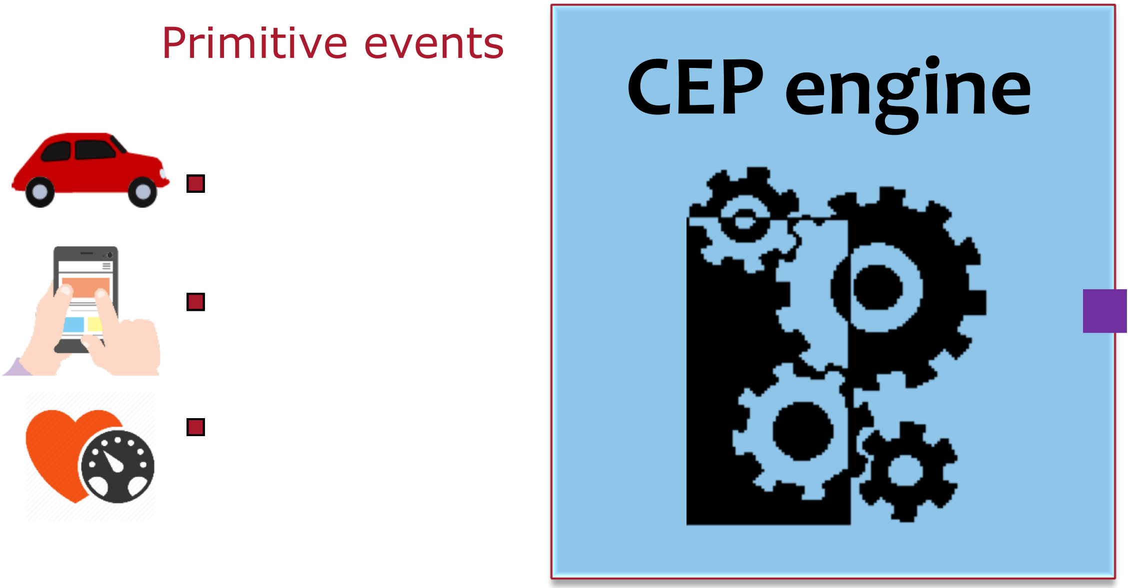
# Sharon: Shared Online Event Sequence Aggregation

Olga Poppe, Allison Rozet, Chuan Lei,
Elke A. Rundensteiner, and David Maier

April 18, 2018

# Complex Event Processing






**Input:** High-rate, potentially unbounded event stream

**Output:** Reliable summarized insights about the current situation in real time

# Motivating Example: Traffic Analytics



INPUT

**Event Sequence Aggregation Queries**

$q_1$: **RETURN COUNT**(*)
**PATTERN** OakSt, MainSt, StateSt
**WHERE** [vehicle] **WITHIN** 10 min **SLIDE** 1 min

$q_2$: **PATTERN** OakSt, MainSt, WestSt
$q_3$: **PATTERN** LindenSt, ParkAve, OakSt, MainSt
$q_4$: **PATTERN** ParkAve, OakSt, MainSt, WestSt

**Event Stream**

**Position report event**

- Vehicle id
- Location
- Time stamp
- Speed

# Problem



**Event Sequence Aggregation Queries**

**Event Stream**

The aggregation of **which sub-patterns** should be shared to process the workload with **minimal latency**?

# State-of-the-Art



| | Non-Shared | Shared |
|---|---|---|
| Two-step | **Flink, SASE, Cayuga, ZStream**<br>1. Event sequence construction<br>2. Event sequence aggregation | **SPASS, ECube**<br>1. Event sequence construction<br>2. Event sequence aggregation |
| Online | **A-Seq, GRETA**<br>Event sequence aggregation | **Sharon**<br>Event sequence aggregation |


