Tutorial:
Complex Event Recognition in the Big Data Era

Nikos Giatrakos[1], Alexander Artikis[2,3], Antonios Deligiannakis[1],
Minos Garofalakis[1,4]

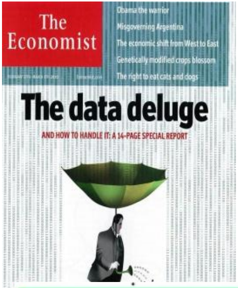
[1]Technical University of Crete, Chania, Greece
[2]University of Piraeus, Greece
[3]NCSR Demokritos, Athens, Greece
[4]ATHENA Research & Innovation Center, Athens, Greece

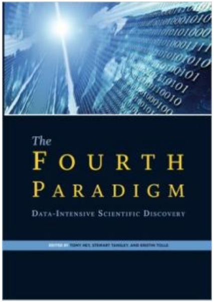
Slides available at:

Big Data is Big News (and Big Business)

- Rapid growth due to several information-generating technologies, such as mobile computing, sensornets, and social networks

- How can we cost-effectively manage and analyze all this data…?





Big Data Challenges: The Four V’s (… and one D)

- Volume: Scaling from Terabytes to Exa/Zettabytes
- Velocity: Processing massive amounts of streaming data
- Variety: Managing the complexity of multiple relational and non-relational data types and schemas
- Veracity: Handling inherent uncertainty and noise in the data
- Distribution: Dealing with massively distributed information

Existing Big Data Platforms

Large computing clusters – **scale out** to 1000s of commodity nodes

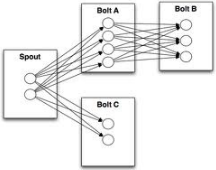
Map/Reduce, Hadoop, Spark

Simple programmatic models, scalable, replication for robustness

BUT: Batch processing of static data

Focus on *relational model* (tables, SQL)

Storm/Heron, Flink, Spark Streaming



Simple, scalable dataflow processing

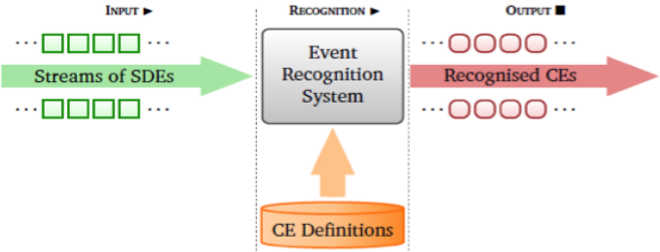
Hard to map from higher level logic and complex analytics tasks!

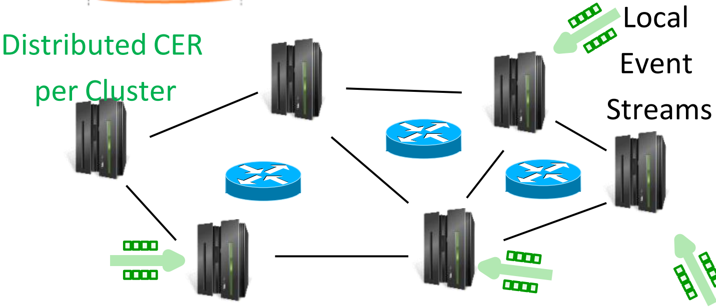
Complex Event Recognition (Event Pattern Matching, CEP)

- **Input**
  - Massive streams of time-stamped *Simple Derived Events (SDEs)* coming from (distributed) sources

- **Output**
  - *Complex/Composite Events (CEs)* – collections of SDEs and/or CEs satisfying some pattern
    - Patterns defined using variety of constraints (temporal, spatial, logical, …)
    - Not restricted to simple aggregation!
    - Complex, multi-level CE hierarchies
    - Inherent uncertainty (SDEs, patterns)

Complex Event Recognition (Event Pattern Matching, CEP)





Complex Event Recognition for Credit Card Fraud Management

Input:

- Credit card transactions from all over the world.

Output:

- Cloned card — a credit card is being used simultaneously in different countries.
- New high use — the card is being frequently used in merchants or countries never used before.
- Potential batch fraud — many transactions from multiple cards in the same point-of-sale terminal in high amounts.

Complex Event Recognition for Credit Card Fraud Management

- Fraud must be detected within 25 milliseconds.
- Fraudulent transactions: 0.1% of the total number of transactions.
- Fraud is constantly evolving.
- Erroneous transactions, missing fields.

Complex Event Recognition for Maritime Surveillance

Fast Approach

- A vessel is moving at a high speed ...
- towards other vessels.

Possible Rendezvous

- Two vessels are suspiciously delayed ...
- in the same location ...
- at the same time.

Complex Event Recognition for Maritime Surveillance

'Sea' of information:

- $>$200,000 vessels operate globally.
- Position signals need to be combined with other data streams
  - Weather forecasts, sea currents, etc.
- ... and static information
  - NATURA areas, shallow waters, coastlines, etc.

Complex Event Recognition for Maritime Surveillance

'Sea' of noisy information:

- GPS manipulation has risen 59% over the past two years.
- There is a 30% increase over the past two years of vessels reporting a false identity.
- 27% of vessels do not report position at least 10% of the time.
  - 19% of vessels are repeat offenders.

This Tutorial: CER + Big Data (4Vs + D)

- Introduction
- Complex Event Recognition Languages
- Handling Uncertainty
- Scalable (Parallel and Distributed) CER
- Outlook

Part 1: Complex Event Recognition Languages

Language Requirements: Credit Card Fraud Management

- Input:
  - Instantaneous events.
  - Context information.
- Output: durative events.
  - Relational & non-relational events.
  - Limited temporal distance between the events comprising fraudulent activity ('WITHIN' constraint).
  - Event sequences.
  - Spatial reasoning for some patterns.

Language Requirements: Maritime Surveillance

- Input:
  - Instantaneous events.
  - Durative events.
  - Context information.
- Output: durative events.
  - The interval may be open.
  - Relational events.
  - No limit on the temporal distance between the events comprising the composite activity.
  - Concurrency constraints.
  - Spatial reasoning.
  - Event hierarchies.

Event Algebra

Core components of an event algebra with point-based semantics:

- Sequencing (SEQ) lists the required event types in temporal order — eg SEQ(A, B, C).
- Kleene closure ($+$) collects a finite yet unbound number of events of a particular type. It is used as a component of SEQ — eg SEQ$(A, B+, C)$.
- Negation (˜ or !) verifies the absence of certain events in a sequence — eg SEQ$(A, !B, C)$.
- Value predicates specify constraints on the event attributes
  - Aggregate functions *max*, *min*, *count*, *sum*, *avg*.

Event Algebra

- Composition refers to:
  - Union of constraints — eg SEQ(A, B, C) $\cup$ SEQ(A, D, E).
  - Negation of a sequence — eg !SEQ(A, B, C).
  - Kleene closure of a constraint — eg SEQ(A, B, C)$+$.
- Windowing (WITHIN) restricts a CE definition to a specific time period.

Event Selection Strategies

- Strict contiguity: No intervening events allowed between two sequence events in the pattern.
- Partition contiguity: Same as above, but the stream is partitioned into substreams according to a partition attribute. Events must be contiguous within the same partition.
- Skip-till-next-match: Intervening events are allowed, but only non-overlapping occurrences of SEQ are detected. E.g. for SEQ(A, B, C) and a_1, b_1, b_2, c_1, only a_1, b_1, c_1 will be detected.
- Skip-till-any-match: Most flexible (and expensive). Detects every possible occurrence. For the previous example, a_1, b_2, c_1 will also be detected.

Example

Fishing pattern:

- A vessel slows down, ...
- begins a series of turns, where, for each pair of successive turns, their difference in heading is more than 90 degrees, ...
- and subsequently the vessel stops moving at a low speed.

PATTERN SEQ($lowSpeedStart\ a,\ turn+\ b,\ lowSpeedEnd\ c$)
WHERE skip-till-next-match
AND *vesselId*
AND $b[i].heading - b[i-1].heading > 90$
WITHIN 21600

Event Calculus

- A logic programming language for representing and reasoning about events and their effects.
- Key components:
  - event (typically instantaneous).
  - fluent: a property that may have different values at different points in time.
- Built-in representation of inertia:
  - $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an event at some earlier time-point, and not *terminated* by another event in the meantime.

Run-Time Event Calculus (RTEC)

Predicate	Meaning
happensAt(E, T)	Event E occurs at time T
initiatedAt$(F = V, T)$	At time T a period of time for which $F = V$ is initiated
terminatedAt$(F = V, T)$	At time T a period of time for which $F = V$ is terminated
holdsFor$(F = V, I)$	I is the list of the maximal intervals for which $F = V$ holds continuously
holdsAt$(F = V, T)$	The value of fluent F is V at time T
union_all$([J_1, \ldots, J_n],\ I)$	$I = (J_1 \cup \ldots \cup J_n)$
intersect_all$([J_1, \ldots, J_n],\ I)$	$I = (J_1 \cap \ldots \cap J_n)$
relative_complement_all $(I', [J_1, \ldots, J_n],\ I)$	$I = I' \setminus (J_1 \cup \ldots \cup J_n)$

Example

CE definition:

$$
\begin{aligned}
&\textbf{initiatedAt}(\mathit{gap}(\mathit{Vessel}) = \text{true},\ T) \leftarrow \\
&\qquad \textbf{happensAt}(\mathit{gapStart}(\mathit{Vessel}),\ T), \\
&\qquad \textbf{holdsAt}(\mathit{coord}(\mathit{Vessel}) = (\mathit{Lon}, \mathit{Lat}),\ T), \\
&\qquad \text{not}\ \mathit{nearPorts}(\mathit{Lon}, \mathit{Lat}) \\
&\textbf{terminatedAt}(\mathit{gap}(\mathit{Vessel}) = \text{true},\ T) \leftarrow \\
&\qquad \textbf{happensAt}(\mathit{gapEnd}(\mathit{Vessel}),\ T)
\end{aligned}
$$

CE recognition: $\textbf{holdsFor}(\mathit{gap}(\mathit{Vessel}) = \text{true},\ I)$

Summary

- Various types of language ...
  - Automata-based
  - Logic-based
  - Tree-based
- ... addressing different requirements.
- Some steps towards a systematic, formal comparison of expressivity and complexity have been taken.

A. Artikis, A. Margara, M. Ugarte, S. Vansummeren, M. Weidlich. Complex Event Recognition Languages: A Tutorial. DEBS, 2017.

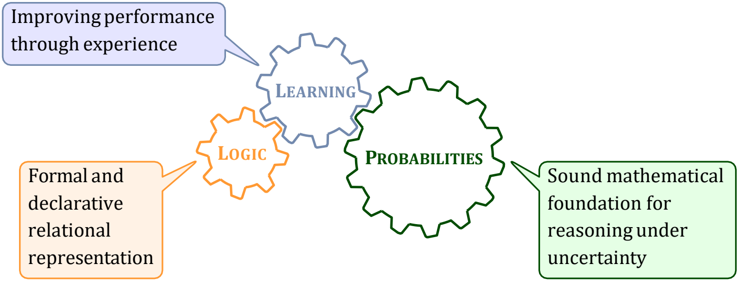
Part 2:
Uncertainty Handling

Common Problems of Complex Event Recognition

- Limited dictionary of SDE and context variables.
  - No explicit representation of oil spillage.
- Incomplete SDE stream.
  - Sharp turn was not detected.
- Erroneous SDE detection.
  - Slow motion was classified as stop.
- Inconsistent ground truth (CE & SDE annotation).
  - Disagreement between (human) annotators.

Therefore, an adequate treatment of uncertainty is required.

