Beyond Analytics

The Evolution of Stream Processing Systems

Prospects

Paris Carbone, Marios Fragkoulis, Vasiliki Kalavri, Asterios Katsifodimos

Slides: **streaming-research.github.io/Tutorial-SIGMOD-2020**

Tutorial overview

- Part I: Introduction & Fundamentals (Vasia)
- Part II: Time, Order, & Progress (Marios)
- Part III: State Management (Paris)
- Part IV: Fault Recovery & High Availability (Marios)
- Part V: Load Management & Elasticity (Vasia)
- Part VI: Prospects (All)

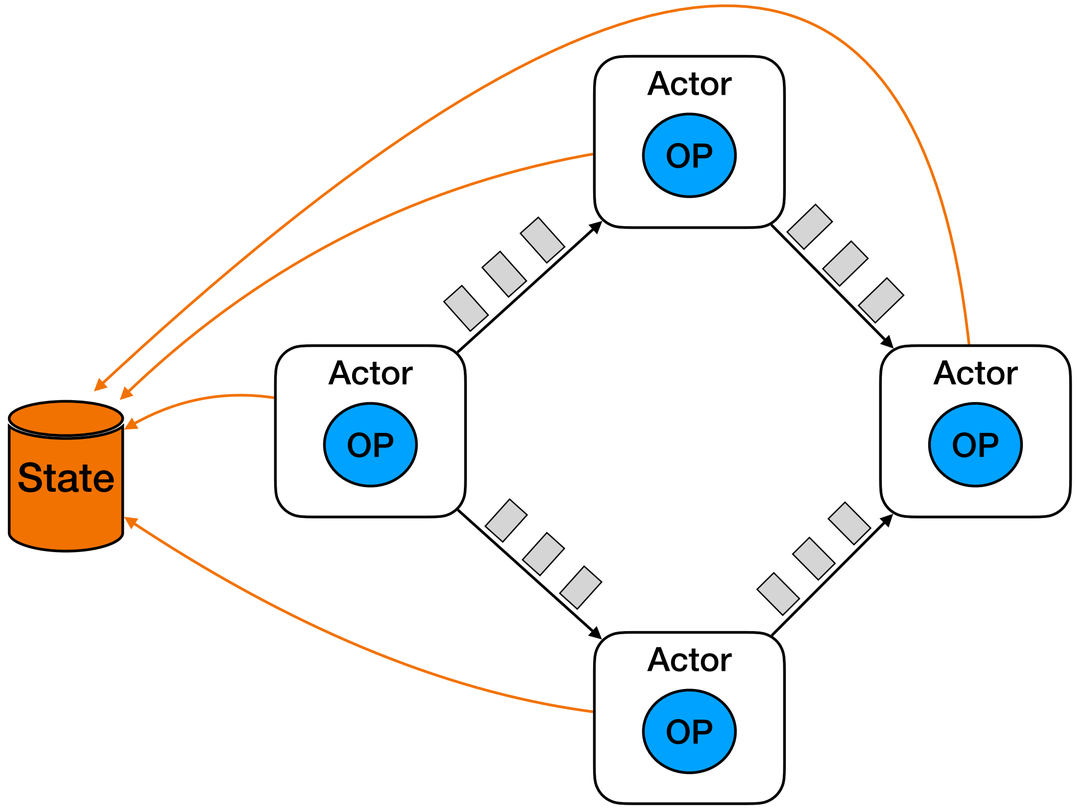
Prospects

Agenda

1. **Emerging applications** (Cloud, ML, Streaming Graphs)
2. **New requirements**
3. Conclusion, references, contacts

Cloud programming frameworks

Actors & Streams

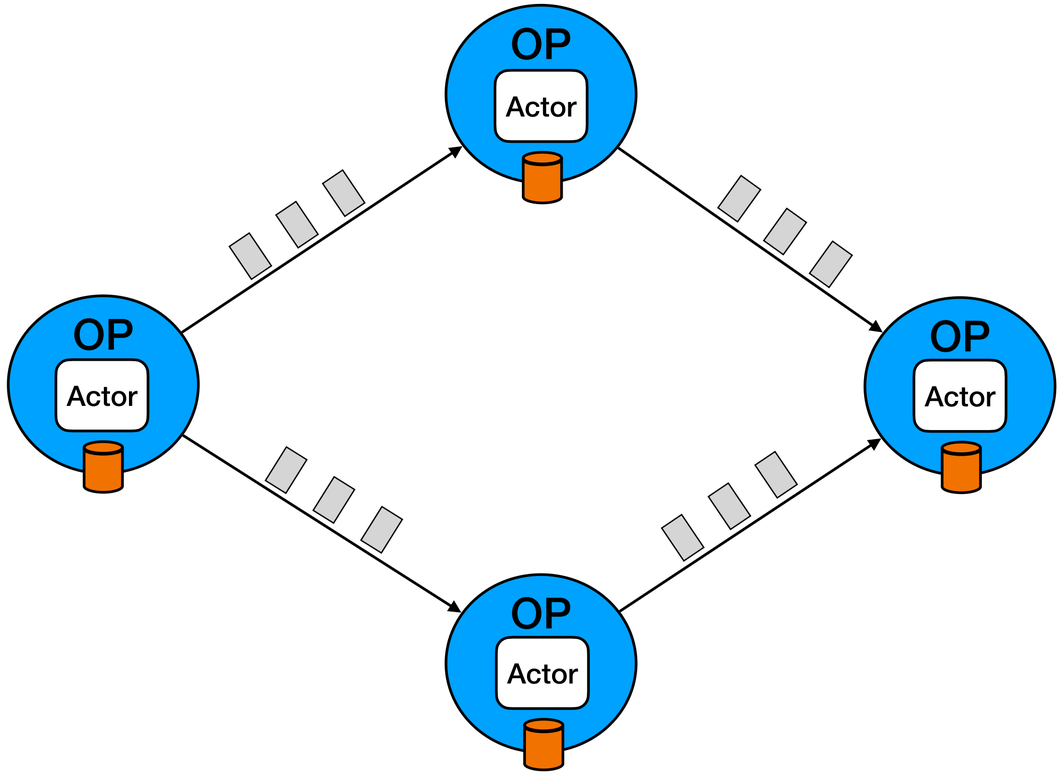


Streams on Actors

Orleans (CIDR'19)

RAY (OSDI'18)

