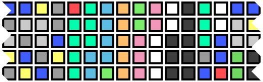
Scalable Online First-Order Monitoring

Joshua Schneider David Basin Frederik Brix
Srđan Krstić Dmitriy Traytel

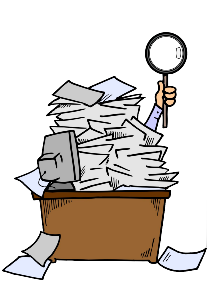
Department of Computer Science

ETH zürich

Online Monitoring



event stream

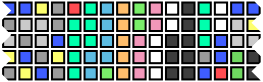


monitor

$$\square\, \forall u.\, \forall dt.\\ \quad \mathrm{insert}(u, \mathrm{db1}, dt) \wedge dt \not\approx \mathrm{unknown} \rightarrow\\ \quad \blacklozenge_{[0,1s)}\, \lozenge_{[0,30h]}\, \exists u'.\, \mathrm{insert}(u', \mathrm{db2}, dt) \vee \mathrm{delete}(u', \mathrm{db1}, dt)$$

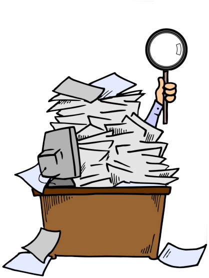
formal specification

Online Monitoring



event stream

in-order
unbounded

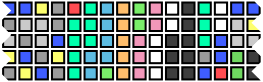


monitor

$$\square\, \forall u.\, \forall dt.\ \ \mathrm{insert}(u, \mathrm{db1}, dt) \wedge dt \not\approx \mathrm{unknown} \rightarrow \blacklozenge_{[0,1s)} \lozenge_{[0,30h]} \exists u'.\, \mathrm{insert}(u', \mathrm{db2}, dt) \vee \mathrm{delete}(u', \mathrm{db1}, dt)$$

formal specification

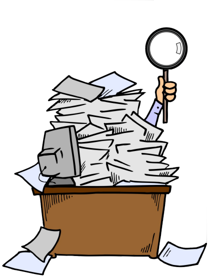
Online Monitoring



event stream

in-order

unbounded



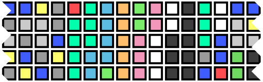
monitor

$$\square\, \forall u.\, \forall dt.\; \mathrm{insert}(u, \mathrm{db1}, dt) \wedge dt \not\approx \mathrm{unknown} \rightarrow \blacklozenge_{[0,1s)} \lozenge_{[0,30h]} \exists u'.\, \mathrm{insert}(u', \mathrm{db2}, dt) \vee \mathrm{delete}(u', \mathrm{db1}, dt)$$

formal specification

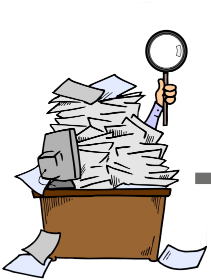
metric first-order temporal logic (MFOTL)

Online Monitoring

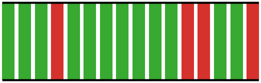


event stream

in-order
unbounded



monitor



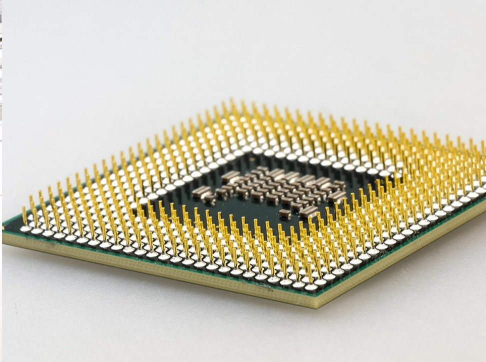
verdict stream

evaluated at every time-point

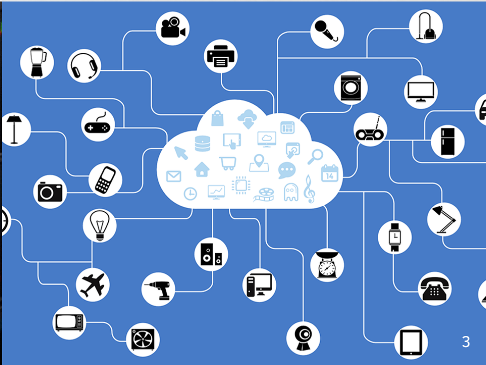
$$\Box \forall u.\, \forall dt.\ \mathrm{insert}(u, \mathrm{db1}, dt) \land dt \not\approx \mathrm{unknown} \rightarrow \blacklozenge_{[0,1s)} \Diamond_{[0,30h]} \exists u'.\, \mathrm{insert}(u', \mathrm{db2}, dt) \lor \mathrm{delete}(u', \mathrm{db1}, dt)$$

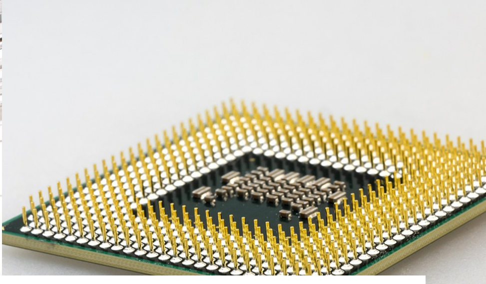
formal specification

metric first-order temporal logic (MFOTL)




DAX - Lang & Schwarz
ZEITEINHEIT
ZEITRAUM
INDIKATOR
SCANS
LS - DAX (846900)
5 Min





Challenge: Big Data

The Goal

Scalable online monitoring in the face of **high-velocity** event streams

The Goal

Scalable online monitoring in the face of **high-velocity** event streams

$$\text{velocity} = \text{event rate} = \frac{\#\text{events}}{\text{time}}$$

The Goal

Scalable online monitoring in the face of **high-velocity** event streams

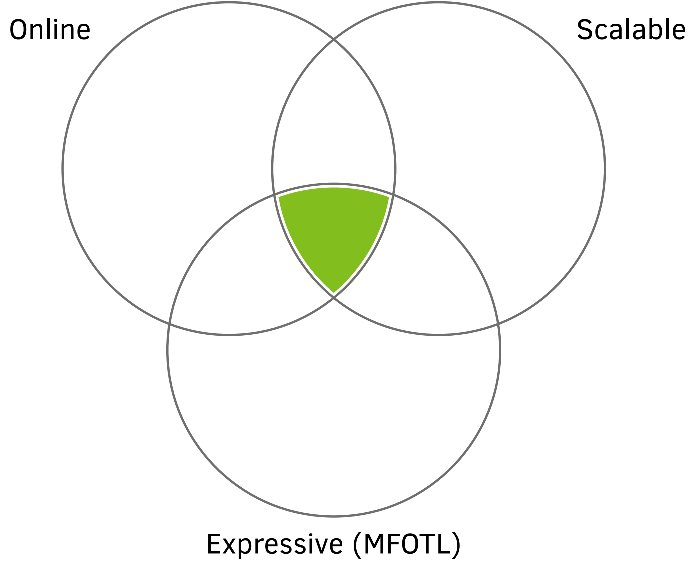
velocity = event rate = $\frac{\#\text{events}}{\text{time}}$