Statistical Relational Learning



Markov Logic Networks (MLN)

SYNTAX: weighted first-order logic formulas (w_i, F_i)

When input events SDE_A **and** SDE_B occur at T,
then the output event CE is initiated:

3.18 $happensAt(SDE_A, T) \wedge happensAt(SDE_B, T) \Rightarrow initiatedAt(CE, T)$

SEMANTICS: (w_i, F_i) represents a probability distribution over possible worlds

SDEs

weight of the i-th formula

$$P(Y{=}\mathbf{y} \mid X{=}\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_i w_i\, n_i(\mathbf{x}, \mathbf{y})\right)$$

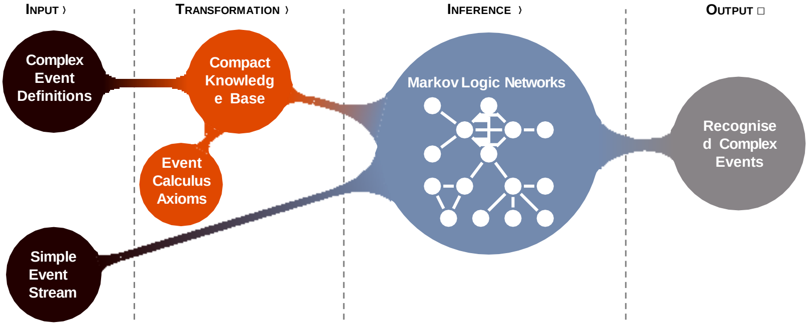
Possible world: CEs

Partition function

number of satisfied groundings

A world violating formulas becomes less probable, but not impossible!

Event Calculus in Markov Logic Networks (MLN-EC)



MLN-EC: Probabilistic Inference

Marginal Inference:

- For all time points T, calculate the probability of each **CE** being true (recognised), given all **input SDEs** (evidence)

$$P(\texttt{holdsAt}(CE,\ T){=}true \mid SDEs)$$

- Marginal inference is #P-complete $\rightarrow$ approximate inference
- MC-SAT algorithm (Markov Chain Monte Carlo techniques with SAT solver)

MLN-EC: Probabilistic Inference

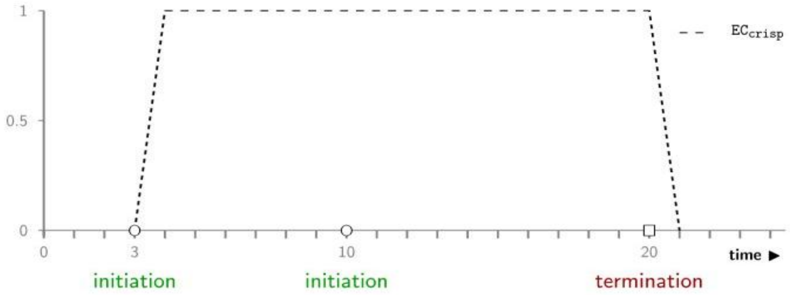
Maximum a Posteriori (MAP) Inference:

- Find the world with the highest probability
- **Input:** truth values for all **input SDEs** (evidence)
- **Output:** truth values of the **output CEs** that maximise the probability (recognition)

$$\underset{\texttt{holdsAt}(CE,\, T)}{\operatorname{argmax}} \Big(P(\texttt{holdsAt}(CE,\, T) | SDEs) \Big)$$

- MAP Inference is NP-hard $\rightarrow$ approximate inference
- Various methods: local search, linear programming, etc.

MLN-EC: Inertia



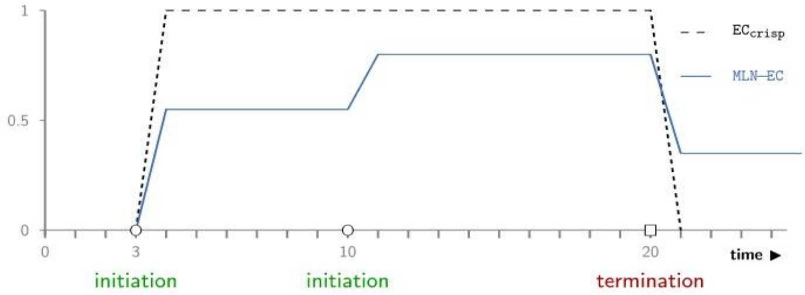
∞ holdsAt(CE, T+1)⇐
[Initiation Conditions]

∞ ¬holdsAt(CE, T+1)⇐
[Termination Conditions]

∞ ¬holdsAt(CE, T+1)⇐
¬holdsAt(CE, T) ∧
¬[Initiation Conditions]

∞ holdsAt(CE, T+1)⇐
holdsAt(CE, T) ∧
¬[Termination Conditions]

MLN-EC: Inertia



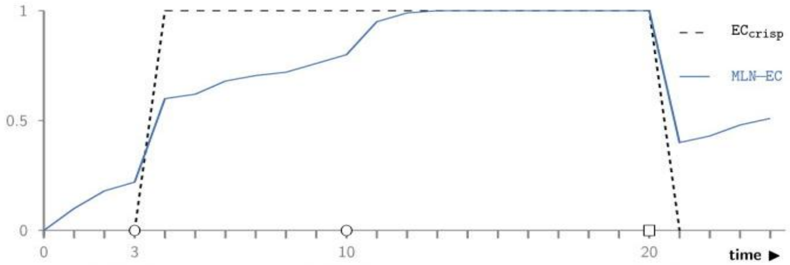
1.2 holdsAt(CE, T+1) ⇐
[Initiation Conditions]

0.7 ¬holdsAt(CE, T+1) ⇐
[Termination Conditions]

∞ ¬holdsAt(CE, T+1) ⇐
¬holdsAt(CE, T) ∧
¬[Initiation Conditions]

∞ holdsAt(CE, T+1) ⇐
holdsAt(CE, T) ∧
¬[Termination Conditions]

MLN-EC: Inertia



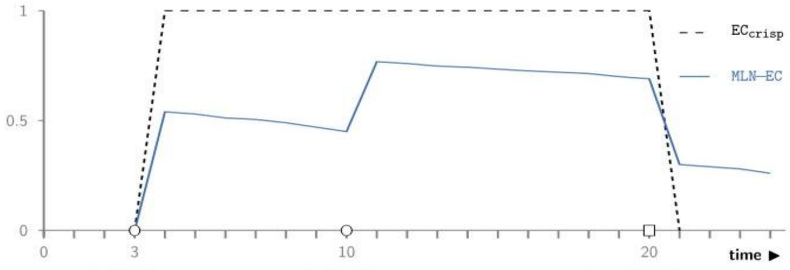
$$1.2 \quad \text{holdsAt(CE, T+1)} \Leftarrow \left[\texttt{Initiation Conditions}\right]$$

$$0.7 \quad \neg\text{holdsAt(CE, T+1)} \Leftarrow \left[\texttt{Termination Conditions}\right]$$

$$2.3 \quad \neg\text{holdsAt(CE, T+1)} \Leftarrow \neg\text{holdsAt(CE, T)} \wedge \neg\left[\texttt{Initiation Conditions}\right]$$

$$\infty \quad \text{holdsAt(CE, T+1)} \Leftarrow \text{holdsAt(CE, T)} \wedge \neg\left[\texttt{Termination Conditions}\right]$$

MLN-EC: Inertia



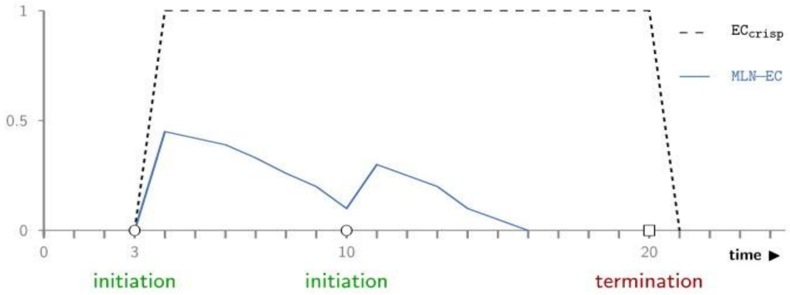
1.2 $\text{holdsAt(CE, T+1)} \Leftarrow$
$\quad$ [`Initiation Conditions`]

0.7 $\neg\text{holdsAt(CE, T+1)} \Leftarrow$
$\quad$ [`Termination Conditions`]

∞ $\neg\text{holdsAt(CE, T+1)} \Leftarrow$
$\quad$ $\neg\text{holdsAt(CE, T)} \wedge$
$\quad$ $\neg$[`Initiation Conditions`]

2.3 $\text{holdsAt(CE, T+1)} \Leftarrow$
$\quad$ $\text{holdsAt(CE, T)} \wedge$
$\quad$ $\neg$[`Termination Conditions`]

MLN-EC: Inertia



1.2 holdsAt(CE, T+1)⇐
[`Initiation Conditions`]

0.7 ¬holdsAt(CE, T+1)⇐
[`Termination Conditions`]

∞ ¬holdsAt(CE, T+1)⇐
¬holdsAt(CE, T) ∧
¬[`Initiation Conditions`]

0.6 holdsAt(CE, T+1)⇐
holdsAt(CE, T) ∧
¬[`Termination Conditions`]

Summary

First-order logic & Probabilistic Graphical Models:

- ✓ Complex temporal patterns, with explicit time constraints. Event hierarchies. Background knowledge. Usually provide a formal Event Algebra.
- × No *Iteration*. Limited support for *Windowing*.
- ✓ Pattern uncertainty. Limited independence assumptions. Hard constraints possible.
- × Often training is required to assign weights to rules. Harder (but not impossible) to express data uncertainty.
- ✓ MAP and approximate inference.
- × Low (or unknown) throughput.

E. Alevizos, A. Skarlatidis, A. Artikis, G. Paliouras. Probabilistic Complex Event Recognition: A Survey. ACM Computing Surveys, 2017.

Summary

Automata:

- ✓ *Iteration*, *Windowing*, formal Event Algebra.
- × Limited support for event hierarchies. No background knowledge. Implicit time representation (hence no explicit constraints on time attribute).
- ✓ Data uncertainty, both with respect to occurrence of events and event attributes.
- × Limited or no support for rule uncertainty. Too many independence assumptions. No hard constraints.
- ✓ Support for confidence thresholds. High throughput values.
- × Throughput figures come from experiments with simplistic event patterns.

E. Alevizos, A. Skarlatidis, A. Artikis, G. Paliouras. Probabilistic Complex Event Recognition: A Survey. ACM Computing Surveys, 2017.

Part 3:
Scalable, Distributed Complex Event Recognition

How to scale CER in the Big Data Era

Scaling out to

- Parallel Architectures: Computer Clusters/Grids, The Cloud
- Networked Settings: Dispersed Clusters, Multi-Cloud Platforms

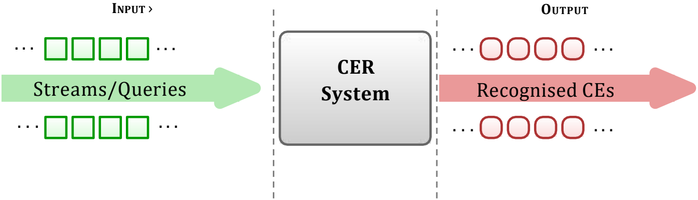
Scalable - Distributed Complex Event Recognition

Why? Well, It’s the Big Data Era

- Volume, Velocity, Variety, Veracity (Uncertainty)

Centralized Architecture

Sequential CER



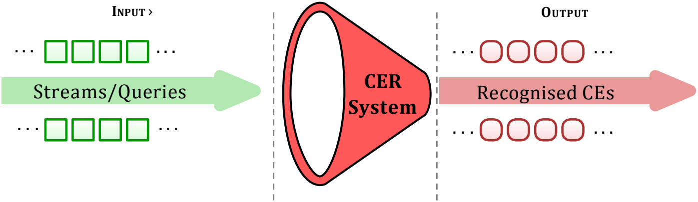
Scalable - Distributed Complex Event Recognition

Why? Well, It's the Big Data Era

- Volume, Velocity, Variety,

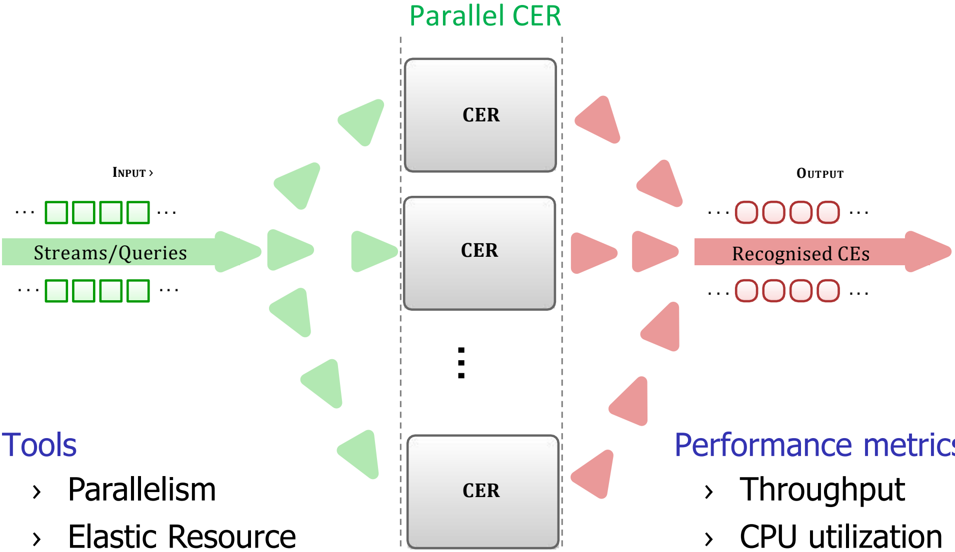
Centralized Architecture

Sequential CER



Scalable - Distributed Complex Event Recognition

Clustered Architecture



Tools

- Parallelism
- Elastic Resource Allocation

Performance metrics

- Throughput
- CPU utilization

Scalable Complex Event Recognition

Parallelization & Elasticity in state-of-the-art DSMSs:

- Horizontal Scalability in Stream Processing by design
- Facilities for Elastic Resource Allocation
- Fault Tolerance in message processing
- Popular Platforms: Apache Storm (Heron/Trident), Spark Streaming

CER Languages & CER Systems:

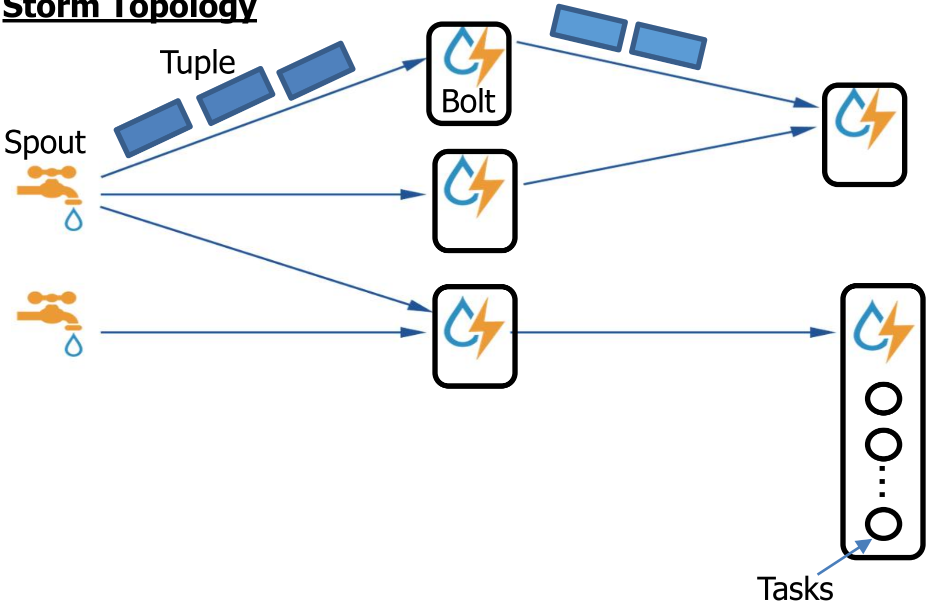
- High-Level CER Language Support
- Uncertainty-aware CER (sometimes)
- Support for various streaming operations (windowing etc.)

How to **bridge the gap**?

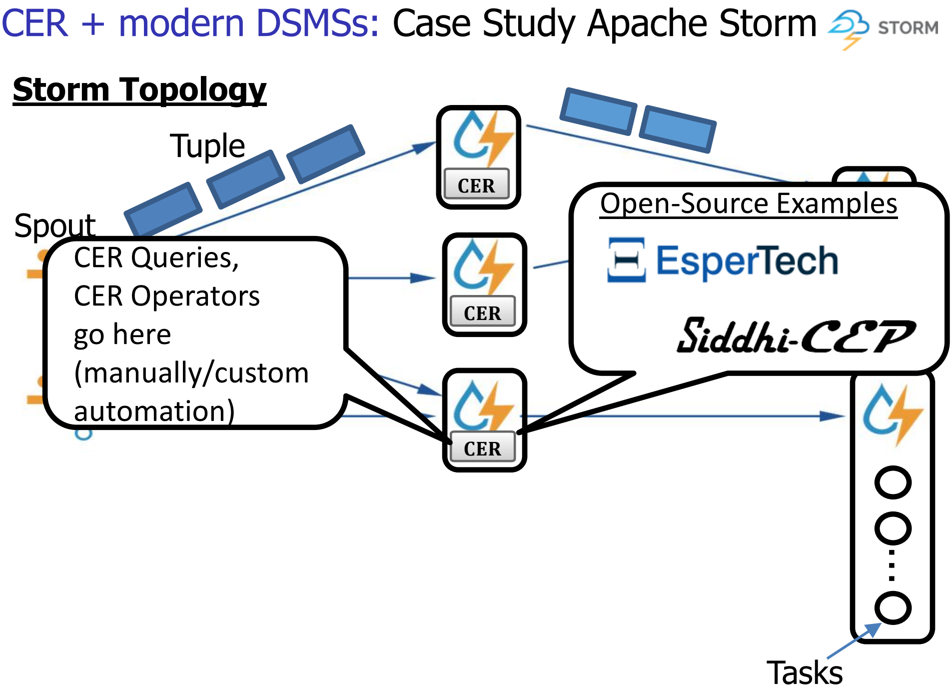
HackerBrucke Munich

CER + modern DSMSs: Case Study Apache Storm

Storm Topology

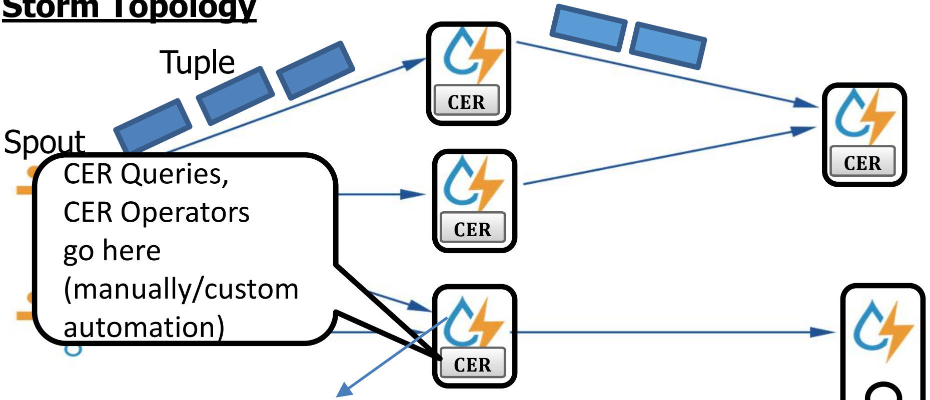


CER + modern DSMSs: Case Study Apache Storm



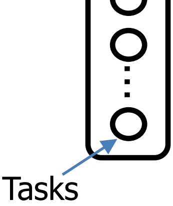
CER + modern DSMSs: Case Study Apache Storm

Storm Topology

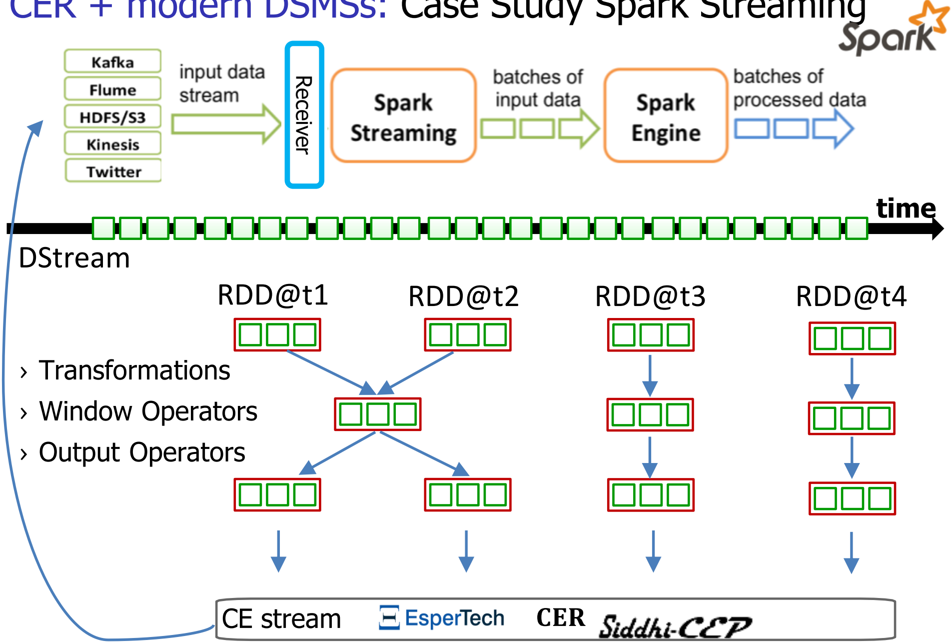


Data Partitioning – Which task a tuple goes to?

- Shuffle Grouping: Random tuple distribution
- Fields Grouping: Partition based on field(s) – keys
- All Grouping: Replicate tuple to all tasks
- Custom: Define your own



CER + modern DSMSs: Case Study Spark Streaming



- Transformations
- Window Operators
- Output Operators

Are we done?

CER Parallelization must guarantee Correctness:

Patterns in Centralized CER ≡ Patterns in Parallel CER

Which parallelization scheme to use?

Criteria – Common Pitfalls

Load (Im)Balance

Parallelization Granularity - Agility

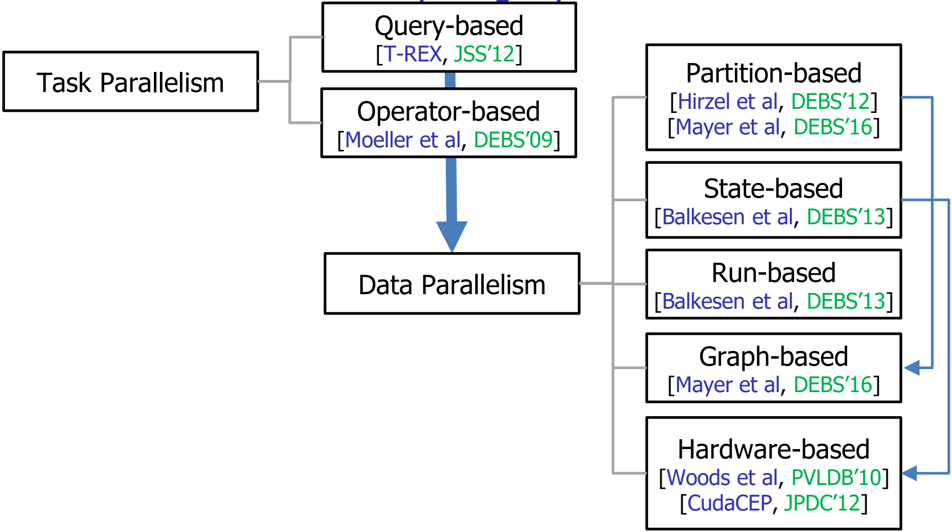
Support for Event Selection Policies

Support for Event Consumption Policies

Support for Parallelization of Windows

Need for Replication/Communication

Categorization of Parallelization Approaches in CER & Parallelization Granularity - Agility



Recap on Event Selection Policies

- Strict contiguity **[Sc]**: No intervening events allowed between two sequence events in the pattern.

- Partition contiguity **[Pc]**: Same as above, but the stream is partitioned into substreams according to a partition attribute. Events must be contiguous within the same partition.

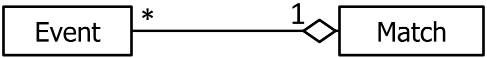
- Skip-till-next-match **[Stnm]**: irrelevant events are skipped until an event matching the next pattern component is encountered. If multiple events in the stream can match the next pattern component, only the first of them is considered.

  E.g. for SEQ(A, B, C) and a_1, b_1, b_2, c_1, only a_1, b_1, c_1 will be detected.

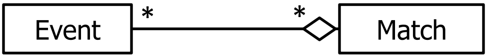
- Skip-till-any-match **[Stam]**: Most flexible (and expensive). Detects every possible occurrence. For the previous example, a_1, b_2, c_1 will also be detected.