akka streams

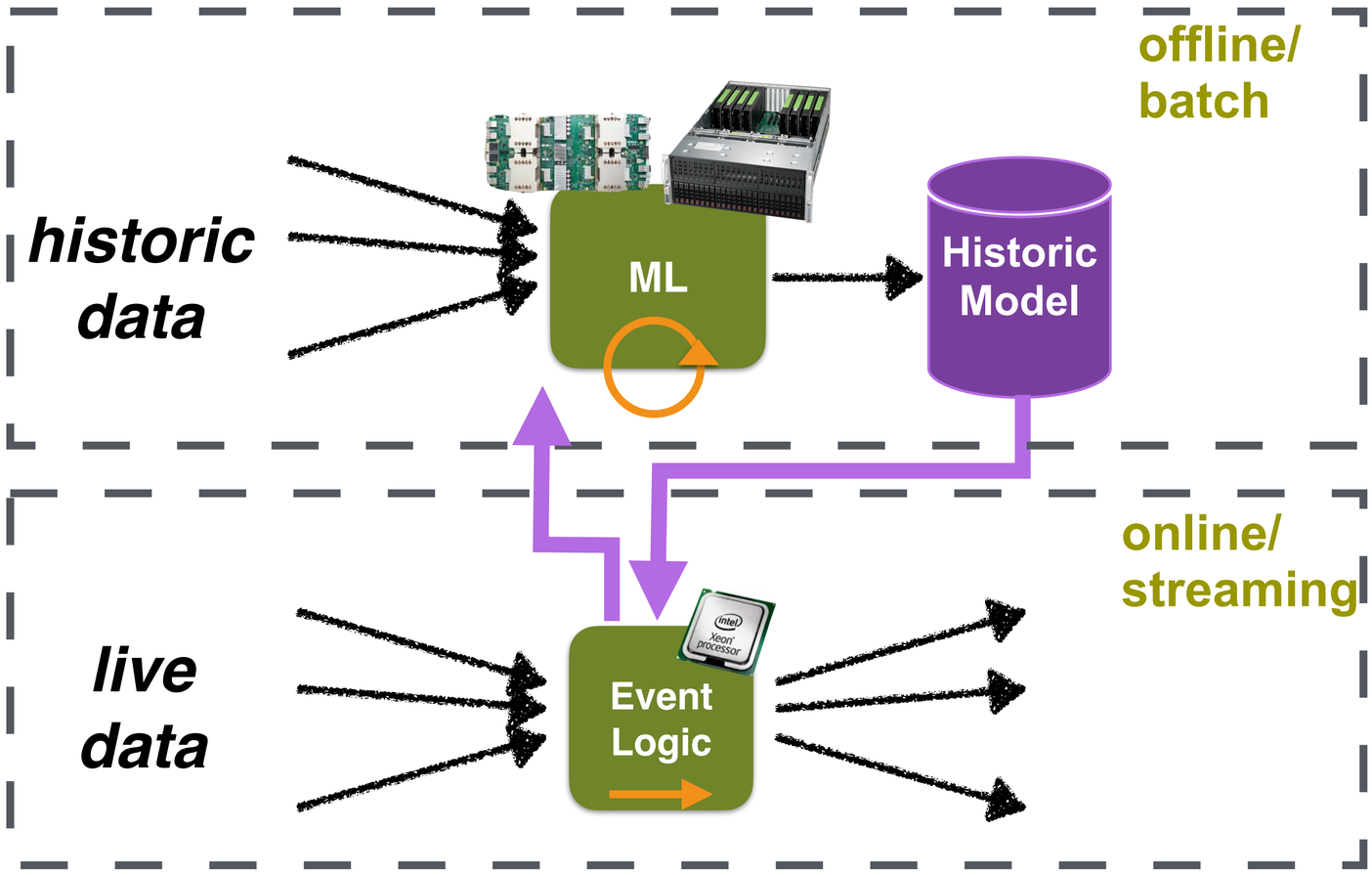


Actors on Streams

sFaaS (VLDB'19)

Stateful Functions

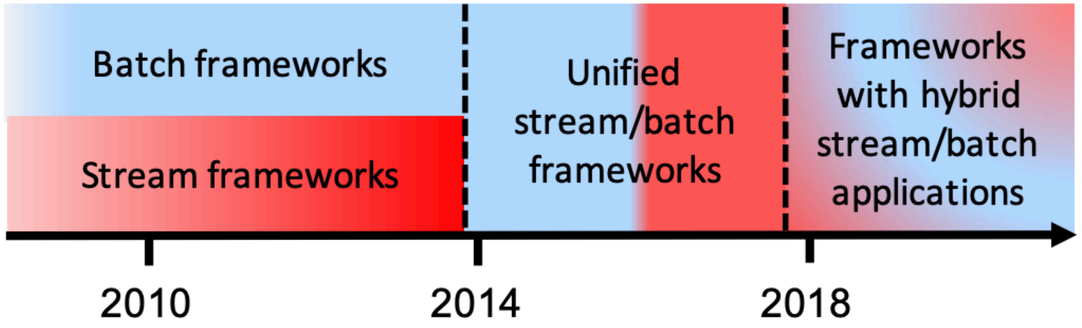
Integrating Shallow+Deep ML Stages



- **Shallow (~stream)**: preprocessing, feature selection, serving a model etc.
- **Deep (~batch)**: SGD, Bulk/State-Iterative Processing, Any Model Training Algorithm
- Shallow phases are implemented today in existing SP systems
- Deep phases require a combination (if not all) of the following:
  - Hybrid short/long-running task execution
  - Consistent model updates (synchronization on side-inputs)
  - Bulk stream iterations

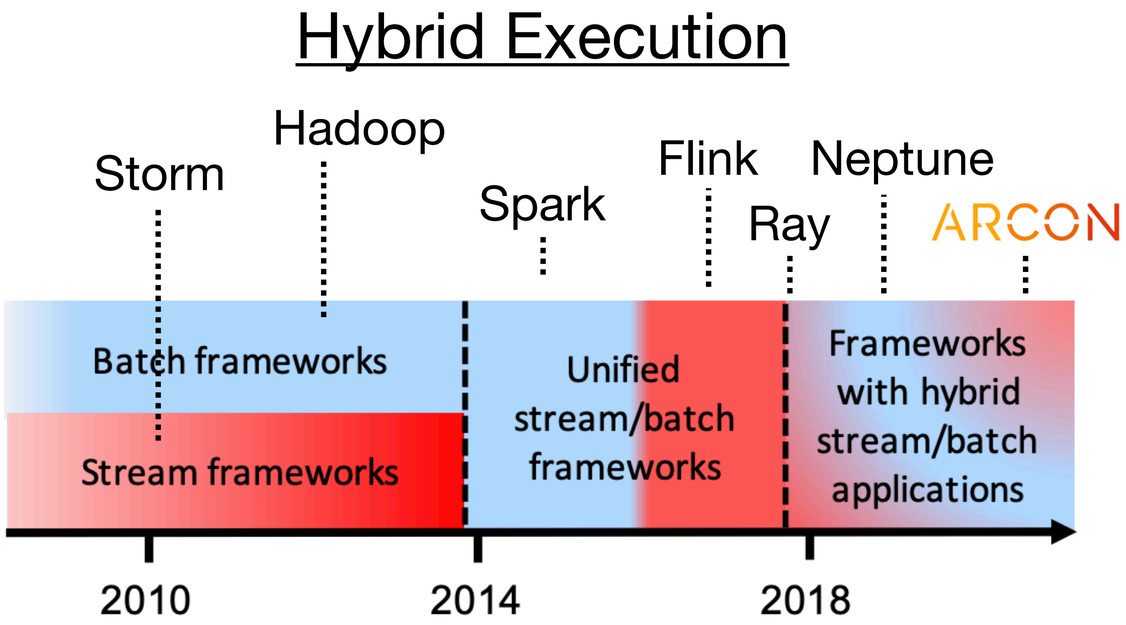
Steps Forward in ML in Data Streaming

Hybrid Execution



Neptune: Scheduling Suspendable Tasks for Unified Stream/Batch Applications
Garefalakis P, Karanasos K, Pietzuch P in SoCC (2019)

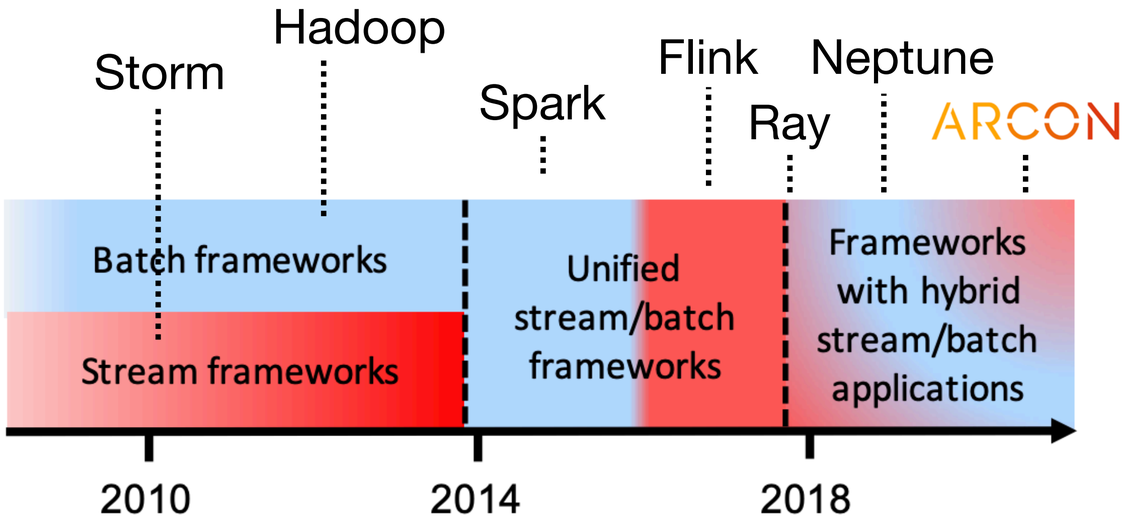
Steps Forward in ML in Data Streaming



Neptune: Scheduling Suspendable Tasks for Unified Stream/Batch Applications
Garefalakis P, Karanasos K, Pietzuch P in SoCC (2019)

