G_2\})\)
- vertex

\{G_1, G_2, G_3\}

- count, sum
- degree

\{G_1, G_2, G_3\}

- count, sum
- degree

\{G_1\}

- vertex

\{G_1\}

- vertex

\{G_2, G_3\}

- vertex

\{G_3\}

- vertex

\{G_1, G_2\}

- count, sum
- degree

\{G_1\}

- vertex

\{G_2\}

- vertex

\{G_3\}

......
Graph Query Execution

Example: average degree of each graph in \{G_1, G_2\}

---

**Graph Query Execution**

- **average**
- **union**
  - \( (1, 2, \{G_1, G_2\}) \)
- **count, sum**
  - \( (a, 2, \{G_1, G_2\}) \)
- **degree**
  - \( (a, \diamond, \{G_1, G_2\}) \)
- **vertex**
  - \( \{G_1, G_2, G_3\} \)

---

**Example:**

- vertex degree count, sum
  - \( \{G_1, G_2, G_3\} \)

---

"NEDB Day 2012"

---

Saturday, July 6, 2013
Example: average degree of each graph in \( \{G_1, G_2\} \)

**Graph Query Execution**

```
average
union
(1,2:\{G_1, G_2\})
count, sum
(a,2:\{G_1, G_2\})
degree
(a, \diamondsuit,\{G_1, G_2\})
vertex

\{G_1, G_2, G_3\}
```

```
\{G_1\}

\{G_1, G_2\}
```

```
\{G_1, G_2\}
\{G_3\}
```

```
\{G_3\}
```

......
Example: average degree of each graph in \{G_1, G_2\}

**Graph Query Execution**

\( (1, 2, \{G_1, G_2\}) \)

\( (a, 2, \{G_1, G_2\}) \)

\( (a, \diamond, \{G_1, G_2\}) \)

\( (b, 1, \{G_1, G_2\}) \)

\( (b, \diamond, \{G_1, G_2\}) \)

\( (1, 1, \{G_1, G_2\}) \)

\( (a, \text{count, sum}) \)

\( (a, \text{degree}) \)

\( (a, \text{vertex}) \)

\( (b, \text{count, sum}) \)

\( (b, \text{degree}) \)

\( (b, \text{vertex}) \)

\( (c, \text{count, sum}) \)

\( (c, \text{degree}) \)

\( (c, \text{vertex}) \)

\( (d, \text{count, sum}) \)

\( (d, \text{degree}) \)

\( (d, \text{vertex}) \)

\( (e, \text{count, sum}) \)

\( (e, \text{degree}) \)

\( (e, \text{vertex}) \)

\( (f, \text{count, sum}) \)

\( (f, \text{degree}) \)

\( (f, \text{vertex}) \)

\( \{G_1, G_2, G_3\} \)

\( \{G_1, G_2, G_3\} \)

\( \{G_1\} \)

\( \{G_1, G_2\} \)

\( \{G_2, G_3\} \)

\( \{G_3\} \)

\( \ldots \)

**NEDB Day 2012**
Example: average degree of each graph in \{G_1,G_2\}

Graph Query Execution

![Graph Query Execution Diagram]
Example: average degree of each graph in \( \{G_1, G_2\} \)

Graph Query Execution

**Diagram:**
- **Average**
- **Union**
- **Count, Sum**
- **Degree**
- **Vertex**

**Graphs:**
- \( G_1 \)
- \( G_2 \)
- \( G_3 \)

**Nodes and Edges:**
- \( a, b, c, d, e, f \)
- \( a \rightarrow c, b \rightarrow d \)

**Sets:**
- \( \{G_1, G_2, G_3\} \)
- \( \{G_1\} \)
- \( \{G_1, G_2\} \)
- \( \{G_2, G_3\} \)
- \( \{G_3\} \)

**Counts and Sums:**
- \( (1, 2, \{G_1, G_2\}) \)
- \( (1, 1, \{G_1, G_2\}) \)
- \( (c, 0, \{G_1\}), (d, 0, \{G_1, G_2\}), (c, 1, \{G_2\}), (e, 0, \{G_2\}) \)

**Example:**
- Calculating the average degree of each graph in \( \{G_1, G_2\} \)
- **Sets:**
  - **Union:** \( \{G_1, G_2, G_3\} \)
  - **Graphs:** \( G_1, G_2, G_3 \)
  - **Nodes:** \( a, b, c, d, e, f \)
  - **Edges:** \( a \rightarrow c, b \rightarrow d \)