**Flink**. https://flink.apache.org/
**SASE**. H. Zhang, Y. Diao, and N. Immerman. On complexity and optimization of expensive queries in Complex Event Processing. In SIGMOD, pages 217-228, 2014.
**Cayuga**. A. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, and W. White. Cayuga: A general purpose event monitoring system. In CIDR, pages 412-422, 2007.
**ZStream**. Y. Mei and S. Madden. ZStream: A Cost-based Query Processor for Adaptively Detecting Composite Events. In SIGMOD, pages 193-206, 2009.
**A-Seq**. Y. Qi, L. Cao, M. Ray, and E. A. Rundensteiner. Complex event analytics: Online aggregation of stream sequence patterns. In SIGMOD, pages 229-240, 2014.
**GRETA.** O.Poppe, C. Lei, E. A. Rundensteiner, and D. Maier. GRETA: Graph-based Real-time Event Trend Aggregation. In VLDB, pages 80-92, 2018.
**SPASS**. M. Ray, C. Lei, and E. A. Rundensteiner. Scalable pattern sharing on event streams. In SIGMOD, pages 495-510, 2016.
**ECube**. M. Liu, E. A. Rundensteiner, et al. E-Cube: Multi-dimensional event sequence analysis using hierarchical pattern query sharing. In SIGMOD, pages 889-900, 2011.

# Challenges



**Online yet shared event sequence aggregation**:

Sharing requires sequence construction $\Rightarrow\Leftarrow$ Online skips sequence construction

**Trade-off between sharing and not sharing**:

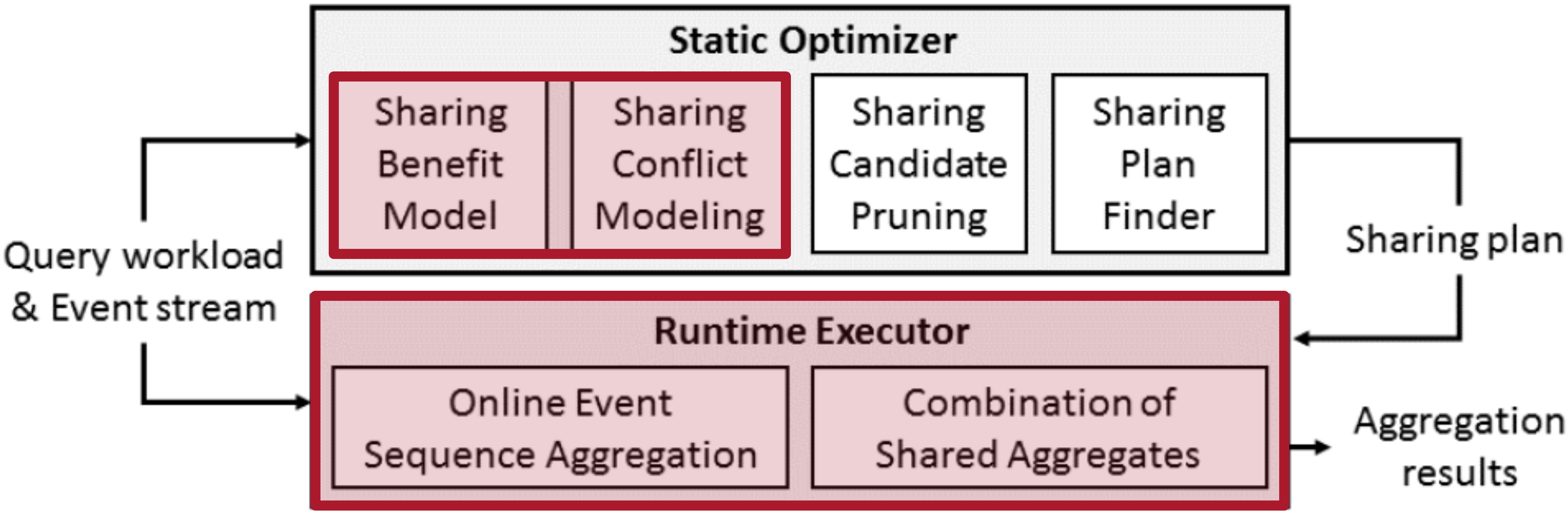
Sharing introduces overhead to combine intermediate aggregates

**Intractable sharing plan search space**:

Exponential in the number of sharing candidates

# Sharon Approach





# Non-Shared Online Aggregation



Pattern from $q_1$: OakSt, MainSt, StateSt

| **Counts** | **Event stream** | | | | |
|---|---|---|---|---|---|
| | o1 | m2 | o3 | m4 | s5 |
| *count(OakSt)* | 1 | | 2 | | |
| *count(OakSt, MainSt)* | | 1 | | 3 | |
| *count(OakSt, MainSt, StateSt)* | | | | | **3** |