▲ number of processors ➡ ▲ throughput

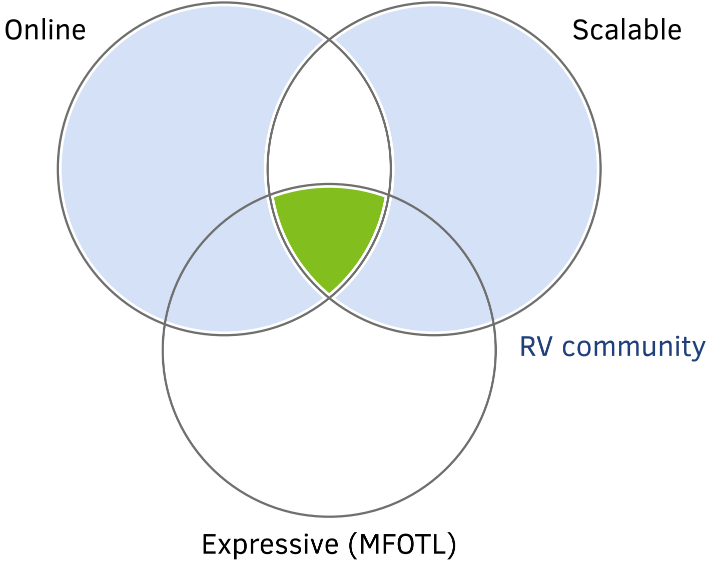
▼ peak latency

▼ memory used per processor

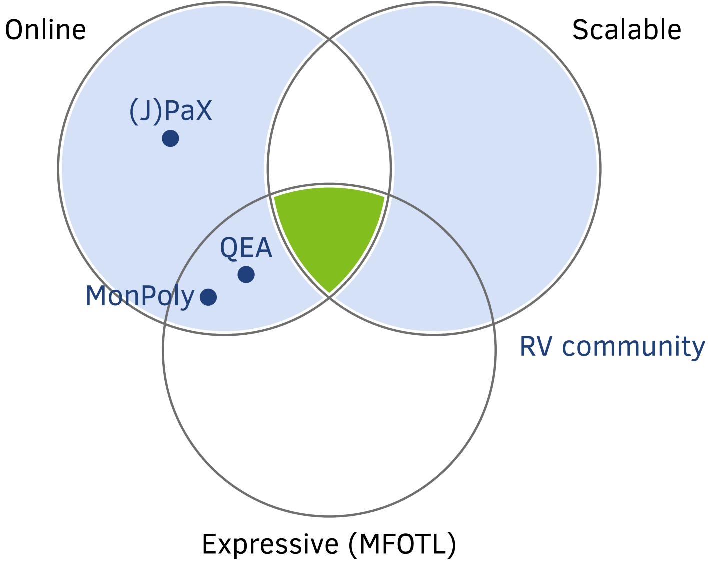
Related Work



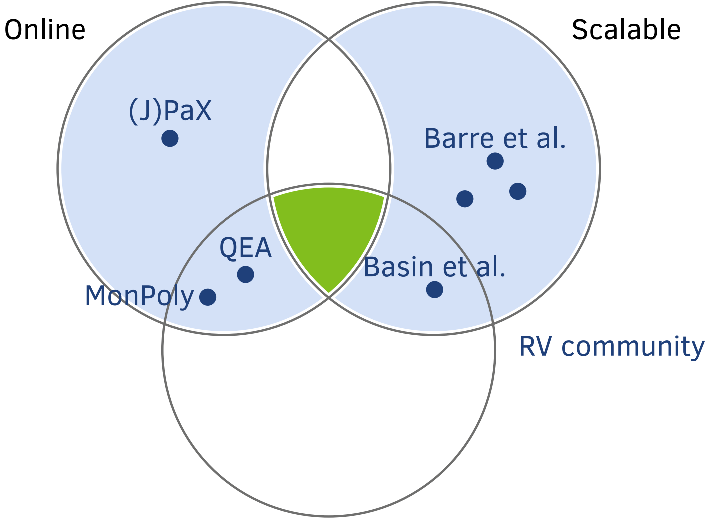
Related Work



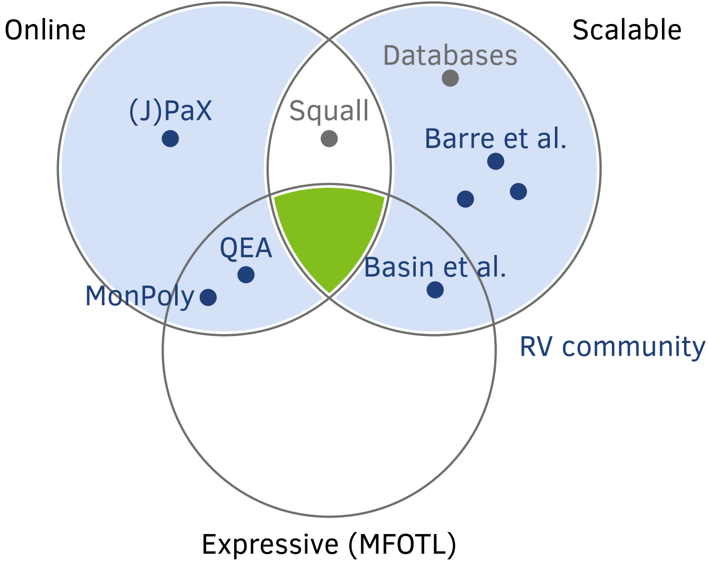
Related Work



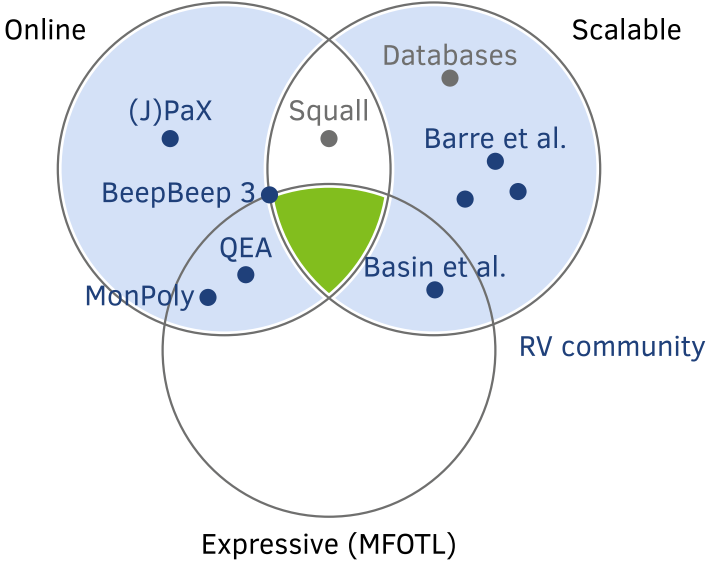
Related Work



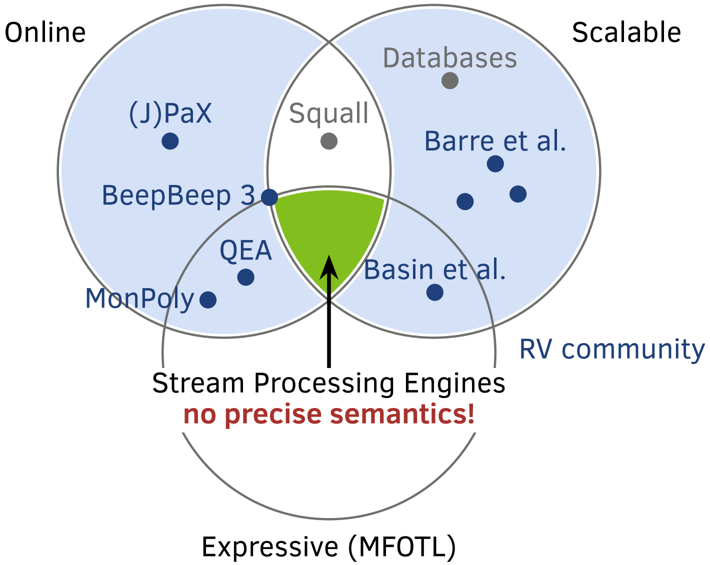
Related Work



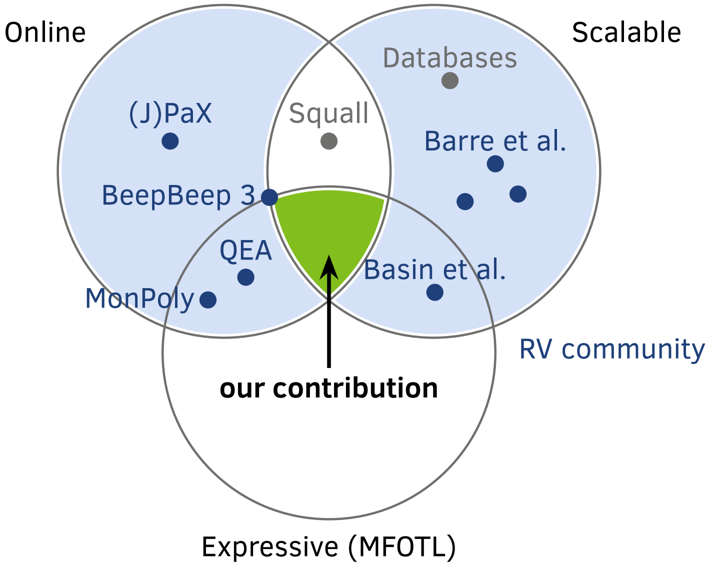
Related Work



Related Work



Related Work

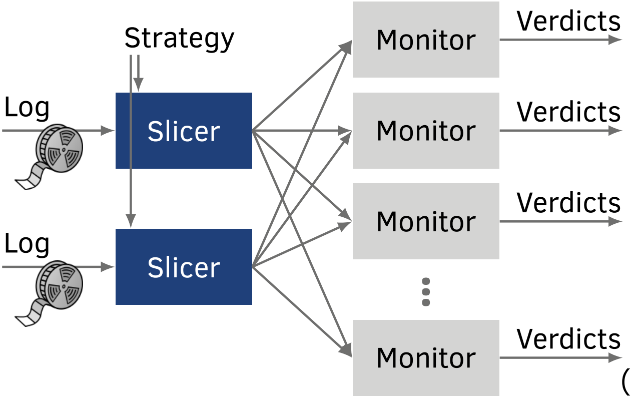


Slicing the Event Stream

- **Reuse** existing monitoring algorithm (MonPoly)

Slicing the Event Stream

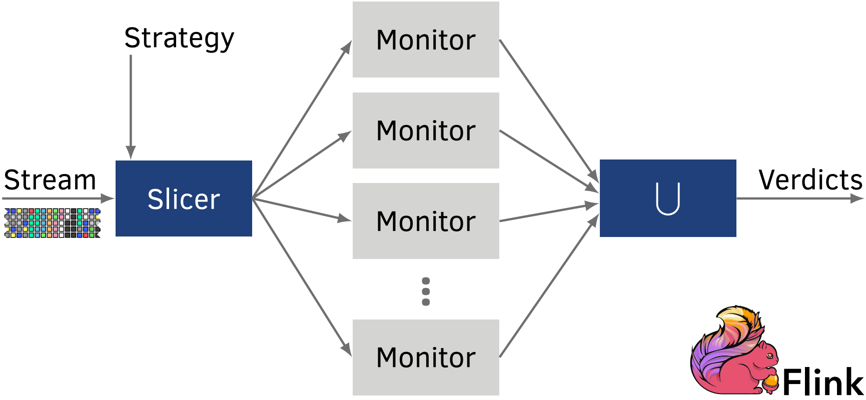
- **Reuse** existing monitoring algorithm (MonPoly)
- **Split** event stream into slices



MapReduce
(Basin et al. 2016)

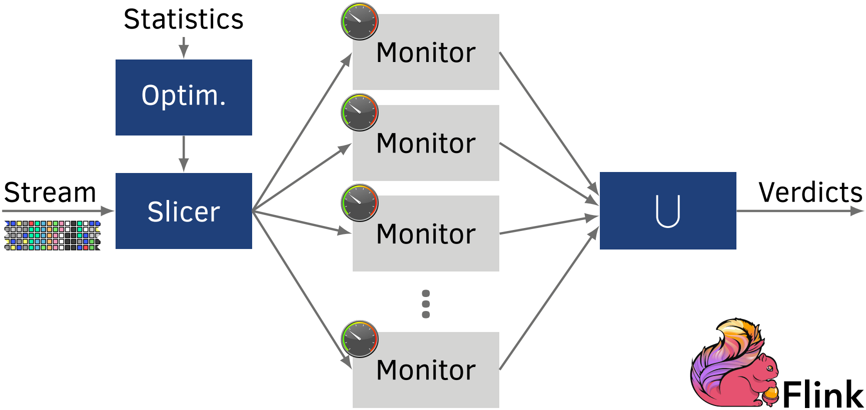
Slicing the Event Stream

- **Reuse** existing monitoring algorithm (MonPoly)
- **Split** event stream into slices



Slicing the Event Stream

