Shared Online Event Trend Aggregation

Olga Poppe, Chuan Lei, Lei Ma, Allison Rozet, Elke A. Rundensteiner

Best short paper in CIKM 2020
Full paper in SIGMOD 2021

Motivation

What are event trends?

Algorithmic Trading

Goal:

Reliable actionable insights about the stream

Solution:

Each event is considered in the context of other events in the stream

Picture source: https://cabotwealth.com/daily/stock-market/stock-market-trends-profit-2018/

Algorithmic Trading

Single event =
Single stock value

Event sequence =
Stock down trend of fixed length

Event trend =
Stock up trend of any length

Picture source: https://cabotwealth.com/daily/stock-market/stock-market-trends-profit-2018/

Algorithmic Trading

Single event =
Single stock value

Event sequence =
Stock down trend of fixed length

Event trend =
Stock up trend of any length

Picture source: https://cabotwealth.com/daily/stock-market/stock-market-trends-profit-2018/

Algorithmic Trading

Single event =
Single stock value

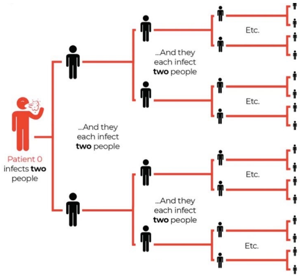
Event sequence =
Stock down trend of fixed length

Event trend =
Stock up trend of any length

Picture source: https://cabotwealth.com/daily/stock-market/stock-market-trends-profit-2018/

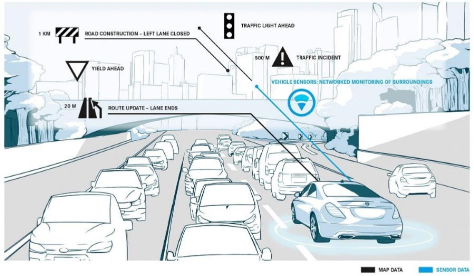
Event Trends

Infection spread



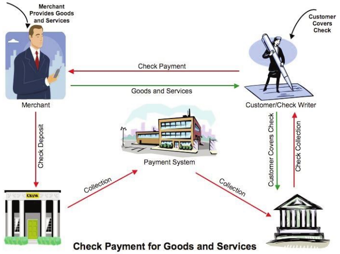
Path of infection spread

Ridesharing



Trajectory of shared ride

Financial fraud



Circular check kite

Performance optimization

Increasing load of a system component

Complexity of Event Trend Analytics

Under Skip-Till-Any-Match Semantics [SIGMOD'08]

Complexity of Event Trend Analytics

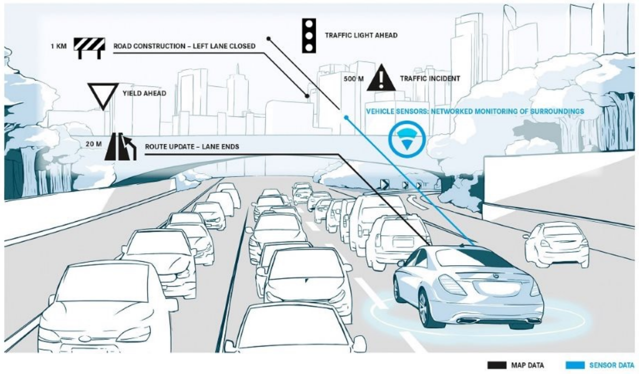
Under Skip-Till-Any-Match Semantics [SIGMOD'08]

Event Trend Aggregation Queries

Ridesharing

```
q1:  RETURN      T.district, COUNT(*), SUM(T.duration)
     PATTERN     Request R, Travel T+, NOT Pickup P
     WHERE       [driver, rider]
     GROUP-BY    T.district
     WITHIN      30 min SLIDE 1 min
```

Number and duration of trips in which driver drove to pickup location but did not pick up the rider

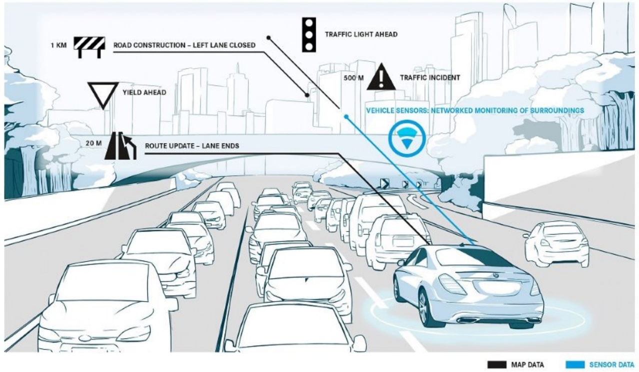


Event Trend Aggregation Queries

Ridesharing

```
q1: RETURN     T.district, COUNT(*), SUM(T.duration)
    PATTERN    Request R, Travel T+, NOT Pickup P
    WHERE      [driver, rider]
    GROUP-BY   T.district
    WITHIN     30 min SLIDE 1 min
```

Number and duration of trips in which driver drove to pickup location but did not pick up the rider

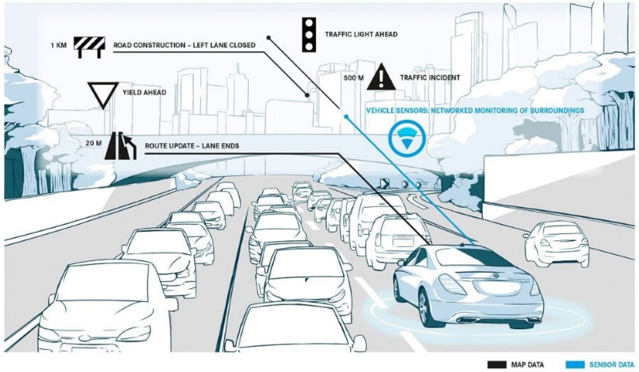


Event Trend Aggregation Queries

Ridesharing

```
q1: RETURN     T.district, COUNT(*), SUM(T.duration)
    PATTERN    Request R, Travel T+, NOT Pickup P
    WHERE      [driver, rider]
    GROUP-BY   T.district
    WITHIN     30 min SLIDE 1 min
```

Number and duration of trips in which driver drove to pickup location but did not pick up the rider

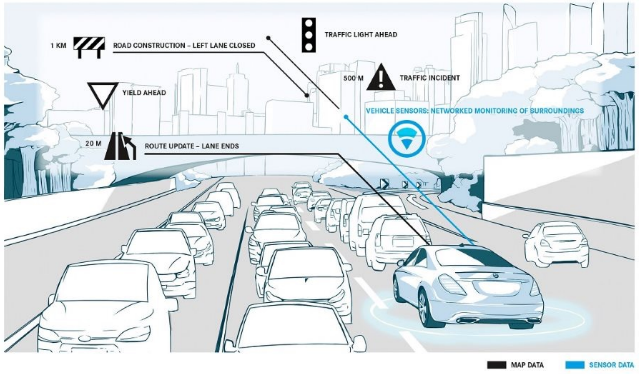


Event Trend Aggregation Queries

Ridesharing

```
q1: RETURN    T.district, COUNT(*), SUM(T.duration)
    PATTERN   Request R, Travel T+, NOT Pickup P
    WHERE     [driver, rider]
    GROUP-BY  T.district
    WITHIN    30 min SLIDE 1 min
```

Number and duration of trips in which driver drove to pickup location but did not pick up the rider

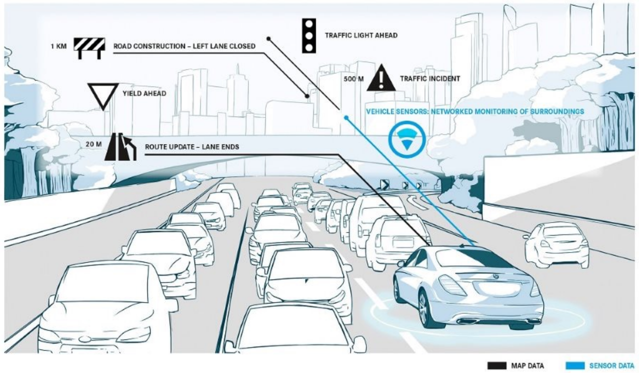


Event Trend Aggregation Queries

Ridesharing

```
q1: RETURN     T.district, COUNT(*), SUM(T.duration)
    PATTERN    Request R, Travel T+, NOT Pickup P
    WHERE      [driver, rider]
    GROUP-BY   T.district
    WITHIN     30 min SLIDE 1 min
```

Number and duration of trips in which driver drove to pickup location but did not pick up the rider

