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

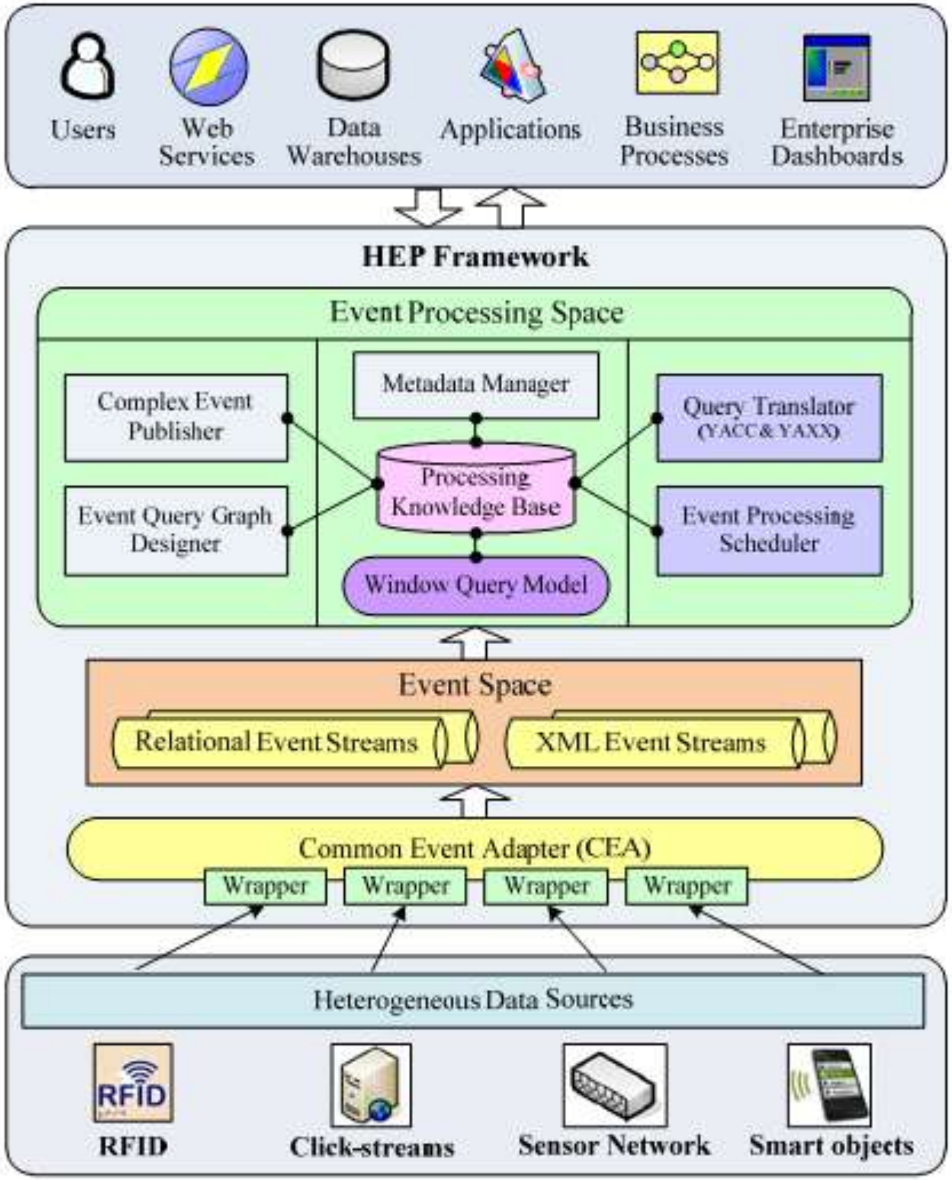

Figure 1 Overview of the HEP framework

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

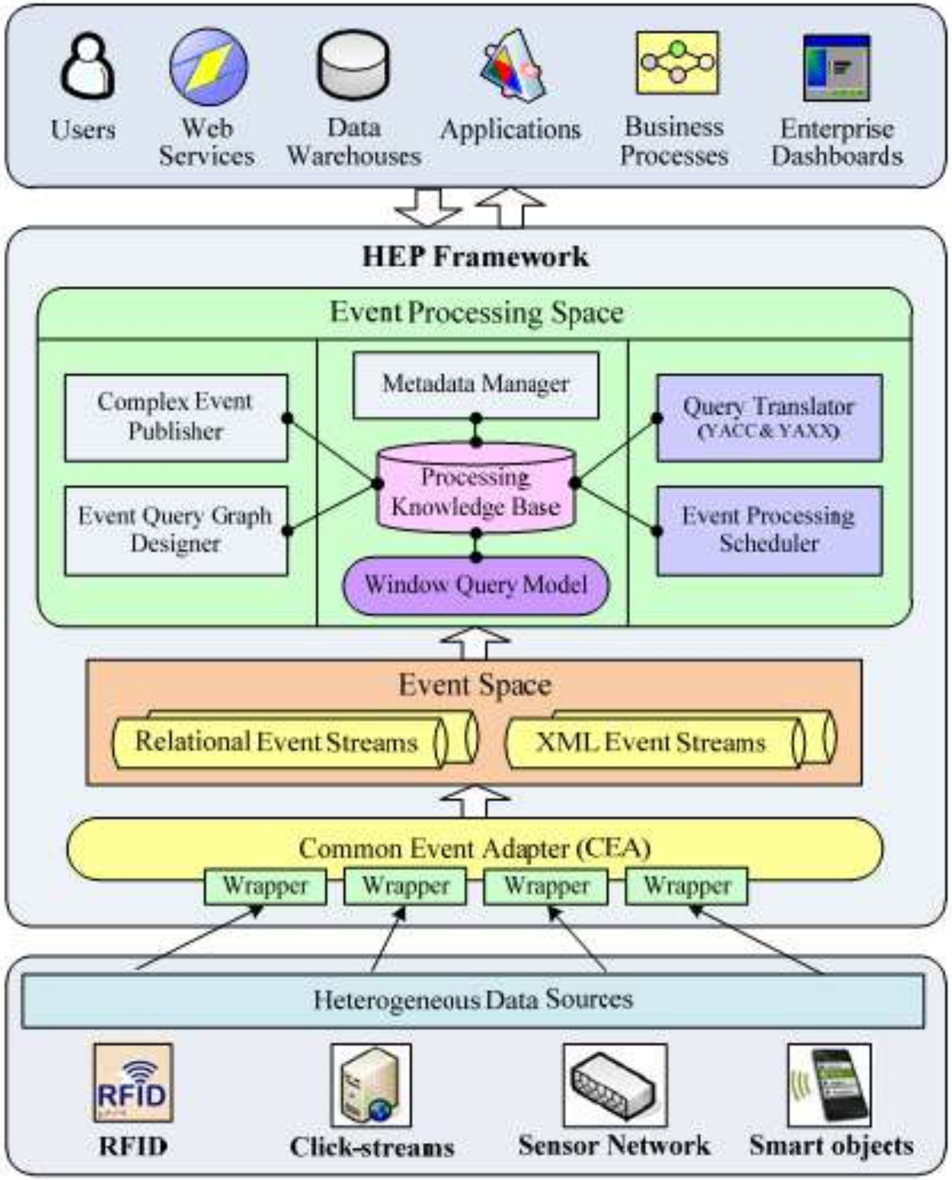

Figure 1 Overview of the HEP framework