- **Reuse** existing monitoring algorithm (MonPoly)
- **Split** event stream into slices



Prior Work: Offline Slicing

(Basin et al. 2016)

Example

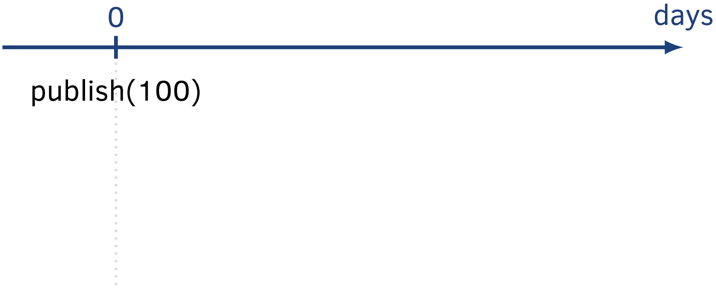
“A report must be published only if it has been approved in the past seven days.”

$$\text{publish}(r) \rightarrow \blacklozenge_{[0,7d)}\, \text{approve}(r)$$

Example

"A report must be published only if it has been approved in the past seven days."

$$\text{publish}(r) \rightarrow \blacklozenge_{[0,7d)}\, \text{approve}(r)$$



Example

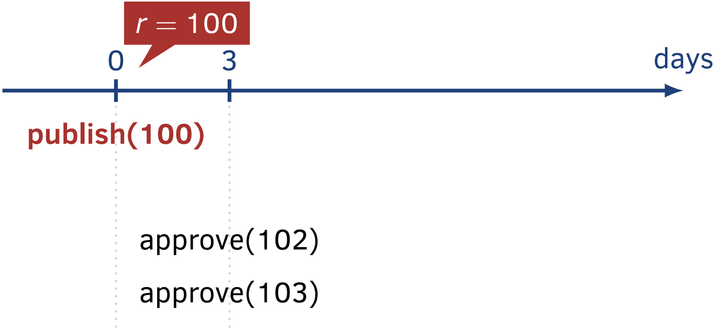
“A report must be published only if it has been approved in the past seven days.”

$$\text{publish}(r) \rightarrow \blacklozenge_{[0,7d)}\, \text{approve}(r)$$

Example

"A report must be published only if it has been approved in the past seven days."

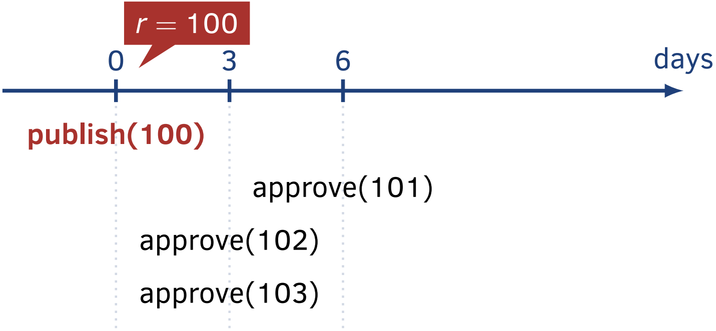
$$\text{publish}(r) \rightarrow \blacklozenge_{[0,7d)}\, \text{approve}(r)$$



Example

"A report must be published only if it has been approved in the past seven days."