Event Consumption Policies

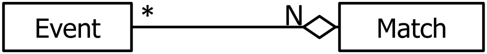
- **Consume [Co]**: Single event is used in a single pattern match



- **Reuse [Re]**: Single event can participate in multiple pattern matches as long as it remains valid e.g. given window constraints



- **Bounded Reuse [BRe]**: Single event can participate in up to N pattern matches as long as it remains valid



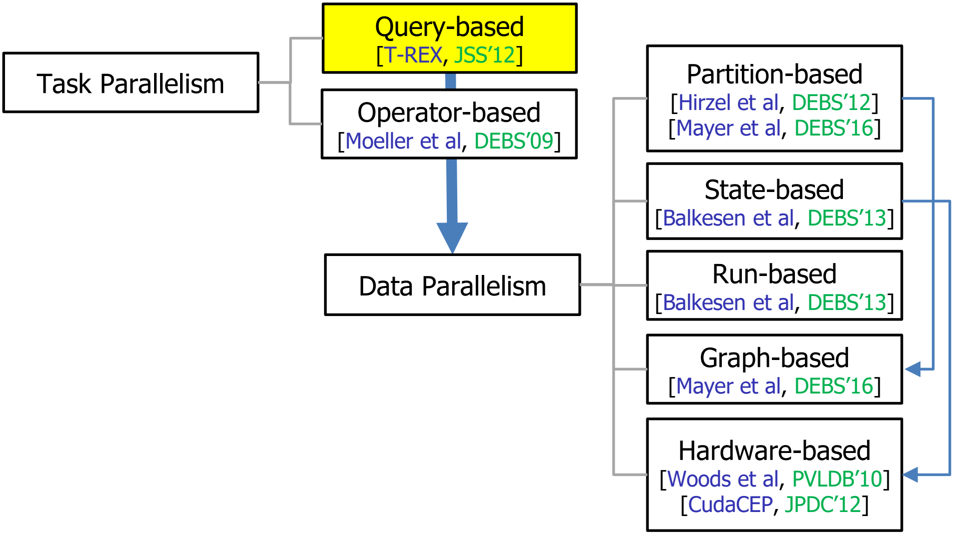
E.g. for SEQ(A, B, C) and a_1, b_1, b_2, c_1
skip-till-any-match & Reuse → $(a_1, b_1, c_1), (a_1, b_2, c_1)$
skip-till-any-match & Consume → (a_1, b_1, c_1)

Generic Stream Window Types

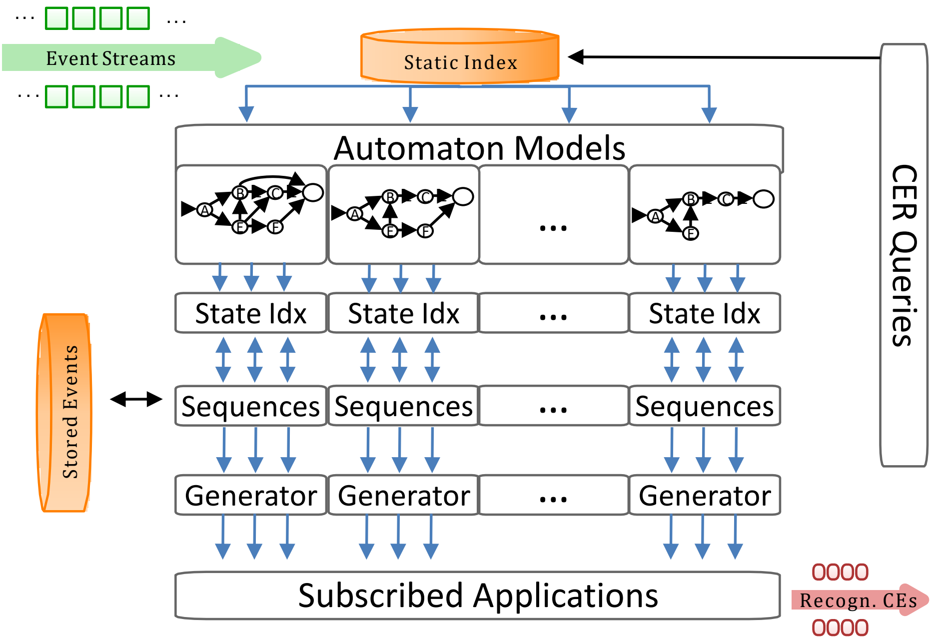
- Time-based Windows **[TiW]**: The upper bound of the <u>current</u> window is the current timestamp while the lower bound is determined based on a given time-interval parameter.

- Tuple-based Windows **[TuW]**: The upper and lower bound of the <u>current</u> window is determined so that it contains a certain amount of tuples

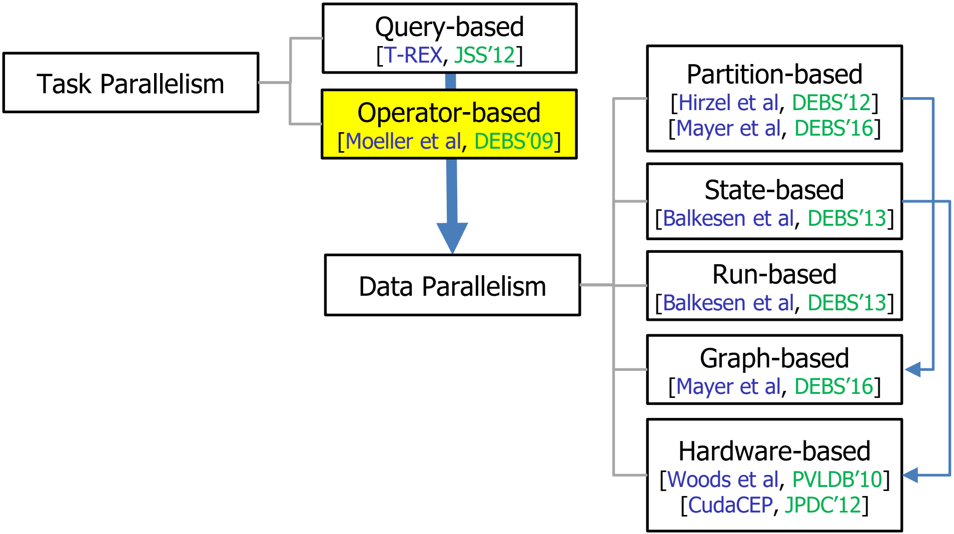
Categorization of Parallelization Approaches in CER



Query-based Parallelization [T-REX, JSS'12]

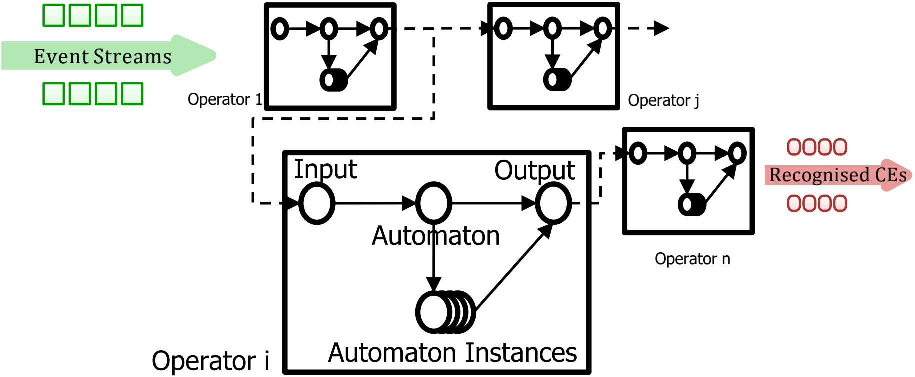


Categorization of Parallelization Approaches in CER

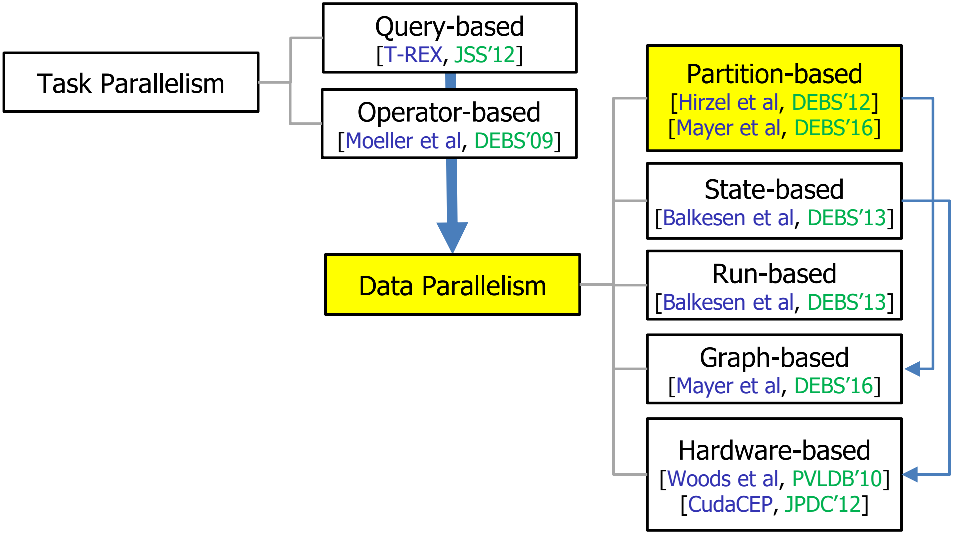


Operator-based Parallelization[Moeller et al, DEBS'09]

- Allows for multi-query and intra-query optimizations
- Intra-query optimizations → Query Rewriting:
  - Commutativity: OP(A,B)=OP(B,A)→ OR
  - Associativity: OP(OP(A,B),C)=OP(A,OP(B,C))→OR, SEQ
  - Evaluate operators with the rarest events first
- Multi-query optimizations → Operator Sharing

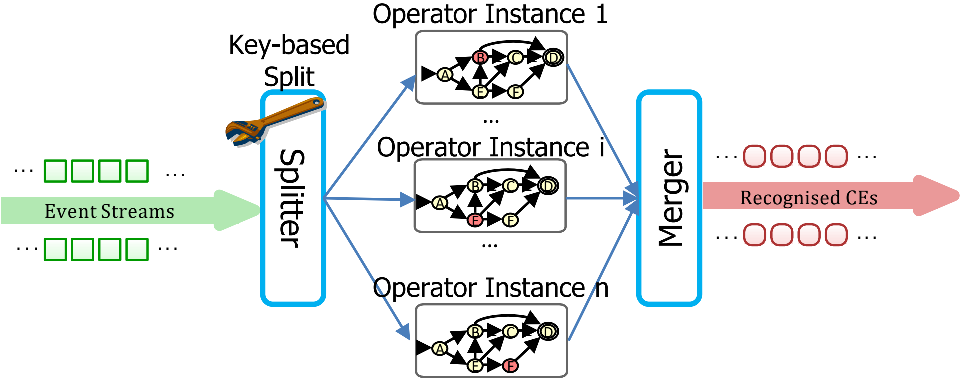


Categorization of Parallelization Approaches in CER

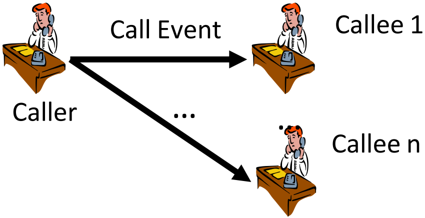


Partition key-based Parallelization [Hirzel et al, DEBS'12]

- Claims CER as a special operator `MatchRegex(Input_Events)`
- Includes a `PARTITION BY(key)` statement for key-based data partitioning
- Partition-isolation and uniqueness of longest match for correctness
- Implemented as an extension of IBM System S



Partition key-based Parallelization - Examples

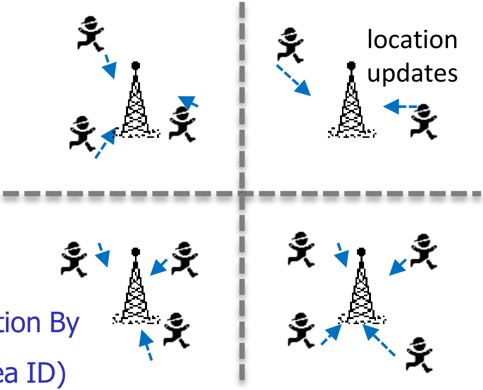


Partition By

(Caller ID)

Partition By

(User ID)