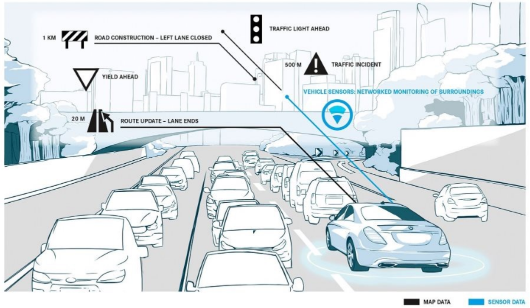


Problem Statement

Event trend aggregation queries

```
q1:  RETURN     T.district, COUNT(*), SUM(T.duration)
     PATTERN    Request R, Travel T+, NOT Pickup P
     WHERE      [driver, rider]
     GROUP-BY   T.district
     WITHIN     30 min SLIDE 1 min

q2:  RETURN     T.district, COUNT(*), AVG(T.speed)
     PATTERN    Request R, Travel T+, Dropoff D
     WHERE      [driver, rider] AND R.type=Pool
     GROUP-BY   T.district
     WITHIN     30 min SLIDE 5 min

q3:  RETURN     T.district, COUNT(*), SUM(T.duration)
     PATTERN    Request R, Travel T+, Cancel C
     WHERE      [driver, rider] AND T.speed<10
     GROUP-BY   T.district
     WITHIN     20 min SLIDE 1 min
```

High-rate event stream



Average query latency of all queries is minimal

Challenges

1. Exponential complexity vs real-time response

Online

Event trend aggregation without event trend construction reduces complexity from exponential to quadratic [VLDB'17, SIGMOD'19]

Shared

Event trend aggregation among multiple queries requires construction of shared sub-trends to ensure correctness

⇒ Correct yet efficient shared online event trend aggregation strategy

Challenges

1. Exponential complexity vs real-time response
2. Benefit vs overhead of sharing

Benefit

Due to avoided re-computations for similar queries in the workload

Overhead

Due to maintenance of intermediate results per query to ensure correctness

⇒ Light-weight yet accurate sharing benefit model

Challenges

1. Exponential complexity vs real-time response
2. Benefit vs overhead of sharing
3. Bursty event streams vs light-weight sharing decisions

Static sharing optimizer

Can do more harm than good if event rate and data distribution fluctuate

➡

Dynamic sharing optimizer

Must adjust its decisions to the changing cost factors at runtime

⇒ Runtime yet light-weight sharing decisions

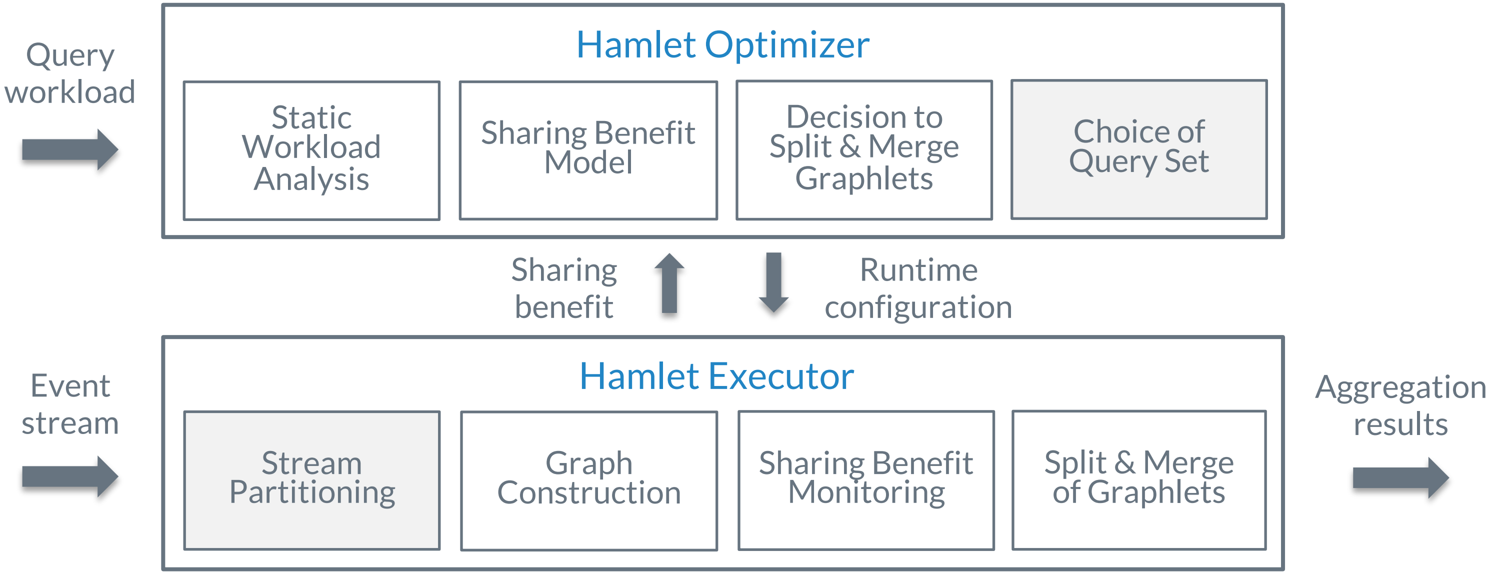
State-of-the-Art

Approach	Kleene closure	Online aggregation	Sharing decisions
MCEP [SIGMOD'19]	✓	-	static
Sharon [ICDE'18]	-	✓	static
Greta [VLDB'17]	✓	✓	not shared

State-of-the-Art

Approach	Kleene closure	Online aggregation	Sharing decisions
MCEP [SIGMOD'19]	✓	-	static
Sharon [ICDE'18]	-	✓	static
Greta [VLDB'17]	✓	✓	not shared
Hamlet dynamically decides to share or not to share online event trend aggregation			

Hamlet Framework



Sharable Queries

```
q1: RETURN     T.district, COUNT(*), SUM(T.duration)
    PATTERN    Request R, Travel T+, Pickup P
    WHERE      [driver, rider]
    GROUP-BY   T.district
    WITHIN     30 min SLIDE 1 min

q2: RETURN     T.district, COUNT(*), AVG(T.speed)
    PATTERN    Request R, Travel T+, Dropoff D
    WHERE      [driver, rider] AND R.type=Pool
    GROUP-BY   T.district
    WITHIN     30 min SLIDE 5 min

q3: RETURN     T.district, COUNT(*), SUM(T.duration)
    PATTERN    Request R, Travel T+, Cancel C
    WHERE      [driver, rider] AND T.speed<10
    GROUP-BY   T.district
    WITHIN     20 min SLIDE 1 min
```

Queries are sharable if their

- Patterns contain at least one sharable Kleene sub-pattern,

Sharable Queries

```
q1: RETURN    T.district, COUNT(*), SUM(T.duration)
    PATTERN   Request R, Travel T+, Pickup P
    WHERE     [driver, rider]
    GROUP-BY  T.district
    WITHIN    30 min SLIDE 1 min

q2: RETURN    T.district, COUNT(*), AVG(T.speed)
    PATTERN   Request R, Travel T+, Dropoff D
    WHERE     [driver, rider] AND R.type=Pool
    GROUP-BY  T.district
    WITHIN    30 min SLIDE 5 min

q3: RETURN    T.district, COUNT(*), SUM(T.duration)
    PATTERN   Request R, Travel T+, Cancel C
    WHERE     [driver, rider] AND T.speed<10
    GROUP-BY  T.district
    WITHIN    20 min SLIDE 1 min
```

Queries are sharable if their

- Patterns contain at least one sharable Kleene sub-pattern,
- Aggregation functions can be shared,

Sharable Queries

```
q1:  RETURN     T.district, COUNT(*), SUM(T.duration)
     PATTERN    Request R, Travel T+, Pickup P
     WHERE      [driver, rider]
     GROUP-BY   T.district
     WITHIN     30 min SLIDE 1 min

q2:  RETURN     T.district, COUNT(*), AVG(T.speed)
     PATTERN    Request R, Travel T+, Dropoff D
     WHERE      [driver, rider] AND R.type=Pool
     GROUP-BY   T.district
     WITHIN     30 min SLIDE 5 min

q3:  RETURN     T.district, COUNT(*), SUM(T.duration)
     PATTERN    Request R, Travel T+, Cancel C
     WHERE      [driver, rider] AND T.speed<10
     GROUP-BY   T.district
     WITHIN     20 min SLIDE 1 min
```

Queries are sharable if their

- Patterns contain at least one sharable Kleene sub-pattern,
- Aggregation functions can be shared,
- Windows overlap, and

