Towards Unified Heterogeneous Event Processing for the Internet of Things

Wei Wang

Department of Computer Science and Technology, Tongji University
Overview

- Background and Introduction
- HEP framework
- Window Specification3E-policy based semantic structure
- Query language
- Experiment
- Conclusion
Background

- IoT technology provide a flood of information
- A large number of diverse, interrelated data sources
- No solution to handle the heterogeneous event streams from diverse sources
- Existing event stream languages lack a consistent semantic for processing data
Introduction

• A framework (HEP) which supports unified event processing

• A general window specification with an understandable and robust stream query language
HEP framework

CEA provides various kinds of wrappers to access heterogeneous event data sources

ES utilized to cache formatted event data, and some pre-processing work is also carried out in it

MM manage all the metadata existing in the framework, including global/local schema, event specification, etc.
HEP framework

WQM provides a standard event stream and query model

QT parse and translate the standard stream language into different vendor-specific stream languages

PKB provides the necessary knowledge for event processing

Figure 1 Overview of the HEP framework
HEP framework

EPS makes a plan for the customer defined stream queries and carries out some necessary optimizations.

EQGD provides users or applications a visualization workbench to define complex event patterns.

CEP provides complex event output adapter to connect with ultimate consumers.
Key points:

– how can the extent of that window be defined?

– should this extent be allowed to change over time?

– when will the query evaluate over the window?
Window Specification

3E-policy:

Extent-Evolution-Evaluation

Extent policy (α-policy):
the content of a Window

Evolution policy (β-policy):
define the way a window evolves or updates

Evaluation policy (χ-policy):
evaluation frequency of a window
Window Specification

```
WINDOW window_identifier ( 
    RANGE α_value 
    [RATTR] {TUPLE(S) | TIME | ON FIELD field | PATTERN patterns} 
    [PARTITION BY fieldlist] 
    SYNC β_value 
    [SATTR] {TUPLE(S) | TIME | WHEN condition1} 
    [EVALUATE χ_value 
    [EATTR] {TUPLE(S) | TIME | WHEN condition2}] )
```

Figure 3 Specification of the unified window
Query language

Example:


SELECT SUM (price) AS SumPrice
FROM S
[RANGE 4 second TIME
SYNC 2 second TIME
EVALUATE 1 second TIME];

The query result is
Q(Time: SumPrice) = (4: 100) (5: 100) (6: 180) (7: 180) (8: 260)
### Various common types of windows:

<table>
<thead>
<tr>
<th>Type</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuple based sliding window</td>
<td>RANGE 5 TUPLES</td>
<td>Basic tuple based sliding window</td>
</tr>
<tr>
<td></td>
<td>SYNC 1 TUPLE</td>
<td></td>
</tr>
<tr>
<td>Time based sliding window</td>
<td>RANGE 5 hour TIME</td>
<td>Basic time based sliding window</td>
</tr>
<tr>
<td></td>
<td>SYNC 1 hour TIME</td>
<td></td>
</tr>
<tr>
<td>Landmark window</td>
<td>RANGE Unbounded TIME</td>
<td>Landmark window without upper bound [4]</td>
</tr>
<tr>
<td></td>
<td>SYNC 1 hour TIME</td>
<td></td>
</tr>
<tr>
<td>Tumbling window</td>
<td>RANGE 5 hour TIME</td>
<td>Tumbling window based on time [2]</td>
</tr>
<tr>
<td></td>
<td>SYNC 5 hour TIME</td>
<td></td>
</tr>
<tr>
<td>Now window</td>
<td>RANGE Now TIME</td>
<td>Window content is based on current time [3]</td>
</tr>
<tr>
<td></td>
<td>SYNC 1 TUPLE</td>
<td></td>
</tr>
<tr>
<td>On field window</td>
<td>RANGE 5 hour</td>
<td>Sliding window based on custom field</td>
</tr>
<tr>
<td></td>
<td>ON FIELD TimestampAttr</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNC 1 hour TIME</td>
<td></td>
</tr>
<tr>
<td>Partition window</td>
<td>RANGE 5 TUPLES</td>
<td>Partition window based on CarId field [3]</td>
</tr>
<tr>
<td></td>
<td>PARTITION BY Carld</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNC 1 TUPLE</td>
<td></td>
</tr>
<tr>
<td>Predicate window</td>
<td>RANGE Temperature &gt; 90</td>
<td>Predicate window for sensor networks [16]</td>
</tr>
<tr>
<td></td>
<td>ON FIELD SensorID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNC 5 second TIME</td>
<td></td>
</tr>
<tr>
<td>Mixed jumping window</td>
<td>RANGE 4 second TIME</td>
<td>A time based jumping window with a tuple based evaluation [17]</td>
</tr>
<tr>
<td></td>
<td>SYNC 4 second TIME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EVALUATE 1 TUPLE</td>
<td></td>
</tr>
<tr>
<td>Sampling window</td>
<td>RANGE 10 TUPLES</td>
<td>Sampling a tuple based sliding window [17]</td>
</tr>
<tr>
<td></td>
<td>SYNC 1 TUPLES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EVALUATE 1 hour TIME</td>
<td></td>
</tr>
</tbody>
</table>
Experiment

Window Specification Effectiveness

S (Time: VID, Spd) = (1: 01, 50) (1: 02, 50) (2: 01, 50) (2: 02, 50) (2: 03, 20)

Q1 = ISTREAM (SELECT avg (Spd) as AvgSpd, Time FROM S[rows 1])

Time based model: Q1 (Time: AvgSpd) = (1: 50) (2: 20)

Tuple based model: Q1 (Time: AvgSpd) = (1: 50) (1: 50) (2: 50) (2:50) (2: 20)
**Experiment**

**Window Specification Effectiveness**

\[ S \ (\text{Time: VID, Spd}) = (1: 01, 50) \ (1: 02, 50) \ (2: 01, 50) \ (2: 02, 50) \ (2: 03, 20) \]

**3E-policy based model:**

\[ Q2 = \text{SELECT} \ \text{avg} \ (\text{Spd}) \ \text{as AvgSpd, Time} \ \text{FORM S} \]

[RANGE Now
SYNC 1 TUPLE
EVALUATE 1 TUPLE];

\[ Q2 \ (\text{Time: AvgSpd}) = (1: 50) \ (1: 50) \ (2: 50) \ (2: 50) \ (2: 40) \]
Experiment

Performance Evaluation

**Figure 4** Evaluation of varying stream size

**Figure 5** Evaluation of space cost
Conclusion

- HEP can correctly handle heterogeneous event data
- HEP supports unified event processing with a unified window specification
- Higher performance with lower cost
Thank You!