Graphflow

A system for active graph processing

Chathura Kankanamge
Siddhartha Sahu
The Twitter Connections Graph
Twitter Follow Suggestions
Twitter Follow Suggestions

A is suggested to B.
Twitter Follow Suggestions

C is suggested to D.
Networked Application Graph

LDAP

Financial Database

Email Service

Student Database
Failure Detection

LDAP

Email Service

Financial Database

Student Database
Problem Statement

Run **continuous queries** on a **large graph database** with **rapidly changing** data and perform user defined actions based on the results.

Examples:

1. New follow suggestions.
2. Cancel transactions which create a circular pattern and email account manager.
3. Send emails to application admins upon server failure.
Previous Solutions

Polling queries

Application-specific optimizations
  Eg. Graphjet at Twitter
Generic Join Algorithm

Worst-case optimal join algorithm

$O(n^{3/2})$ space complexity
MATCH QUERIES

Graph:

Query:
MATCH QUERIES

Graph:

Query:

Goal: find all motifs that match abc

{5,6,4} and {2,4,3}
**Naive Join**

R1
  a->b

R2
  b->c

- {1, 2}
- {1, 5}
- {2, 5}
- {2, 4}
- {3, 2}
- {4, 3}
- {4, 5}
- {5, 6}
- {6, 4}
- {7, 4}

Space complexity $O(n^2)$, Time complexity $O(n^2)$
**Generic Join Algorithm**

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a→b</td>
<td>b→c</td>
<td>c→a</td>
</tr>
<tr>
<td>{1, 2}</td>
<td>{1, 2}</td>
<td>{1, 2}</td>
</tr>
<tr>
<td>{1, 5}</td>
<td>{1, 5}</td>
<td>{1, 5}</td>
</tr>
<tr>
<td>{2, 5}</td>
<td>{2, 5}</td>
<td>{2, 5}</td>
</tr>
<tr>
<td>{2, 4}</td>
<td>{2, 4}</td>
<td>{2, 4}</td>
</tr>
<tr>
<td>{3, 2}</td>
<td>{3, 2}</td>
<td>{3, 2}</td>
</tr>
<tr>
<td>{4, 3}</td>
<td>{4, 3}</td>
<td>{4, 3}</td>
</tr>
<tr>
<td>{4, 5}</td>
<td>{4, 5}</td>
<td>{4, 5}</td>
</tr>
<tr>
<td>{5, 6}</td>
<td>{5, 6}</td>
<td>{5, 6}</td>
</tr>
<tr>
<td>{6, 4}</td>
<td>{6, 4}</td>
<td>{6, 4}</td>
</tr>
<tr>
<td>{7, 4}</td>
<td>{7, 4}</td>
<td>{7, 4}</td>
</tr>
</tbody>
</table>
**Generic Join Algorithm**

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a-&gt;b</td>
<td>b-&gt;c</td>
<td>c-&gt;a</td>
</tr>
<tr>
<td>{1,2}</td>
<td>{1,2}</td>
<td>{1,2}</td>
</tr>
<tr>
<td>{1,5}</td>
<td>{1,5}</td>
<td>{1,5}</td>
</tr>
<tr>
<td>{2,5}</td>
<td>{2,5}</td>
<td>{2,5}</td>
</tr>
<tr>
<td>{2,4}</td>
<td>{2,4}</td>
<td>{2,4}</td>
</tr>
<tr>
<td>{3,2}</td>
<td>{3,2}</td>
<td>{3,2}</td>
</tr>
<tr>
<td>{4,3}</td>
<td>{4,3}</td>
<td>{4,3}</td>
</tr>
<tr>
<td>{4,5}</td>
<td>{4,5}</td>
<td>{4,5}</td>
</tr>
<tr>
<td>{5,6}</td>
<td>{5,6}</td>
<td>{5,6}</td>
</tr>
<tr>
<td>{6,4}</td>
<td>{6,4}</td>
<td>{6,4}</td>
</tr>
<tr>
<td>{7,4}</td>
<td>{7,4}</td>
<td>{7,4}</td>
</tr>
</tbody>
</table>

The diagram shows the relationship between the entities a, b, and c, with arrows indicating the join operations:
- R1: a->b
- R2: b->c
- R3: c->a

The additional sets shown are:
- {2, _}
- {3, _}
- {4, _}
- {5, _}
- {6, _}
## Generic Join Algorithm