Non-shared:
- Maintains a count for each prefix of each **query pattern**
- Events are discarded
- Re-computation overhead

# Shared Online Aggregation



Pattern from $q_1$: [OakSt, MainSt], StateSt

| Counts | Event stream | | | | |
|---|---|---|---|---|---|
| | o1 | m2 | o3 | m4 | s5 |
| *count(OakSt)* | 1 | | 2 | | |
| *count(OakSt, MainSt)* | | 1 | | 3 | |
| *count(StateSt)* | | | | | 1 |

Shared:

- Maintains a count for each prefix of each **sub-pattern**
- Events are still discarded
- Count combination overhead

# Sharing Candidates



Pattern from $q_1$: OakSt, MainSt, StateSt

Pattern from $q_2$: OakSt, MainSt, WestSt

Pattern from $q_3$: LindenSt, ParkAve, OakSt, MainSt

Pattern from $q_4$: ParkAve, OakSt, MainSt, WestSt

**Pattern**: p1=(OakSt, MainSt)
**Queries**: q1,q2,q3,q4 **Benefit**: 25

Benefit =
Cost of not sharing
- Cost of sharing

# Sharing Conflict



Pattern from $q_1$: OakSt, MainSt, StateSt

Pattern from $q_2$: OakSt, MainSt, WestSt

Pattern from $q_3$: LindenSt, ParkAve, OakSt, MainSt

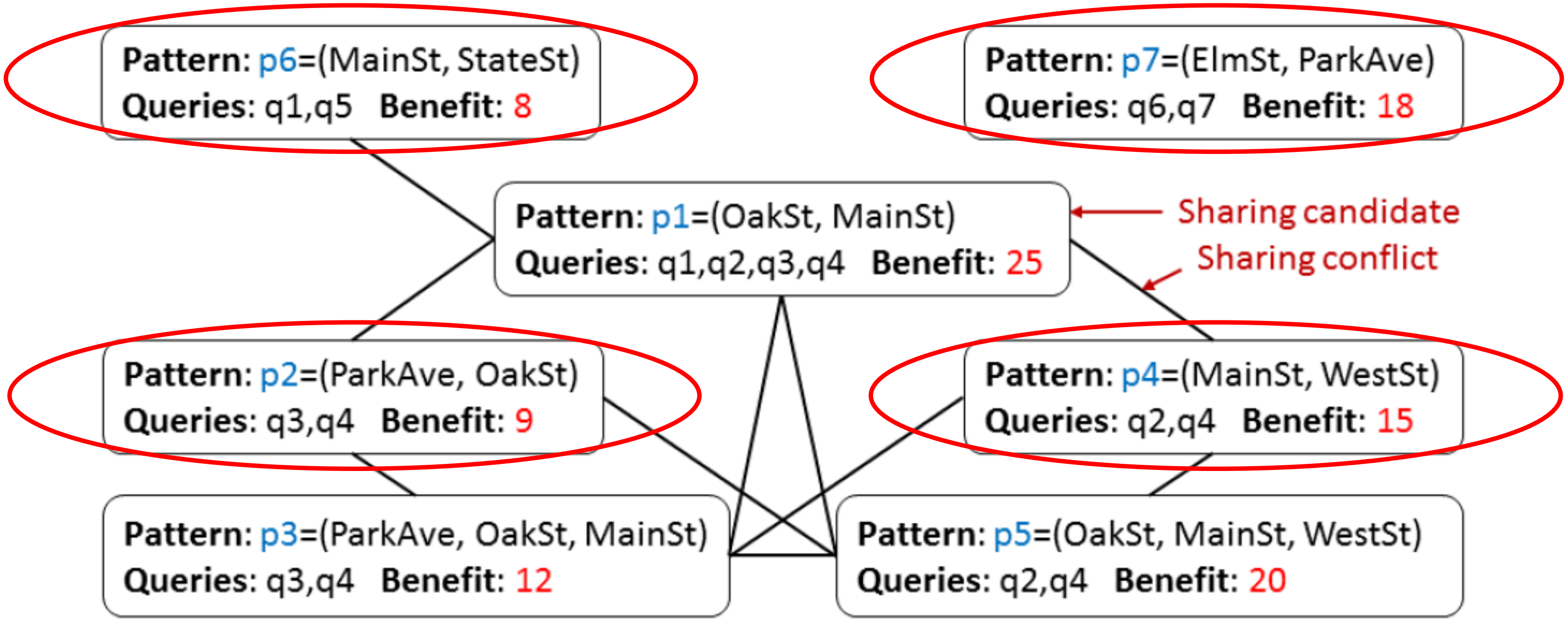
Pattern from $q_4$: ParkAve, OakSt, MainSt, WestSt

