Efficient Complete Event Trend Detection over High-Velocity Streams

Huiyao Mei\textsuperscript{1}, Hanhua Chen\textsuperscript{1}, Hai Jin\textsuperscript{1}, Qiang-shen Hua\textsuperscript{1}, Bing Bing Zhou\textsuperscript{2}

\textsuperscript{1}Huazhong University of Science and Technology
\textsuperscript{2}The University of Sydney
Complete Event Trend Detection

- E-commerce
- Financial Transaction
- Stock Trading
- Cluster Monitoring

- Anti-terrorism Monitoring
- Campus loan
- Price rise or fall
- Load imbalance

• According to the user-defined event pattern, detecting all matching event sequences from high-speed event streams,
Complete Event Trend Detection

Q1:  
\[ \text{PATTERN} \quad \text{check + c[i]} \]
\\* WHERE 
\[ c\.type = 'not covered' \text{ AND c[i].src = c[i - 1].dest} \]
\\* WITHIN 
\[ 1 \text{ week SLIDE 1 day} \]

Q2:  
\[ \text{PATTERN} \quad \text{stock + s[i]} \]
\\* WHERE 
\[ [id] \text{ AND s[i].price > ratio * s[i - 1].price} \]
\\* WITHIN 
\[ 60 \text{ min SLIDE 10 min} \]
Challenges

- Massive Partial Results
  - Partial results must be stored in memory to match with ongoing events
  - One partial result can match any number of events
- High Performances Requirement
- Difficult to be parallized
State-of-the-art Works

• Detection - CET Graph[1]
  • High graph construction latency
  • Impractical graph partition method

• Parallelization Solution - Coarse-grained
  • pipeline-based [2]
  • Key-based[3]

[1]: Olga Poppe, et al. Complete Event Trend Detection in High-Rate Event Streams. SIGMOD’2017
[3]: flink CEP. https://flink.apache.org/
CET Graph — Construction

- CET Graph: share common event sequences by paths
- Traverse to find matches
- $O(|V|^2)$ time and space costs
CET Graph - Extraction

- Only connect by last and first vertices
- Imbalanced graphlets

\[ |\text{partition}| < \text{avg} + |g_1| \]
- Exponential time complexity
ABI Graph Model

- Event Vertex $\rightarrow$ Attribute Vertex (to-edge): the event is relevant to the attribute value
- Attribute Vertex $\rightarrow$ Event Vertex (from-edge): the attribute value of the event
Dynamic Graph Construction

- Composite Attribute Vertex
  - respond to many attribute values
  - most of attribute values are unnecessary
- A Stipulation: values of a composite attribute vertex must only:
  - be all relevant to an event
  - or all irrelevant to an event
- Violate (Composite) Attribute Vertex is split

\[ e[i].\text{value} > 2 \times e[i - 1].\text{value} \]

<table>
<thead>
<tr>
<th>event</th>
<th>( e_1 )</th>
<th>( e_2 )</th>
<th>( e_3 )</th>
<th>( e_4 )</th>
<th>( e_5 )</th>
<th>( e_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>32</td>
<td>7</td>
<td>15</td>
<td>35</td>
<td>40</td>
<td>17</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{av}_1:0-65 & \quad 32 \\
\text{av}_2:65-100 & \quad e_1 \\
\text{av}_1:0-15 & \quad 7 \\
\text{av}_2:15-65 & \quad 32 \\
\text{av}_3:65-100 & \quad e_1 \\
\end{align*}
\]
Dynamic Graph Construction

- **from-edge**: assigned by the value of its source attribute vertex
- **to-edge**: copied and pointed to all child attribute vertex

\[
e[i].value > 2 \times e[i-1].value
\]

<table>
<thead>
<tr>
<th>event</th>
<th>(e_1)</th>
<th>(e_2)</th>
<th>(e_3)</th>
<th>(e_4)</th>
<th>(e_5)</th>
<th>(e_6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>32</td>
<td>7</td>
<td>15</td>
<td>35</td>
<td>40</td>
<td>17</td>
</tr>
</tbody>
</table>
Parallel Graph Construction

- Read-Write Conflict
  - Write: split violated attribute vertices
  - Read: search attribute vertices
- Parallel Solution – postponed construction
  - the split point only relate to event and predicate
  - construct the graph after all split points are calculated

![Graph Diagram]
CET Path In ABI Graph

- two condition for relevance of two events:
  - their corresponding event vertices are connected by an attribute vertex
  - following the direction of edges, the timestamp of the later event is greater than that of the previous
- start (end) event vertex: the start or end event vertex of CET paths
Anchor-based Extraction – BFS stage

- anchors: evenly selected attribute vertices
- start anchor: points to all start event vertices
- end anchor: pointed by all end event vertices
- BFS stage: between two anchors
Anchor-based Extraction – DFS stage

\[
\text{sub}_{s2} & (e_1) \\
\text{sub}_{s4} & (e_1, e_1, e_3) \\
\text{sub}_{se} & (e_1, e_3, e_5) \\
\text{sub}_{24} & (e_2, e_2, e_3) \\
\text{sub}_{2e} & (e_2, e_5) \\
\text{sub}_{4e} & (e_4, e_5, e_6)
\]
Broadcast Join-based Extraction

- General Graph Processing
  - large scale graph, not large scale message
  - transfer by general join
- CET Extraction
  - small scale graph, large scale message
  - transfer by broadcast join
Experiment Setup

<table>
<thead>
<tr>
<th></th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Xeon(Cascade Lake) Platinum 8269CY 16cores</td>
</tr>
<tr>
<td>Memory</td>
<td>64GB</td>
</tr>
<tr>
<td>Disk</td>
<td>SATA 40GB</td>
</tr>
<tr>
<td>Operating System</td>
<td>CentOS 8.0</td>
</tr>
<tr>
<td>Machines</td>
<td>6</td>
</tr>
<tr>
<td>Spark Version</td>
<td>3.0.1</td>
</tr>
</tbody>
</table>

- Dataset 1: crawled stock data and copy for 5000 times
  - 17M events
  - timestamp, price
- Dataset 2: generated check dataset
  - timestamp, source bank account, destination bank account
Graph Construction

![Graph Construction Diagram]

- **p8**
- **cet**
- **seq**

**Diagram Details:**
- **Y-axis:** Time (ms) on a logarithmic scale.
- **X-axis:** Number of events.

**Legend:**
- **p8**
- **cet**
- **seq**

50th International Conference on Parallel Processing (ICPP) August 9-12, 2021 in Virtual Chicago, IL
Graph Construction
CET Extraction

\[ 
\begin{array}{c}
\text{number of events} \\
0 & 100k & 200k & 300k & 400k & 500k \\
10^2 & 10^3 & 10^4 & 10^5 & 10^6 & 10^7 \\
\hline
\text{time (ms)} \\
\end{array} 
\]
CET Extraction

![Graph showing the relationship between time (ms) and number of events for different processor counts (p2, p4, p8, p16). The graph illustrates the increase in time with the number of events, with different processor counts showing varying slopes and points.]
CET Extraction
Thank you for your attention!

Source code available at: https://github.com/CGCL-codes/DynamicTG