WQM
provides a standard
event stream and query model

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

PKB
provides the necessary
knowledge for event processing

HEP framework

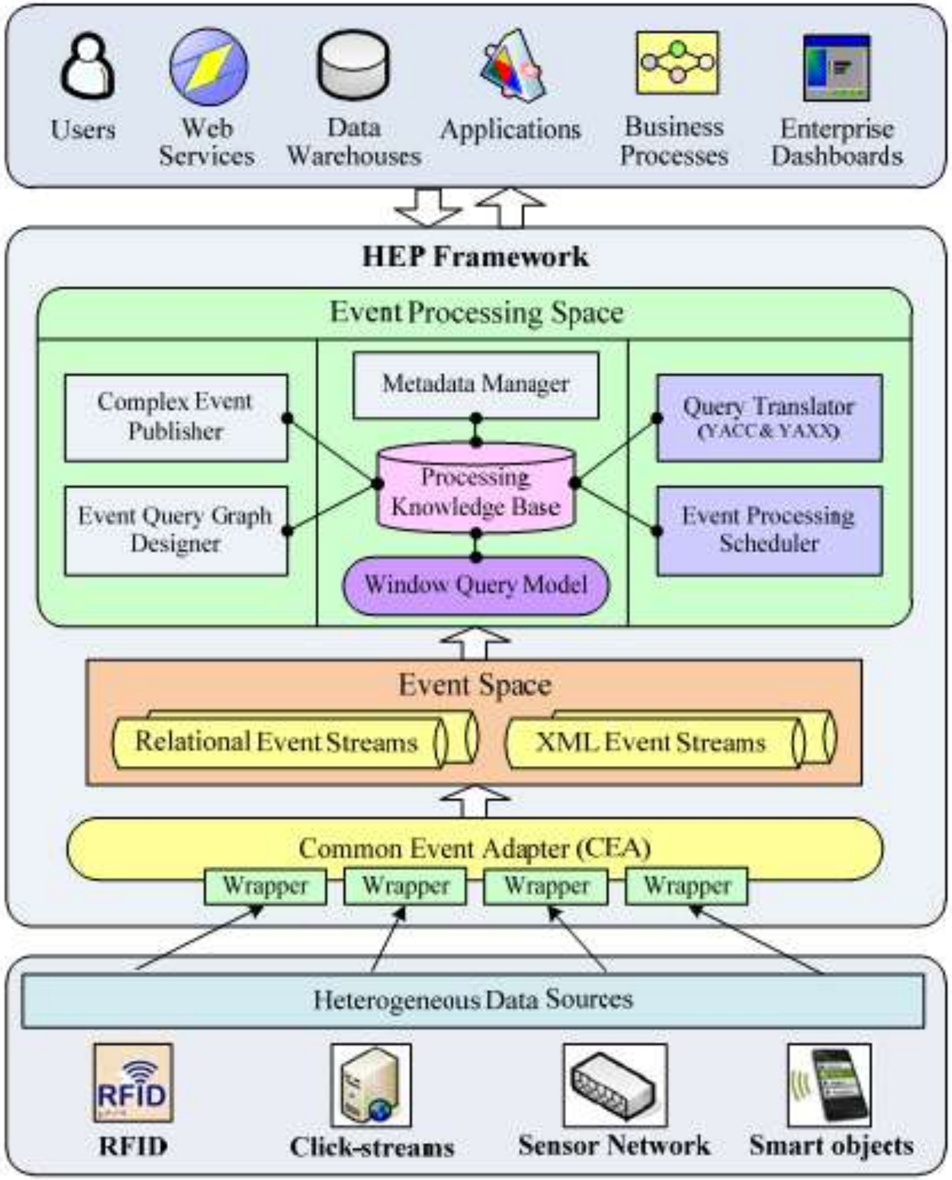

Figure 1 Overview of the HEP framework