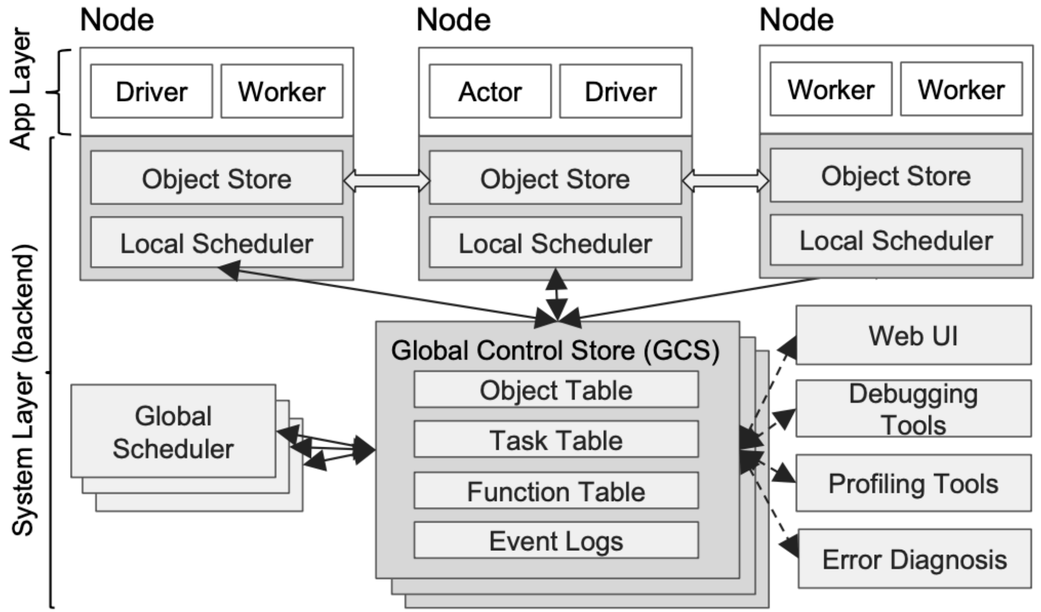
Steps Forward in ML in Data Streaming

Hybrid Execution



Neptune: Scheduling Suspendable Tasks for Unified Stream/Batch Applications
Garefalakis P, Karanasos K, Pietzuch P in SoCC (2019)

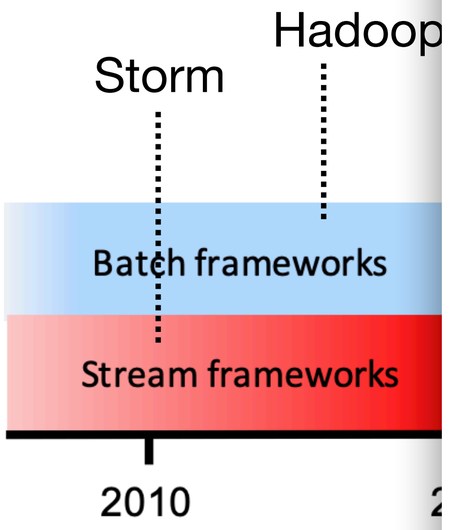
Ray : Actors + Remote Functions for ML



Ray: A Distributed Framework for Emerging AI Applications
Moritz P, Nishihara R, Wang S, et. al. in USENIX OSDI (2018)

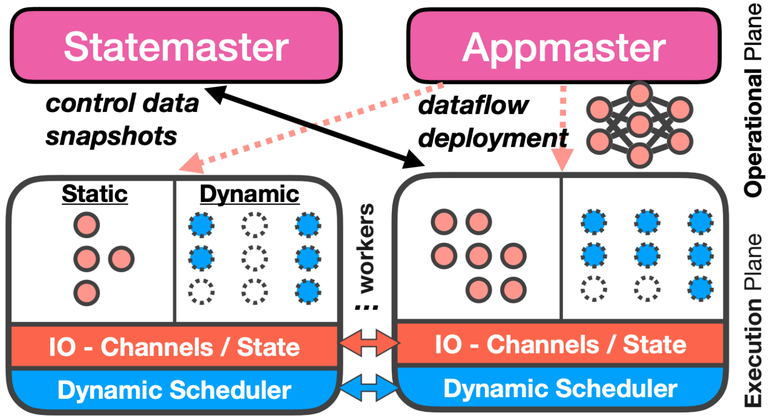
Steps Forward in ML in Data Streaming

Hybri



Neptune: Scheduling Suspen
Garefalakis P, Karanasos K, Pietzuch

Example: Batch+Stream Hybrid Execution



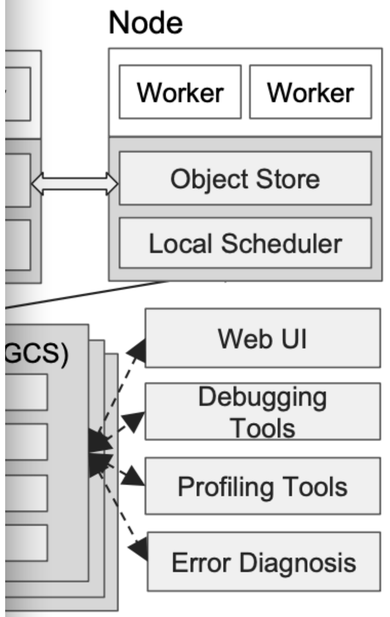

```
|source: Stream[i32],
 odd_sink: StreamAppender[i32],
 even_sink: StreamAppender[i32]|
  let mapped = result(
    for(source, StreamAppender[i32], |sink, element|
        merge(sink, element + 5)));
  for(mapped,  odd_sink, |sink, e|
    if(e % 2 != 0, merge(sink, e), sink));
  for(mapped, even_sink, |sink, e|
    if(e % 2 == 0, merge(sink, e), sink))
```

Arc: intermediate representation for batch+stream computation

Arcon: Continuous and deep data stream analytics
Meldrum M, Segeljakt K, Kroll L, et. al. in BIRTE (2019)

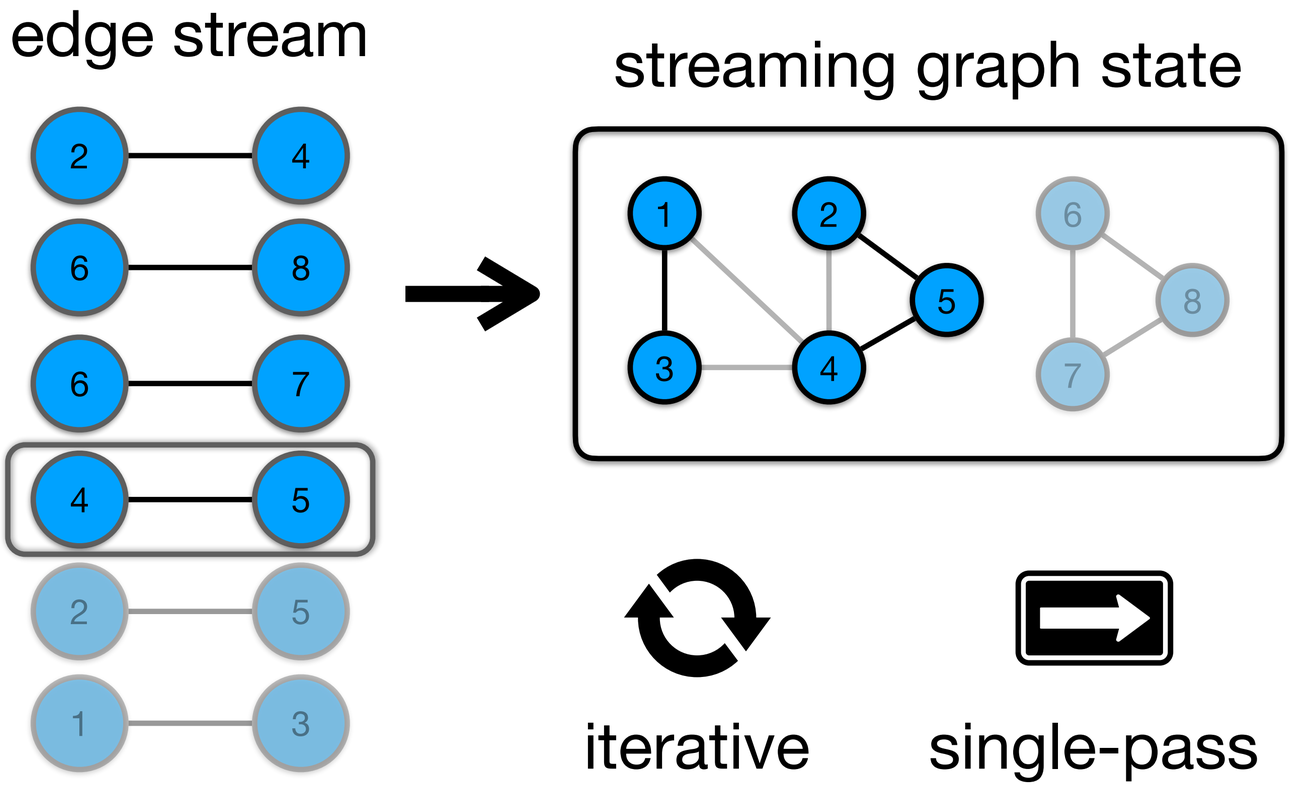
Arc: An IR for batch and stream programming
Kroll L, Segeljakt K, Schulte C, et. al. in Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation

unctions for ML



Emerging AI Applications
Moritz P, Nishihara R, Wang S, et. al. in USENIX OSDI (2018)