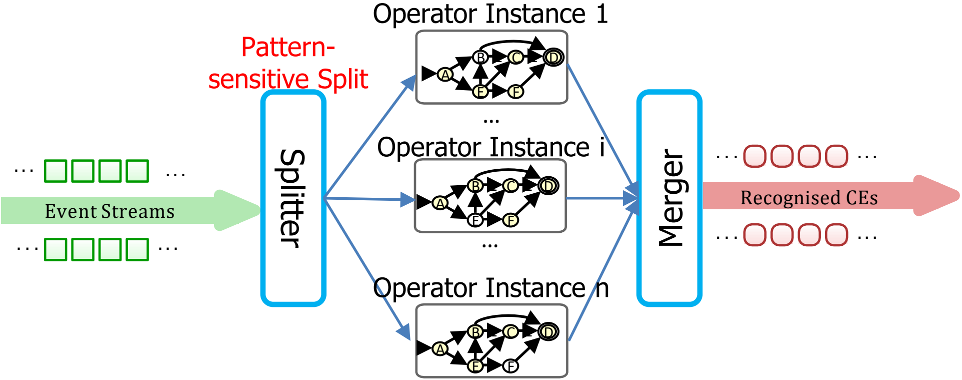


Partition By

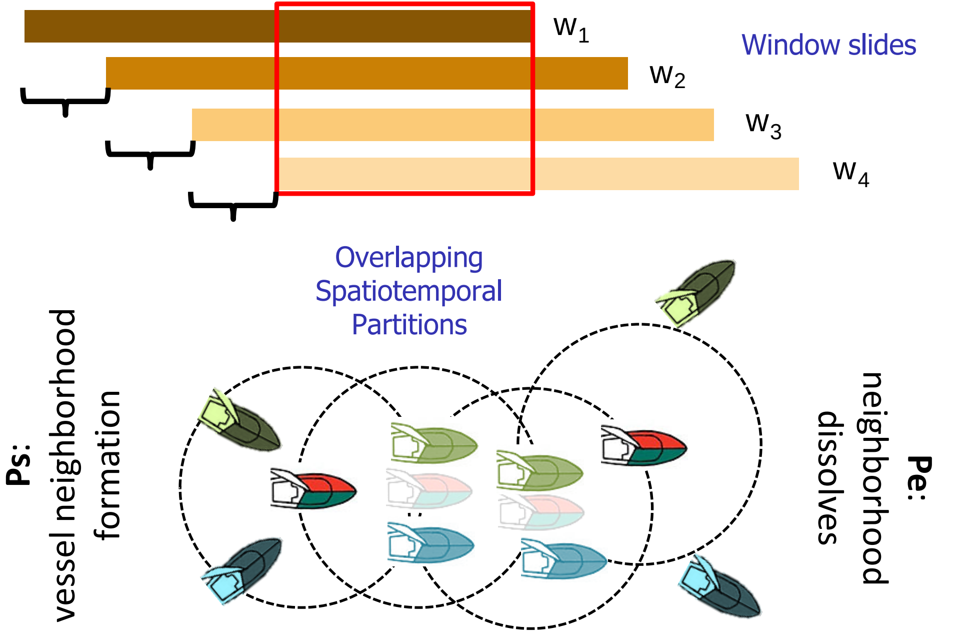
(Area ID)

Pattern-sensitive Partition-based Parallelization [Mayer et al, DEBS'16]

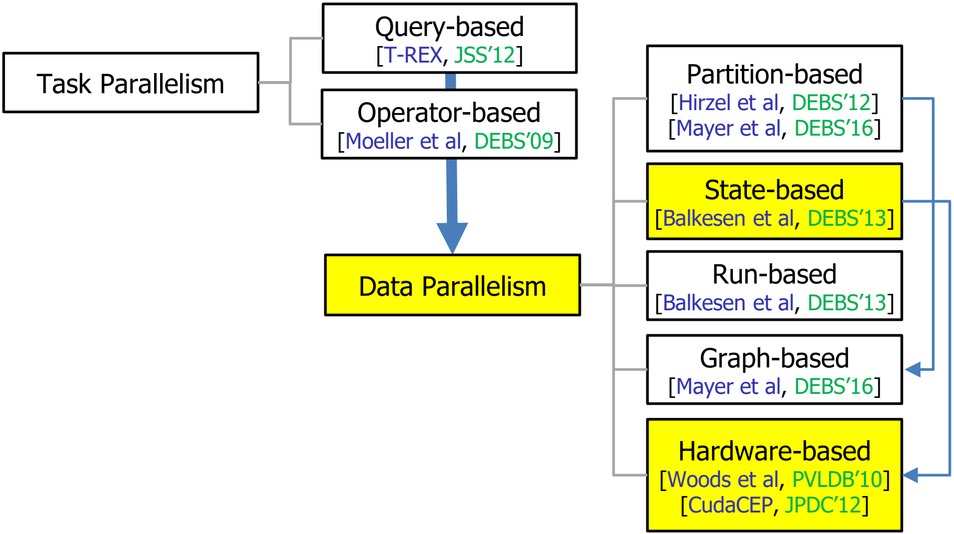
- Introduces pattern-sensitive data partitioning apart from key-based
- Partition Start: *e→BOOL* Partition End: *(partition, e)→BOOL*
- New event may start, be part of, or terminate a partition
- No partition isolation → replication of event to multiple partitions
- Can be used to parallelize sliding windows!



Pattern-sensitive Partition - Examples

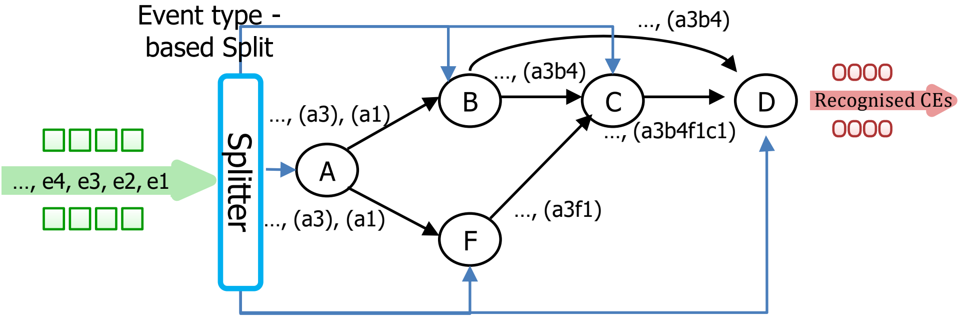


Categorization of Parallelization Approaches in CER

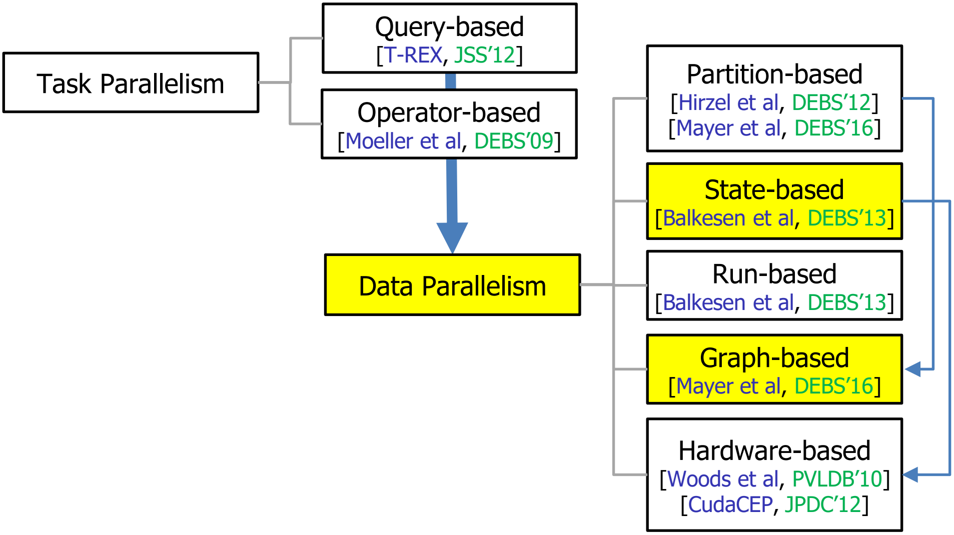


State-based Parallelization [Balkesen et al, DEBS'13]

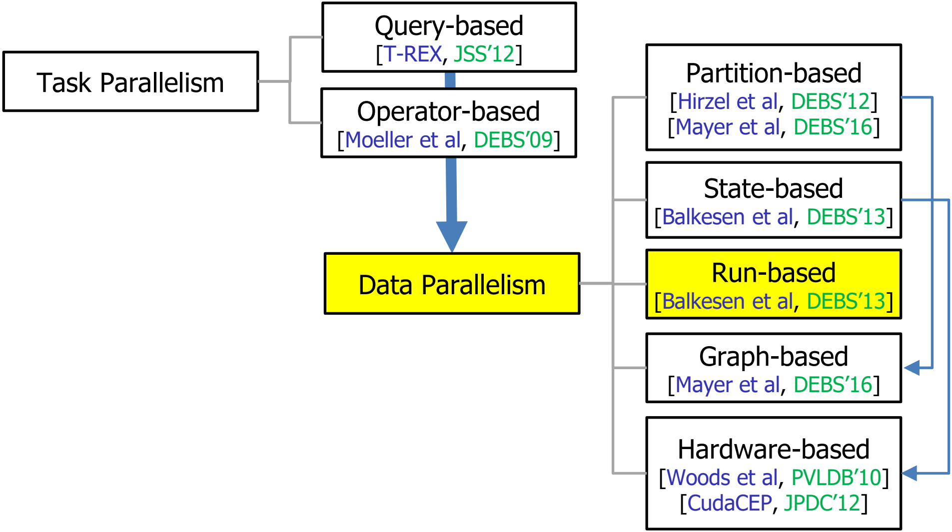
- NFA states (A,B,…)→Processing Units (PUs), NFA edges → Pipelines
- Event type-based data partitioning
- Filtering and predicate evaluation per state
- Pipeline the results among states on NFA structure
- Evaluation load towards final state
- FPGAs [Woods et al, PVLDB'10]
- GPUs [CudaCEP,JPDC'12] Column-based Delayed Processing (CDP)



Categorization of Parallelization Approaches in CER

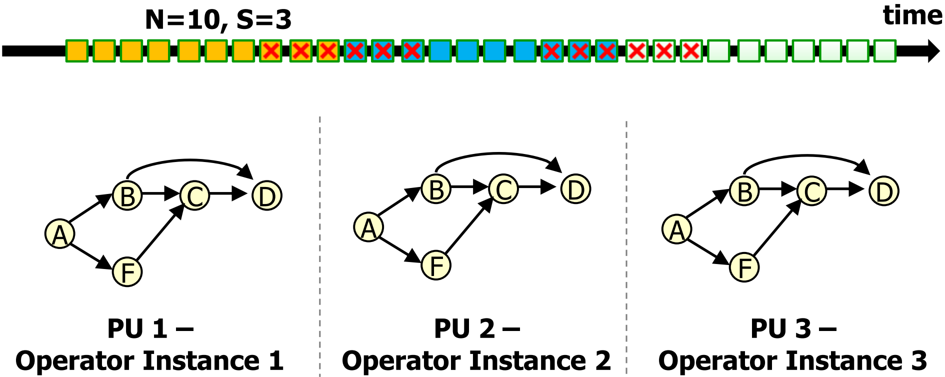


Categorization of Parallelization Approaches in CER



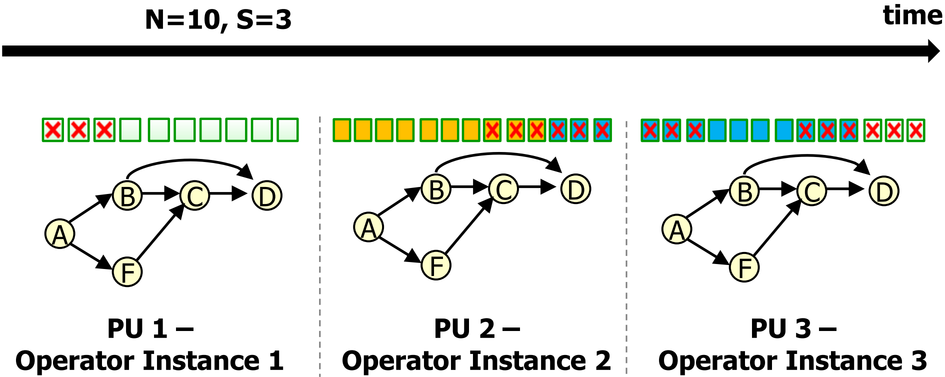
Run-based Parallelization[Balkesen et al, DEBS'13]

