Raising the Level of Abstraction for Time State Analytics With the Timeline Framework

Henry Milner, Yihua Cheng, Jibin Zhan, Hui Zhang, Vyas Sekar, Junchen Jiang, Ion Stoica

Conviva, Carnegie Mellon, UChicago, UC Berkeley

10 January 2023
Example Analytics Query Intents

Common feature:
Stateful context-sensitive metrics computed over continuous time

- **Video**
  - How much time did this session spend buffering while using CDN C1?
  - Change CDN

- **Cybersecurity**
  - Which credit card users made purchases at geographically separate locations in the last 5 minutes?
  - Block URL
  - Quarantine host

- **Manufacturing**
  - Which Android users sent a sequence of anomalous DNS requests after visiting website xyz.com in the last hour?
  - Rebalance load, Repair

- **Manufacturing**
  - How many machines from vendor X are showing degrading health status over time?
  - Rebalance load, Repair

...
Streaming Video QoE: Connection-Induced Rebuffering

Stateful
Context-sensitive
Continuous time

How much time did this session spend in a connection-induced rebuffering state while using CDN C1?

Count the duration where:
1. Currently buffering &
2. Play has already initialized &
3. Hasn’t seeked in last 5 seconds &
4. Using CDN C1
How much time did this session spend buffering while using CDN C1?

Which credit card users made purchases at geographically separate locations in the last 5 minutes?

Which Android users sent a sequence of anomalous DNS requests after visiting website xyz.com in the last hour in ATL?

How many machines from vendor X are showing poor health status in Florida?

Compute **stateful, context-sensitive** metrics over **continuous time**, using **measurements at scale**.
TSA isn’t served well by existing data processing systems

- High dev effort
- High cost

Count the duration where:
1. Currently buffering &
2. Play has already initialized &
3. Hasn’t seeked in last 5 seconds &
4. Using CDN C1
Our work: Timeline abstraction for Time-State Analytics

Writing time-state queries becomes intuitive visual operations
→ Reduced dev effort

Enables new opportunities for structure-aware optimizations
→ Up to 10x improvement in cost

Scope of this work: Focus on the single user-session intent modeling problem
Outside our scope: Supporting scale-out and aggregation
Outline for talk

• What is Time-State Analytics

• Time-State Analytics not well supported by status quo

• Introducing the Timeline abstraction

• Early Wins + Next Steps
Where's the problem?

- High dev effort
- High cost

Count the duration where:
1. Currently buffering &
2. Play has already initialized &
3. Hasn’t seeked in last 5 seconds &
4. Using CDN C1
Tabular model isn’t well-suited for Time-State

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Player State</th>
<th>Bitrate</th>
<th>CDN</th>
<th>Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td>Player State</td>
<td>Bitrate</td>
<td>CDN</td>
<td>Seek</td>
</tr>
<tr>
<td>t3</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t4</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t5</td>
<td>C1 B1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t6</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t8</td>
<td>t_1 t_2 t_3</td>
<td>t_4 t_5</td>
<td>t_6 t_7 t_8</td>
<td>t_9 t_10 t_11 t_12 t_13 t_14 t_15</td>
</tr>
<tr>
<td>t9</td>
<td>Paused</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t10</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t11</td>
<td>B3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t12</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t13</td>
<td>C2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t14</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t15</td>
<td>B1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Count the duration where:
1. Currently buffering &
2. Play has already initialized &
3. Hasn’t seeked in last 5 seconds &
4. Using CDN C1
State and Context over Continuous Time is Hard

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Player State</th>
<th>Bitrate</th>
<th>CDN</th>
<th>Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t3</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t4</td>
<td>Play</td>
<td></td>
<td>C1</td>
<td></td>
</tr>
<tr>
<td>t5</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t6</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t7</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t8</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t9</td>
<td>Paused</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t10</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t11</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t12</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t13</td>
<td>Buffer</td>
<td>B3</td>
<td>C2</td>
<td></td>
</tr>
<tr>
<td>t14</td>
<td>Play</td>
<td>B3</td>
<td>C2</td>
<td></td>
</tr>
<tr>
<td>t15</td>
<td>Play</td>
<td>B1</td>
<td>C2</td>
<td></td>
</tr>
</tbody>
</table>