Sharable Queries

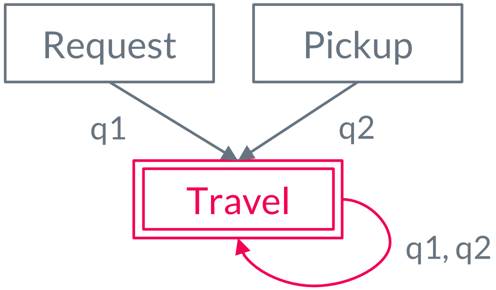
```
q1: RETURN     T.district, COUNT(*), SUM(T.duration)
    PATTERN    Request R, Travel T+, Pickup P
    WHERE      [driver, rider]
    GROUP-BY   T.district
    WITHIN     30 min SLIDE 1 min

q2: RETURN     T.district, COUNT(*), AVG(T.speed)
    PATTERN    Request R, Travel T+, Dropoff D
    WHERE      [driver, rider] AND R.type=Pool
    GROUP-BY   T.district
    WITHIN     30 min SLIDE 5 min

q3: RETURN     T.district, COUNT(*), SUM(T.duration)
    PATTERN    Request R, Travel T+, Cancel C
    WHERE      [driver, rider] AND T.speed<10
    GROUP-BY   T.district
    WITHIN     20 min SLIDE 1 min
```

Queries are sharable if their

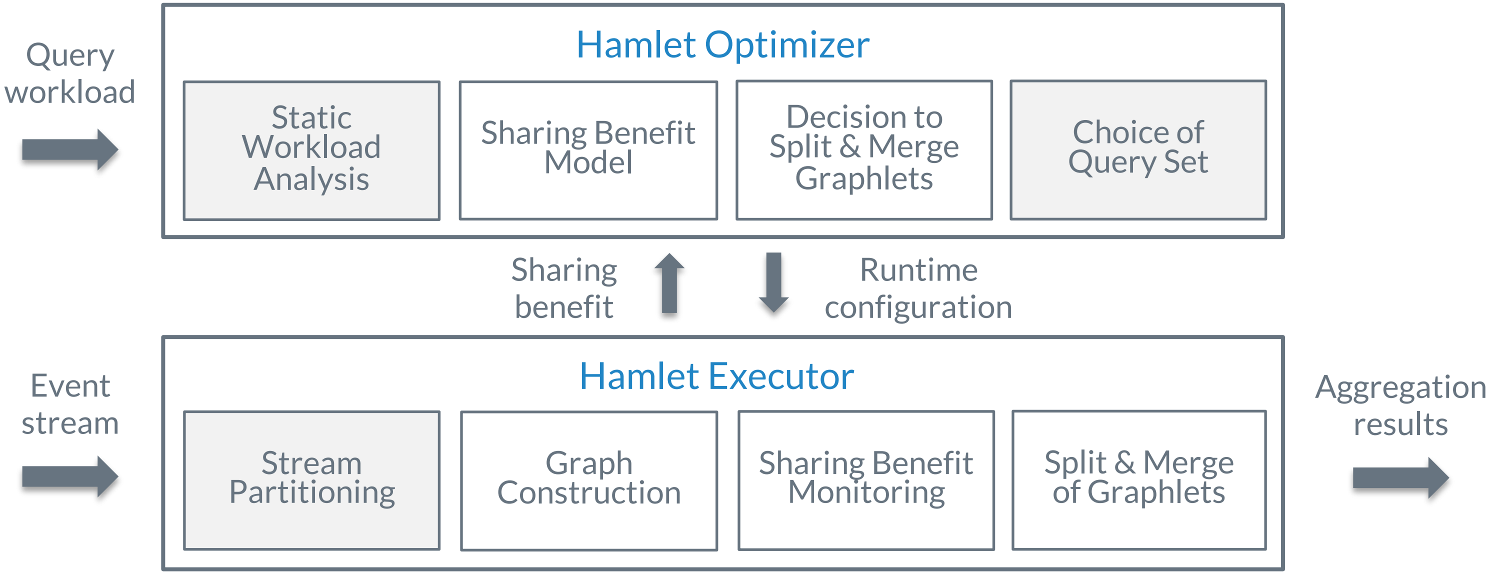
- Patterns contain at least one sharable Kleene sub-pattern,
- Aggregation functions can be shared,
- Windows overlap, and
- Grouping attributes are the same.

Hamlet Template

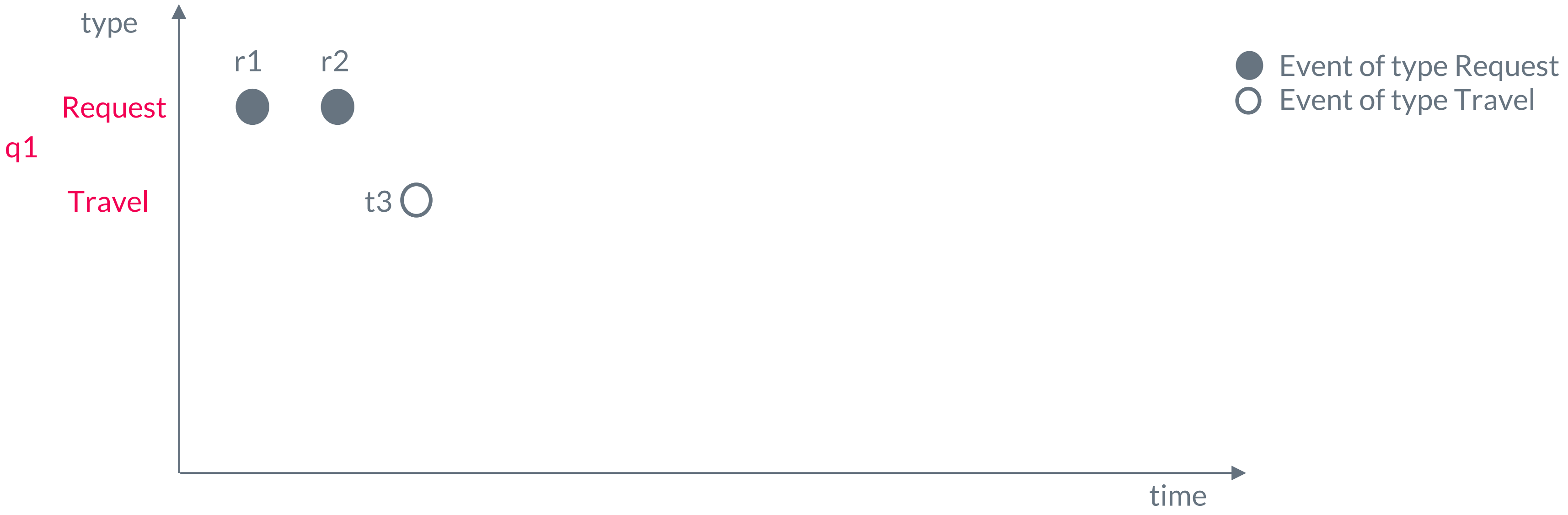
```
q1: RETURN    T.district, COUNT(*), SUM(T.duration)
    PATTERN   Request R, Travel T+
    WHERE     [driver, rider]
    GROUP-BY  T.district
    WITHIN    10 min SLIDE 5 min

q2: RETURN    T.district, COUNT(*), AVG(T.speed)
    PATTERN   Pickup P, Travel T+
    WHERE     [driver, rider] AND P.type=Pool
    GROUP-BY  T.district
    WITHIN    15 min SLIDE 5 min
```