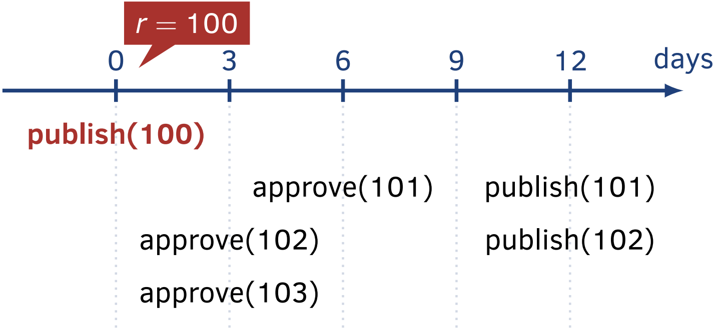
$$\text{publish}(r) \rightarrow \blacklozenge_{[0,7d)}\, \text{approve}(r)$$



Example

“A report must be published only if it has been approved in the past seven days.”

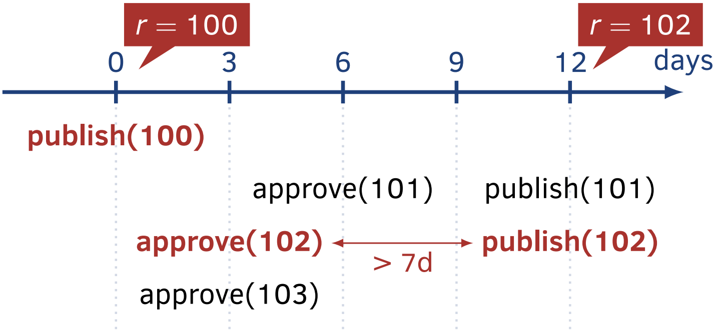
$$\mathrm{publish}(r) \rightarrow \blacklozenge_{[0,7d)}\, \mathrm{approve}(r)$$



Example

"A report must be published only if it has been approved in the past seven days."

$$\text{publish}(r) \rightarrow \blacklozenge_{[0,7d)}\, \text{approve}(r)$$



Data Slicer

"A report must be published only if it has been approved in the past seven days."

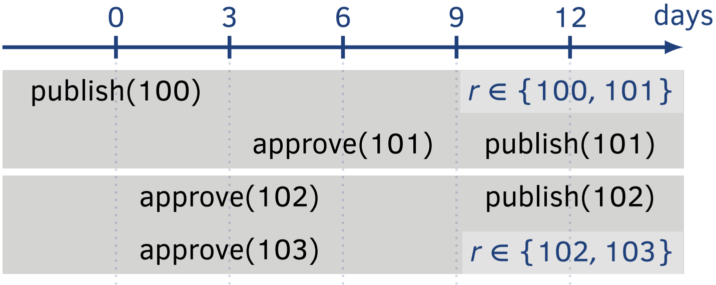
$$\text{publish}(r) \rightarrow \blacklozenge_{[0,7d)}\, \text{approve}(r)$$

slicing variable

Data Slicer

"A report must be published only if it has been approved in the past seven days."

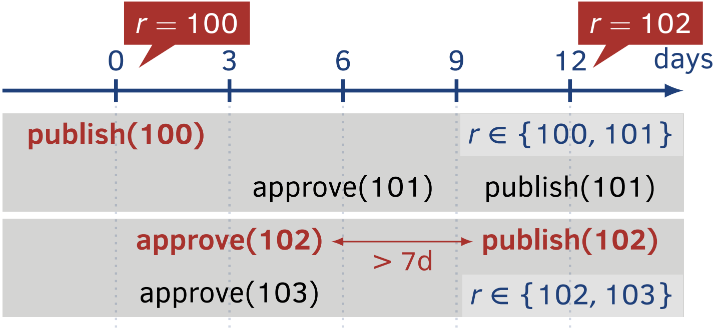
$$\text{publish}(r) \to \blacklozenge_{[0,7d)}\, \text{approve}(r)$$



Data Slicer

"A report must be published only if it has been approved in the past seven days."

$$\text{publish}(r) \rightarrow \blacklozenge_{[0,7d)}\, \text{approve}(r)$$



Problem 1: Too Much Data Duplication

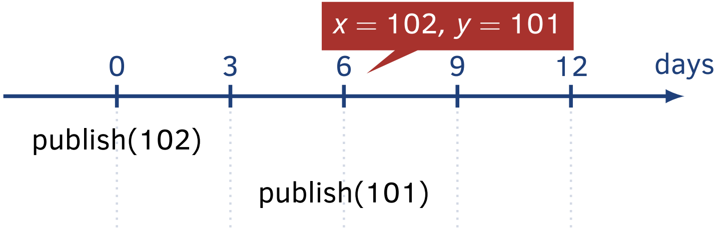
“IDs of published reports must increase over time.”

$$(\blacklozenge\, \text{publish}(x)) \wedge \text{publish}(y) \rightarrow x \leq y$$

Problem 1: Too Much Data Duplication

"IDs of published reports must increase over time."

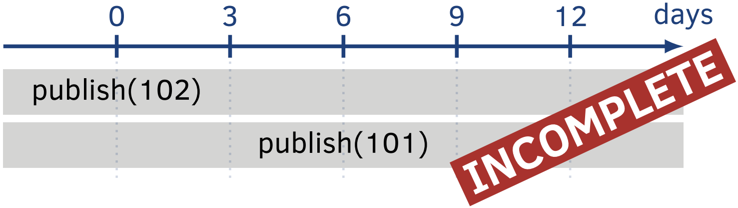
$$(\blacklozenge\,\mathrm{publish}(x)) \land \mathrm{publish}(y) \to x \leq y$$



Problem 1: Too Much Data Duplication

"IDs of published reports must increase over time."

$$(\blacklozenge\, \text{publish}(x)) \wedge \text{publish}(y) \rightarrow x \leq y$$

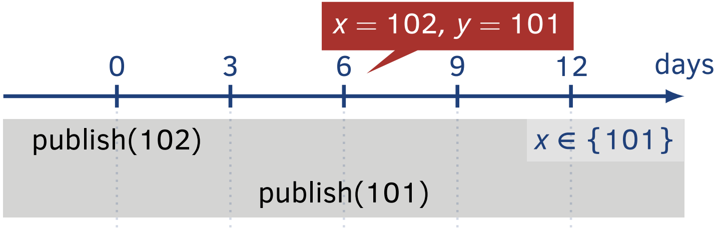


Completeness: all violations are detected
Soundness: all detected violations are true

Problem 1: Too Much Data Duplication

"IDs of published reports must increase over time."

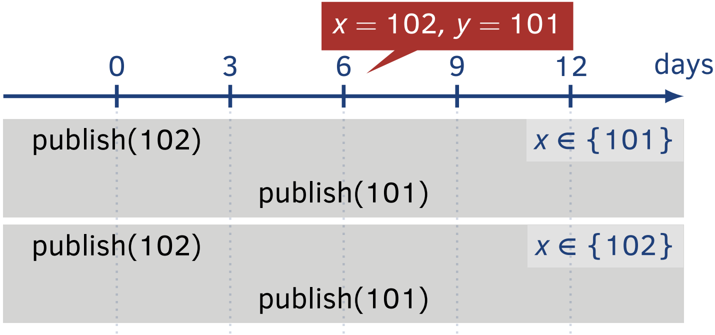
$$(\blacklozenge\, \text{publish}(x)) \land \text{publish}(y) \rightarrow x \leq y$$



Problem 1: Too Much Data Duplication

"IDs of published reports must increase over time."

$$(\blacklozenge\,\text{publish}(x)) \land \text{publish}(y) \rightarrow x \leq y$$



Problem 2: How to Slice?