**Pattern**: p1=(OakSt, MainSt)
**Queries**: q1,q2,q3,q4 **Benefit**: 25

**Pattern**: p2=(ParkAve, OakSt)
**Queries**: q3,q4 **Benefit**: 25

# Sharing Conflict Modeling

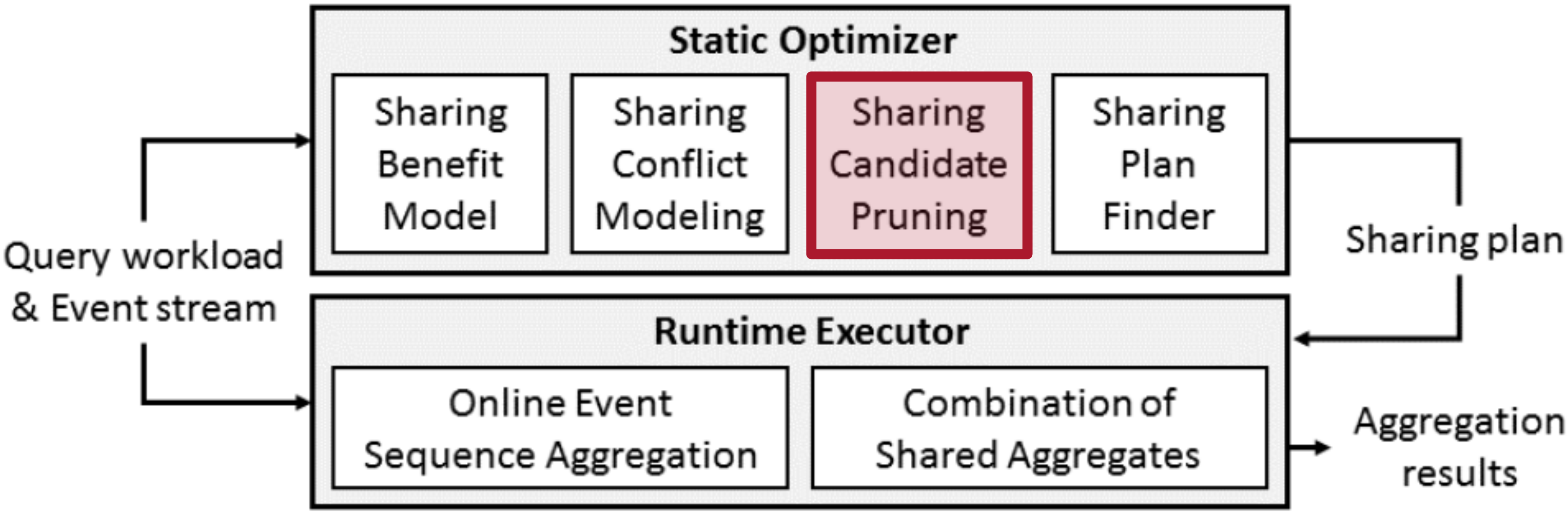





Optimal sharing plan = Maximum Weight Independent Set

# Sharon Approach





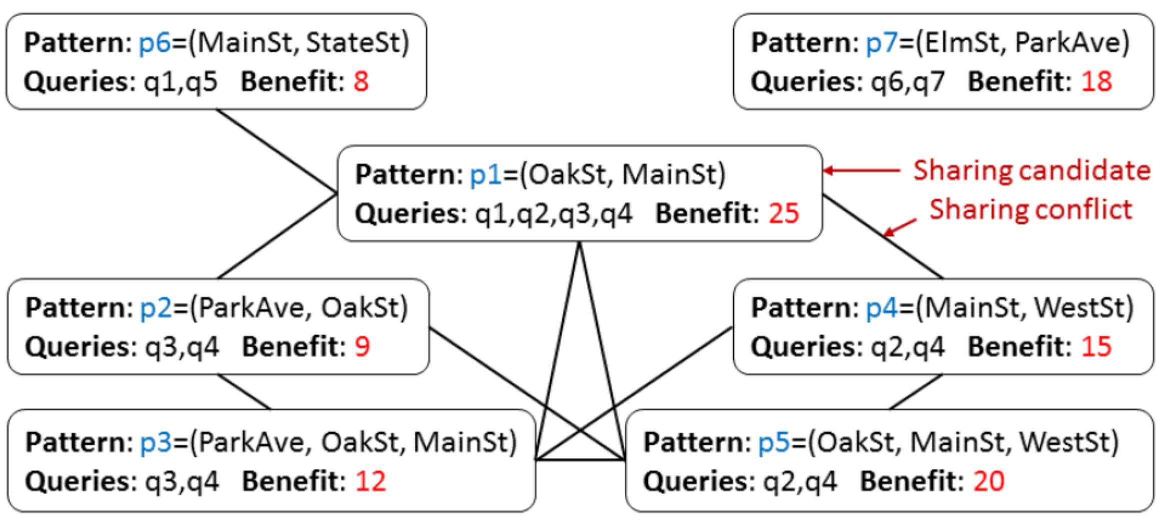
# Sharing Candidate Pruning



**Challenge:** Finding the optimal sharing plan is exponential in the number of vertices