Graph streams model interactions as events that update an underlying graph structure



Edge events:

A purchase, a movie rating, a like on an online post, a bitcoin transaction, a packet routed from a source to destination

Vertex events:

A new product, movie, user

Open problems in graph streaming

- Streaming graph **state** management
  - *See: **LiveGraph: a transactional graph storage system with purely sequential adjacency list scans**. (VLDB'20).*
- Streaming graph **partitioning** (of unbounded graphs)
  - *See: **Streaming graph partitioning: an experimental study**. (VLDB'18)*
- Graph streaming **algorithms**
  - *See: **Regular Path Query Evaluation on Streaming Graphs**. (SIGMOD'20 Research 16: Graph and Stream Processing, Thursday 1:30 PM – 3:00 PM).*
- **Languages** and operator semantics

For an overview, see: ***Practice of Streaming and Dynamic Graphs: Concepts, Models, Systems, and Parallelism*** *(arxiv.org/abs/1912.12740).*

Novel APIs and abstractions

Cloud Apps, ML, Graphs

functional APIs

```
val maxTemp = sensorData
  .map(r => Reading(r.id, r.time, r.temp)
  .keyBy(_.id)
  .max("temp")
maxTemp.print()
```

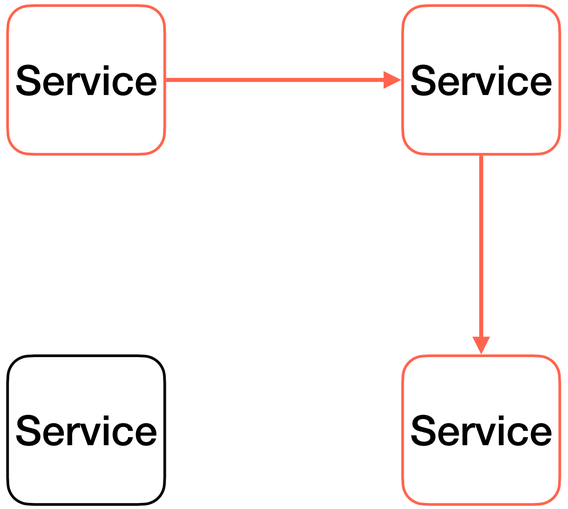
streaming SQL

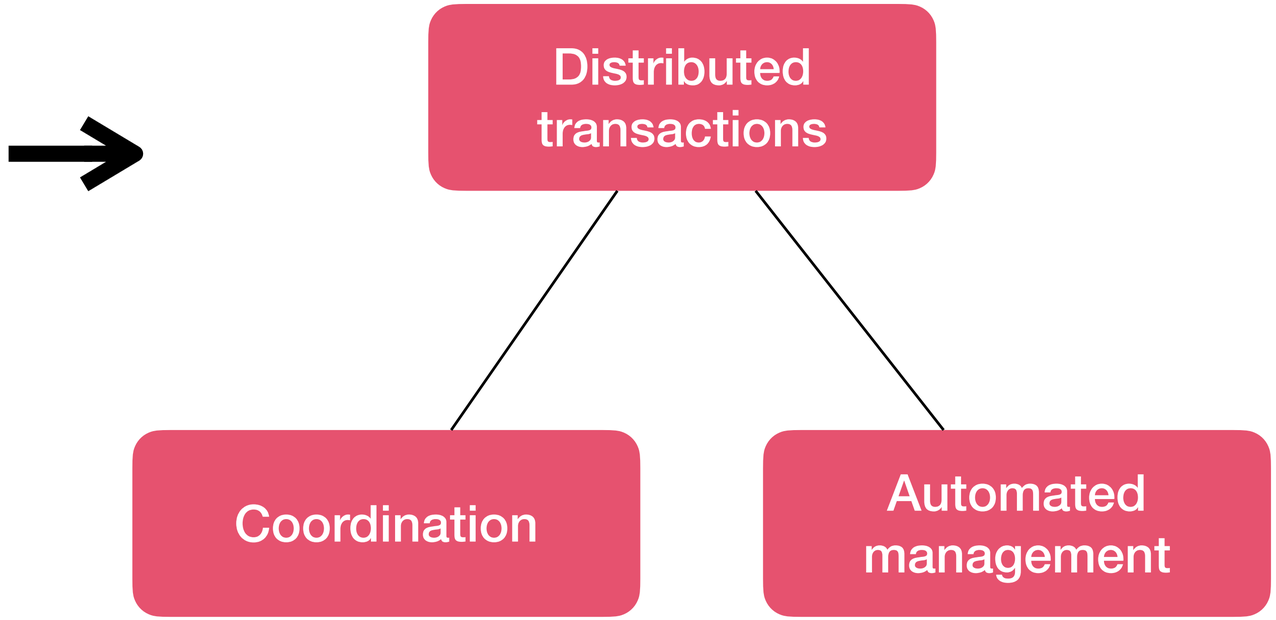
```
SELECT *
FROM Tumble(
    data = > TABLE ( Bid ),
    timecol = > DESCRIPTOR ( bidtime ),
    dur = > INTERVAL '10' MINUTES,
    offset = > INTERVAL '0' MINUTES );
```

- Actor-like APIs
- Streaming ML APIs
- Streaming graph abstractions

Application transactions

Cloud Apps



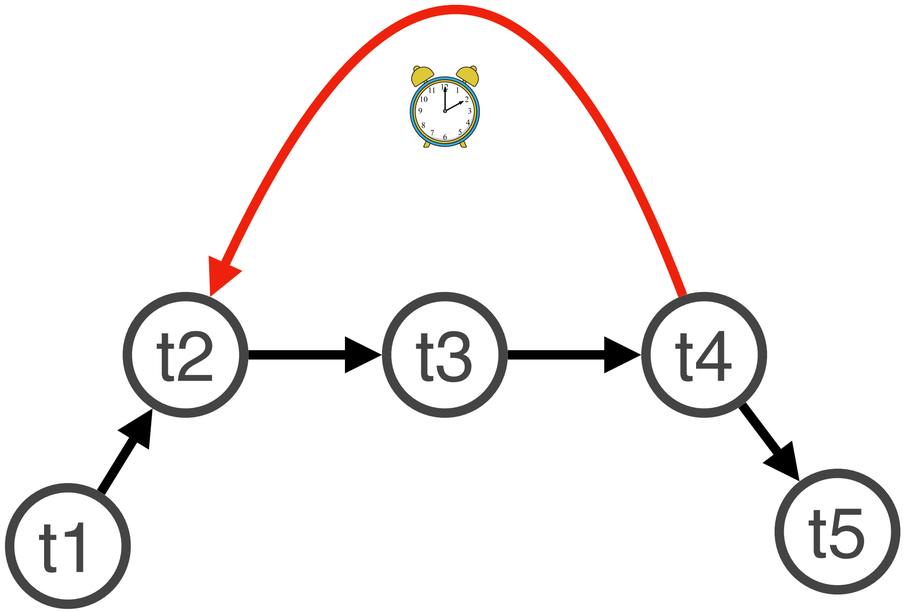


S-Store (VLDB ’14)

