SuSe: Summary Selection for Regular Expression Subsequence Aggregation over Streams

Steven Purtzel and Matthias Weidlich

Humboldt-Universität zu Berlin

purtzesc@hu-berlin.de

Problem Setting

Given ... :

- stream s = <e_1,e_2,..>
- summary size *n*
- RegEx query (⩛, τ, A)
  - ➜ ⩛ = AB*C
  - ➜ τ = 10 time units
  - ➜ A = COUNT
- (unknown) evaluation time points $\mathcal{E}$

... select a substream ...

Stream s:

A_1	A_2	B_3	C_4	B_5	C_6	A_7	B_8	C_9	A_{10}	$\cdots$

Summary Size: 5; **RegEx:** AB^*C; **Eval. Time Point:** 10; **Matches:** 28

Random $\mathcal{S}$:

B_3	B_5	C_6	C_9	A_{10}

Matches: 0

FIFO $\mathcal{S}$:

C_6	A_7	B_8	C_9	A_{10}

Matches: 2
$\{(A_7,C_9),(A_7,B_8,C_9)\}$

SuSe $\mathcal{S}$:

A_1	A_2	B_3	B_5	C_6

Matches: 8
$\{(A_1,C_6),\ (A_2,C_6),\ (A_1,B_3,C_6),$
$(A_2,B_3,C_6),\ (A_1,B_5,C_6),\ (A_2,B_5,C_6),$
$(A_1,B_3,B_5,C_6),\ (A_2,B_3,B_5,C_6)\}$

... such that:

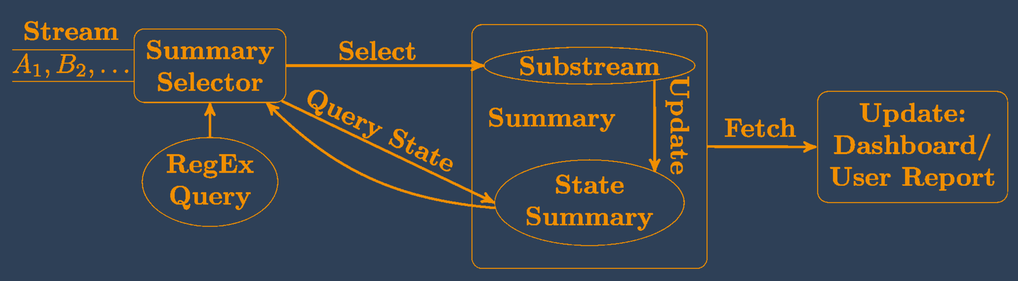
- at each evaluation point, the number of subsequence matches is maximized to minimize aggregation loss

$$minimize \sum_{\varepsilon \in \mathcal{E}} \left(|CM_{\mathcal{S}(\varepsilon)}| - |CM_{\mathcal{S}(\varepsilon,n)}| \right)$$

SuSe Architecture

1 Two Main Components:

1) **summary selector** & 2) **summary** (substream & StateSummary)



summary

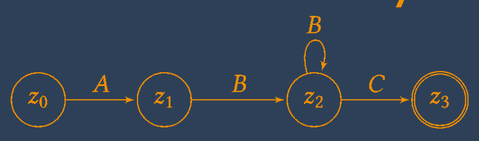
- ➜ consists of the selected substream and StateSummary
  - ➜ StateSummary holds aggregated information on (partial) matches within the substream and for each of its selected elements
  - ➜ maintain state consistency during element insertions and removals

summary selector

- ➜ queries the StateSummary to decide element insertions/replacements

2 StateSummary:

- ⩛ = AB^+C



$\#z_0 = \#z_1 = \#z_2 = \#z_3 = 0$;
$A : \#z_1 \leftarrow \#z_1 + 1$;
$B : \#z_2 \leftarrow \#z_2 + \#z_2 + \#z_1$;
$C : \#z_3 \leftarrow \#z_3 + \#z_2$

- count matches per state
  - ➜ global state counters
    - ➜ to quantify the current quality
- count matches per element per state
  - ➜ (active) local state counters
    - ➜ to quantify an element's current (and future) match participation
- derive count rules; state counter updates

Element Stream	A_1	A_1B_2	A_1B_3 / $A_1B_2B_3$	$A_1B_2C_4$ / $A_1B_3C_4$ / $A_1B_2B_3C_4$
	A_1	B_2	B_3	C_4

$A_1 : \#z_1 \leftarrow \#z_1 + 1 = 1$ $\quad B_3 : \#z_2 \leftarrow \#z_2 + \#z_2 + \#z_1 = 3$

$B_2 : \#z_2 \leftarrow \#z_2 + \#z_2 + \#z_1 = 1$ $\quad C_4 : \#z_3 \leftarrow \#z_3 + \#z_2 = 3$

Global State Counters
(0,0,0,0) A_1➜ (0,1,0,0) B_2➜ (0,1,1,0) B_3➜ (0,1,3,0) C_4➜ (0,1,3,3)
Local State Counters B_2
(0,0,1,0) B_3➜ (0,0,2,0) C_4➜ (0,0,2,2)

Operation	Time complexity	Space complexity
Initialize	$O(\|S\| \cdot \|Z\|)$	$O(\|S\| \cdot \|Z\|)$
Insert	$O(min(q_\tau, \|S\|) \cdot \|Z\|^2)$	$O(\|Z\|)$
Remove	$O(min(q_\tau, \|S\|)^2 \cdot \|Z\|^2)$	$O(min(q_\tau, \|S\|) \cdot \|Z\|)$