in the Sharon graph

**Sharon graph reduction principles**:

- Non-beneficial candidates
- Conflict-ridden candidates
- Conflict-free candidates

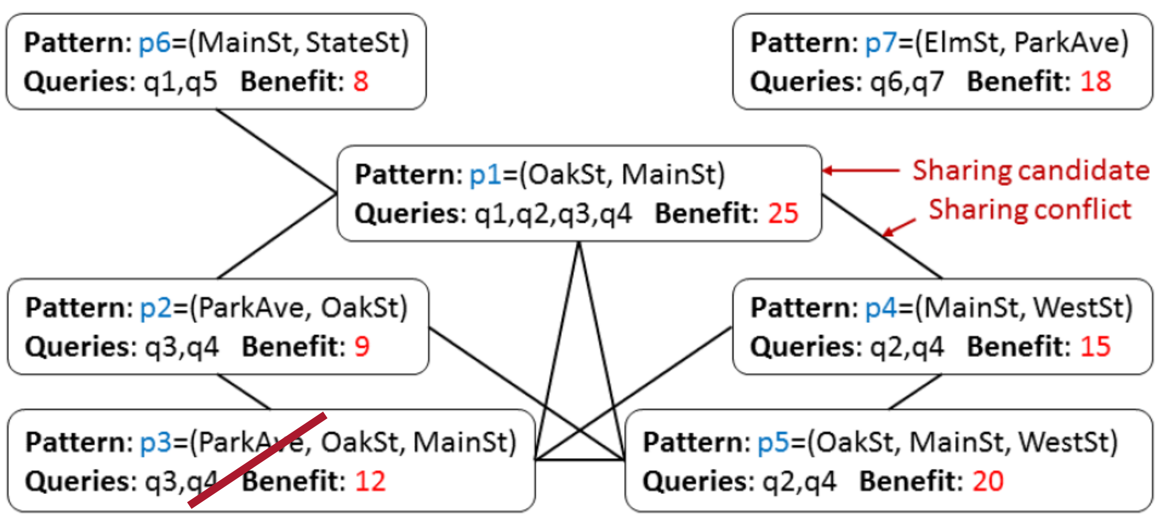


# Sharing Candidate Pruning



**Challenge:** Finding the optimal sharing plan is exponential in the number of vertices in the Sharon graph

**Sharon graph reduction principles**:
- Non-beneficial candidates
- Conflict-ridden candidates
- Conflict-free candidates