Iterative computation on streams

Cloud Apps, ML, Graphs

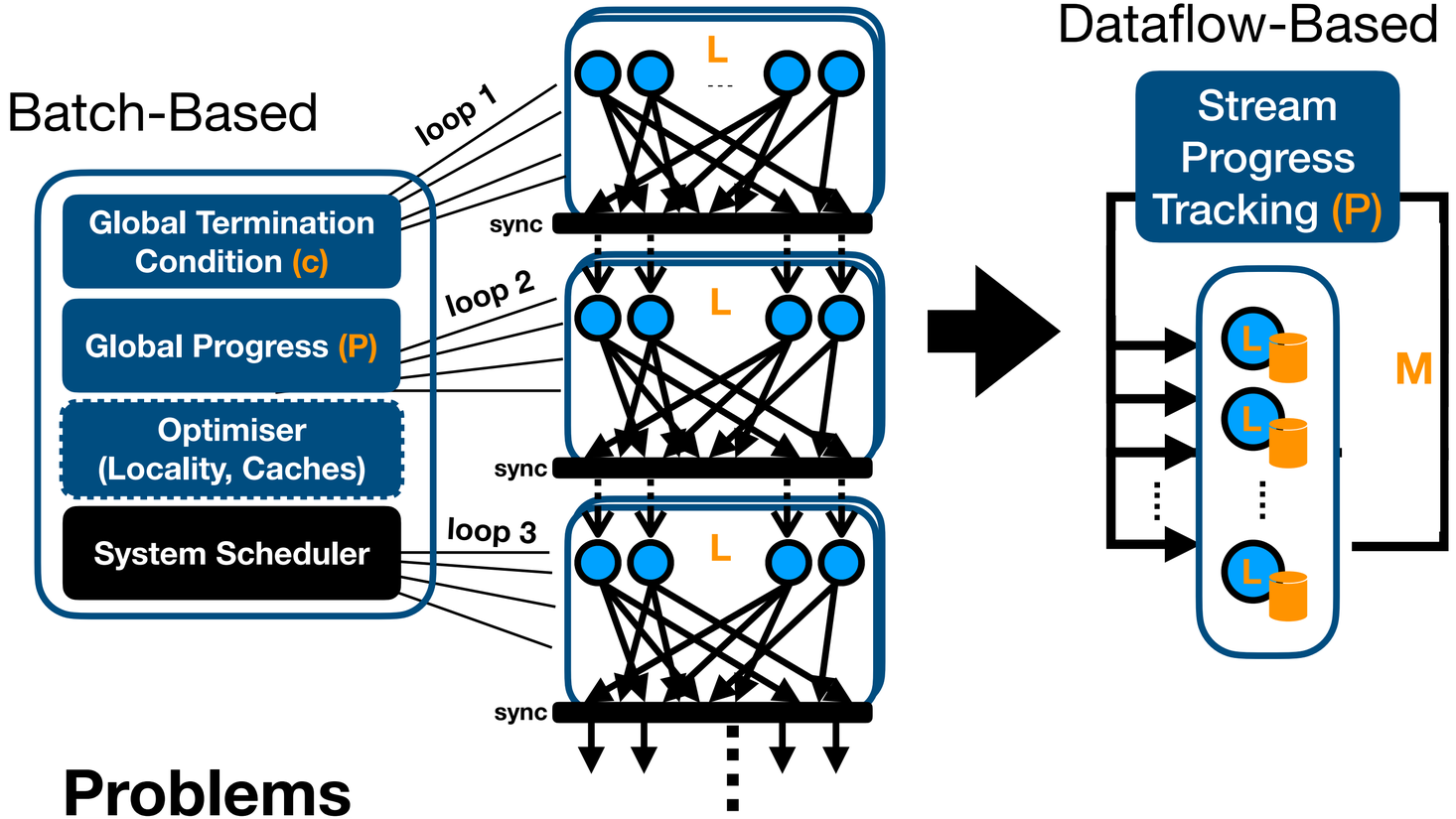
Asynchronous Loops



Problems

- Deadlocks - Cyclic Flow Control
- Time Progress?

Synchronous Loops



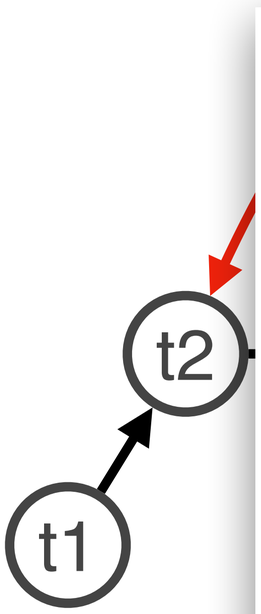
Problems

- Decentralised Global Progress Tracking
- Semantics + Arbitrary Nesting

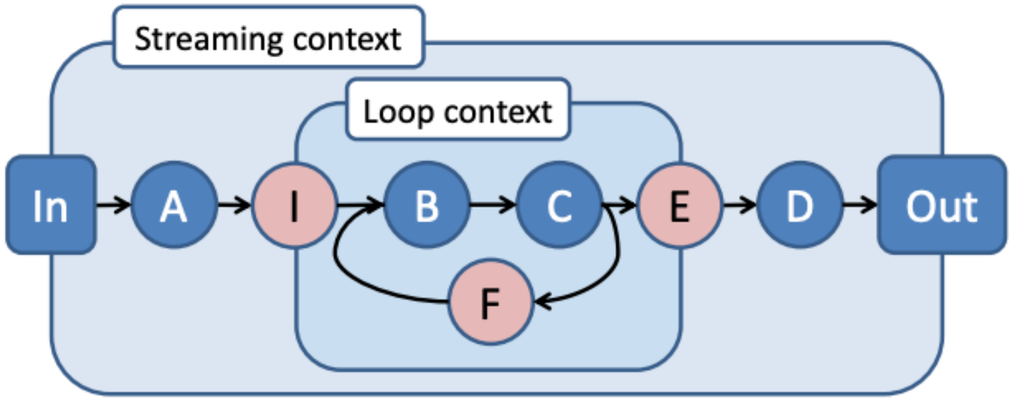
Iterative computation on streams

Cloud Apps, ML, Graphs

Asynchronous Loops



Naiad, Timely Dataflow



Naiad: A timely dataflow system

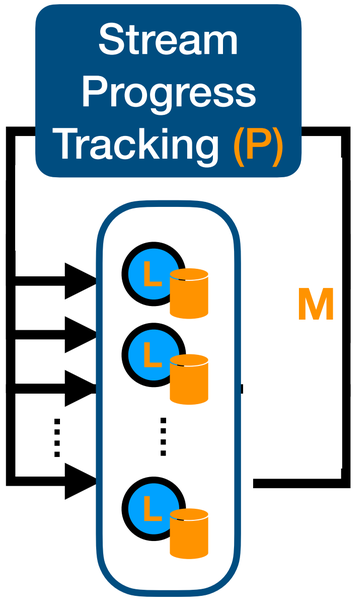
Murray D, McSherry F, Isaacs R, et. al. in SOSP 2013 - Proceedings of the 24th ACM Symposium on Operating Systems Principles (2013)

Based on Microbatch Model (no event-time)

- Deadlocks - Cyclic Flow Control
- Time Progress?