3 Active Time Window:

- substream segment within the current (sliding) time window
  - ➜ contains (active) partial matches with match potential
  - ➜ used to initialize (active) local state counters
  - ➜ crucial for state consistency and estimating future matches

Query $q = (\gamma, \tau, \mathscr{A})$; $q_\gamma = A(B^*C)^*D$; $q_\tau = 10$; $q_{\mathscr{A}} = \textsc{Count}$

Time:42 — Selected stream elements in $\mathcal{S}$

$\omega' = [\ldots, A_{11}, B_{13}, A_{14}, D_{15}, B_{17}, C_{18}, D_{19}]$
$\lambda' = [\ldots, 32, 35, 37, 38, 40, 41, 42]$
Active time window $\mathcal{A}$ of size q_τ

$|PM_\mathcal{A}| = 12$, $|CM_\mathcal{A}| = 10$, $\mathcal{A}_{\#z} = (0, 8, 4, 10)$

Time:43

$\omega' = [\ldots, A_{11}, B_{13}, A_{14}, D_{15}, B_{17}, C_{18}, D_{19}]$
$\lambda' = [\ldots, 32, 35, 37, 38, 40, 41, 42]$
Update $\mathcal{A}$ due to time step increase

$|PM_\mathcal{A}| = 4$, $|CM_\mathcal{A}| = 4$, $\mathcal{A}_{\#z} = (0, 3, 1, 4)$

Time:43 (B_{20} arrives with timestamp 43)

$\omega' = [\ldots, B_{13}, A_{14}, D_{15}, B_{17}, C_{18}, D_{19}, B_{20}]$
$\lambda' = [\ldots, 35, 37, 38, 40, 41, 42, 43]$
Update $\mathcal{A}$ due to adding B_{43}

$|PM_\mathcal{A}| = 8$, $|CM_\mathcal{A}| = 4$, $\mathcal{A}_{\#z} = (0, 3, 5, 4)$, $B_{20\#z} = (0, 0, 4, 0)$, $\mathcal{A}_{B_{20}\#z} = (0, 0, 4, 0)$

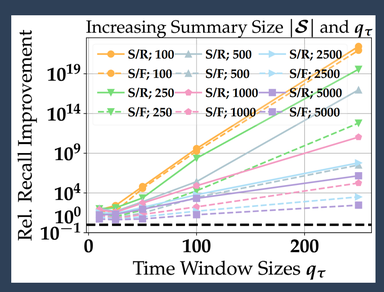
4 Selection Strategy

- goal: keep elements in substream that maximize RegEx matches
  - benefit(*e*) = present benefit(*e*) + expected benefit(*e*)
    - ➜ present: #complete matches *e* participates in
    - ➜ expected: #complete matches *e* *will* participate in
      - ➜ partial matches including *e* with future match potential
- idea: simulate how *e*'s (active) partial matches evolve with future arrivals
  - ➜ for the remaining time span where *e* stays within the window, e.g., Δt=2
    - ➜ t=0: (0, 5, 10, **15**); t=1: (0, 7.75, 13, **15.75**); t=2: (0, 10, 16.5, **17**)
      - ➜ accumulate expected match completions via counters on final states

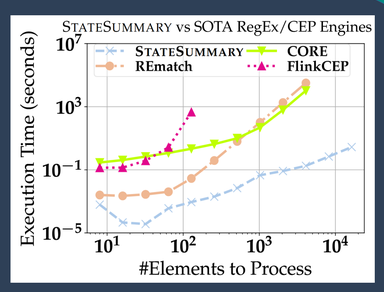
(Some) Evaluation Results

Effectiveness

- Random & FIFO baselines
  - ➜ $\text{RRI} = \frac{1}{|\mathcal{E}|} \sum_{\varepsilon \in \mathcal{E}} \frac{|CM_{\text{SuSe}}(\varepsilon)|}{|CM_{\text{base}}(\varepsilon)|}$

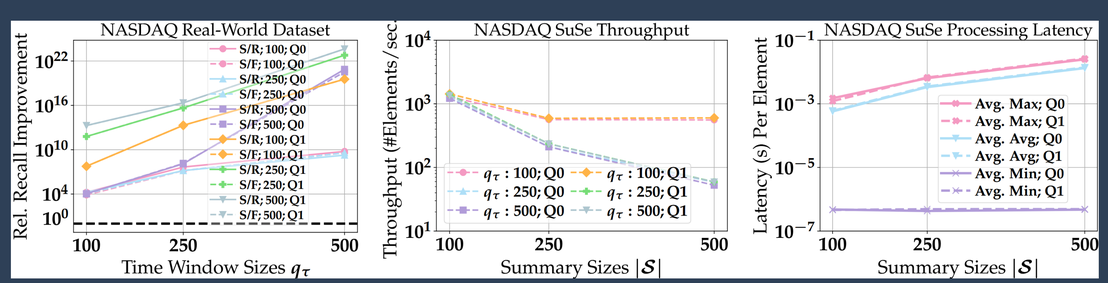


Efficiency



- FlinkCEP, CORE, REmatch
- NASDAQ ~ 450k elements
- Q0 = A(B|G)*A
- Q1 = A*G(A|B)*G*A

Real-world Dataset



Take Away

- SuSe: architecture for efficient RegEx subsequence summarization over streams
- leverages StateSummary to maintain query-specific, aggregated match summaries
- enables substream selection that minimizes aggregation loss by maximizing subsequence matches
- SuSe is orders of magnitude faster than leading RegEx and CEP engines, while producing richer substream aggregates than baselines