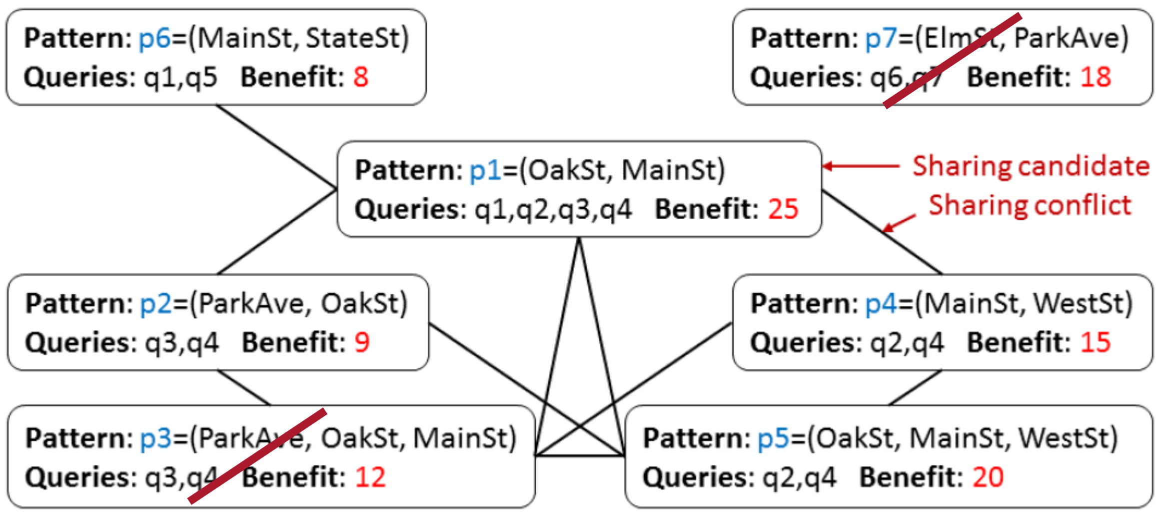


# Sharing Candidate Pruning



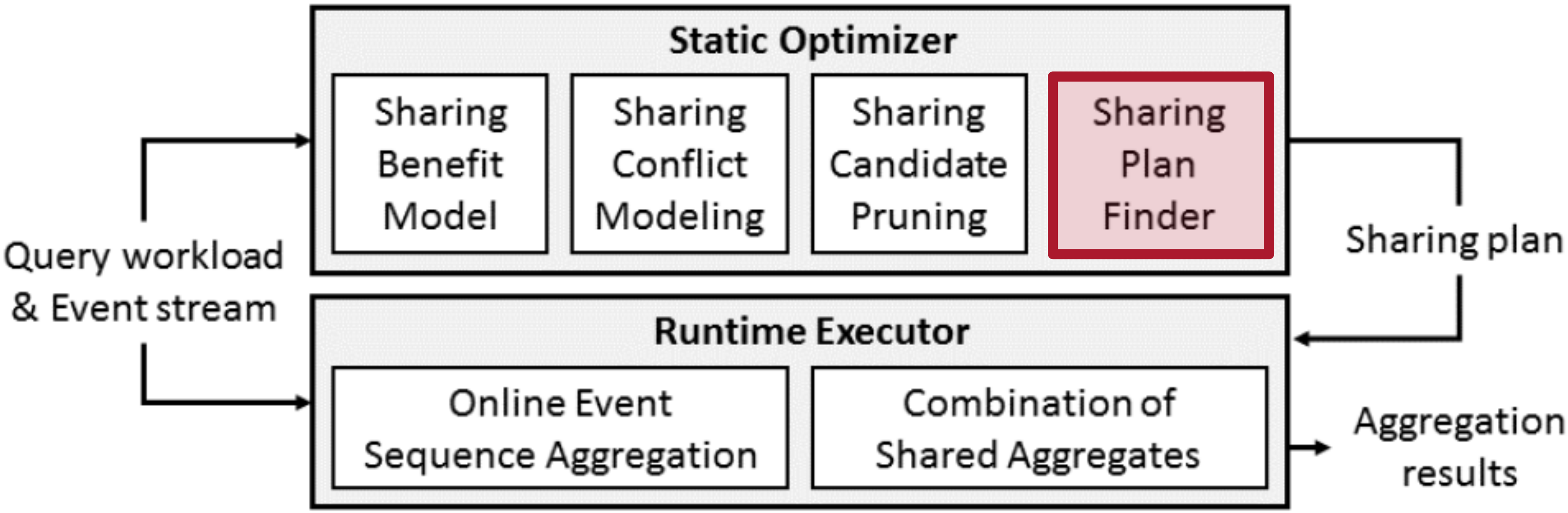
**Challenge:** Finding the optimal sharing plan is exponential in the number of vertices in the Sharon graph

**Sharon graph reduction principles**:
- Non-beneficial candidates
- Conflict-ridden candidates
- Conflict-free candidates