Count the duration where:
1. Currently buffering &
2. Play has already initialized &
3. Hasn’t seeked in last 5 seconds &
4. Using CDN C1

\[ t7 + 5 \text{ seconds} \]
Poor abstraction $\rightarrow$ Complex code

1. Currently buffering &
2. Play has already initialized &
3. Hasn’t seeked in last 5 seconds &
4. Using CDN C1

**Difficult to develop**

**Semantic bugs**

```sql
WITH SeekAsPlayerState(T, P) as (
    SELECT T, P FROM heartbeats WHERE P IS NOT NULL
    UNION SELECT T, "Seek_st" FROM heartbeats WHERE A IS NOT NULL
    UNION SELECT T + 5, "Seek_ed" FROM heartbeats WHERE A IS NOT NULL ),
IgnoreBufBeforePlay(T, P) as (
    SELECT T, P FROM ( 
        SELECT T, P, Max(If(P == 'play', 1, 0)) OVER (PARTITION BY 1 ORDER BY T) as H
        FROM SeekAsPlayerState ) WHERE H == True ),
DuringBufferTable(T, P, DB) as ( 
    SELECT T, P, LAST(tmp1) IGNORE NULLS OVER (PARTITION BY 1 ORDER BY T)
    FROM ( 
        SELECT T, P, CASE P WHEN 'buffer' THEN True WHEN 'Seek_st' THEN NULL WHEN 'Seek_ed' THEN NULL ELSE FALSE END as tmp1
        FROM IgnoreBufBeforePlay ) ,
    DuringSeekTable(T, P, DB, DS) as ( 
        SELECT T, P, DB, (T - Max(If(P == 'Seek_st', T, 0)) OVER (PARTITION BY 1 ORDER BY T)) < 5 as tmp2
    FROM DuringBufferTable ) ,
IgnoreBufSeek(T, P) as ( 
    SELECT T, P FROM ( 
        SELECT T, DS, If(P == 'Seek_ed' and DB, 'buffer', P) as P
        FROM DuringSeekTable ) WHERE NOT (P == 'buffer' AND DS) ,
    WithCDNAndQuery(T, P, C) as ( 
        SELECT T, P, NULL FROM IgnoreBufSeek
    UNION SELECT T, NULL, C FROM heartbeats WHERE C IS NOT NULL
    UNION SELECT 2022-07-21 10:05, NULL, NULL s, 
    Intervals(Ed, St, State, CDN) as ( 
        SELECT T, LEAD(T, 1) OVER (PARTITION BY 1 ORDER BY T), P, C 
        FROM ( 
            SELECT T, LAST(P) IGNORE NULLS OVER (PARTITION BY 1 ORDER BY T) as P,
            LAST(C) IGNORE NULLS OVER (PARTITION BY 1 ORDER BY T) as C
            FROM WithCDNAndQuery ) )
    SELECT SUM(St - Ed) as result FROM Intervals
    WHERE Ed < 2022-07-21 10:05 AND State == 'buffer' AND CDN == 'CDN1'
)
Poor abstraction → High cost