Hamlet Framework

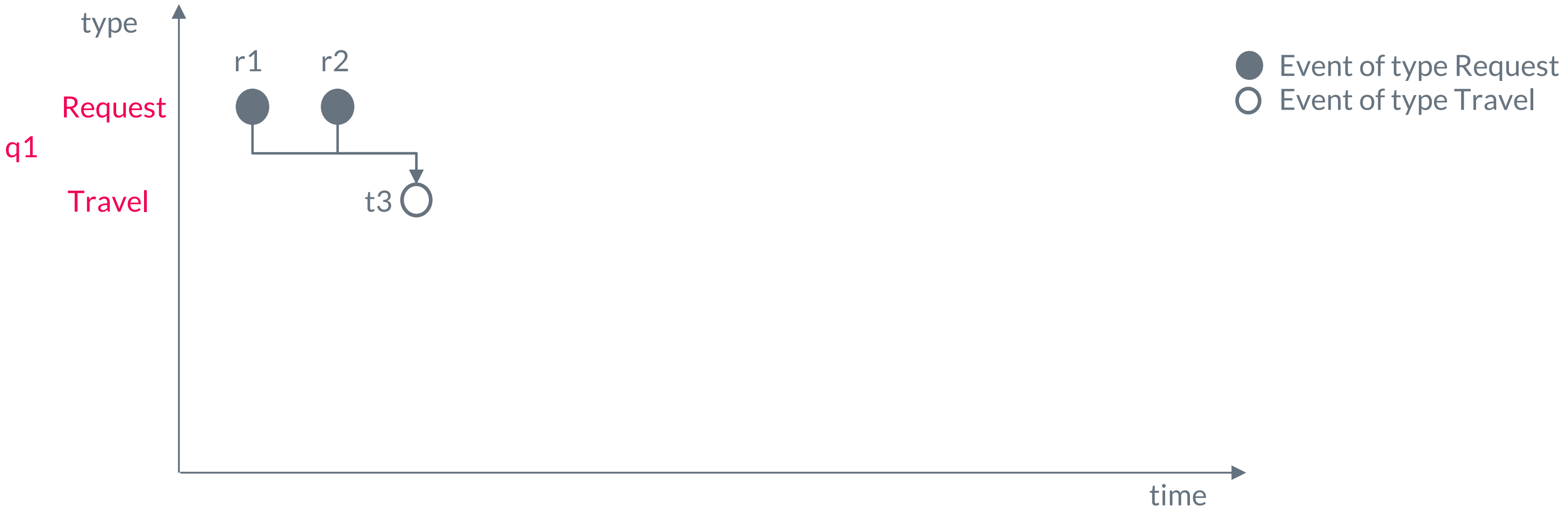


Non-Shared Graph Construction



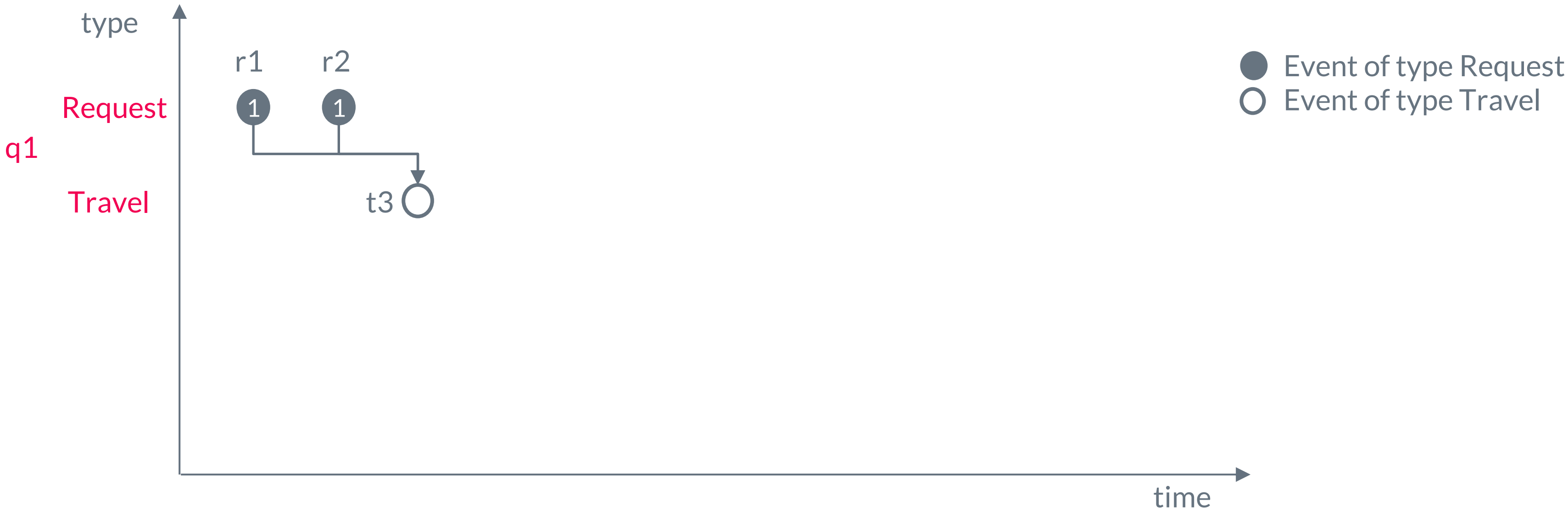
2 event trends:
r1,t3
r2,t3

Non-Shared Graph Construction



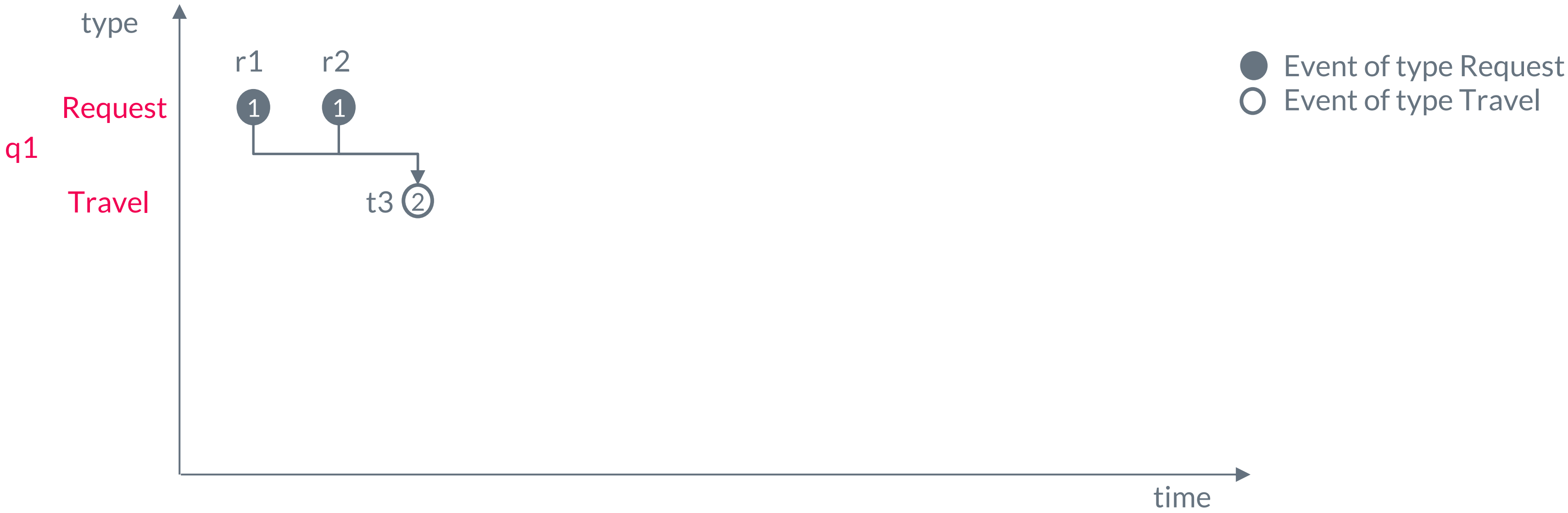
2 event trends:
r1,t3
r2,t3

Non-Shared Graph Construction



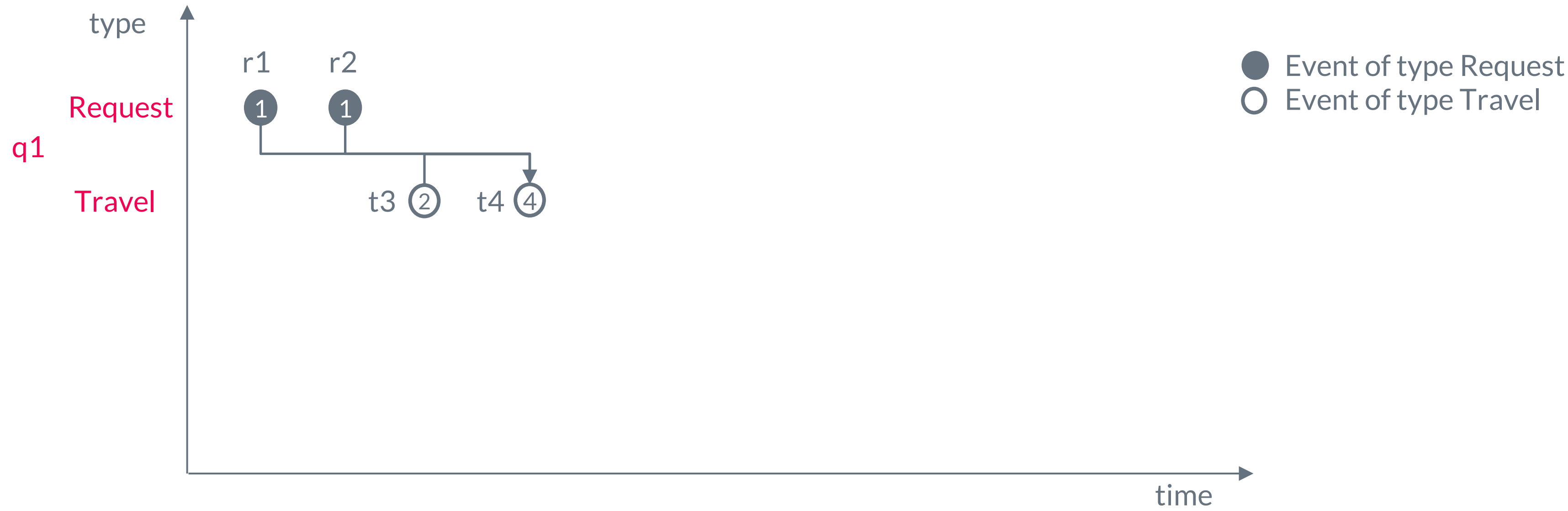
2 event trends:
r1,t3
r2,t3

Non-Shared Graph Construction



2 event trends:
r1,t3
r2,t3