- Split stream into overlapping batches of B size
- Size of overlap S = maximal_match_length-1 ≤ B/2
- Assign a batch to one PU
- A PU detects all matches that start in the first B--S events in a batch
- Batch-based data partitioning → Load Balancing



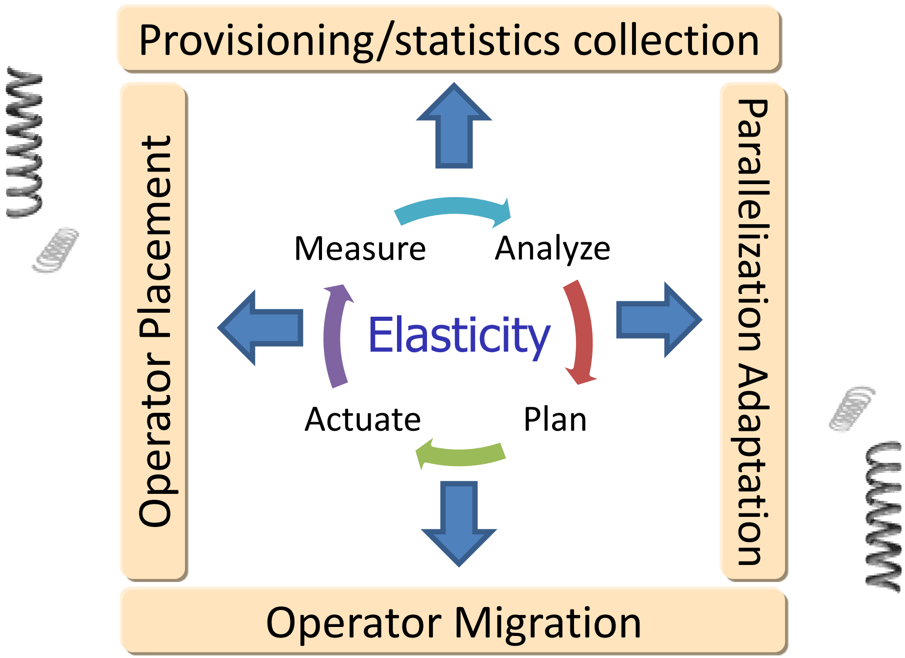
Run-based Parallelization[Balkesen et al, DEBS’13]

- Split stream into overlapping batches of B size
- Size of overlap S = maximal_match_length-1 ≤ B/2
- Assign a batch to one PU
- A PU detects all matches that start in the first B--S events in a batch
- Batch-based data partitioning → Load Balancing



No one size fit all solution!

		Task Parallelism		Data Parallelism				
	Criterion	Query-based	Operator-based	Partition Key-based	Pattern sensitive	State-based	Run-based	Hybrid
Selection Policies	Sc	✓	✓	✗	✗	✓	✓	AND
	Pc	✓	✓	✓	✓	✓	✗✓	
	Stnm	✓	✓	✗	✗	✓	✓	
	Stam	✓	✗	✗	✗	✓	✓	
Consumptio n Policies	Co	✓	✓	✓	✗	✓	✗✓	
	Re	✓	✓	✓	✓	✓	✓	
	BRe	✓	✓	✓	✗	✓	✗✓	
Window Parallel	TuW	✗	✗	✗	✓	✗	✓	OR
	TiW	✗	✗	✗	✓	✗	✗	
Agility	LB	✗	✗	✗	✗	✗	✓	
	Rep/ Comm	✗	✗	✓	✗	✗	✗✓	


Provisioning/statistics collection
Operator Placement
Parallelization Adaptation
Measure
Analyze
Elasticity
Actuate
Plan
Operator Migration

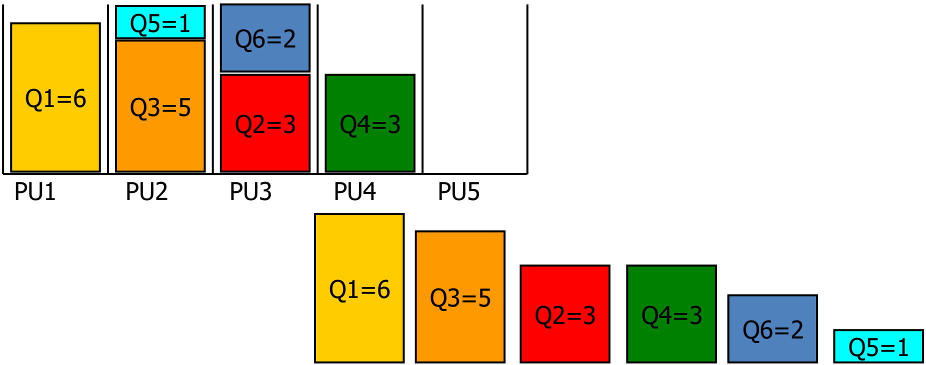
Elastic Resource Allocation in CER – FUGU Approach [Heinze et al, DB3@VLDB '13, DEBS'14]

Key Concepts

› First Fit Bin Packing for Operator Placement

› Elastic, Workload Unaware, Resource Allocation

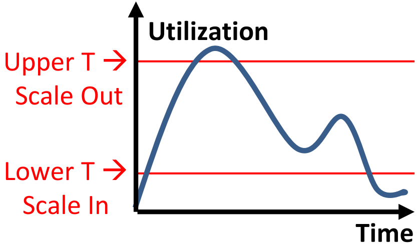
- ➢ Local & Global Threshold-based Approach
- ➢ Reinforcement Learning Approach



Elastic Resource Allocation in CER – FUGU Approach [Heinze et al, DB3@VLDB '13, DEBS'14]

Key Concepts

- First Fit Bin Packing for Operator Placement
- Elastic, Workload Unaware, Resource Allocation
  - Threshold-based Approach



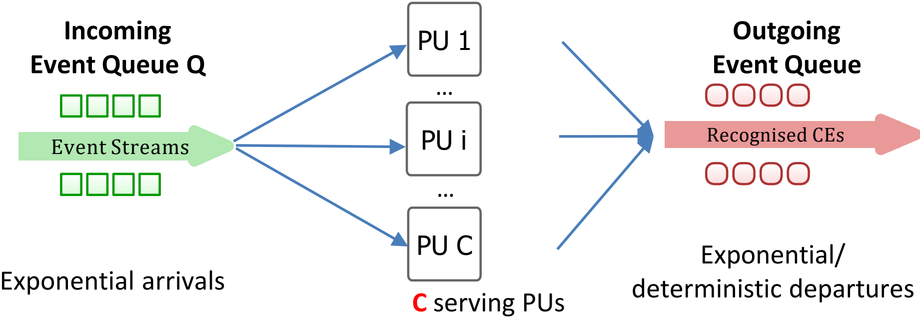
  - Reinforcement Learning Approach
    - Look up table describing "benefit" of each action based on recent experience

Utilization	Scale In	No Action	Scale Out
80%	0.28	0.7	0.88
90%	0.28	0.5	0.9
100%	0.1	0.4	1.0

Elastic Resource Allocation in CER – Queueing Models [Mayer et al, IEEE BigData'14]

Key Concepts

- Workload-, Latency-, Load-shedding Aware Scheme
- Choices based on probabilistic buffer limit (BL)



$_{now}P = P(Q(t) \leq BL) < P_{thres}$?

Yes → $C \leftarrow C+1$

No → $_{last}P > P_{thres}$?

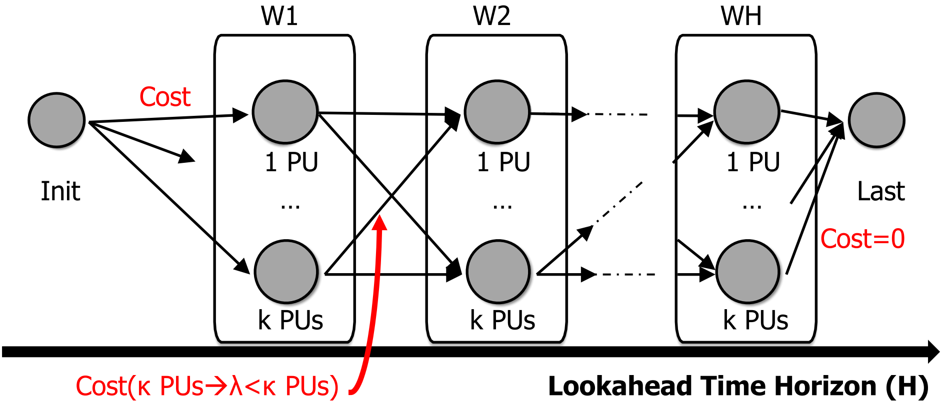
Yes → $C \leftarrow C-1$

No → Return C

Elastic Resource Allocation in CER – Time Series-based [Zacheilas et al, IEEE BigData'15]

Key Concepts

- Monitor event input rate and processing latency
- Predict their values (<u>Gaussian Processes</u>, SVM, NNs)
- Construct state graph and compute shortest path



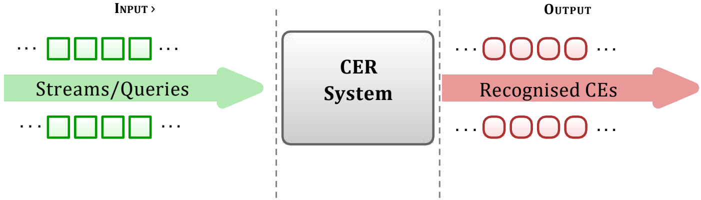
Scalable - Distributed Complex Event Recognition

Why? Well, It's the Big Data Era

- Volume, Velocity, Variety, Veracity

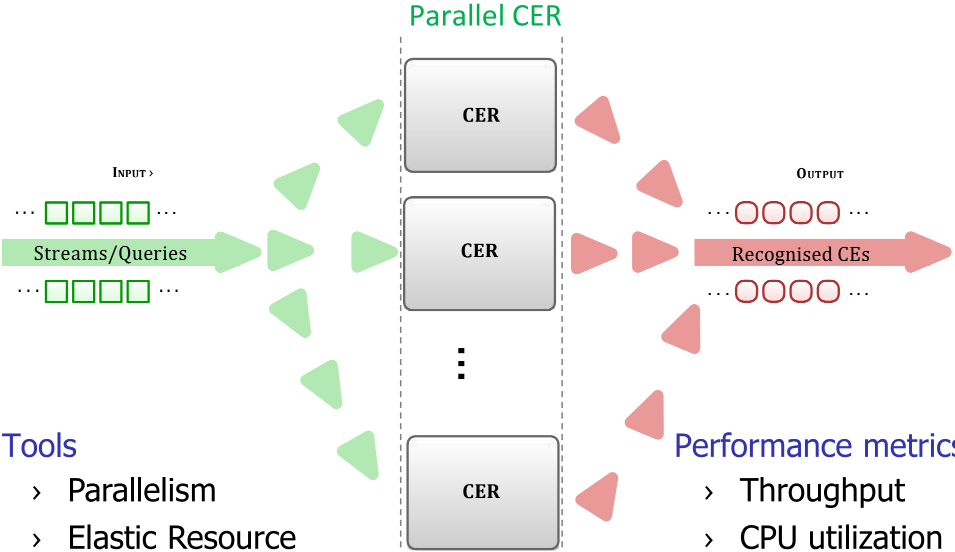
Centralized Architecture

Sequential CER



Scalable - Distributed Complex Event Recognition

Clustered Architecture



Tools

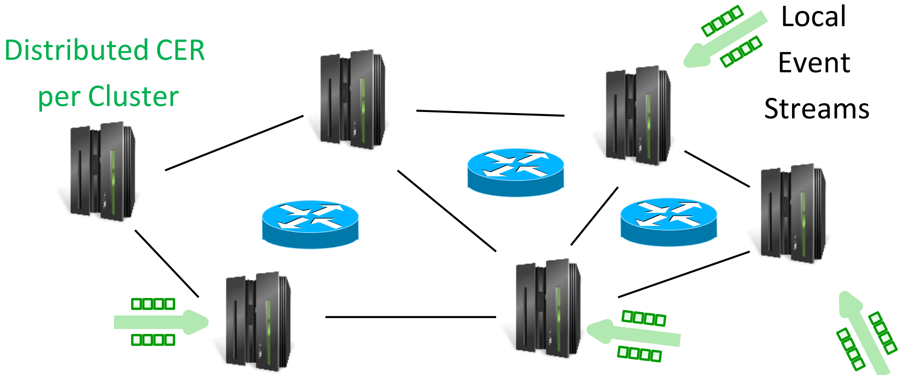
- Parallelism
- Elastic Resource Allocation