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

Window Specification

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

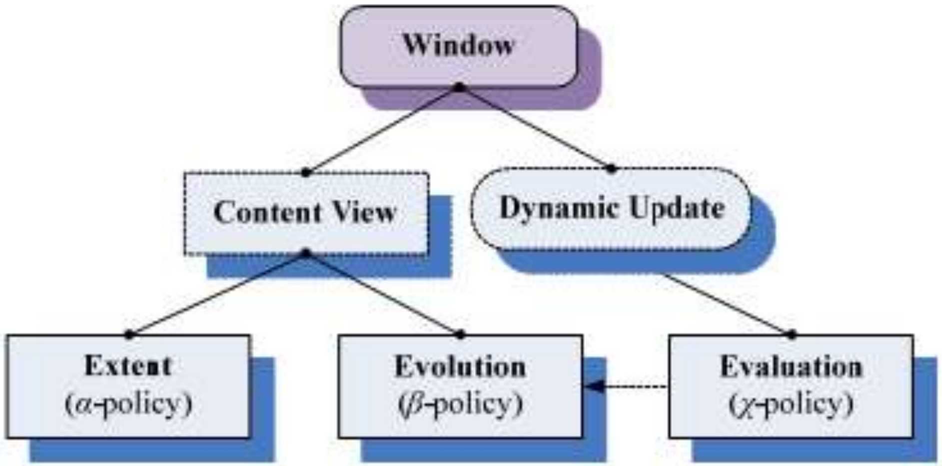

Figure 2 Semantic structure of window

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:

