Moving Range Queries in Distributed Complex Event Processing

Boris Koldehoffe, Beate Ottenwälder, Kurt Rothermel, Umakishore Ramachandran
Moving Range Queries (MRQs)

- Key paradigm for mobile users
  - Range query parameterized to user’s location
  - Traffic, logistics, simulation
- Focal object triggers updates for range of interest
- Range updates determine event streams propagated to the user

State of the art:
- Primary events streams:
  [Jayaram et al. 2010, Benzing et al. 2011, Cheema et al. 2010]
- **Difference for complex events?**
Moving Range Queries on Complex Events

- **Operator tree** $T$
  - Defines event streams and processing elements
  - Operators possibly distributed

**Key challenge:** Stateful Operators

- **Semantics** for event delivery
  - Range updates
- **Efficient deployment**
  - Low overhead at runtime
  - Low overhead for reconfiguration
Contributions

1) Semantics
   ◦ Highly accurate event processing
   ◦ Completeness and ordering in the presence of range updates

2) Efficient reconfigurations
   ◦ Multiple dependent operators

3) Optimizations
   ◦ Account temporal and spatial overlap of events

4) Event processing model
   ◦ Efficiently support for optimization
Outline

• Deployment problem of Mobile CEP
• Delivery semantics for mobile event streams
• Dynamic reconfigurations of operator trees
• Optimizations
• Conclusion
Deployment Possibilities

• Preconfiguration of operator trees
  ◦ No reconfigurations of operators
  ◦ Wasteful due to unnecessary processing
  ◦ German road network:
    ➔ 23,100,000 ranges in Germany
    ➔ traffic variations: 40% of all ranges, idle 95% of their deployment time

• One tree per consumer
  ◦ Only regions with a query are covered
  ◦ Stateful operators need to be “correctly” initialized
Example: Event processing on ranges

**BlockedLeft:** the left lane of a road is blocked for a defined threshold \( m \), if \( \geq m(3) \) out of \( n(4) \) events indicate a right movement.

**BlockedRight:** analogously.

---

State of \( \omega_\# \) at time \( t \) depends on time when the range update occurred.

---

Research Group
"Distributed Systems"

Universität Stuttgart
State of operator without reconfiguration

Initialization

Update at $t'$

Update at $t$

1) No range Update:

2) Immediate Range Update:

3) Update at $t'$:

Observe: State at time $t$ should only depend on events of the same range
Operator reinitialization

Reinitialization at \( t' \):

- Historic events: reconfiguration overhead
- Reconfiguration overhead in operator hierarchy?

historic

live

BlockedLeft

\[\emptyset\]
Dynamic Interest Queries (diq)

diq = \{ T, mo, R, \delta \}

- **T** = operator tree
- **mo** = mobile focal object
- **R** = range of interest
- **\delta** = lifetime

**Requirements:**

- **Temporal order** \( \forall e \in [t'(R_i), end(R_i)] \).
  - Subsequent processing of events
- **Spatial order.** Events are ordered by occurrence of range updates
  - Marker messages to separate events
- **Completeness.** Produced event stream is a maximum covering sequence of \([t'(R_i), end(R_i)]\)

\[
\begin{align*}
t'(R_{i+1}) & \quad \delta \\
\omega & \Rightarrow \\
ω_# & \quad ω_s \\
R_i & \quad R_{i+1} & \quad R_{i+2}
\end{align*}
\]
Finding initialization points

• Single operator
  ◦ Count
    ▪ BlockLeft requires m events
  ◦ Timespan
    ▪ (timestamp of earliest event of relevance)

• Operator tree
  ◦ Initialization of primary event streams
  ◦ Dependencies successors/predecessor

\[
\begin{align*}
\Delta(\omega_0) & \Delta(\omega_{\rightarrow}) \\
\Rightarrow R_{i+1} & \text{ produces (historic) events ahead} \\
t'(R_{i+1}) - \Delta(\omega_{\rightarrow}) & \text{ for a covering sequence}
\end{align*}
\]
Basic Operator reconfiguration process

Upon initialization points of primary streams are determined:

1. Sources produce marker message $M$
2. Operator $\omega$ receiving $M$ completes processing on stream until $M$
3. Operator $\omega$ produces itself marker $M$
Evaluations of deployment possibilities

• Accident in Stuttgart
  ◦ Simulated with Omnet++ and Sumo
    ▪ Realistic traffic and network simulations
  ◦ Area 7.7km x 3km
• Duration approximately 15 minutes (1000s)
  ◦ Single diq
  ◦ 1000 cars passed through an area of size
    ▪ ~0.8 cars/km in average (throughout the whole simulation)
    ▪ ~250 events/second of primary events
• **Goal:** Comparison of deployment overhead
Comparison of predeployment to reconfiguration

- Possible to accommodate up to 100-1000 diqs at cost of predeployment
- Higher side length also increases the overall event load for detected events
Optimizations

- Only few assumptions on event processing model so far
  - Timespan, count
  - Operator idle because a marker is in the selection
- Knowledge on operator properties
  - Use temporal overlap of incoming streams after range update
  - Use spatial overlap after range update
- Knowledge on operator state can yield significant benefits
  - Improve the estimate for the operator initialization
  - Detect dependencies between incoming event streams
Improving Reconfigurations

Upon initialization points of primary streams are determined:

1. Sources produce marker message $M$
2. Operator $\omega$ receiving $M$ completes processing on stream until $M$
3. Operator $\omega$ produces itself marker $M$

Analysis of operator’s state to avoid useless reconfigurations:

- E.g. no primary streams of $\omega_a$ available
- Only reconfigurations on the path $(\omega_s, \omega_\#, \omega_\rightarrow)$
- Extend protocol by a collect phase ($C$)
Gain from optimizations

- Additional reduction on event streams by up to 90%
Conclusions and Future Work

Approach

• Enables efficient reconfiguration of CEP operators with movements of consumers
• Even without special event processing model high potential
• Additional state information helps significantly to reduce reconfiguration overhead
  ◦ Partitioned window model (cf. paper) allows to describe such state information

Future: Optimizations on saving energy and bandwidth
• Placement of operators to reduce bandwidth
• Strong semantics to enable reuse between multiple operators
Questions?

Contact
Boris Koldehofe und Beate Ottenwälder,
IPVS
Universität Stuttgart
boris.koldehofe@ipvs.uni-stuttgart.de

Related Projects
CEPiL, SpoVNet, SimTech