Performance metrics

- Throughput
- CPU utilization

Scalable - Distributed Complex Event Recognition

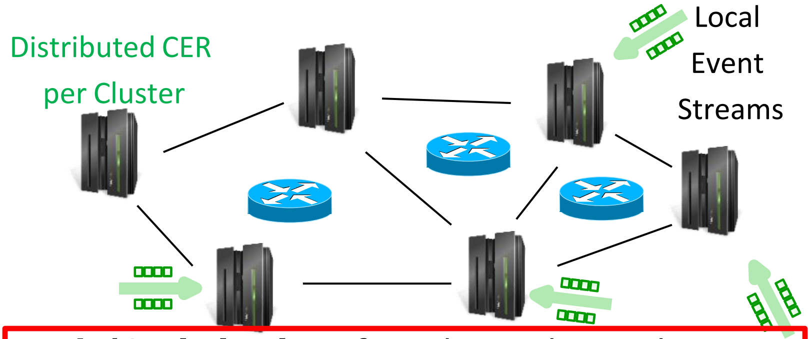
Networked Architecture: Geographically Distributed CER



- Business User Poses CER queries (business logic)
- The business logic is **<u>independent</u>** of geographic locations
  - Does not specify which operations are performed at each site
- Goal: Use business logic and perform **"efficient"** CER
  - Data Centralization often not possible in Big Data Applications

Key Ingredients for Distributed CER in Big Data

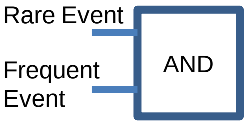
Networked Architecture: Geographically Distributed CER



- **<u>Tools/Optimizations</u>** for reducing data exchange between clusters
- **<u>Architectures</u>** that support these tools
- An **<u>optimizer</u>**: decide best way to distribute business logic given tools & architecture

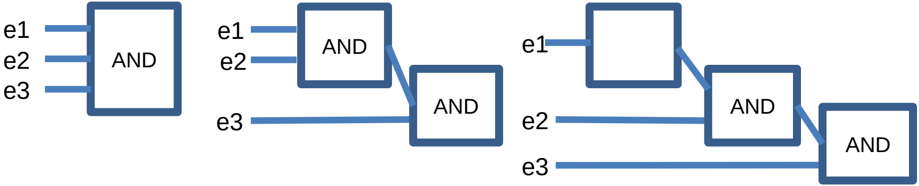
Tool 1 for In-Situ Processing: Push-Pull Paradigm

Key Concept: Do not transmit frequent events, unless rare events occur. May increase latency but decreases network cost



- Decreases Network Cost
- Increases Latency
- Increase Buffer Requirements (for cached events that may be pulled later)
- Same idea can speed up CER WITHIN a cluster [Kolchinsky et al, DEBS'15]

Example: Different ways of evaluating AND

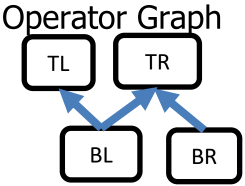


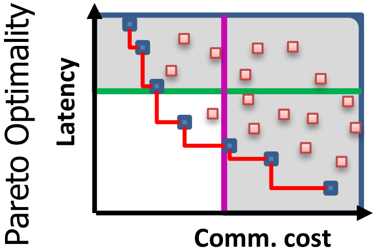
e3 is pulled when e1 and e2 appear

e2 is pulled when e1 appears
e3 is pulled when e1 and e2 appear

Push-Pull Approach for CER [Adkere et al, PVLDB'08]

Single site





Key Ideas:

- All operators evaluated at a <u>central</u> site/cluster
- Data pushed/pulled to central location based on desired optimization criteria
    - Bandwidth Cost, Latency, Available Memory
- DP + Greedy Algorithms provided

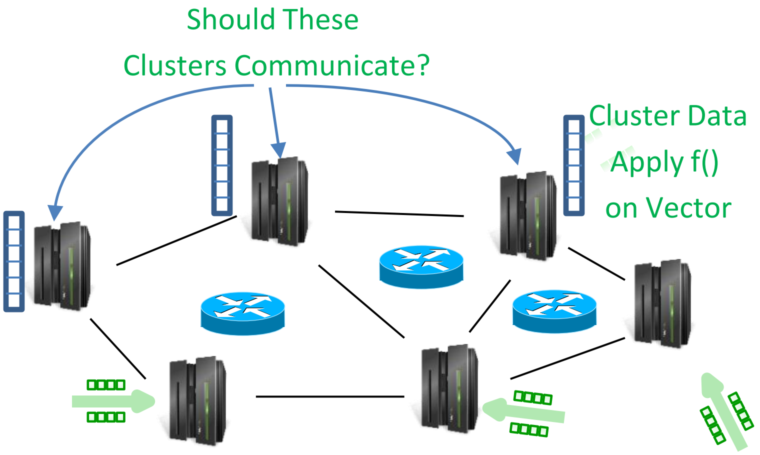
Sufficient for Big Data CER?

- Processing not actually pushed inside the network
- May not be suitable for large scale distributed topologies

Tool 2: Distributed Function Monitoring (DFM)

Key Idea:

- Define a function f() over the data of different clusters
- Communicate only when function f() crosses a threshold



Tool 2: Distributed Function Monitoring (DFM)

Key Idea:

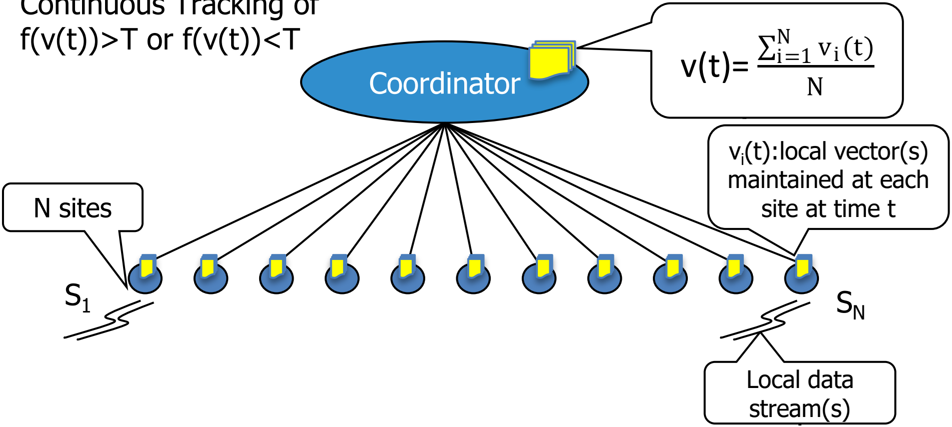
- Define a function f() over the data of different clusters
- Communicate only when function f() crosses a threshold
- Definition of function depends on desired task
  - Simple aggregates of data cross a threshold (i.e., SUM)
  - Event frequency statistics have changed significantly (i.e., Cosine Similary, Pearson Coefficient etc)
  - The global model of the data has changed significantly (Distributed Machine Learning)
  - The variance of some data has changed significantly
  - And many more…

Key Tool: Geometric Monitoring

- **Generic tool**
- DFM problem much simpler for linear functions
- One may derive more efficient solutions for specific functions

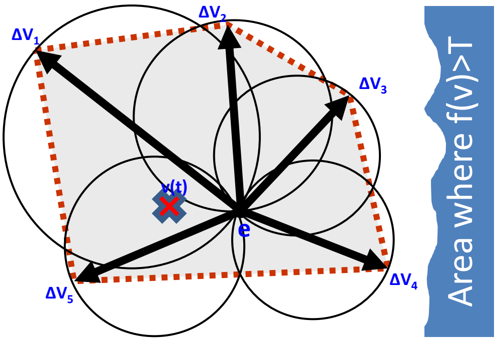
Basic Tool: Geometric Monitoring (GM) - Setup

Continuous Tracking of
$f(v(t))>T$ or $f(v(t))<T$



› Track if $f(v(t))>T$
› Works for **any** f() over the (weighted) average of local $v_i(t)$

Basic GM Scheme [Sharfman et al, SIGMOD'06]



- e(t): Last known average vector
- Sites check f() within $B(e+ \Delta v_i/2, ||\Delta v_i||/2)$
- If union of $B(e+ \Delta v_i/2, ||\Delta v_i||/2)$ crosses the threshold, v(t) **may have crossed the threshold**

Key Points

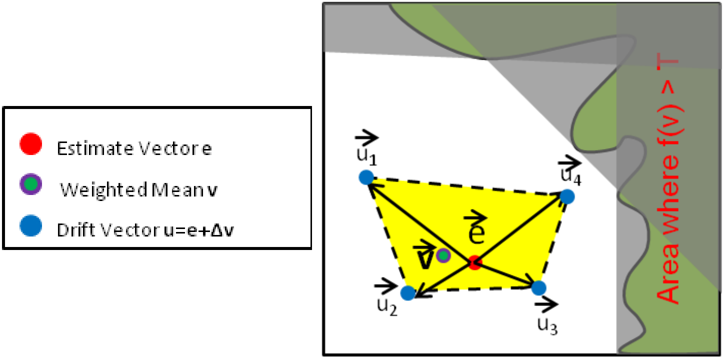
- Monitoring done in a distributive way
- Sites perform **local tests** to see if f() may have crossed T
  - Test: find min/max of f() over a sphere (costly!)

- Many improvements have followed…

GM Scheme – Key Advances

Key Problems & Solutions (at a glance)

- Make the local test much simpler and more efficient
- Safe Zones [Keren et al, TKDE’12]
  - Check if e+ Δv_i is inside a “safe” convex region
- Convex Decomposition + Convex Bounds [Lazerson et al, PVLDB’15, KDD’16]
  - Methodology to help find a good safe zone

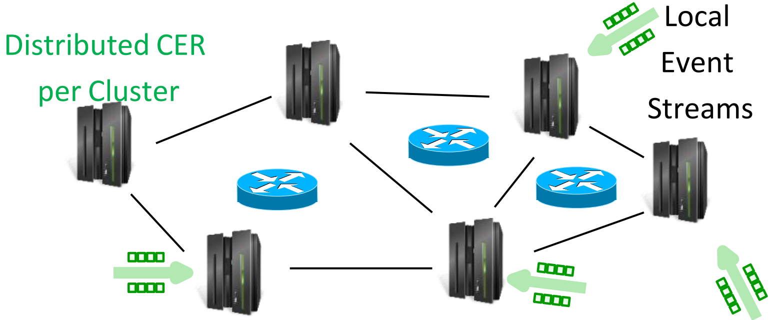


GM Scheme – Key Advances (cont)

Key Problems & Solutions (cont.)

- Prediction Models [Giatrakos et al, SIGMOD'12, TODS'14]
  - If we can predict the values of the local vectors, can we do better?
- Sampling [Giatrakos et al, SIGMOD'16]
  - For many sites, chances of communication increases $\Rightarrow$ use sampling
- Sketches [Garofalakis et al, PVLDB'13]
  - How to combine GM with sketches if vectors are too large

Key Ingredients for Distributed CER in Big Data

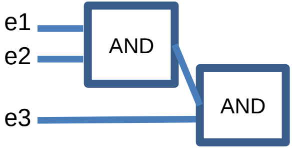


- **<u>Tools/Optimizations</u>** for reducing data exchange between clusters
  - Push-pull paradigm (for regular event operators)
  - Distributed Function Monitoring/GM
- **<u>Architectures</u>** that support these tools
- An **<u>optimizer</u>**: decide best way to distribute business logic given tools & architecture

Architectures for Distributed CER in Big Data

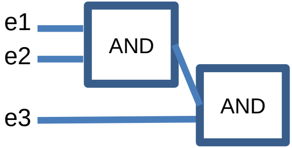
- **No current support** for desired tools for CER
  - Push-pull paradigm, Distributed Function Monitoring/GM

- How hard is it to develop them? Simplest approach
  - Take a CER engine for distributed (intra-cluster) CER
  - Move Distributed Function Monitoring **outside** the CER engine
    - Easier to write custom code this way



Architectures for Distributed CER in Big Data (cont.)