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a-&gt;b</td>
<td>b-&gt;c</td>
<td>c-&gt;a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{2,5}</td>
<td>{1,2}</td>
<td>{1,2}</td>
</tr>
<tr>
<td>{4,5}</td>
<td>{2,5}</td>
<td>{2,5}</td>
</tr>
<tr>
<td>{2,4}</td>
<td>{2,4}</td>
<td>{2,4}</td>
</tr>
<tr>
<td>{6,4}</td>
<td>{3,2}</td>
<td>{3,2}</td>
</tr>
<tr>
<td>{3,2}</td>
<td>{4,3}</td>
<td>{4,3}</td>
</tr>
<tr>
<td>{4,3}</td>
<td>{4,5}</td>
<td>{4,5}</td>
</tr>
<tr>
<td>{5,6}</td>
<td>{5,6}</td>
<td>{5,6}</td>
</tr>
<tr>
<td></td>
<td>{6,4}</td>
<td>{6,4}</td>
</tr>
<tr>
<td></td>
<td>{7,4}</td>
<td>{7,4}</td>
</tr>
</tbody>
</table>

- R1: a->b
- R2: b->c
- R3: c->a

- R1: \{2,5\} \{4,5\} \{2,4\} \{6,4\} \{3,2\} \{4,3\} \{5,6\}
- R2: \{1,2\} \{1,5\} \{2,5\} \{2,4\} \{3,2\} \{4,3\} \{4,5\} \{5,6\}
- R3: \{1,2\} \{1,5\} \{2,5\} \{2,4\} \{3,2\} \{4,3\} \{4,5\} \{5,6\}
## Generic Join Algorithm

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{2, 5}</td>
<td>{1, 2}</td>
</tr>
<tr>
<td>a-&gt;b</td>
<td></td>
<td>{2, 4}</td>
</tr>
<tr>
<td>{2, 5}</td>
<td>{6, 4}</td>
<td>{2, 5}</td>
</tr>
<tr>
<td>{4, 5}</td>
<td></td>
<td>{3, 2}</td>
</tr>
<tr>
<td>{2, 4}</td>
<td>{6, 4}</td>
<td>{2, 5}</td>
</tr>
<tr>
<td>{6, 4}</td>
<td></td>
<td>{4, 3}</td>
</tr>
<tr>
<td>{3, 2}</td>
<td>{4, 5}</td>
<td>{3, 2}</td>
</tr>
<tr>
<td>{4, 3}</td>
<td></td>
<td>{5, 6}</td>
</tr>
<tr>
<td>{5, 6}</td>
<td>{5, 6}</td>
<td>{5, 6}</td>
</tr>
<tr>
<td></td>
<td>{6, 4}</td>
<td>{6, 4}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{7, 4}</td>
</tr>
</tbody>
</table>

{3, 2, 4}
{5, 6, 4}
{4, 3, 2}
{6, 4, 5}
{2, 4, 3}
{4, 5, 6}
GraphFlow

A new graph database implemented from scratch.

One-time MATCH queries
   Uses Generic Join algorithm

Continuous MATCH queries
   Uses Delta Generic Join algorithm
GraphFlow Architecture

- Clients
- Query Processor
- Cypher++ Query Parser

Diagram:
- Clients communicate with Query Processor via gRPC.
- Query string is passed to Cypher++ Query Parser.
- Cypher++ Query Parser uses ANTLR.
Cypher++

CREATE (1)->(2), (2)->(3), (2)->(4), (4)->(8);
DELETE (2)->(3);
MATCH (a)->(b),(a)->(c),(b)->(d),(c)->(d);
CONTINUOUS MATCH (a)->(b),(b)->(c),(c)->(a) FILE triangle_query;
GraphFlow Architecture

Clients → gRPC → Query Processor

Query Processor → query string → Cypher++ Query Parser

Cypher++ Query Parser → parsed query → Query Planner

Query Planner → parsed query → Continuous Query Executor

Continuous Query Executor → Emerging or disappearing motifs

Continuous Query Executor → Graph Update Query Executor

Graph Update Query Executor → Graph Store

Graph Store → in-memory batching

Query Executor → Generic Join Query Executor

Generic Join Query Executor → in-memory
Future Work

- Engineering
  - Add support for vertex and edge labels.
  - Plan viewer
  - UI to visualize continuous query results.

- Performance evaluation and comparisons with:
  - Neo4j
  - Incremental view maintenance.

- Transactions support
  - Abort transactions based on output motifs.

- Streaming support
  - CQL-like windowing semantics
Future Work

- **Engineering**
  - Add support for vertex and edge labels.
  - Plan viewer
  - UI to visualize continuous query results.

- **Performance evaluation and comparisons with:**
  - Neo4j
  - Incremental view maintenance.

- **Transactions support**
  - Abort transactions based on output motifs.

- **Streaming support**
  - CQL-like windowing semantics

Thank you.