Synchronous Loops

Dataflow-Based



- Decentralised Global Progress Tracking
- Semantics + Arbitrary Nesting

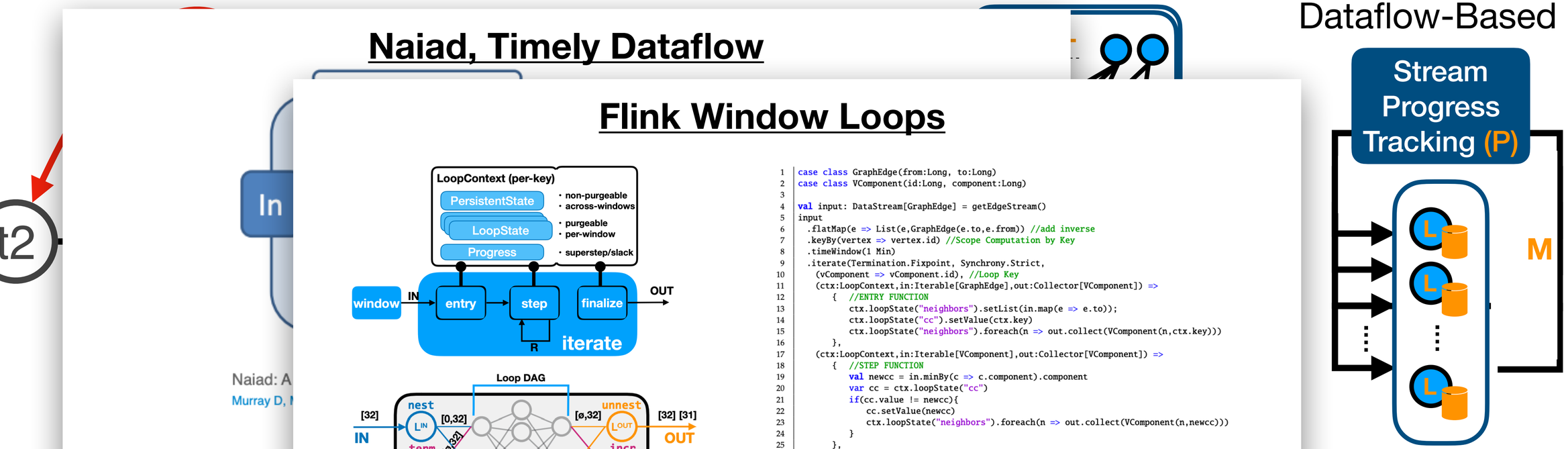
Iterative computation on streams

Cloud Apps, ML, Graphs

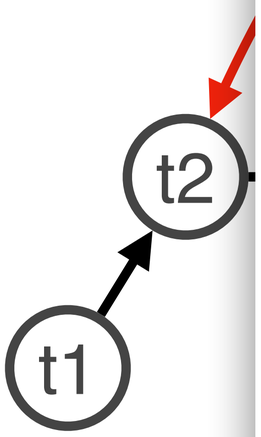
Asynchronous Loops

Synchronous Loops

Dataflow-Based



Naiad, Timely Dataflow



Naiad: A
Murray D, M

Based c

Flink Window Loops


```
case class GraphEdge(from:Long, to:Long)
case class VComponent(id:Long, component:Long)

val input: DataStream[GraphEdge] = getEdgeStream()
input
  .flatMap(e => List(e,GraphEdge(e.to,e.from)) //add inverse
  .keyBy(vertex => vertex.id) //Scope Computation by Key
  .timeWindow(1 Min)
  .iterate(Termination.Fixpoint, Synchrony.Strict,
    (vComponent => vComponent.id), //Loop Key
    (ctx:LoopContext,in:Iterable[GraphEdge],out:Collector[VComponent]) =>
        {  //ENTRY FUNCTION
           ctx.loopState("neighbors").setList(in.map(e => e.to));
           ctx.loopState("cc").setValue(ctx.key)
           ctx.loopState("neighbors").foreach(n => out.collect(VComponent(n,ctx.key)))
        },
    (ctx:LoopContext,in:Iterable[VComponent],out:Collector[VComponent]) =>
        {  //STEP FUNCTION
           val newcc = in.minBy(c => c.component).component
           var cc = ctx.loopState("cc")
           if(cc.value != newcc){
              cc.setValue(newcc)
              ctx.loopState("neighbors").foreach(n => out.collect(VComponent(n,newcc)))
           }
        },
    (ctx:LoopContext,out:Collector[VComponent]) =>
        {  //FINALIZE FUNCTION
           out.collect(VComponent(ctx.key, ctx.loopState("cc").value))
        });
```

Based on out-of-order event time model

Scalable and Reliable Data Stream Processing
P Carbone
KTH Royal Institute of Technology

- Deadlocks - Cyclic Fl
- Time Progress?

Semantics + Arbitrary Nesting

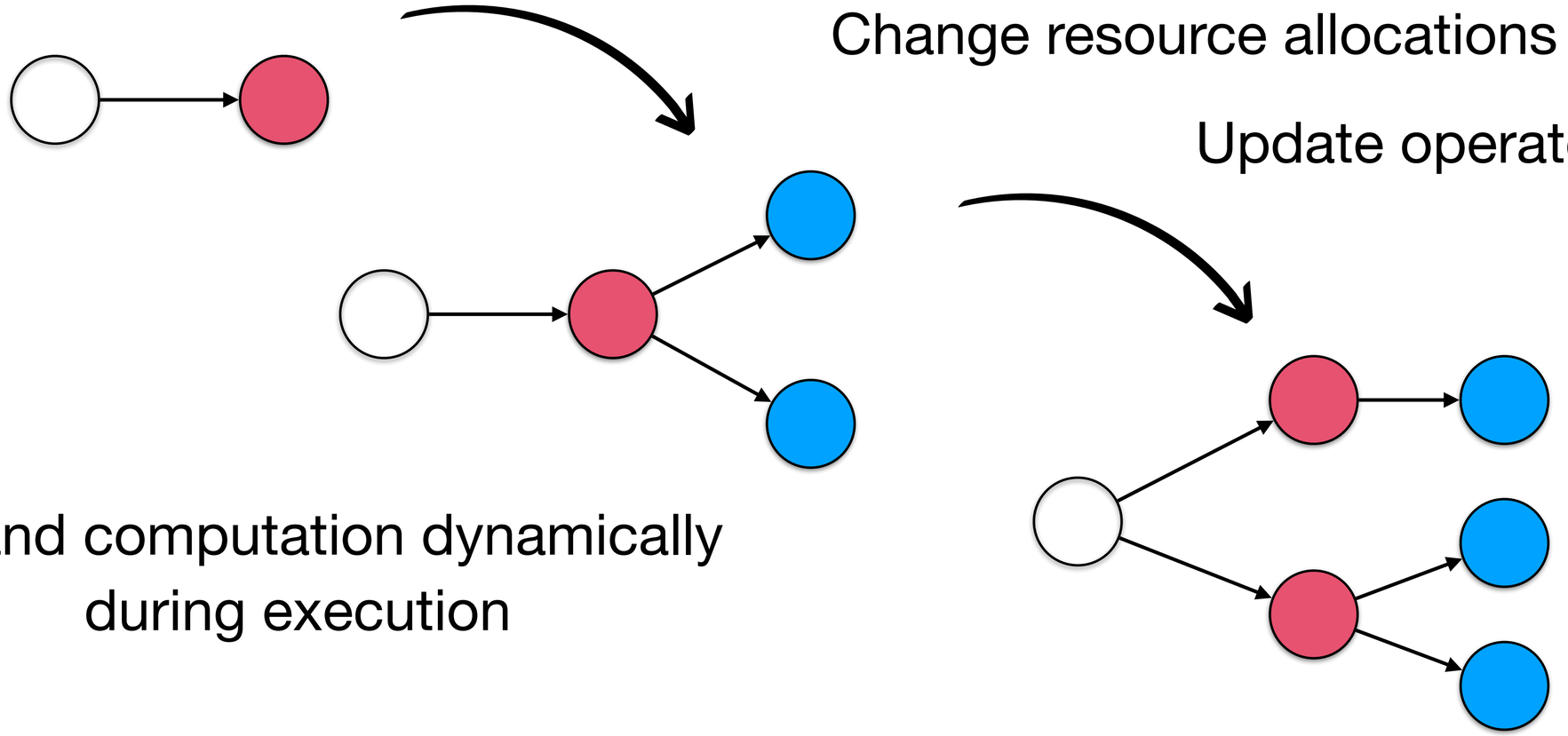
Elasticity, adaptivity, dynamic plans

Cloud Apps, ML

Spawn instances on demand

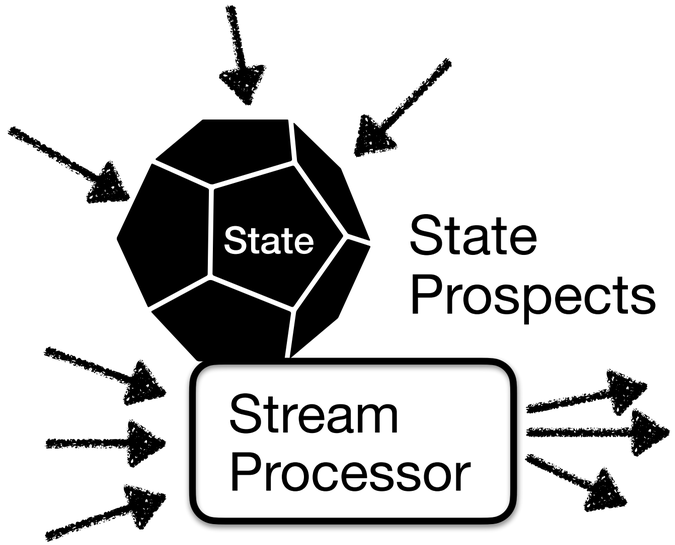
Change resource allocations

Update operator logic



Expand computation dynamically during execution

Taking Stateful Processing Further



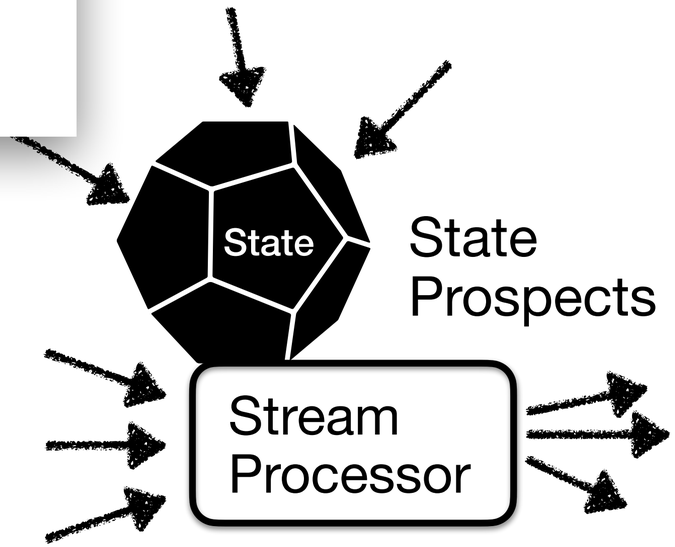
Taking Stateful Processing Further

Workload-Aware State Backends

Databases : Optimal plans using different indexes (e.g., B+Trees)

Stream Processors : Use LSM for every operator.