- How hard is it to develop them? Simplest approach
  - The CER engine must emit an event on pull requests
    - Event must be handled **<u>outside</u>** the CER engine
    - Emitting events is simple and done for output events
    - Pull requests can only occur on state transitions
    - Not too much code to add
    - Hardest task: out of order data
    - Let’s see an example…

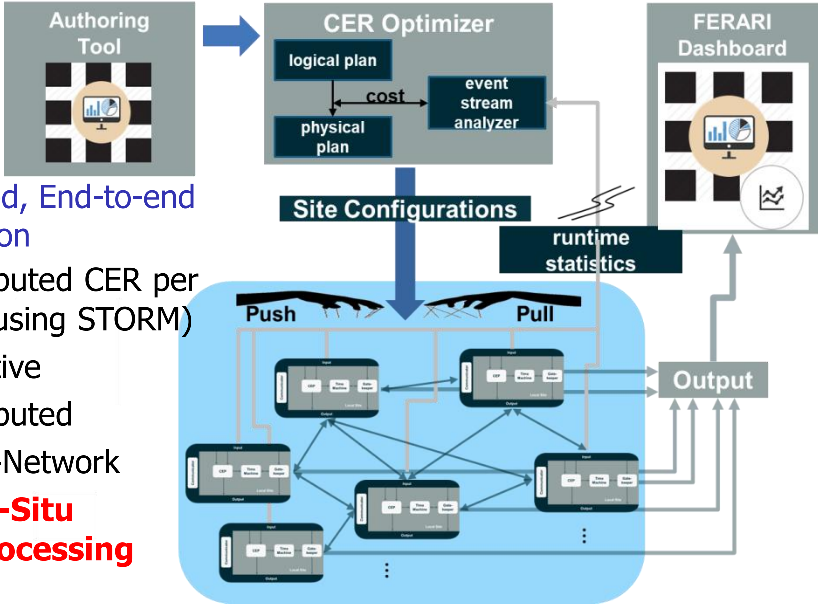


An Architecture for CER in Big Data Applications

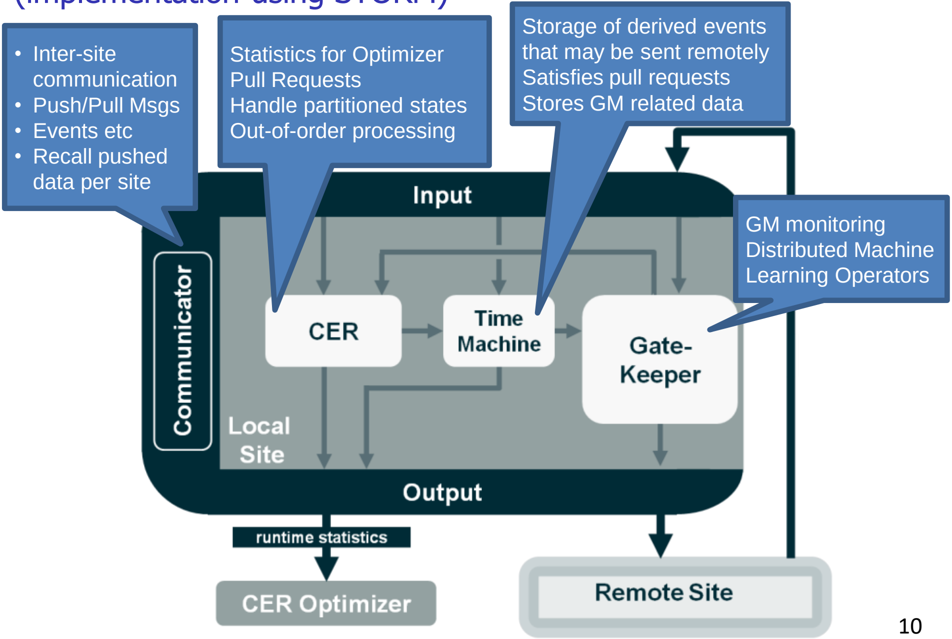
The FERARI Approach [Flouris et al, SIGMOD'16]

Full-fledged, End-to-end CER solution

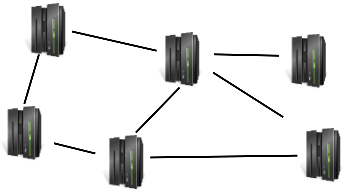
- Distributed CER per site (using STORM)
- Adaptive
- Distributed
  - In-Network
  - **In-Situ Processing**



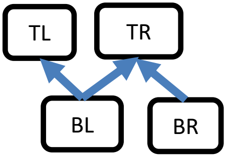
FERARI [Flouris et al, SIGMOD'16]: Inside each Cluster
(implementation using STORM)



Optimizer Inputs



Network of Sites



Operator Graph

Inputs

- Business Logic
- Network Parameters
- Event Frequency Statistics
- Optimization Goals

In-Network Processing → Operator Placement Problem

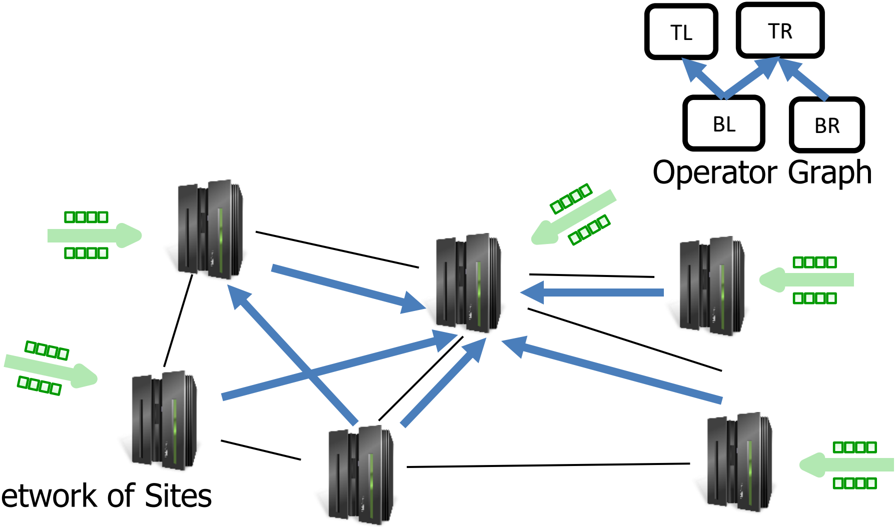
Goals:

- exploit data Variety,
- ***push computation to sites***

Distributed Complex Event Recognition

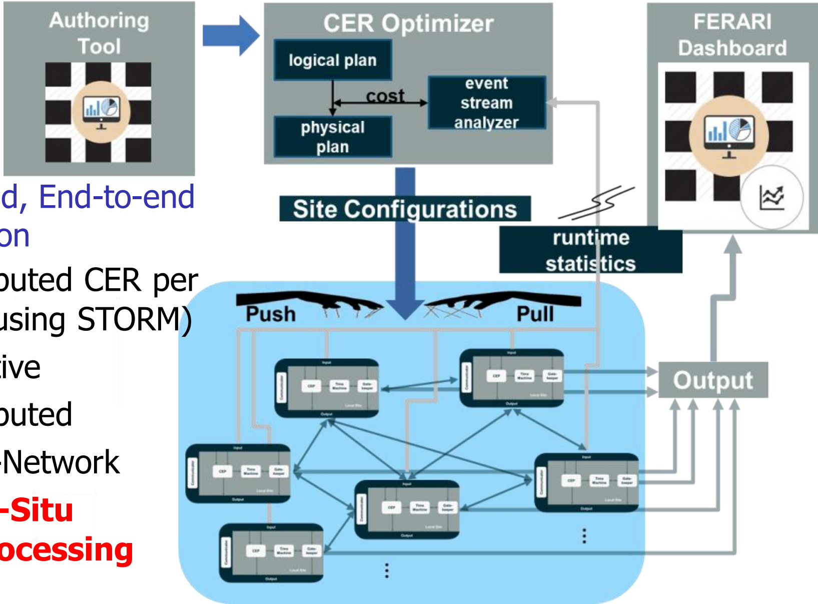
In-Network Processing → Operator Placement Problem in Traditional Streaming Settings

- Key Concept: exploit data Variety, push computation to sites



An Architecture for CER in Big Data Applications

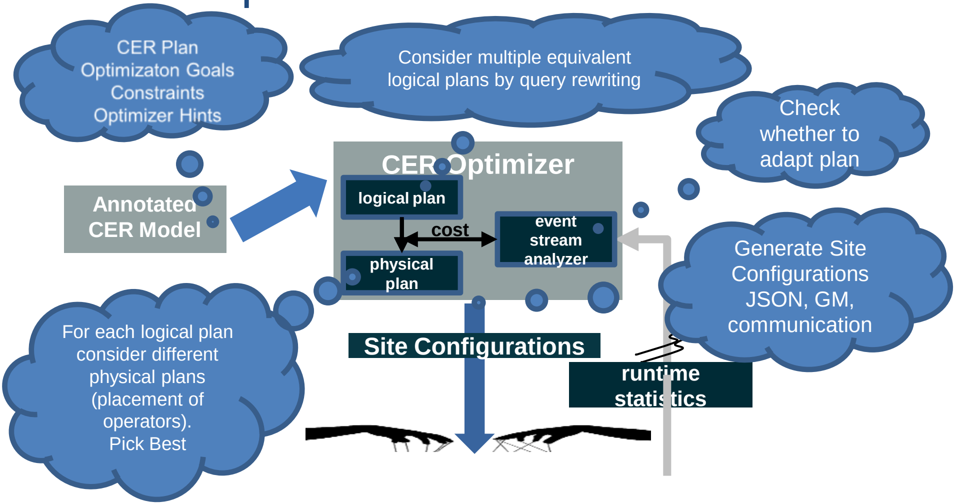
The FERARI Approach [Flouris et al, SIGMOD'16]



Full-fledged, End-to-end CER solution

- Distributed CER per site (using STORM)
- Adaptive
- Distributed
  - In-Network
  - **In-Situ Processing**

FERARI Optimizer



Optimizer mostly independent of underlying CER engine

Outlook

Future Exciting Research Domains

- **IoT Domain**
  - 100,000s of nodes
  - Heterogeneous capabilities
  - Not data centers
  - How to detect complex events?
  - In-situ processing extremely crucial

- **Automatic Learning & Adaptation of CER patterns**
  - Patterns of interest change over time

- **Effective Support for Complex Analytics Operators**
  - E.g., time series analysis, machine learning

Additional Readings (beyond what is in tutorial's abstract)

- G. Cugola, A. Margara. Processing Flows of Information: From Data Stream to Complex Event Processing. ACM Computing Surveys, 2012.
- E. Alevizos, A. Skarlatidis, A. Artikis, G. Paliouras. Probabilistic Complex Event Recognition: A Survey. ACM Computing Surveys, 2017.
- G. Cugola, A. Margara. Low latency complex event processing on parallel hardware. J. Parallel Distrib. Comput., 2012.
- T. Heinze, V. Pappalardo, Z. Jerzak, C. Fetzer. Auto-scaling techniques for elastic data stream processing. In DEBS, 2014.
- R. Mayer, B. Koldehofe, K. Rothermel. Meeting predictable buffer limits in the parallel execution of event processing operators. In IEEE BigData, 2014.
- I. Kolchinsky, I. Sharfman, A. Schuster. Lazy evaluation methods for detecting complex events. In DEBS, 2015.

Additional Readings (beyond what is in tutorial's abstract)

- N. Giatrakos, A. Deligiannakis, M. Garofalakis. Scalable Approximate Query Tracking over Highly Distributed Data Streams. In SIGMOD, 2016.
- D. Keren, I. Sharfman, A. Schuster, A. Livne: Shape Sensitive Geometric Monitoring. IEEE Trans. Knowl. Data Eng., 2012.
- A. Lazerson, I. Sharfman, D. Keren, A. Schuster, M. Garofalakis, V. Samoladas: Monitoring Distributed Streams using Convex Decompositions. PVLDB, 2015.
- A. Lazerson, D. Keren, A. Schuster: Lightweight Monitoring of Distributed Streams. In KDD, 2016.
- M. Garofalakis, D. Keren, V. Samoladas: Sketch-based Geometric Monitoring of Distributed Stream Queries. PVLDB, 2013.