Complex Event Processing

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

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

**Motivation** | **Optimizer** | **Evaluation** | **Conclusion**
Motivating Example: Traffic Analytics

Event Sequence Aggregation Queries

INPUT

Event Stream

Position report event
- Vehicle id
- Location
- Time stamp
- Speed

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

$q_2$: \textbf{PATTERN OakSt, MainSt, WestSt}

$q_3$: \textbf{PATTERN LindenSt, ParkAve, OakSt, MainSt}

$q_4$: \textbf{PATTERN ParkAve, OakSt, MainSt, WestSt}
The aggregation of which sub-patterns should be shared to process the workload with minimal latency?
State-of-the-Art

<table>
<thead>
<tr>
<th></th>
<th>Non-Shared</th>
<th>Shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-step</td>
<td>Flink, SASE, Cayuga, ZStream</td>
<td>SPASS, ECube</td>
</tr>
<tr>
<td>1. Event sequence construction</td>
<td>1. Event sequence construction</td>
<td></td>
</tr>
<tr>
<td>2. Event sequence aggregation</td>
<td>2. Event sequence aggregation</td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td>A-Seq, GRETA</td>
<td>Sharon</td>
</tr>
<tr>
<td>Event sequence aggregation</td>
<td>Event sequence aggregation</td>
<td></td>
</tr>
</tbody>
</table>

**Flink.** [https://flink.apache.org/](https://flink.apache.org/)
Challenges

**Online yet shared event sequence aggregation:**
Sharing requires sequence construction \(\Rightarrow\) 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

Motivation

Optimizer

Evaluation

Conclusion
Non-Shared Online Aggregation

Pattern from $q_1$: OakSt, MainSt, StateSt

<table>
<thead>
<tr>
<th>Counts</th>
<th>Event stream</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>o1</td>
</tr>
<tr>
<td>$\text{count(OakSt)}$</td>
<td></td>
</tr>
<tr>
<td>$\text{count(OakSt, MainSt)}$</td>
<td></td>
</tr>
<tr>
<td>$\text{count(OakSt, MainSt, StateSt)}$</td>
<td></td>
</tr>
</tbody>
</table>

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]

<table>
<thead>
<tr>
<th>Counts</th>
<th>Event stream</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>o1</td>
</tr>
<tr>
<td>$count(OakSt)$</td>
<td>1</td>
</tr>
<tr>
<td>$count(OakSt, MainSt)$</td>
<td></td>
</tr>
<tr>
<td>$count(StateSt)$</td>
<td></td>
</tr>
</tbody>
</table>

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: $p_1$=(OakSt, MainSt)
Queries: $q_1,q_2,q_3,q_4$  Benefit: 25

Benefit = Cost of not sharing - Cost of sharing
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: $p_1$** = (OakSt, MainSt)
**Queries:** q1, q2, q3, q4  **Benefit:** 25

**Pattern: $p_2$** = (ParkAve, OakSt)
**Queries:** q3, q4  **Benefit:** 25
Sharing Conflict Modeling

Optimal sharing plan = Maximum Weight Independent Set
Sharon Approach
**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
**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

Motivation

Optimizer

Evaluation

Conclusion

Worcester Polytechnic Institute
**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

- Motivation
- Optimizer
- Evaluation
- Conclusion
Sharing Plan Finder

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

Sharing Plan Selection Algorithm

Motivation  Optimizer  Evaluation  Conclusion
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

Sharon versus State-of-the-Art

Latency of two-step approaches

- 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.

Latency of online approaches
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

<table>
<thead>
<tr>
<th>Phases</th>
<th>GO: Greedy</th>
<th>EO: Exhaustive</th>
<th>SO: Sharon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph construction</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Graph expansion</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Graph reduction</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Sharing plan finder</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

- **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

- 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)