Non-Shared Graph Construction

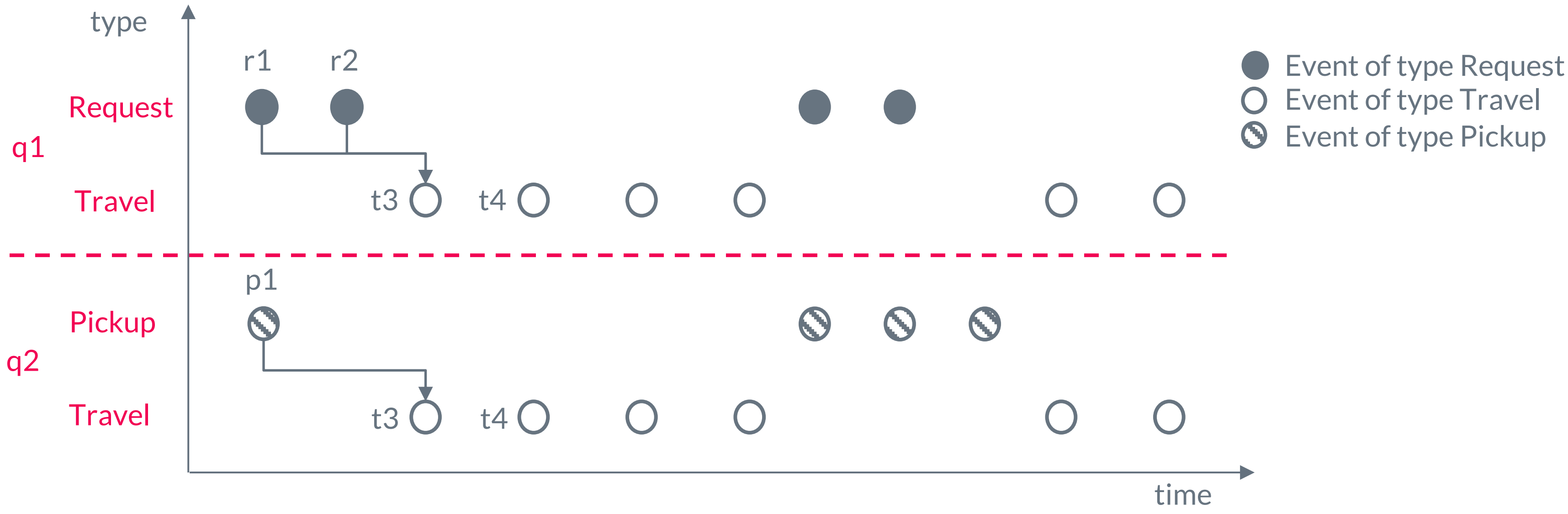


6 event trends:
r1,t3
r1,t4
r1,t3,t4
r2,t3
r2,t4
r2,t3,t4

$$NonShared(Q) = O(n^2)$$

where n – # events in a window

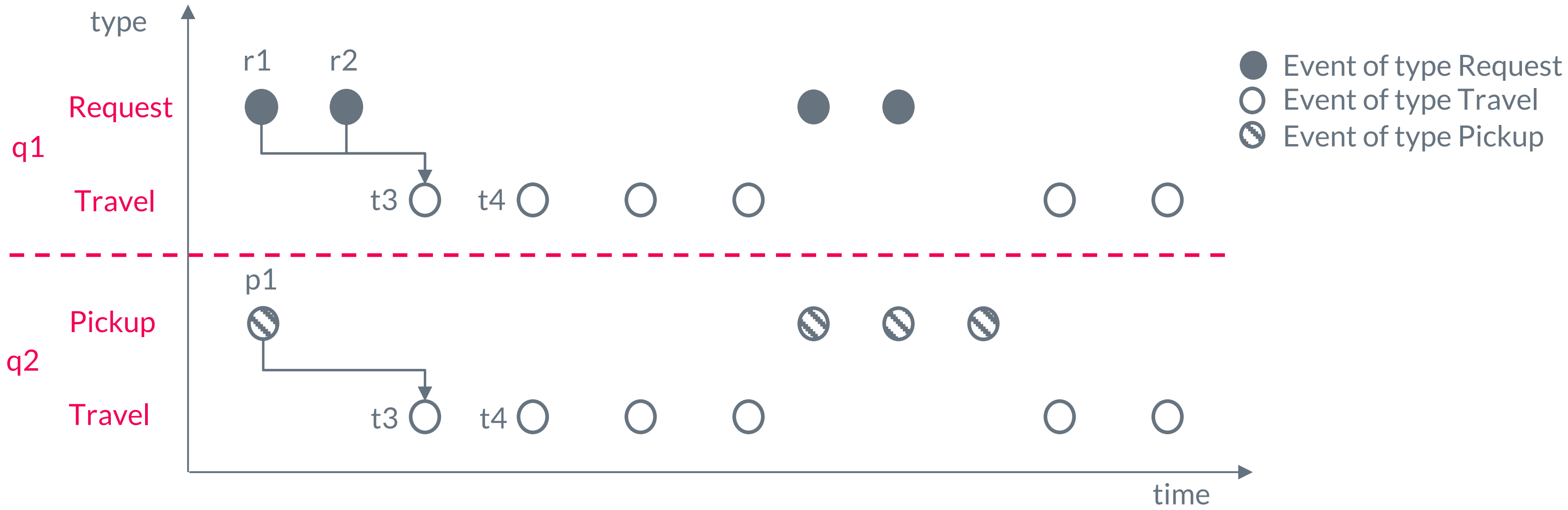
Non-Shared Graph Construction



$$NonShared(Q) = O(n^2 * k)$$

where n – # events in a window,
k – # queries

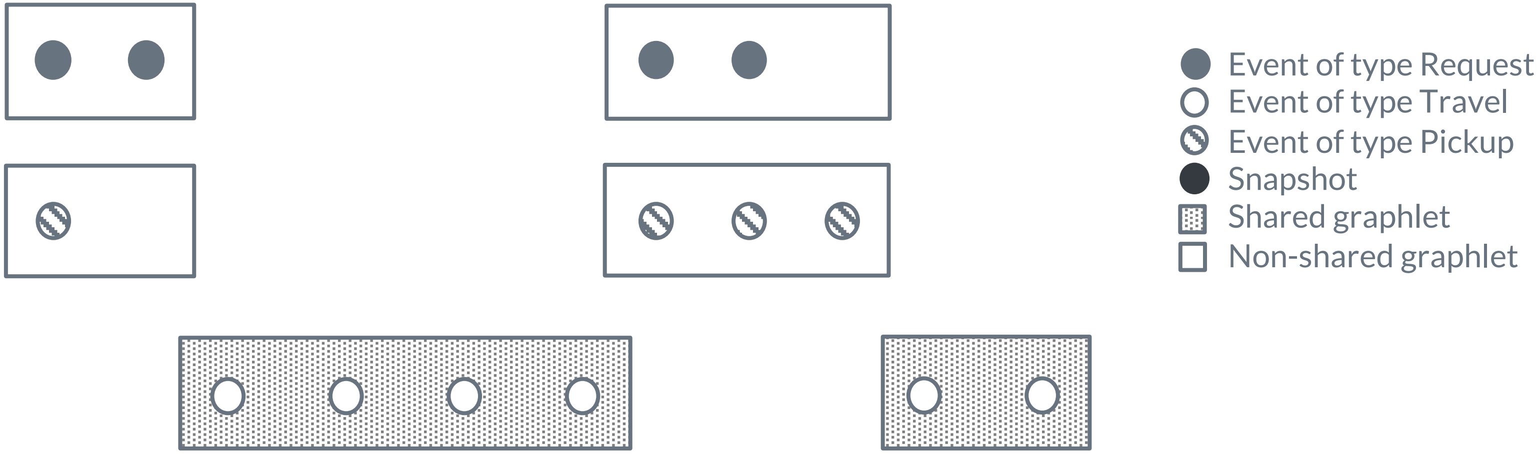
Non-Shared Graph Construction



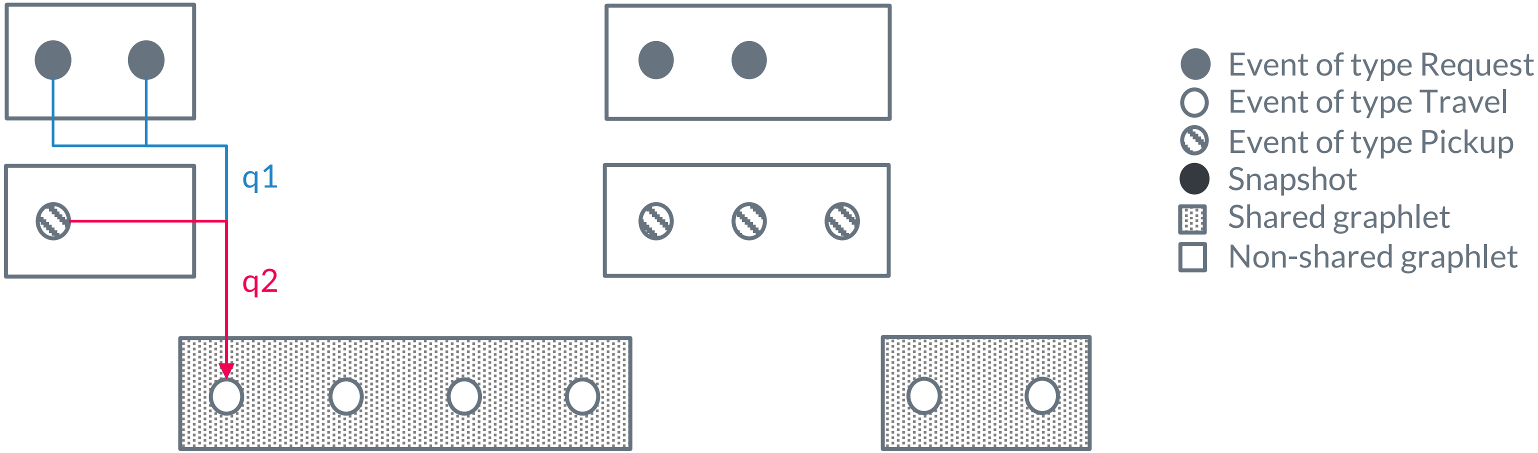