# Sharon Approach





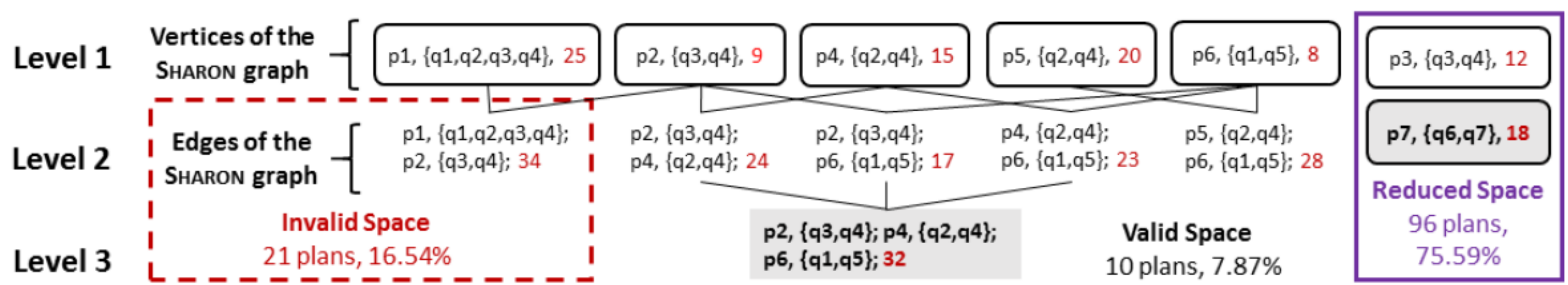
# Sharing Plan Finder



**Optimal sharing plan**

(p2, {q3,q4}), (p4, {q2,q4}), (p6, {q1,q5}), (p7, {q6,q7}): 50

**Sharing Plan Selection Algorithm**



# Experimental Setup



**Execution infrastructure**:

Java 7, 1 Linux machine with 16-core

3.4 GHz CPU and 128GB of RAM

**Data sets**:

- **TX**: NYC taxi real data set [1]

  Event sequences = Vehicle trajectories

- **LR**: Linear road benchmark data set [2]

  Event sequences = Vehicle trajectories

- **EC**: E-commerce synthetic data set

  Event sequences = Items added

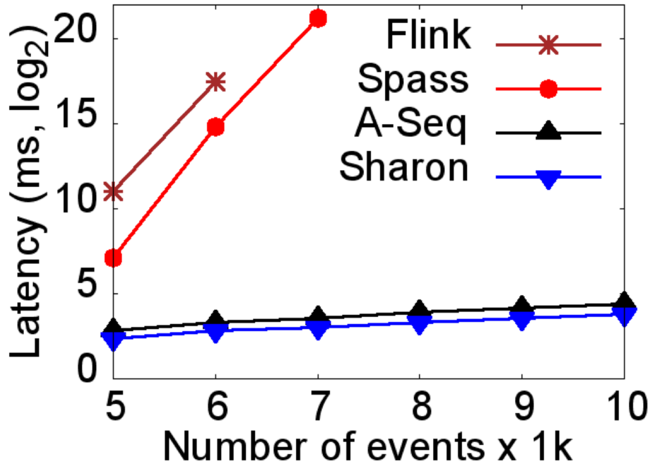
[1] Unified New York City Taxi and Uber data. https://github.com/toddwschneider/nyc-taxi-data

[2] A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. S. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. Linear road: A stream data management benchmark. In VLDB, pages 480-491, 2004.