In support of workload-aware streaming state management
V Kalavri, J Liagouris
12th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 20)



Taking Stateful Processing Further

Workload-Aware State Backends

Databases : Optimal plans using different indexes (e.g., B+Trees)

Stream Processors : Use LSM for every operator.

In support of workload-aware streaming state management
V Kalavri, J Liagouris
12th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 20)

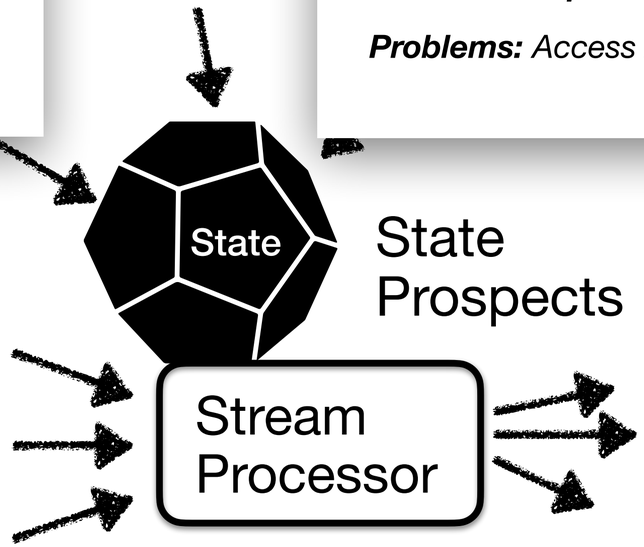
Queryable State

Currently : State is Black Box to the User

Queryable:

- ***Materialized Views***
- ***Exploratory Data Analysis***

Problems: *Access Isolation? Indexes?*



Taking Stateful Processing Further

Workload-Aware State Backends

Databases : Optimal plans using different indexes (e.g., B+Trees)

Stream Processors : Use LSM for every operator.

In support of workload-aware streaming state management
V Kalavri, J Liagouris
12th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 20)

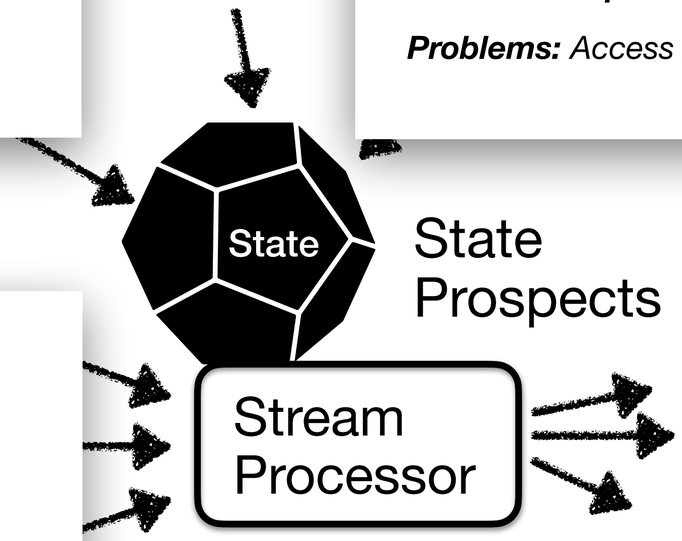
Queryable State

Currently : State is Black Box to the User

Queryable:

- ***Materialized Views***
- ***Exploratory Data Analysis***

Problems: *Access Isolation? Indexes?*



State Versioning

- **Schema Evolution**
- **Time-Travel Stream State Access**
- **MVCC on Stream State**

Taking Stateful Processing Further

Workload-Aware State Backends

Databases : Optimal plans using different indexes (e.g., B+Trees)

Stream Processors : Use LSM for every operator.

In support of workload-aware streaming state management
V Kalavri, J Liagouris
12th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 20)

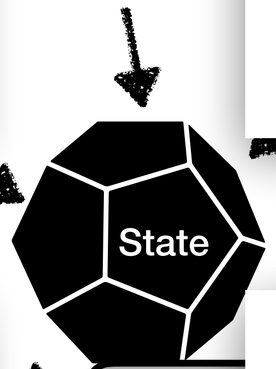
Queryable State

Currently : State is Black Box to the User

Queryable:

- ***Materialized Views***
- ***Exploratory Data Analysis***

Problems: *Access Isolation? Indexes?*



State
Prospects

Strea
Proce

State Versioning

- **Schema Evolution**
- **Time-Travel Stream State Access**
- **MVCC on Stream State**

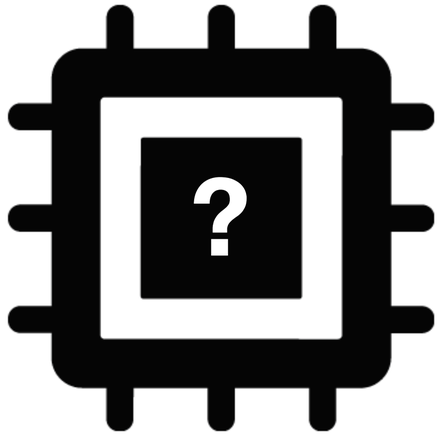
Inter-Query/Task State Sharing

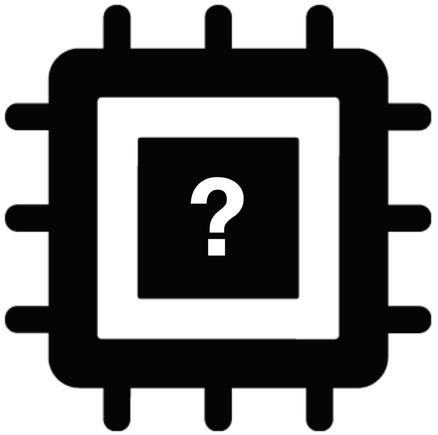
Currently : State restricted to dedicated task access

Shared Access:

- Mutable shared state for ML applications (Ray)
- Inter-Query Optimisation (Aurora, Shared Arrangements)

Shared Arrangements: practical inter-query sharing for streaming dataflows
McSherry F, Lattuada Malte Schwarzkopf A, Roscoe T, et. al.

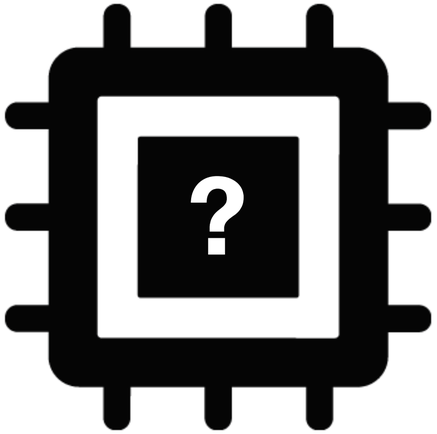




Hardware-conscious **stream processing**: A survey
S Zhang, F Zhang, Y Wu, B He, P Johns - ACM **SIGMOD Record**, 2020 - dl.acm.org

GPUs

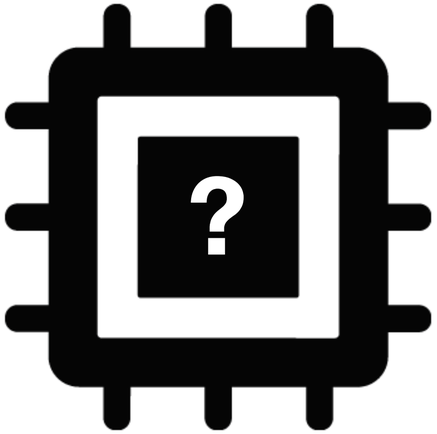
Saber: Window-based hybrid stream processing for heterogeneous architectures
A Koliousis, M Weidlich, R Castro Fernandez, AL Wolf, P Costa, ...
Proceedings of the 2016 International Conference on Management of Data, 555-569