S (Time: Price) = (1: 10) (2: 20) (3:30) (4: 40) (5: 50) (6: 60) (7: 70) (8: 80)

```
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)

Query language

Various common types of windows:

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

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

3E-policy based model:

Q2= SELECT avg (Spd) as AvgSpd, Time FORM S
[RANGE Now
SYNC 1 TUPLE
EVALUATE 1 TUPLE];

Q2 (Time: AvgSpd) = (1: 50) (1: 50) (2: 50) (2: 50) (2: 40)

Experiment

Performance Evaluation

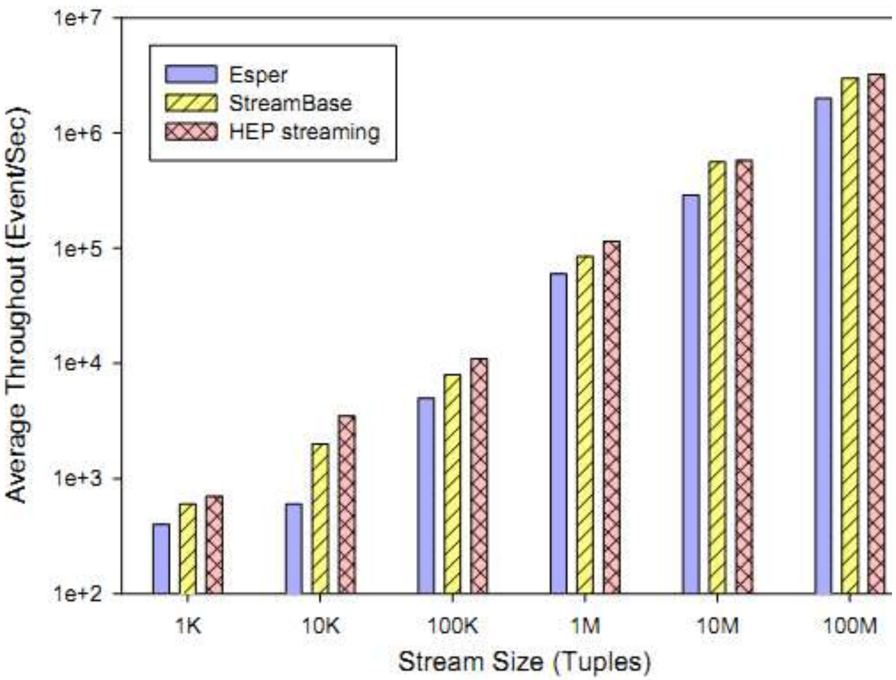

Figure 4 Evaluation of varying stream size

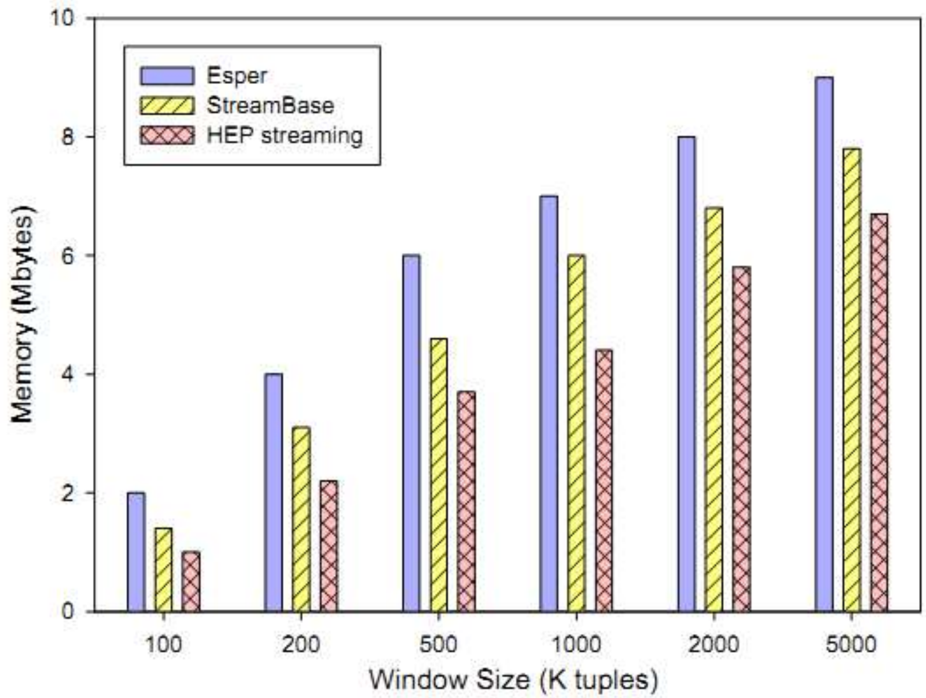

Figure 5 Evaluation of space cost

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 !