Complex formulas:

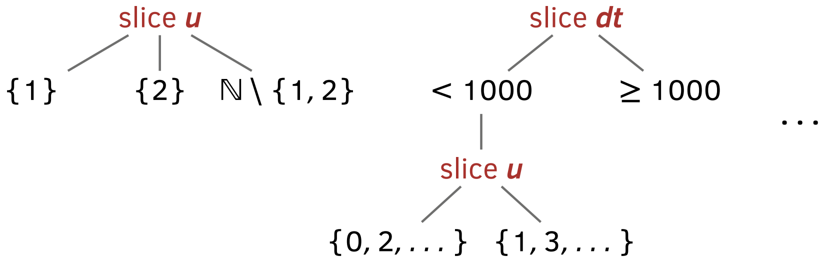
$$\begin{array}{l} \mathrm{delete}(\boldsymbol{u}, \mathrm{db1}, \boldsymbol{dt}) \wedge \boldsymbol{dt} \not\approx \mathrm{unknown} \rightarrow \\ \big(\blacklozenge_{[0,1s)} \lozenge_{[0,30h)} \exists u'.\, \mathrm{delete}(u', \mathrm{db2}, \boldsymbol{dt}) \big) \vee \\ \big((\lozenge_{[0,1s)} \blacklozenge_{[0,30h)} \exists u'.\, \mathrm{insert}(u', \mathrm{db1}, \boldsymbol{dt})) \wedge \\ \quad (\blacksquare_{[0,30h)} \square_{[0,30h)} \neg \exists u'.\, \mathrm{insert}(u', \mathrm{db2}, \boldsymbol{dt})) \big) \end{array}$$

Problem 2: How to Slice?

Complex formulas:

$$\mathrm{delete}(\boldsymbol{u}, \mathrm{db1}, \boldsymbol{dt}) \land \boldsymbol{dt} \not\approx \mathrm{unknown} \rightarrow$$
$$\big(\blacklozenge_{[0,1s)} \lozenge_{[0,30h)} \exists u'.\, \mathrm{delete}(u', \mathrm{db2}, \boldsymbol{dt})\big) \lor$$
$$\big((\lozenge_{[0,1s)} \blacklozenge_{[0,30h)} \exists u'.\, \mathrm{insert}(u', \mathrm{db1}, \boldsymbol{dt})) \land$$
$$(\blacksquare_{[0,30h)} \square_{[0,30h)} \neg \exists u'.\, \mathrm{insert}(u', \mathrm{db2}, \boldsymbol{dt}))\big)$$

Many possible choices:

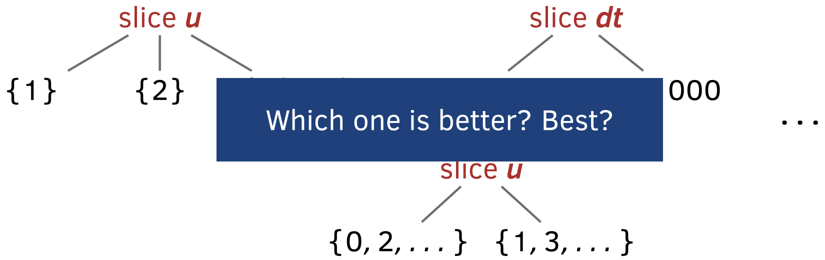


Problem 2: How to Slice?

Complex formulas:

$$\mathrm{delete}(\mathbf{u}, \mathrm{db1}, \mathbf{dt}) \land \mathbf{dt} \not\approx \mathrm{unknown} \rightarrow \\ \left(\blacklozenge_{[0,1s)} \lozenge_{[0,30h)} \exists u'.\, \mathrm{delete}(u', \mathrm{db2}, \mathbf{dt})\right) \lor \\ \left(\left(\lozenge_{[0,1s)} \blacklozenge_{[0,30h)} \exists u'.\, \mathrm{insert}(u', \mathrm{db1}, \mathbf{dt})\right) \land \\ \left(\blacksquare_{[0,30h)} \square_{[0,30h)} \neg\exists u'.\, \mathrm{insert}(u', \mathrm{db2}, \mathbf{dt})\right)\right)$$

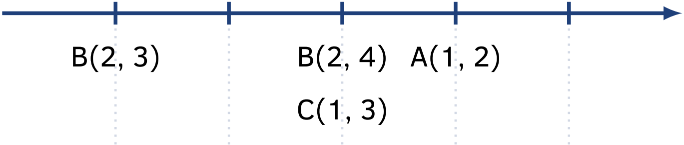
Many possible choices:



Our Solution

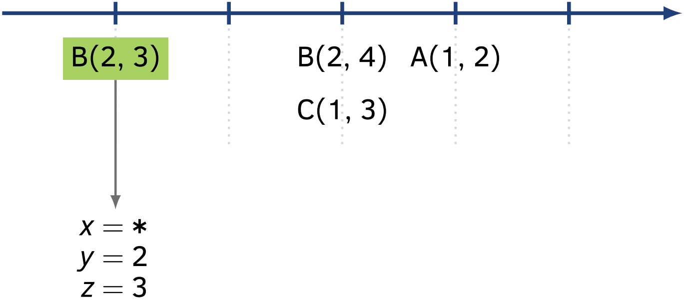
Solution 1: Joint Data Slicer

$$A(x, y) \land (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$



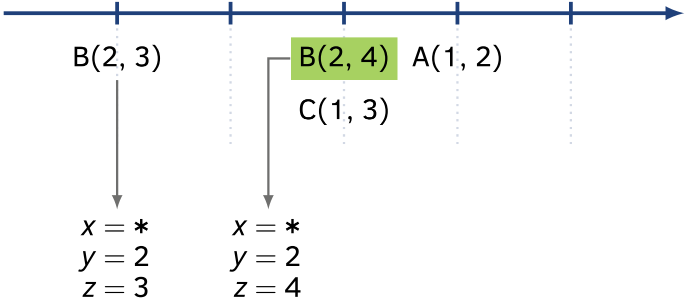
Solution 1: Joint Data Slicer

$$A(x, y) \land (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$



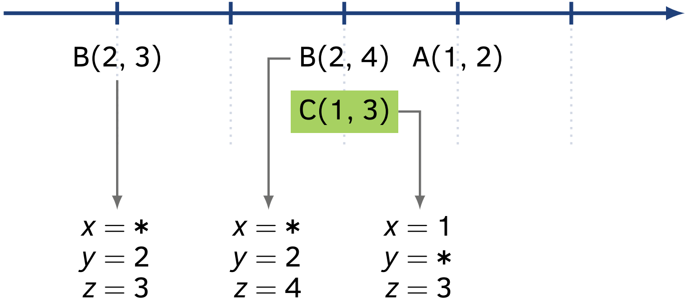
Solution 1: Joint Data Slicer

$$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$



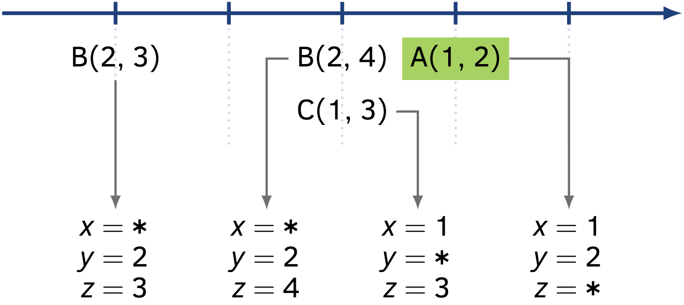
Solution 1: Joint Data Slicer

$$A(x,y) \wedge (\blacklozenge B(y,z)) \rightarrow \blacklozenge C(x,z)$$



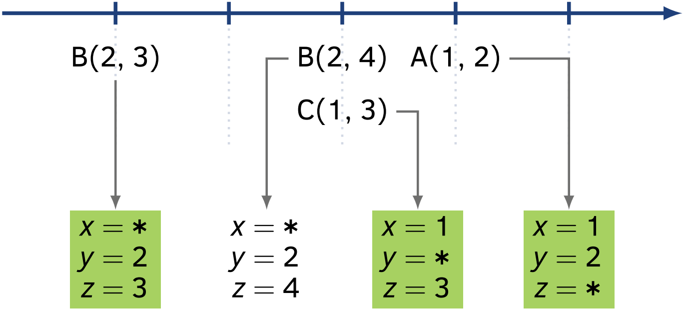
Solution 1: Joint Data Slicer

$$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$



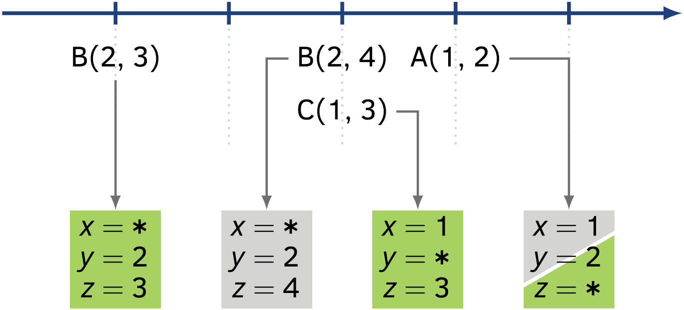
Solution 1: Joint Data Slicer

$$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$



Solution 1: Joint Data Slicer

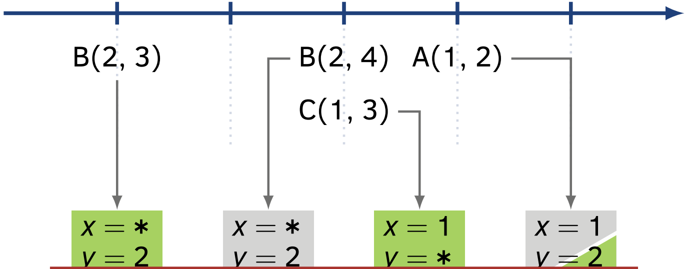
$$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$



sound & complete if ■ and ■ are each sent to the same slice

Solution 1: Joint Data Slicer

$$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$



need a mapping from assignments to slices

sound & complete if [green] and [grey] are each sent to the same slice

Key Observation

A slicing strategy for an MFOTL formula like

$$A(x, y) \land (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$

is sound & complete

Key Observation

A slicing strategy for an MFOTL formula like

$$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$

is sound & complete
if it is sound & complete for the conjunction of all atoms

$$A(x, y) \wedge B(y, z) \wedge C(x, z).$$

Key Observation

A slicing strategy for an MFOTL formula like

$$A(x, y) \land (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$

is sound & complete
if it is sound & complete for the conjunction of all atoms

$$A(x, y) \land B(y, z) \land C(x, z).$$

- This problem has been studied by the database community!
- Hypercube algorithm (Afrati and Ullman 2011, and others)

Key Observation

A slicing strategy for an MFOTL formula like

$$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$$

is sound & complete
if it is sound & complete for the conjunction of all atoms

$$A(x, y) \wedge B(y, z) \wedge C(x, z).$$

- This problem has been studied by the database community!
- Hypercube algorithm (Afrati and Ullman 2011, and others)
- Condition is only sufficient, not necessary

Solution 2: Hypercube

$\mathsf{A}(x, y) \wedge (\blacklozenge\, \mathsf{B}(y, z)) \rightarrow \blacklozenge\, \mathsf{C}(x, z)$

Free variables x, y, z

n monitors

Solution 2: Hypercube

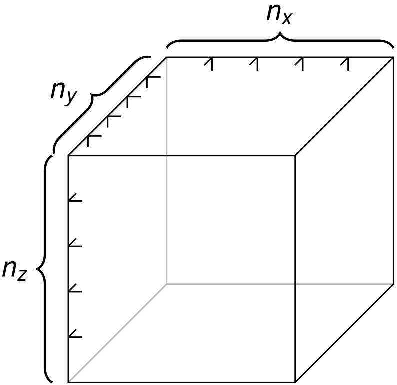
$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$

Free variables x, y, z

n monitors

Shares n_x, n_y, n_z

$n = n_x \cdot n_y \cdot n_z$



Solution 2: Hypercube

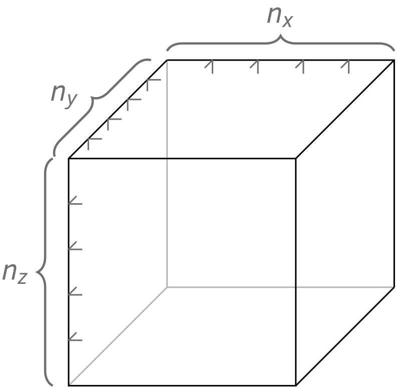
$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$