```sql
WITH SeekAsPlayerState(T, P) as (  
    SELECT T, P FROM heartbeats WHERE P IS NOT NULL
    UNION SELECT T, "Seek_st" FROM heartbeats WHERE A IS NOT NULL
    UNION SELECT T + 5, "Seek_ed" FROM heartbeats WHERE A IS NOT NULL ),
  IgnoreBufBeforePlay(T, P) as (  
    SELECT T, P FROM (  
      SELECT T, P, Max(If(P == 'play', 1, 0)) OVER (PARTITION BY 1 ORDER BY T)  
        as H
    ) FROM SeekAsPlayerState) WHERE H == True ,
  DuringBufferTable(T, P, DB) as (  
    SELECT T, P, LAST(tmp1) IGNORE NULLS OVER (PARTITION BY 1 ORDER BY T)  
        as tmp1
  )
  SELECT T, P FROM (  
    CASE P WHEN 'buffer' THEN True WHEN 'Seek_st' THEN NULL WHEN 'Seek_ed'  
      THEN NULL ELSE FALSE END as tmp1
  ) FROM IgnoreBufBeforePlay ) ,
  DuringSeekTable(T, P, DB, DS) as (  
    SELECT T, P, DB, (T - Max(If(P == 'Seek_st', T, 0)) OVER (PARTITION BY 1 ORDER BY T)  
      ) / 5 as tmp2
  )
  SELECT T, P FROM (  
    SELECT T, DS, IF(P == 'Seek_ed' and DB, 'buffer', P) as P
  ) FROM DuringSeekTable ) WHERE NOT (P == 'buffer' AND DS ) ,
  WithCDNAndQuery(T, P, C) as (  
    SELECT T, P, NULL FROM IgnoreBufInSeek  
    UNION SELECT T, NULL, C FROM heartbeats where C IS NOT NULL
    UNION SELECT 2022-07-21 10:05, NULL, NULL s,
    Intervals(Ed, St, State, CDN) as (  
      SELECT T, LEAD(T, 1) OVER (PARTITION BY 1 ORDER BY T), P, C  
        FROM (  
          SELECT T,
            LAST(P) IGNORE NULLS OVER (PARTITION BY 1 ORDER BY T) as P,
            LAST(C) IGNORE NULLS OVER (PARTITION BY 1 ORDER BY T) as C
        ) FROM WithCDNAndQuery )
    SELECT SUM(St - Ed) AS result FROM Intervals
    WHERE Ed < 2022-07-21 10:05 AND State == 'buffer' AND CDN == 'CDN1'
```

Lacks structure:

Difficult for query engines to optimize

High cost

Count the duration where:
1. Currently buffering &
2. Play has already initialized &
3. Hasn’t seeked in last 5 seconds &
4. Using CDN C1
Outline for talk

• What is Time-State Analytics

• Time-State Analytics not well supported by status quo

• *Introducing the Timeline abstraction*

• Early Wins + Next Steps
Stepping back
What’s a Timeline?

“Geometric abstractions are powerful tools” – Fred Brooks

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Player State</th>
<th>Bitrate</th>
<th>CDN</th>
<th>Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td></td>
<td>C1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t3</td>
<td></td>
<td>B1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t4</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t5</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t6</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t7</td>
<td></td>
<td></td>
<td></td>
<td>Seek</td>
</tr>
<tr>
<td>t8</td>
<td></td>
<td>B2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t9</td>
<td>Paused</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t10</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t11</td>
<td></td>
<td>B3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t12</td>
<td>Buffer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t13</td>
<td></td>
<td>C2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t14</td>
<td>Play</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t15</td>
<td></td>
<td>B1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An equivalent geometric view:
Timeline of each measurement

- **Type: Step Function**
  - Column X
  - Type: Event
  - Column Y
  - Type: Continuous Value
  - Column Z

Metric
Whiteboarding Timelines: the Player State over Time

Count the duration where:
1. Currently buffering &
2. Play has already initialized &
3. Hasn’t seeked in last 5 seconds &
4. Using CDN C1

Operator: LatestEventToState (PlayerState)
Whiteboarding Timelines: When has a Seek recently happened?

Count the duration where:
1. Currently buffering &
2. Play has already initialized &
3. **Hasn’t seeked in last 5 seconds** &
4. Using CDN C1

Operator:
TimeSince(Seek) > 5 seconds

Didn’t recently seek

- True
- False

- $t_7$
- $t_7 + 5$
Connection-induced rebuffering

- Currently buffering &
- Play has already initialized &
- Hasn’t seeked in last 5 seconds &
- Using CDN C1

CDN = C1

Connection-induced rebuffering *while CDN = C1*
Count the duration where:

1. Currently buffering &
2. Play has already initialized &
3. Hasn’t seeked in last 5 seconds &
4. Using CDN C1

LatestEventToState(PlayerState) = “Buffer”

HasBeenTrue(PlayerState = “Play”)

TimeSince(Seek) > 5 seconds

LatestEventToState(CDN) = “C1”
Timeline, in a nutshell

Data abstraction with 3 types of timeline dynamics

- Event
- Step function
- Continuously-evolving

Compositional language for defining DAG of operators

Library of operators

Timeline generalizations of classical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\equiv, &lt;, &gt;$ [constant]</td>
<td>Compare each update or state with a fixed value, producing True or False</td>
</tr>
<tr>
<td>$&amp;,</td>
<td>$ [timeline]</td>
</tr>
<tr>
<td>$\sim$</td>
<td>Logically invert each update or state</td>
</tr>
</tbody>
</table>

Timeline-specific Operators

- **TL\_HASEXISTED**
  - A StateDynamics timeline of the cumulative OR

- **TL\_HASEXISTEDWITHIN**
  - As TL\_HASEXISTED, but resets to False after a specified duration $D$ without True values

- **TL\_LATESTEVENTTOSTATE**
  - A StateDynamics Timeline of the latest update

- **TL\_DURATIONWHERE**
  - A Numerical Timeline of the cumulative duration where the state was True

- **TL\_DURATIONINCURSTATE**
  - A Numerical Timeline of the duration since the last state change

Connectors with external data sources/sinks
Outline for talk

• What is Time-State Analytics

• Time-State Analytics not well supported by status quo

• Introducing the Timeline abstraction

• Early Promise + Next Steps
Timeline Reduced Dev Effort at Conviva

Onboarding: Weeks $\rightarrow$ Days

Semantic Bugs: Dropped by 80%

Prototype query language
Timeline Offers Reduced Cost

2-10X Faster Execution Time!
Future Outlook

• Applications to many domains
  • Cybersecurity, IoT, logistics, manufacturing, ...

• Visual interfaces to democratize TSA

• Even better performance

• Streaming implementation
Takeaways

• Growing need for Time-State Analytics across different domains

• Fundamentally hard problem:
  Stateful, Context-Sensitive, Continuous

• State-of-art systems (e.g., streaming systems, data warehouses, RDBMS) ill suited
  • Why: Classical tabular model for data processing is ineffective for Time-State Analytics
    • Great for simple stateless filter/aggregation but not Time-State Analytics
    • High cost, low performance + High dev effort, many bugs

• Our work: Timeline ➔ A geometric abstraction for Time-State Analytics
  • Early promise: Up to 10X better cost/performance AND 10X reduced effort
  • New opportunities: Generality, No-code Intents