$$NonShared(Q) = O(n^2 * k) = 14^2 * 2 = 392$$

where n – # events in a window,
k – # queries

Shared Graph Construction



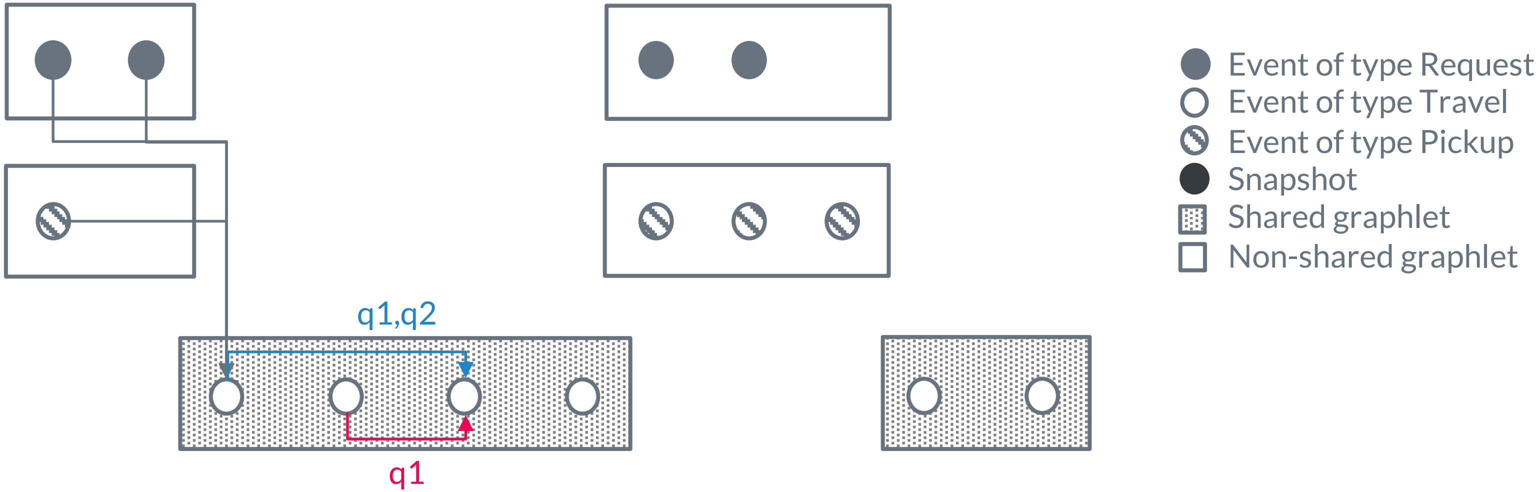
Shared Graph Construction



The set of predecessor events is different for q1 and q2 due to:

- Different patterns

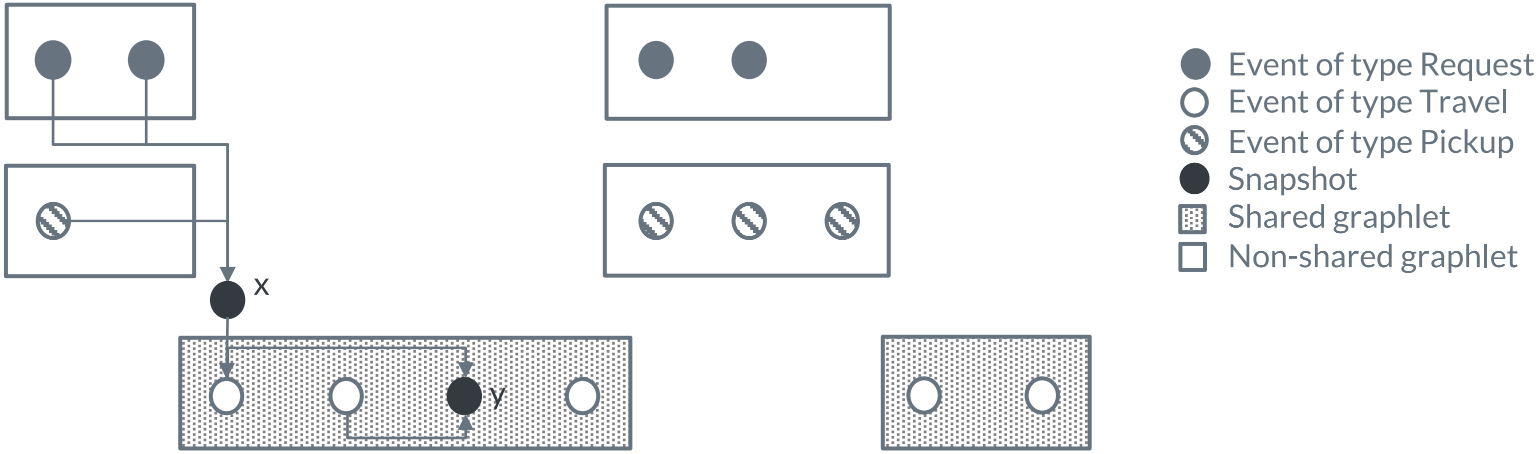
Shared Graph Construction



The set of predecessor events is different for q1 and q2 due to:

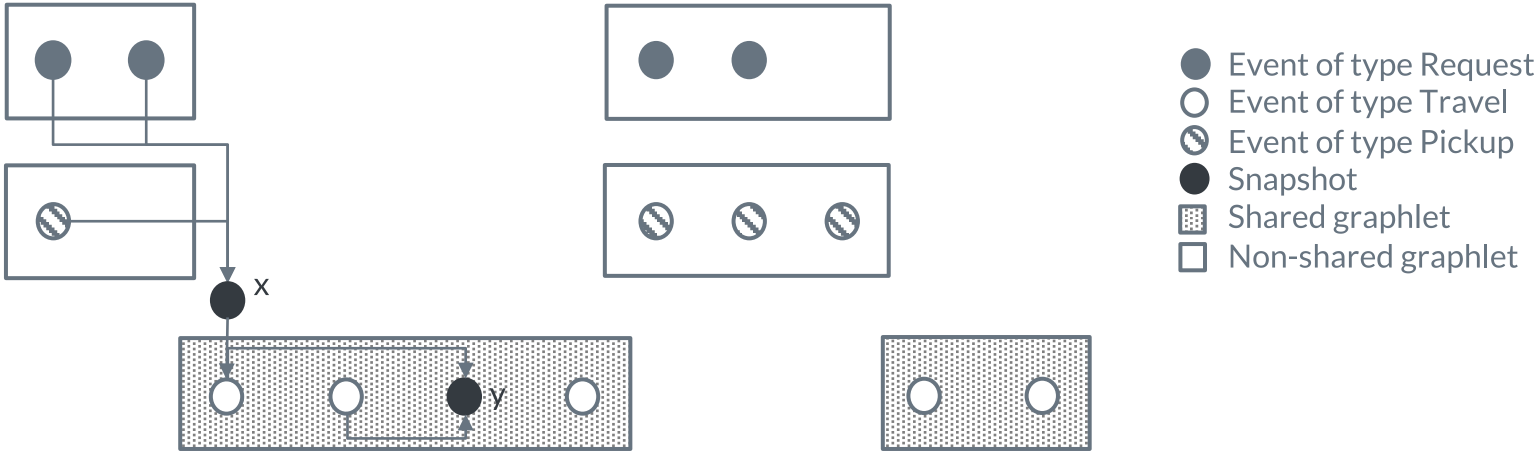
- Different patterns
- Predicates

Shared Graph Construction



Snapshot	q1	q2
x	2	1
y	8	4

Shared Graph Construction



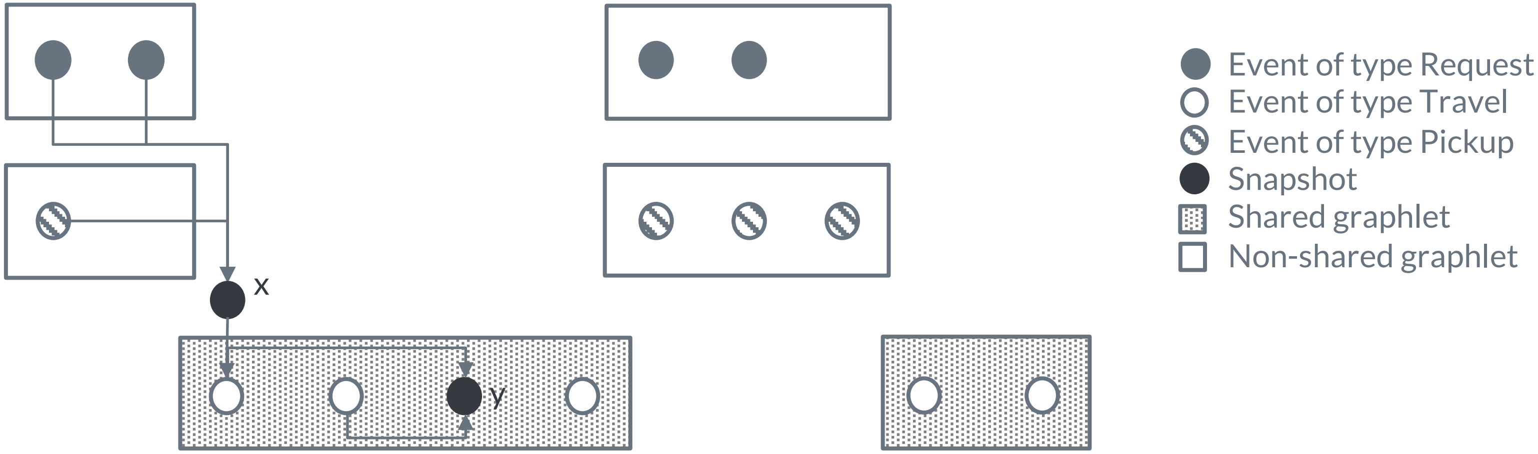
$$Shared(Q) = O(n^2 * s + s * k * g * t)$$

where n – # events in a window,

k – # queries, g – # events per graphlet,

s – # snapshots, t – # types per query

Shared Graph Construction



$Shared(Q) = O(n^2 * s + s * k * g * t) = 14^2 * 2 + 2 * 2 * 4 * 2 = 424$ > $NonShared(Q) = O(n^2 * k) = 14^2 * 2 = 392$

where n – # events in a window,

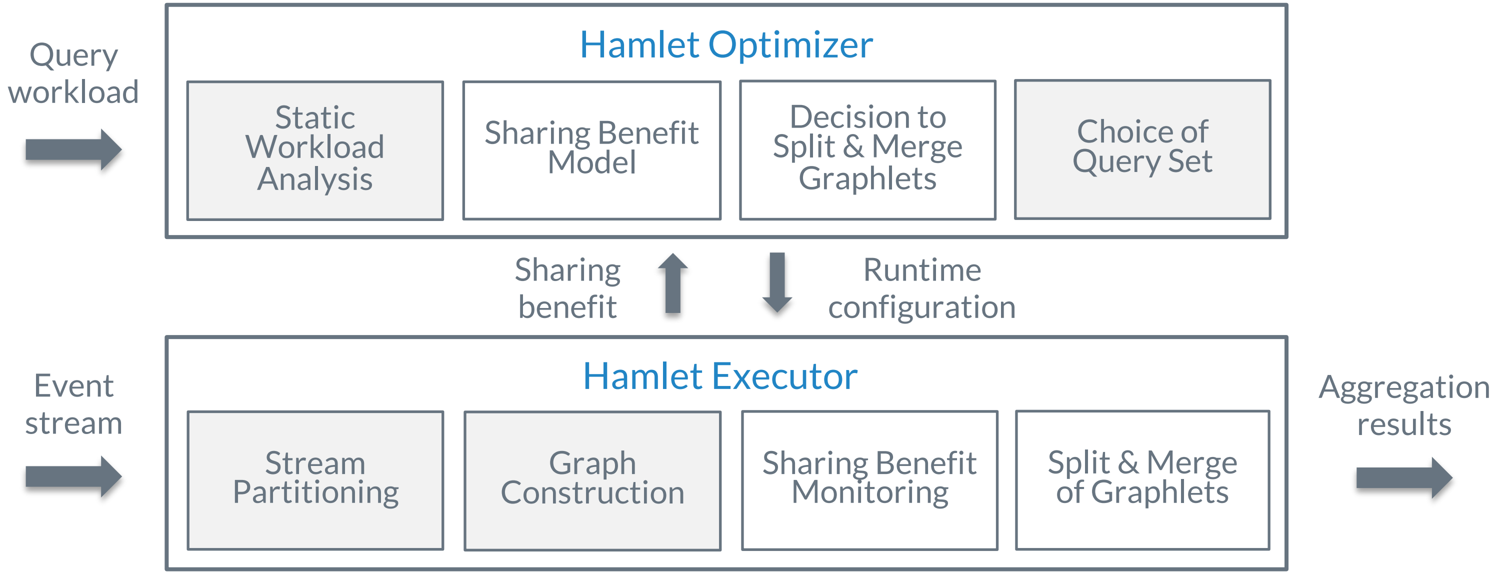
k – # queries,

s – # snapshots,

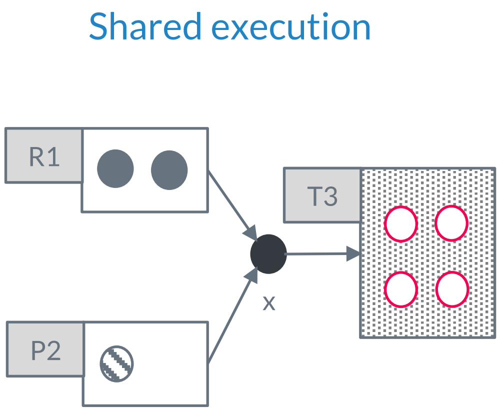
g – # events per graphlet,

t – # types per query

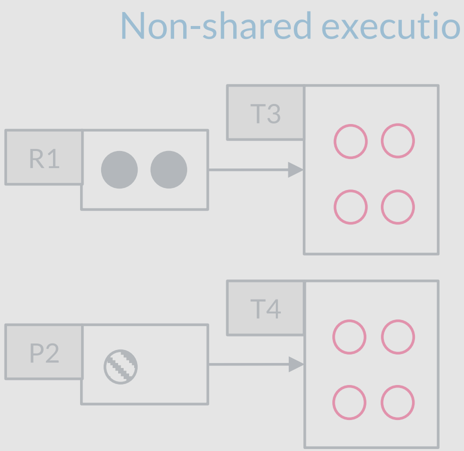
Hamlet Framework



Dynamic Sharing Decision



$$Shared(T_3, Q_T) = 4 * 7 * 1 + 1 * 2 * 4 * 2 = 44$$

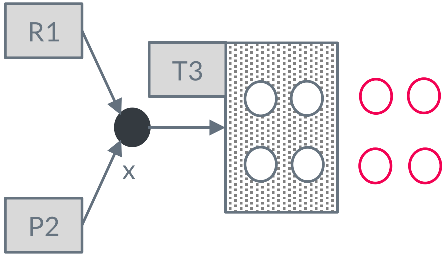


$$NonShared(\{T_3, T_4\}, Q_T) = 2 * 4 * 7 = 56$$

A burst is a set of consecutive events of type T, the processing of which can be shared by queries Q_T that contain a Kleene sub-pattern $T+$.
|Single event| ≤ |Burst| ≤ |Window|