Free variables x, y, z

n monitors

Shares n_x, n_y, n_z

$n = n_x \cdot n_y \cdot n_z$

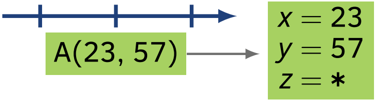


Solution 2: Hypercube

$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$

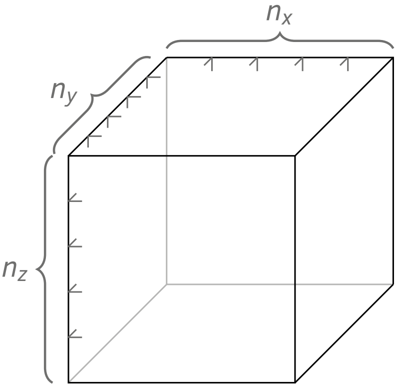
Free variables x, y, z

n monitors



Shares n_x, n_y, n_z

$n = n_x \cdot n_y \cdot n_z$

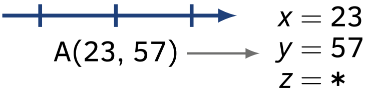


Solution 2: Hypercube

$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$

Free variables x, y, z

n monitors



Shares n_x, n_y, n_z

$n = n_x \cdot n_y \cdot n_z$

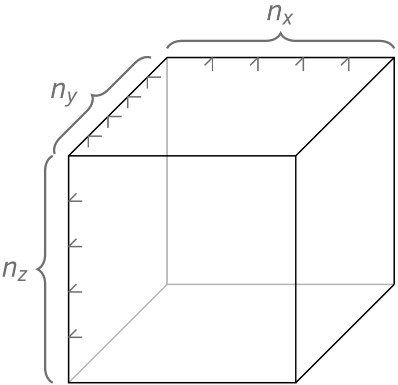
Hash functions

$f_x : D \rightarrow \{1, \ldots, n_x\}$

$f_y : D \rightarrow \{1, \ldots, n_y\}$

$f_z : D \rightarrow \{1, \ldots, n_z\}$

D: set of data values

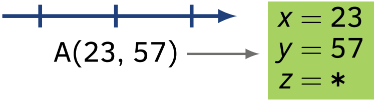


Solution 2: Hypercube

$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$

Free variables x, y, z

n monitors



Shares n_x, n_y, n_z

$n = n_x \cdot n_y \cdot n_z$

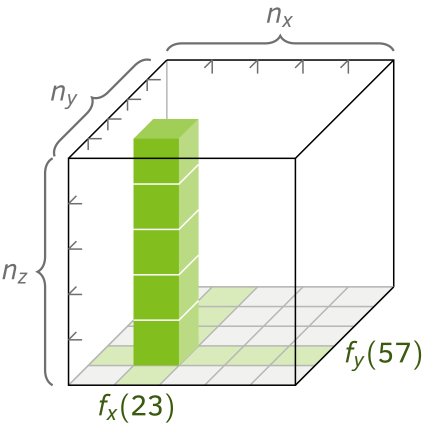
Hash functions

$f_x : D \rightarrow \{1, \ldots, n_x\}$

$f_y : D \rightarrow \{1, \ldots, n_y\}$

$f_z : D \rightarrow \{1, \ldots, n_z\}$

D: set of data values

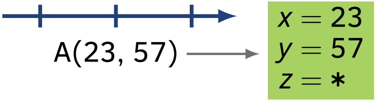


Solution 2: Hypercube

$A(x, y) \wedge (\blacklozenge B(y, z)) \rightarrow \blacklozenge C(x, z)$

Free variables x, y, z

n monitors



Shares n_x, n_y, n_z

$n = n_x \cdot n_y \cdot n_z$

optimized using event frequency statistics

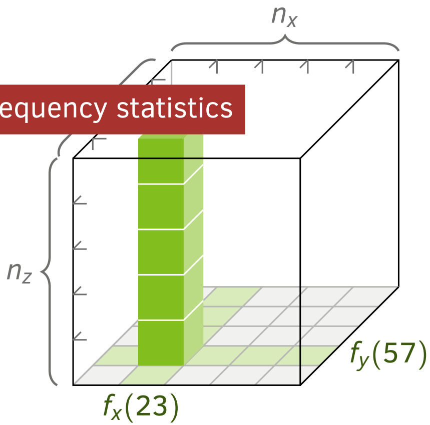
Hash functions

$f_x : D \rightarrow \{1, \ldots, n_x\}$

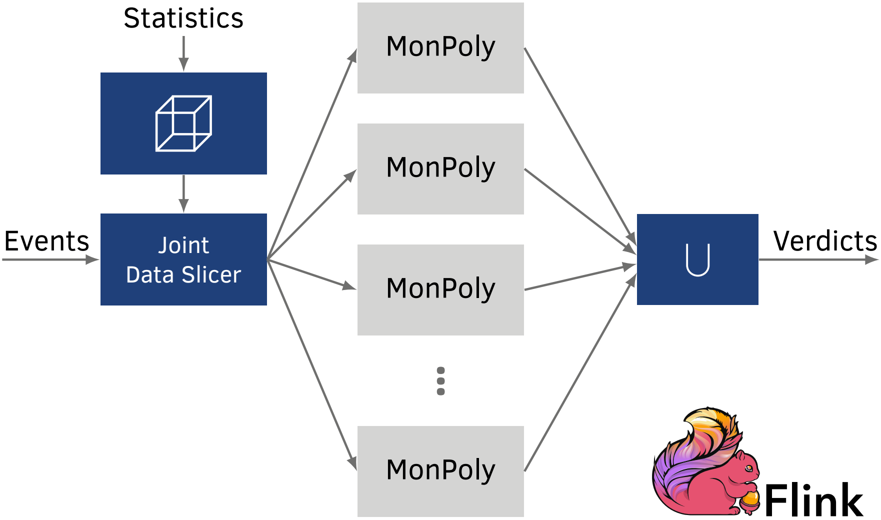
$f_y : D \rightarrow \{1, \ldots, n_y\}$

$f_z : D \rightarrow \{1, \ldots, n_z\}$

D: set of data values

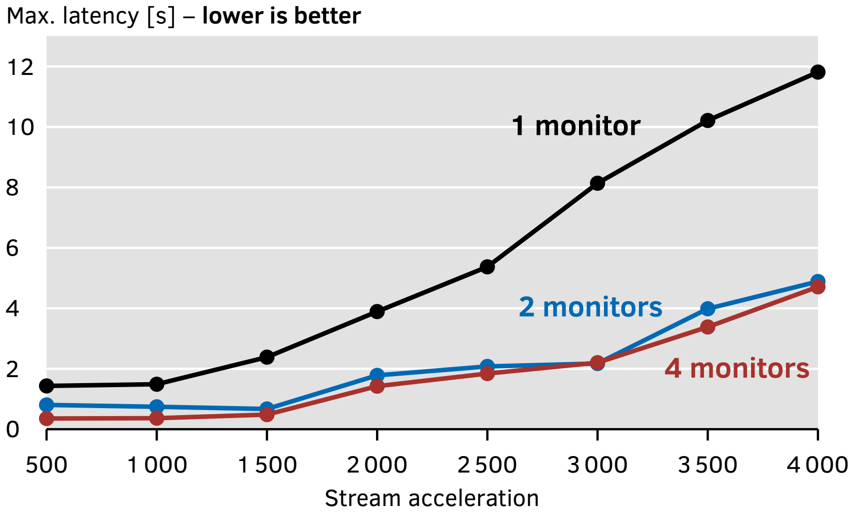


Our Current Implementation



Evaluation (1)

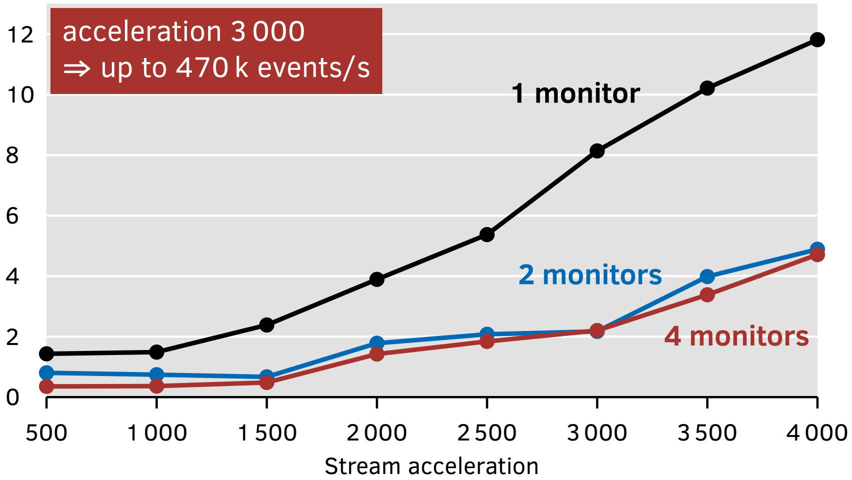
"Nokia" data excerpt, *insert* formula, with fault-tolerance



Evaluation (1)

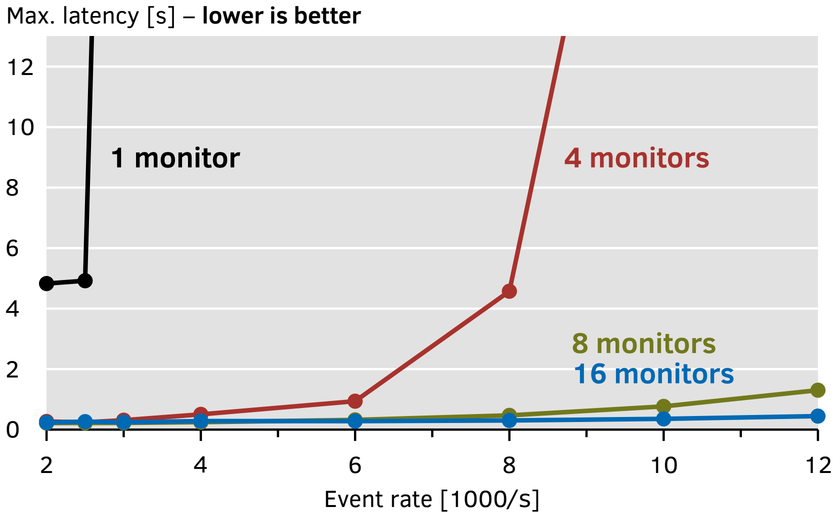
“Nokia” data excerpt, *insert* formula, with fault-tolerance

