Distributed Graph Analytics with Gradoop

Martin Junghanns
University of Leipzig – Database Research Group
„An open-source graph dataflow system for declarative analytics of heterogeneous graph data.“
<table>
<thead>
<tr>
<th>Gradoop</th>
<th>Extended Property Graphs</th>
<th>Evaluation</th>
<th>Summary</th>
</tr>
</thead>
</table>

**Summary**

- **Graph Dataflow Operators**
- **Extended Property Graph Model (EPGM)**
- **I/O**
- **Apache Flink Operator Implementation**
- **Distributed Operator Execution (Apache Flink)**
- **Distributed Graph Storage (Apache HDFS)**
Extended Property Graphs

- Vertices and directed Edges
- Logical Graphs
- Identifiers
- Type Labels
- Properties

Extended Property Graph Model (EPGM)
## Extended Property Graphs

### EPGM Operators/Transformations

<table>
<thead>
<tr>
<th>Logical Graph</th>
<th>Unary</th>
<th>Binary</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aggregation</td>
<td>Combination</td>
<td>Flink Gelly Library</td>
</tr>
<tr>
<td></td>
<td>Pattern Matching</td>
<td>Overlap</td>
<td>BTG Extraction</td>
</tr>
<tr>
<td></td>
<td>Transformation</td>
<td>Exclusion</td>
<td>Adaptive Partitioning</td>
</tr>
<tr>
<td></td>
<td>Grouping</td>
<td>Equality</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Subgraph</td>
<td>Fusion</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Call</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selection</td>
<td>Union</td>
<td>Frequent Subgraph Mining</td>
</tr>
<tr>
<td></td>
<td>Pattern Matching</td>
<td>Intersection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distinct</td>
<td>Difference</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Limit</td>
<td>Equality</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reduce</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Call</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graph Collection</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Basic Binary Operators**

**Combination**

\[ \text{LogicalGraph } graph3 = \text{graph1.combine(graph2)}; \]

**Overlap**

\[ \text{LogicalGraph } graph4 = \text{graph1.overlap(graph2)}; \]

**Exclusion**

\[ \text{LogicalGraph } graph5 = \text{graph1.exclude(graph2)}; \]
Graph4 = graph3.subgraph(
  (vertex => vertex.getLabel().equals('Green')),
  (edge => edge.getLabel().equals('orange')));

LogicalGraph

Subgraph
Aggregation

graph3 = graph3.aggregate(new VertexCount());
Gradoop

Extended Property Graphs

Evaluation

Summary

Grouping

LogicalGraph grouped = graph3.groupBy()
    .useVertexLabel()
    .useEdgeLabel()
    .addVertexAggregate(new CountAggregator())
    .addEdgeAggregate(new MaxAggregator('a'));
Pattern Matching (Single Graph Input)

```
GraphCollection collection = graph3.match('(:Green)-[:orange]->(:Orange)');
```
Pattern Matching (Graph-Transaction Setting)

GraphCollection filtered = coll.match('v0:Green)-[:blue]->(:Orange),(v0)-[:blue]->(:Orange');
Call (e.g. Clustering)

```java
GraphCollection clustering = graph.callForCollection(new ClusteringAlgorithm());
```
1. Extract **subgraph** containing only *Persons* and *knows* relations

2. **Transform** *Persons* to necessary information

3. Find communities using **Label Propagation**

4. **Aggregate** vertex count for each community

5. **Select** communities with more than 50K users

6. **Combine** large communities to a single graph

7. **Group** graph by *Persons* *location* and *gender*

8. **Aggregate** vertex and edge count of grouped graph

http://ldbcouncil.org/
Overall performance [3]

1. **Extract** subgraph containing only *Persons* and *knows* relations
2. **Transform** *Persons* to necessary information
3. **Find** communities using **Label Propagation**
4. **Aggregate** vertex count for each community
5. **Select** communities with more than 50K users
6. **Combine** large communities to a single graph
7. **Group** graph by Persons *location* and *gender*
8. **Aggregate** vertex and edge count of grouped graph

https://git.io/vD2Ii
Overall performance [3]

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Vertices</th>
<th># Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphalytics.1</td>
<td>61,613</td>
<td>2,026,082</td>
</tr>
<tr>
<td>Graphalytics.10</td>
<td>260,613</td>
<td>16,600,778</td>
</tr>
<tr>
<td>Graphalytics.100</td>
<td>1,695,613</td>
<td>147,437,275</td>
</tr>
<tr>
<td>Graphalytics.1000</td>
<td>12,775,613</td>
<td>1,363,747,260</td>
</tr>
<tr>
<td>Graphalytics.10000</td>
<td>90,025,613</td>
<td>10,872,109,028</td>
</tr>
</tbody>
</table>