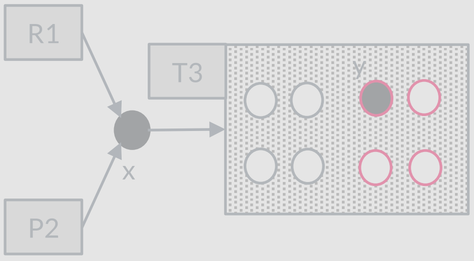
Dynamic Sharing Decision

Shared execution



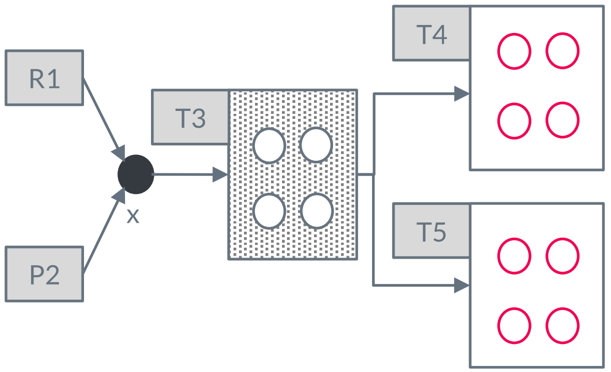
Dynamic Sharing Decision

Shared execution



$$Shared(T_3, Q_T) = 4 * 11 * 2 + 1 * 2 * 8 * 2 = 120$$

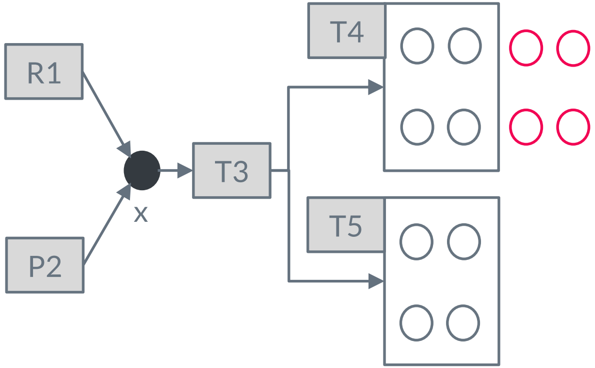
Non-shared execution



$$NonShared(\{T_4, T_5\}, Q_T) = 2 * 4 * 11 = 88$$

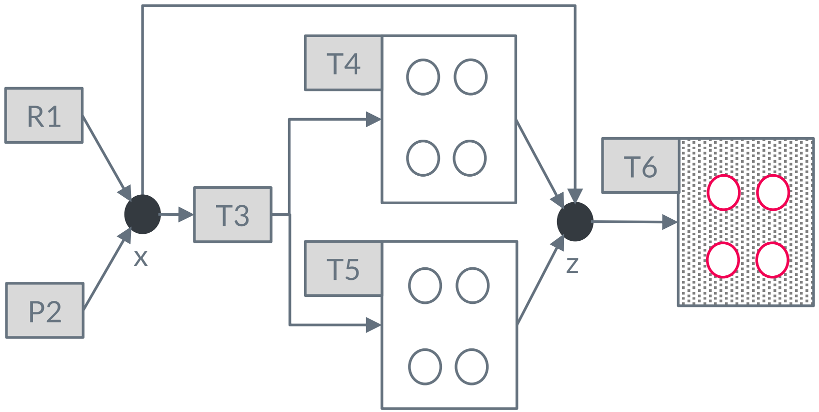
Dynamic Sharing Decision

Non-shared execution

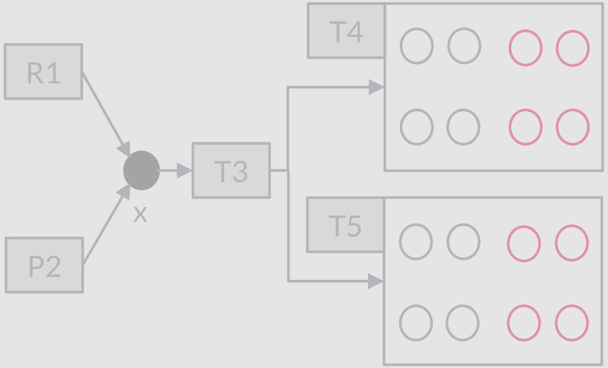


Dynamic Sharing Decision

Shared execution



Non-shared execution



$Shared(T_6, Q_T) = 4 * 15 * 1 + 1 * 2 * 4 * 2 = 76$

$NonShared(\{T_4, T_5\}, Q_T) = 2 * 4 * 15 = 120$

Merge creates one snapshot

Linear in # events per graphlet

Split comes for free!

Experiments

Experimental Setup

Infrastructure
Java 8, Ubuntu 14.04, 16 cores, 128GB

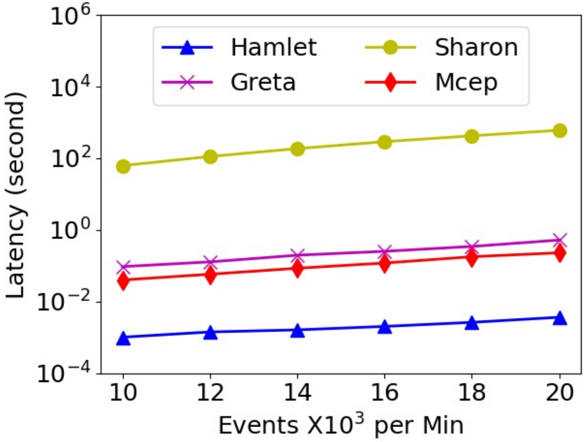
Data sets
- NYC taxi and Uber real data set
- Smart home real data set
- Stock real data set
- Ridesharing data set

Metrics
- Latency
- Throughput
- Peak memory

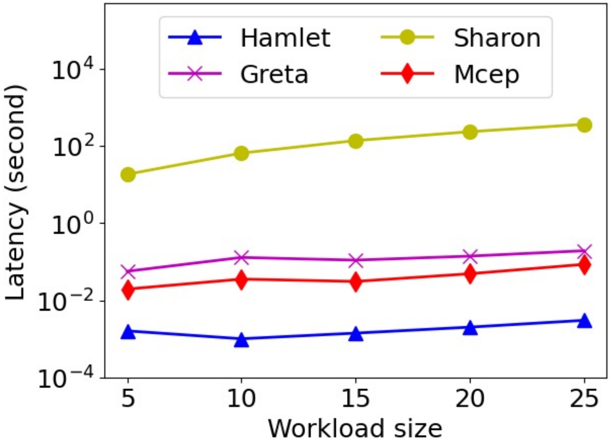
Cost factors
- Number of events per minute
- Number of queries

Approach	Kleene closure	Online aggregation	Sharing decisions
MCEP [SIGMOD'19]	✓	-	static
Sharon [ICDE'18]	-	✓	static
Greta [VLDB'17]	✓	✓	not shared
Hamlet [SIGMOD'21]	✓	✓	dynamic

Hamlet vs State-of-the-Art



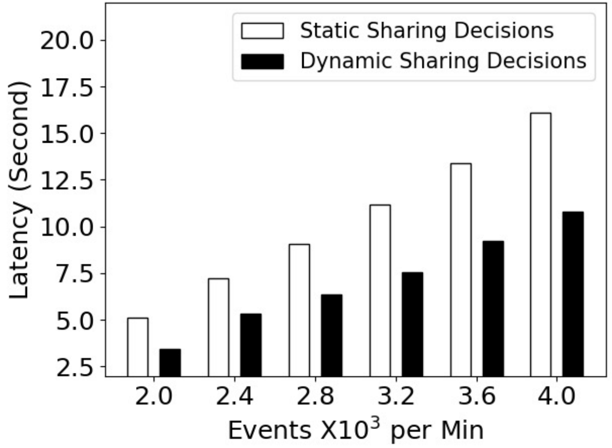
Ridesharing data set



Ridesharing data set

- Hamlet outperforms Sharon by 3-5 orders of magnitude, Greta by 1-2 orders of magnitude, and MCEP by 7-76X
- Hamlet terminates within 25 ms, Sharon – 50 min, Greta – 3 sec, MCEP – 1 sec

Dynamic vs Static Sharing Decisions



Stock real data

120 events per shared burst of event on avg
Number of graphlets is 400-600
Number of shared graphlets is 360-500

Static optimizer

Shared execution during the entire window
⇒ Number of snapshots is 10K-20K
⇒ Sharing overhead

Dynamic optimizer

10% of bursts is not shared
⇒ Number of snapshots is reduced by 50% (4K-8K)
⇒ 21-34% speed-up compared to static optimizer

Overhead:
400-600 sharing decisions per window within 20ms
0.2% of total latency per window

Conclusions

Hamlet integrates:

- Shared online trend aggregation strategy
- Dynamic sharing optimizer
  - Makes fine-grained sharing decisions per each
    - Sharable Kleene sub-pattern,
    - Burst of events, and
    - Subset of queries.
  - Switches between shared and non-shared execution at runtime

Hamlet achieves substantial performance gains compared to state-of-the-art

Acknowledgements

Chuan Lei
Researcher
IBM

Lei Ma
PhD student
WPI

Allison Rozet
SWE
MathWorks

Elke A. Rundensteiner
Professor
WPI

Funding agencies:
- NSF grants IIS-1815866, IIS-1018443, CRI-1305258
- U.S. Department of Education grant P200A150306
- U.S. Department of Agriculture grant 1023720

Thanks!

Questions?