Max. latency [s] – **lower is better**



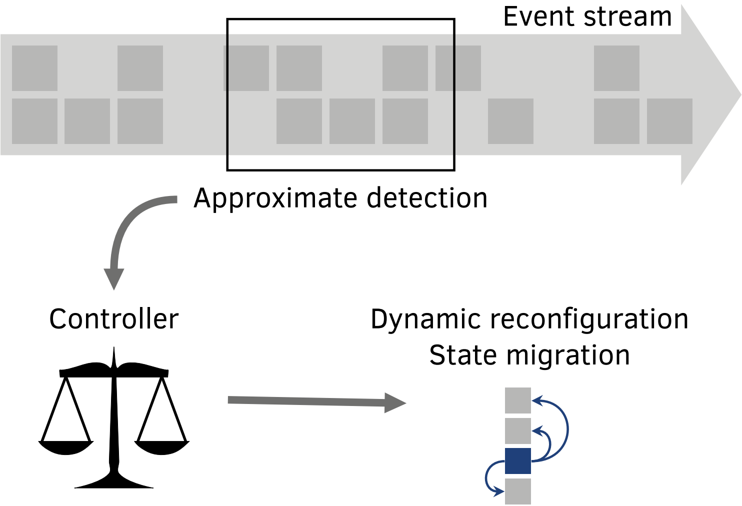
Evaluation (2)

Random data, *triangle* formula, with fault-tolerance & statistics

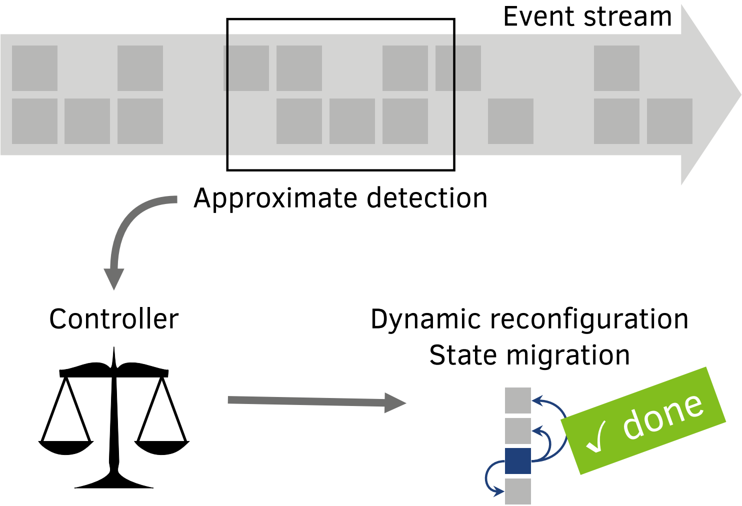


Future Work & Conclusion

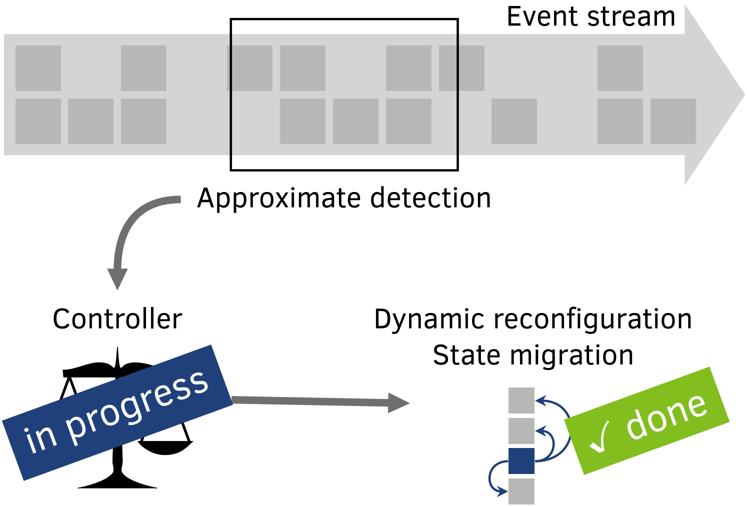
Adapting to Changing Statistics



Adapting to Changing Statistics

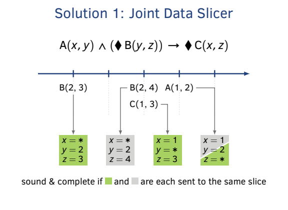


Adapting to Changing Statistics

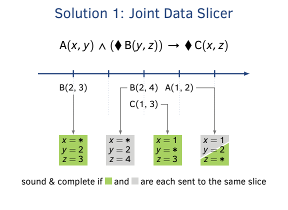


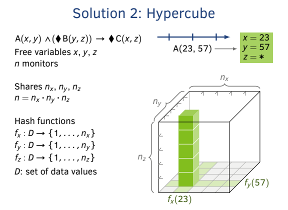
Scalable Online First-Order Monitoring

Scalable Online First-Order Monitoring

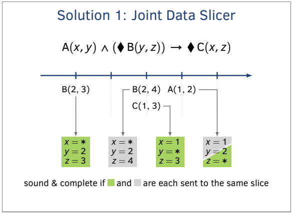


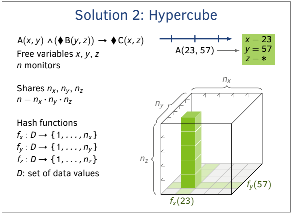
Scalable Online First-Order Monitoring

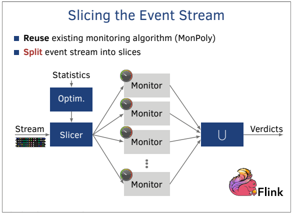




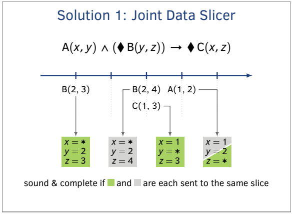
Scalable Online First-Order Monitoring

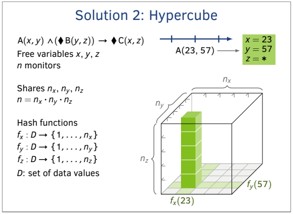


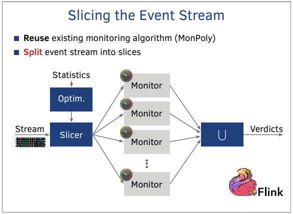


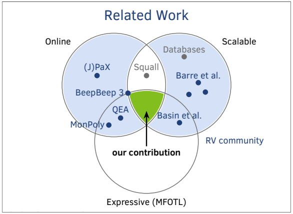


Scalable Online First-Order Monitoring

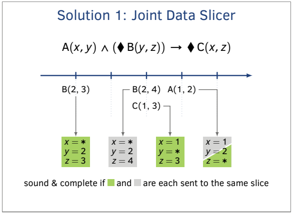


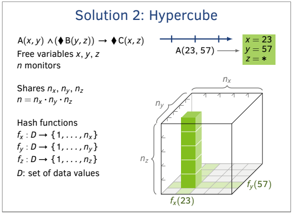


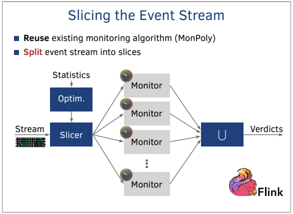


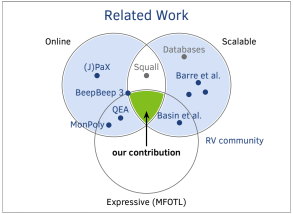


Scalable Online First-Order Monitoring









Joshua Schneider David Basin Frederik Brix
Srđan Krstić Dmitriy Traytel
ETH zürich