- 16x Intel(R) Xeon(R) 2.50GHz 6 (12)
- 16x 48 GB RAM
- 1 Gigabit Ethernet
- Hadoop 2.6.0
- Flink 1.0-SNAPSHOT
  - slots (per worker) 12
  - jobmanager.heap.mb 2048
  - taskmanager.heap.mb 40960
**Summary**

- **Dataset**
  - Graphalytics.1: 61,613 vertices, 2,026,082 edges
  - Graphalytics.10: 260,613 vertices, 16,600,778 edges
  - Graphalytics.100: 1,695,613 vertices, 147,437,275 edges
  - Graphalytics.1000: 12,775,613 vertices, 1,363,747,260 edges
  - Graphalytics.10000: 90,025,613 vertices, 10,872,109,028 edges

- **Evaluation Environment**
  - 16x Intel(R) Xeon(R) 2.50GHz 6 (12)
  - 16x 48 GB RAM
  - 1 Gigabit Ethernet
  - Hadoop 2.6.0
  - Flink 1.0-SNAPSHOT
    - slots (per worker): 12
    - jobmanager.heap.mb: 2048
    - taskmanager.heap.mb: 40960

**Evaluation**

- Graphalytics.100

**Overall performance [3]**

- **Graph-analytics.100**

- **Runtime [s]**
  - 1 worker: 1200
  - 2 workers: 600
  - 4 workers: 300
  - 8 workers: 150
  - 16 workers: 75

- **Speedup**
  - Linear
  - Number of workers
  - 1 worker: 1
  - 2 workers: 2
  - 4 workers: 4
  - 8 workers: 8
  - 16 workers: 16

Distributed Graph Analytics with Gradoop – 9th LDBC TUC Meeting – Martin Junghanns
### Evaluation

#### Overall performance [3]

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Vertices</th>
<th># Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphalytics.1</td>
<td>61,613</td>
<td>2,026,082</td>
</tr>
<tr>
<td>Graphalytics.10</td>
<td>260,613</td>
<td>16,600,778</td>
</tr>
<tr>
<td>Graphalytics.100</td>
<td>1,695,613</td>
<td>147,437,275</td>
</tr>
<tr>
<td>Graphalytics.1000</td>
<td>12,775,613</td>
<td>1,363,747,260</td>
</tr>
<tr>
<td>Graphalytics.10000</td>
<td>90,025,613</td>
<td>10,872,109,028</td>
</tr>
</tbody>
</table>

- 16x Intel(R) Xeon(R) 2.50GHz 6 (12)
- 16x 48 GB RAM
- 1 Gigabit Ethernet
- Hadoop 2.6.0
- Flink 1.0-SNAPSHOT
  - slots (per worker) 12
  - jobmanager.heap.mb 2048
  - taskmanager.heap.mb 40960
Grouping [1]

(a) Runtime on Graphalytics 1000
(b) Runtime on Pokec
(c) Speedup for configuration 1 (RG)

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Vertices</th>
<th># Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphalytics.1</td>
<td>61,613</td>
<td>2,026,082</td>
</tr>
<tr>
<td>Graphalytics.10</td>
<td>260,613</td>
<td>16,600,778</td>
</tr>
<tr>
<td>Graphalytics.100</td>
<td>1,695,613</td>
<td>147,437,275</td>
</tr>
<tr>
<td>Graphalytics.1000</td>
<td>12,775,613</td>
<td>1,363,747,260</td>
</tr>
<tr>
<td>Pokec</td>
<td>1,632,803</td>
<td>30,622,564</td>
</tr>
</tbody>
</table>

- 16x Intel(R) Xeon(R) 2.50GHz 6 (12)
- 16x 48 GB RAM
- 1 Gigabit Ethernet
- Hadoop 2.6.0
- **Flink 1.0.3**
  - slots (per worker) 12
  - jobmanager.heap.mb 2048
  - taskmanager.heap.mb 40960
• Extended Property Graph Model
  • Schema flexible: Type Labels and Properties
  • Logical Graphs / Graphs Collection

• Graph and Collection Operators
  • Combination to analytical workflows

• Implemented on Apache Flink
  • Built-in scalability
  • Combine with other libraries


www.gradoop.com