# Sharon versus State-of-the-Art

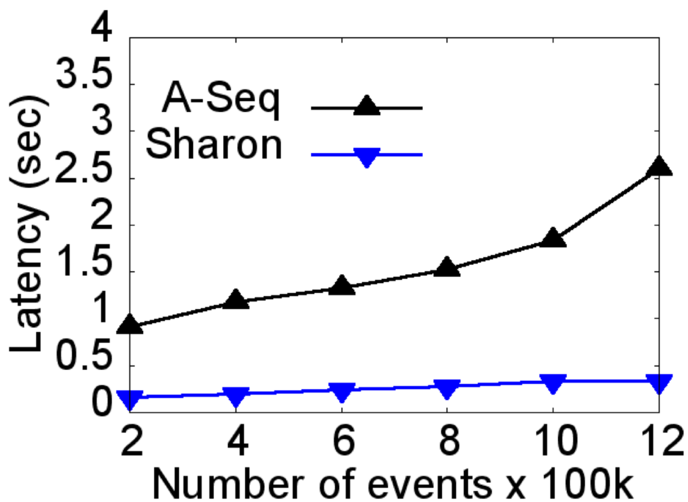


**Latency of two-step approaches**




Linear Road data set

**Latency of online approaches**




Taxi real data set

- The online approaches achieve **5 orders of magnitude** speed-up compared to the two-step approaches
- Sharon achieves up to **18-fold** speed-up compared to A-Seq

# Conclusions



- **Real-time** processing of event sequence aggregation queries due to
  - Sharing of intermediate aggregates
  - Online aggregation
- Effective **pruning principles** reduce the search space of sharing plans
- **Optimal plan** guides the executor at runtime
- **18-fold speed-up** compared to state-of-the-art approaches

**Thank You**

# Supplementary Slides

# Optimizer Algorithms



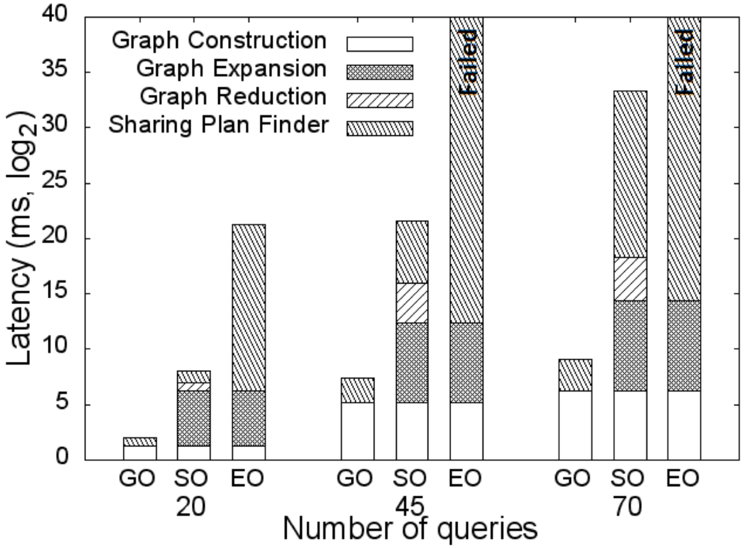
| **Phases** | **GO**: Greedy | **EO**: Exhaustive | **SO**: Sharon |
|---|---|---|---|
| Graph construction | + | + | + |
| Graph expansion | - | + | + |
| Graph reduction | - | - | + |
| Sharing plan finder | + | + | + |

- Greedy selects vertices in the graph with maximal ratio of benefit to number of conflicts
- Exhaustive traverses the entire search space
- Sharon reduces the graph and excludes the invalid search space

# Sharing Plan Selection Algorithms

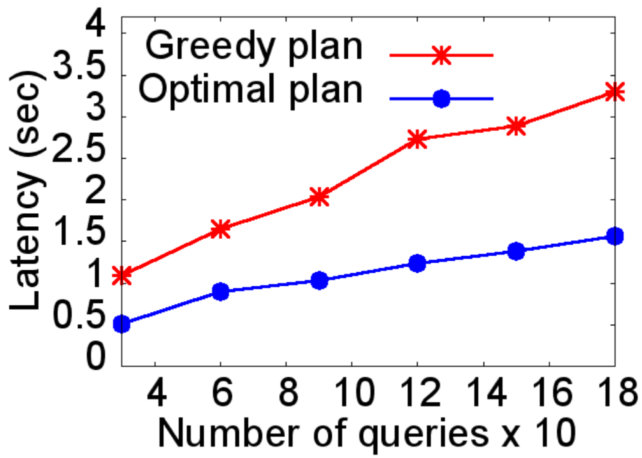


## Optimizer algorithms




E-commerce data set

## Quality of sharing plan




Taxi real data set

- Sharon optimizer is **3 orders of magnitude faster** than exhaustive search (20 queries) but **3 orders of magnitude slower** than greedy (70 queries)
- Executor latency is reduced **2-fold** when processed with an optimal plan rather than a greedy plan (180 queries)