**Saturday, July 6, 2013**
Example: average degree of each graph in \{G_1, G_2\}

**Graph Query Execution**

![Graph Execution Diagram]

- **Average**
- **Union**
- **Count, Sum**
- **Degree**
- **Vertex**

**Example:** average degree of each graph in \{G_1, G_2\}
Graph Query Execution

Example: average degree of each graph in \{G_1, G_2\}

NEDB Day 2012

Saturday, July 6, 2013
Example: average degree of each graph in \{G_1,G_2\}

**Graph Query Execution**

(3/4,\{G_1\}), (4/5,\{G_2\})

average

(1,2,\{G_1,G_2\}), (1,1,\{G_1,G_2\}), (2,0,\{G_1\}), (3,1,\{G_2\})

union

(1,2,\{G_1,G_2\})

count, sum

(a,2,\{G_1,G_2\})

degree

(a,\diamond,\{G_1,G_2\})

vertex

......

(2,0,\{G_1\}), (3,1,\{G_2\})

count, sum

(c,0,\{G_1\}), (d,0,\{G_1,G_2\}), (c,1,\{G_2\}), (e,0,\{G_2\}),

degree

(b,1,\{G_1,G_2\})

degree

(b,\diamond,\{G_1,G_2\})

vertex

......

(S1, G_1, G_2, G_3)

(S2, G_1, G_2, G_3)

(S3, G_1, G_2, G_3)

......

NEDB Day 2012

Saturday, July 6, 2013
Example: average degree of each graph in \{G_1, G_2\}
G* vs Phoebus (an open-source version of Pregel)

- **Data**
  - 10 complete binary trees
  - i’th tree = (i-1)’th tree
    + new 25K vertices (and edges)

- **Query**
  - single-source shortest paths

---

**Query time (sec)**

<table>
<thead>
<tr>
<th>number of graphs</th>
<th>Phoebus (all graphs)</th>
<th>Phoebus (last graph)</th>
<th>G* (all graphs)</th>
<th>G* (last graph)</th>
</tr>
</thead>
</table>
G* vs Phoebus
(an open-source version of Pregel)

- **Data**
  - 10 complete binary trees
  - i’th tree = (i-1)’th tree
    + new 25K vertices (and edges)

- **Query**
  - single-source shortest paths
G* vs Phoebus

(an open-source version of Pregel)

- **Data**
  - 10 complete binary trees
  - i’th tree = (i-1)’th tree
    + new 25K vertices (and edges)

- **Query**
  - single-source shortest paths

---

**Phoebus (all graphs)**

**Phoebus (last graph)**

**G* (all graphs)**

**G* (last graph)**

---

**query time (sec)**

**number of graphs**
G* vs Phoebus

- **Data**
  - 10 complete binary trees
  - i’th tree = (i-1)’th tree
    + new 25K vertices (and edges)

- **Query**
  - single-source shortest paths

(an open-source version of Pregel)
G* vs Phoebus

(Data)
- 10 complete binary trees
- i’th tree = (i-1)’th tree + new 25K vertices (and edges)

(Query)
- single-source shortest paths

(an open-source version of Pregel)
Scale-up Test

• Platform
  - 64 core server cluster

• Data
  - 1 complete binary tree
  - 1 billion vertices (edges)
  - 64 million vertices / server

• Queries
  - vertex degree
  - single-source shortest paths

---

NEDB Day 2012

Saturday, July 6, 2013
Scale-up Test

- **Platform**
  - 64 core server cluster

- **Data**
  - 1 complete binary tree
  - 1 billion vertices (edges)
  - 64 million vertices / server

- **Queries**
  - vertex degree
  - single-source shortest paths

![Graph showing query time vs. number of cores for degree and SSSP queries.]
Scale-up Test

- **Platform**
  - 64 core server cluster

- **Data**
  - 1 complete binary tree
  - 1 billion vertices (edges)
  - 64 million vertices / server

- **Queries**
  - vertex degree
  - single-source shortest paths

---

**Graph**

- x-axis: number of cores (relative data size)
- y-axis: query time (sec)
- Lines: degree (blue), SSSP (red)

---

*NEDB Day 2012*
Conclusion

• Convenient and efficient queries on graphs
  - distributed, deduplicate storage of graphs
  - parallel query processing that share computations across graphs

• Supported by NSF CAREER award IIS-1149372

• 30K lines of Java code

• 1st code release in August 2012

• Future work
  - disk optimization / graph distribution / query optimization / high availability
Thank you!