Hardware-conscious **stream processing**: A survey
S Zhang, F Zhang, Y Wu, B He, P Johns - ACM **SIGMOD Record**, 2020 - dl.acm.org

GPUs

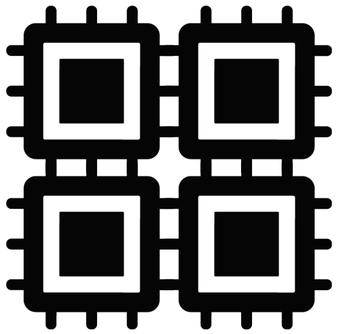
Saber: Window-based hybrid stream processing for heterogeneous architectures
A Koliousis, M Weidlich, R Castro Fernandez, AL Wolf, P Costa, ...
Proceedings of the 2016 International Conference on Management of Data, 555-569



Analyzing Efficient Stream Processing on Modern Hardware
S Zeuch, B Del Monte, J Karimov, C Lutz, M Renz, J Traub, S Breß, T Rabl, ...
Proceedings of the VLDB Endowment (PVLDB)

LightSaber: Efficient Window Aggregation on Multi-core Processors
G Theodorakis, A Koliousis, P Pietzuch, H Pirk
Proceedings of the 2020 ACM SIGMOD International Conference on Management of …

Hardware-conscious **stream processing**: A survey
S Zhang, F Zhang, Y Wu, B He, P Johns - ACM **SIGMOD Record**, 2020 - dl.acm.org

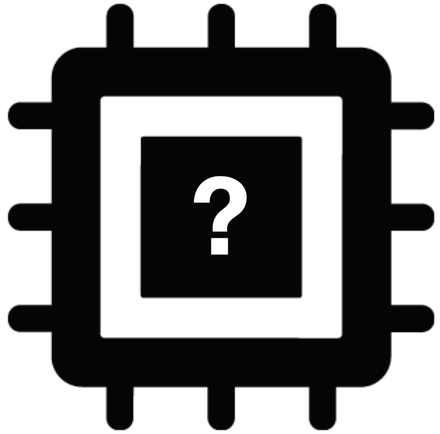


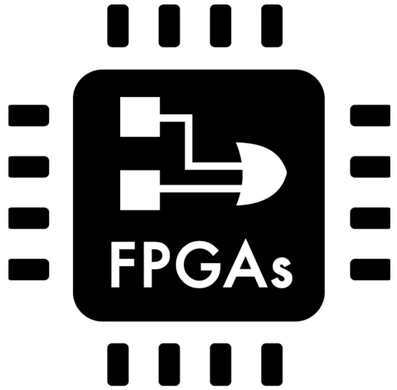
Multicore

Hardware acceleration

GPUs

Saber: Window-based hybrid stream processing for heterogeneous architectures
A Koliousis, M Weidlich, R Castro Fernandez, AL Wolf, P Costa, ...
Proceedings of the 2016 International Conference on Management of Data, 555-569

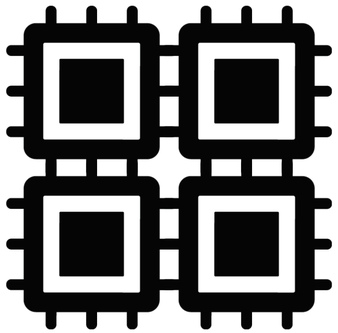




Fleet: A framework for massively parallel streaming on FPGAs
J Thomas, P Hanrahan, M Zaharia
Proceedings of the Twenty-Fifth International Conference on Architectural …

Analyzing Efficient Stream Processing on Modern Hardware
S Zeuch, B Del Monte, J Karimov, C Lutz, M Renz, J Traub, S Breß, T Rabl, ...
Proceedings of the VLDB Endowment (PVLDB)

LightSaber: Efficient Window Aggregation on Multi-core Processors
G Theodorakis, A Koliousis, P Pietzuch, H Pirk
Proceedings of the 2020 ACM SIGMOD International Conference on Management of …



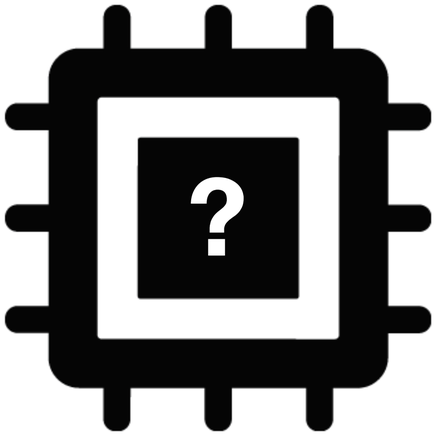
Multicore

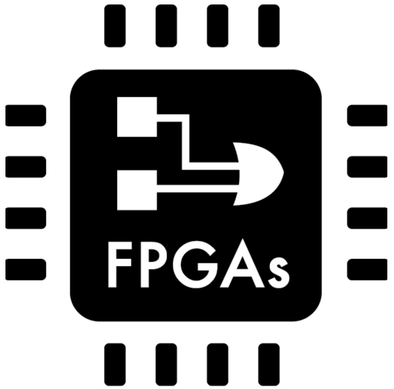
Hardware-conscious **stream processing**: A survey
S Zhang, F Zhang, Y Wu, B He, P Johns - ACM **SIGMOD Record**, 2020 - dl.acm.org

Hardware acceleration

GPUs

Saber: Window-based hybrid stream processing for heterogeneous architectures
A Koliousis, M Weidlich, R Castro Fernandez, AL Wolf, P Costa, ...
Proceedings of the 2016 International Conference on Management of Data, 555-569

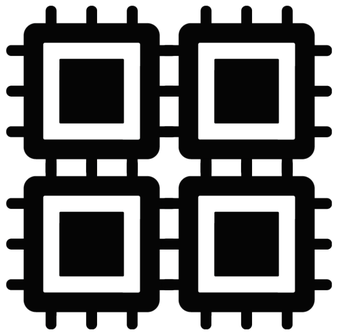




Fleet: A framework for massively parallel streaming on FPGAs
J Thomas, P Hanrahan, M Zaharia
Proceedings of the Twenty-Fifth International Conference on Architectural …

Analyzing Efficient Stream Processing on Modern Hardware
S Zeuch, B Del Monte, J Karimov, C Lutz, M Renz, J Traub, S Breß, T Rabl, ...
Proceedings of the VLDB Endowment (PVLDB)

LightSaber: Efficient Window Aggregation on Multi-core Processors
G Theodorakis, A Koliousis, P Pietzuch, H Pirk
Proceedings of the 2020 ACM SIGMOD International Conference on Management of …



Multicore

Hardware-conscious **stream processing**: A survey
S Zhang, F Zhang, Y Wu, B He, P Johns - ACM **SIGMOD Record**, 2020 - dl.acm.org

NVMe?

RDMA?

Requirements overview

	Cloud Apps	ML	Graphs
Programming models	✓	✓	✓
Transactions	✓		
Advanced state backends	✓	✓	✓
Loops and cycles	✓	✓	✓
Elasticity & reconfiguration	✓		
Dynamic topologies	✓	✓	
Shared mutable state		✓	✓
Queryable state	✓	✓	
State versioning	✓	✓	
Hardware acceleration		✓	✓

Stream processing @SIGMOD'20

Research 28: Stream Processing, Thursday 10:30 AM – 12:00 PM

- Ahmed S. Abdelhamid, Ahmed R. Mahmood, Anas Daghistani, and Walid G. Aref. **Prompt: Dynamic Data-Partitioning for Distributed Micro-batch Stream Processing Systems**.
- Bonaventura Del Monte, Steffen Zeuch, Tilmann Rabl, and Volker Markl. Rhino: **Efficient Management of Very Large Distributed State for Stream Processing Engines**.
- Philipp M. Grulich, Breß Sebastian, Steffen Zeuch, Jonas Traub, Janis von Bleichert, Zongxiong Chen, Tilmann Rabl, and Volker Markl. **Grizzly: Efficient Stream Processing Through Adaptive Query Compilation**.
- Georgios Theodorakis, Alexandros Koliousis, Peter Pietzuch, and Holger Pirk. **LightSaber: Efficient Window Aggregation on Multi-core Processors**.
- Amirhesam Shahvarani and Hans-Arno Jacobsen. **Parallel Index-based Stream Join on a Multicore CPU**.

Beyond Analytics

The Evolution of Stream Processing Systems

Prospects

Paris Carbone, Marios Fragkoulis, Vasiliki Kalavri, Asterios Katsifodimos

Slides: **streaming-research.github.io/Tutorial-SIGMOD-2020**