Position Paper: Bitemporal Graph Analytics

Hassan Halawa, Matei Ripeanu
University of British Columbia

Hassan Halawa
NetSysLab
<hhalawa@ece.ubc.ca>
<http://netsyslab.ece.ubc.ca>

GRADES-NDA’21
20 June 2021
Context

- **Model** the real-world
  - Computations done by the graph analytics system on the data
- **Bidirectional channel**
  - Update model based on real-world events
  - Decisions made affect the real-world!
- **Arbitrary event delays**
  - No delay/ordering guarantees!
- **Evolution** tracked through time-stamped events
  - Graph is *dynamic*!
  - Streaming Systems
  - Historical Systems
- **Explicit Temporal Modeling**
  - *Uni*-temporal: Valid time
  - *Bi*-temporal: Valid time vs. Transaction time
Use Cases & Key Shared Application Properties

Real-World Use Cases
- Infrastructure Modeling and Planning
- Regulatory Compliance in the Crypto-Currency Market
- Financial Services

Key Shared Properties
- Time Evolving
- Interactive Use
- Explicit Temporal Modeling
- Arbitrary Event Delays
- Complex Query Types

Our Space

Typical Assumptions
- Static Graphs
- Events Propagate Immediately
- High Event Velocity
- Long Tail Event Delays
- Inconsistent States Viewed by Operators
Query Types (Point-in-time)

- **Current state**
  - What is the state of the graph *now*, given everything the system knows up to *now*?

- **Historical**
  - What was the state of the graph at some historical point, given everything the system knows up to *now*?

- **Audit**
  - What was the state of the graph at some historical point, given everything the system knew at a later historical point?
Example: Uni-Temporal Modeling Scenarios

**Real-World Domain**

**Initial Graph State**

A → B
C → D

**Online Graph Analytics**

**Event Queue**

<table>
<thead>
<tr>
<th>Head</th>
<th>Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV = 1</td>
<td>TV = 2</td>
</tr>
<tr>
<td>✔️ BD</td>
<td>✗ AB</td>
</tr>
</tbody>
</table>

**Event Stream**

**In Order**

**Out of Order**

**Decisions**

**Events**

**Incorrect Last State**

**Missing State**

**Missing State**

**Correct Last State**

TV = 3

TV = 2

TV = 1
Example: Bi-Temporal Modeling Scenario

Initial Graph State

A

C

D

Event Queue

Head

Tail

\( T_T = 3 \)

\( T_T = 4 \)

\( T_T = 5 \)

\( T_V = 3 \)

\( T_V = 2 \)

\( T_V = 1 \)

\( BD \)

\( AB \)

\( BD \)

ingested @

\( T_T = 3 \)

\( T_T = 4 \)

\( T_T = 5 \)

Event Stream

Historical

Current State

Audit

Out of Order

Initial Graph State

A

B

C

D

Event Queue

Head

Tail

\( T_T = 3 \)

\( T_T = 4 \)

\( T_T = 5 \)

\( T_V = 3 \)

\( T_V = 2 \)

\( T_V = 1 \)

\( BD \)

\( AB \)

\( BD \)

ingested @

\( T_T = 3 \)

\( T_T = 4 \)

\( T_T = 5 \)

Event Stream

Historical

Current State

Audit
All that is needed: two low-level data structures

- Model the evolution of a *single* entity
  - (e.g., a specific attribute of an existing vertex/edge)
- Model the evolution of a *set* of entities
  - (e.g., the vertices of a graph, or the neighbour set of a vertex)
- Combine the two to model the evolution of a *graph*!
Challenges & Our Solutions

- **Correctness**
  - Out of order event ingestion

- **Ingestion performance**
  - High dynamicity and velocity

- **Query performance**
  - Diverse query types

- **Auditability**

- **Space efficiency**
  - All data kept indefinitely

### Design Choices
- **Bitemporality**
- **Data Structures**
- **Segment Stabbing**

### Persistent Data Structures
- **Immutability**
- **Copy-on-write**
- **Structural Sharing**
Bitemporal Graph Analytics
Q/A

Contact: Hassan Halawa
Research Group: NetSysLab
<hhalawa@ece.ubc.ca>
<http://netsyslab.ece.ubc.ca>
Backup Slides
Related Work

Temporal Model
- Non-Temporal
- Uni-Temporal
- Bi-Temporal

Graph Dynamicity
- Static
- Streaming
- Dynamic

Querying Model
- Offline (Batch)
- Online (Trigger)
- Online (Point-in-Time)
Limitations

■ Design
  □ Locality
    ◦ Persistent Data Structures and Copy-on-Write
  □ Real-World Graphs and Domain Specific Optimizations
    ◦ Sparse / Scale-free Graphs

■ Prototype
  □ Scale-up vs. Scale-out
    ◦ SCMs
  □ Fault-Tolerance / Durability
Ongoing Work

- Proof of concept system implementation
- Ongoing
  - Scale-up: SCM
  - Durability: NVDIMMs / NVM
  - Auditability: Tamper-evidence
  - Domain-specific optimizations
Bitemporal Value

- Enables efficient bi-temporal set/get operations on a single value
  - (i.e., provides simple value semantics).
- Supported operations:
  - Setting a value
  - Getting a value

<table>
<thead>
<tr>
<th>$T_V$</th>
<th>$T_T$</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>E</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>A</td>
</tr>
</tbody>
</table>
Bitemporal Set

- Enables efficient bi-temporal set/get operations on a set of elements
- Supported operations:
  - Adding an element to the set
  - Removing an element from the set
  - Determining membership of an element in the set
  - Iterating over all elements in the set
- **Segment Stabbing** from Computational Geometry!
Bonus: Persistent Data Structures

- Guarantee historical data is never modified
  - Audit requirements
- Properties
  - Immutable
  - Copy-on-write
  - Efficient
- Persistent bitemporal value and bitemporal set data structures
  - Composite bitemporal graph data structure also persistent!
# Complexity Analysis

Notation. \( C \): The total number of commits to the data structure (i.e., the total number of tracked historical/audit states). \( K \): The number of reported results (i.e., the total number of elements that exist at \((T_V, T_T)\)). \( L \): The number of element lifetime intervals (i.e., segments) stored in the dynamic segment stabbing data structure. \( S \): The number of times the bi-temporal value was set.

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Operation</th>
<th>Average-Case Runtime Complexity</th>
<th>Worst-Case Runtime Complexity</th>
<th>Space Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-Temporal Value</td>
<td>setValue((T_V, T_T, \text{Value}))</td>
<td>( O(\log S) )</td>
<td></td>
<td>( O(S \log S) )</td>
</tr>
<tr>
<td></td>
<td>getValue((T_V, T_T))</td>
<td>( O(\log S) )</td>
<td>( O(S) )</td>
<td></td>
</tr>
<tr>
<td>Bi-Temporal Set</td>
<td>addElement((T_V, T_T, \text{Element}))</td>
<td>( O(\log C + \log L) )</td>
<td>( \approx O(\log C) )</td>
<td>( O(C \log C + L \log L) ) ( \approx O(C \log C) )</td>
</tr>
<tr>
<td></td>
<td>removeElement((T_V, T_T, \text{Element}))</td>
<td>( O(\log C + K \log L) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>containsElement((T_V, T_T, \text{Element}))</td>
<td>( O(\log C + K \log L) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>iterateElements((T_V, T_T))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>