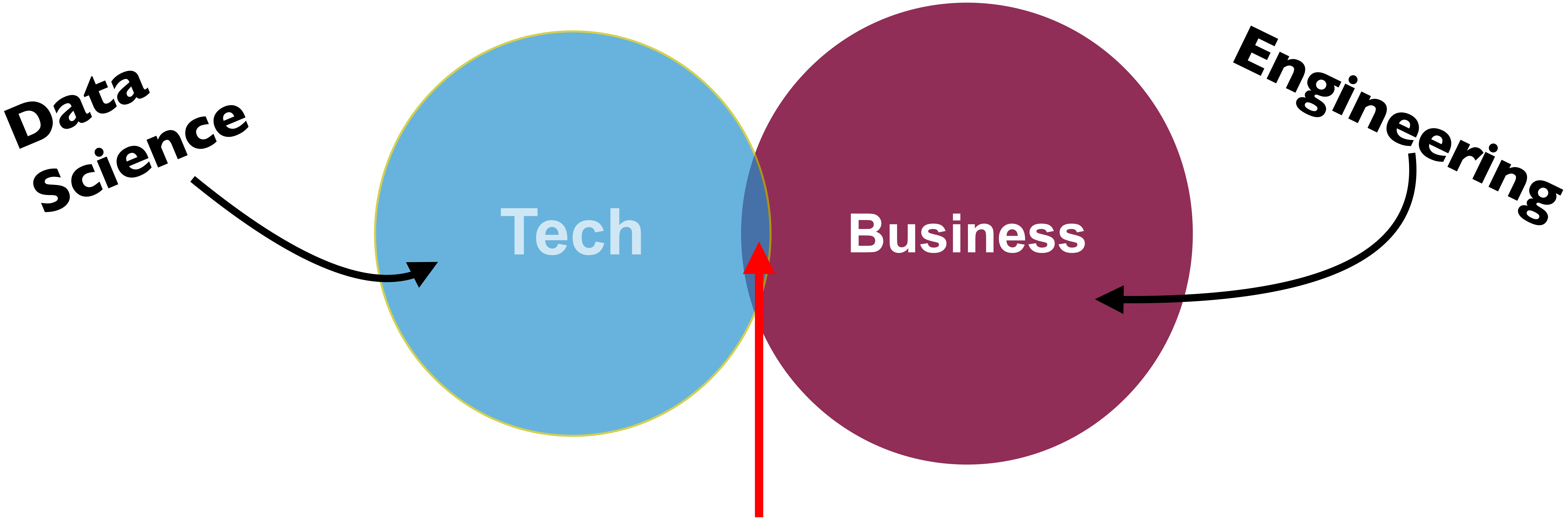
Continuous Intelligence

Through Computation Sharing With Arcon

Paris Carbone
Senior Researcher @ RISE
Committer @ Apache Flink
<paris.carbone@ri.se>

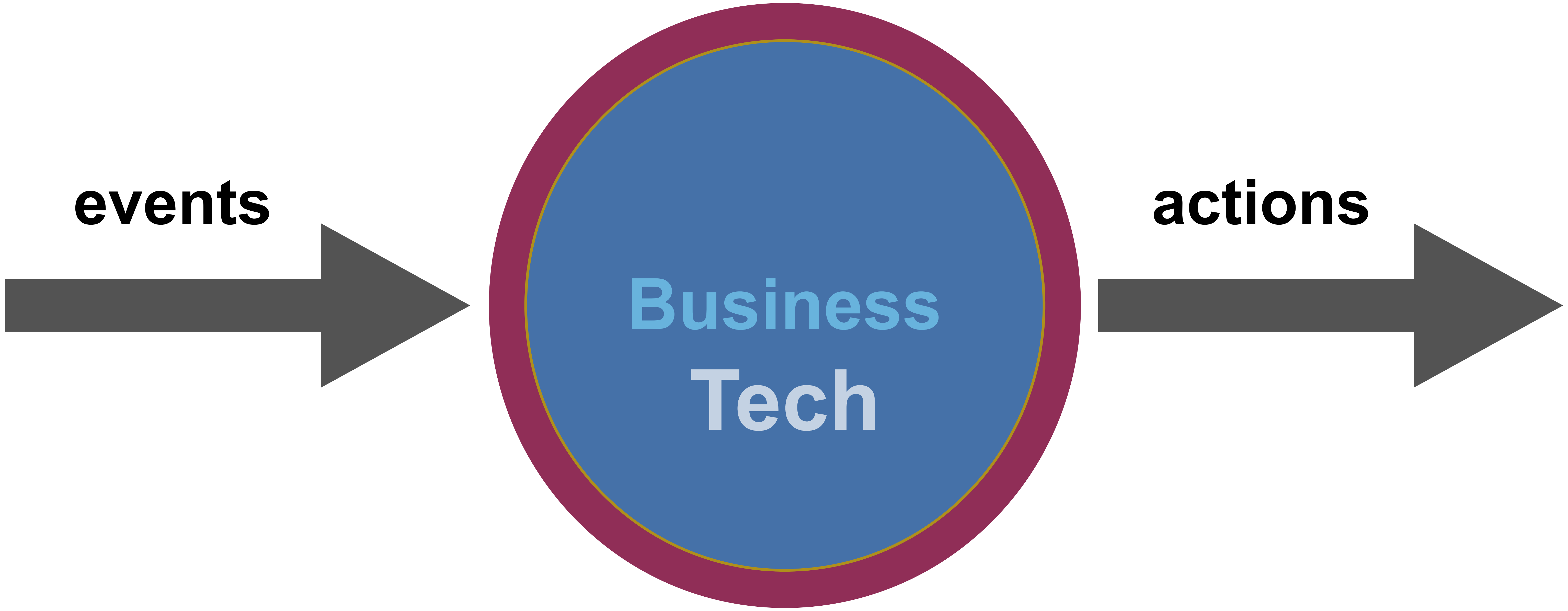
Castor
Software Days



A **Lot** is going on in Tech (Deep Learning, Scalable Processing etc.)
Little contribution to critical real-time decision making

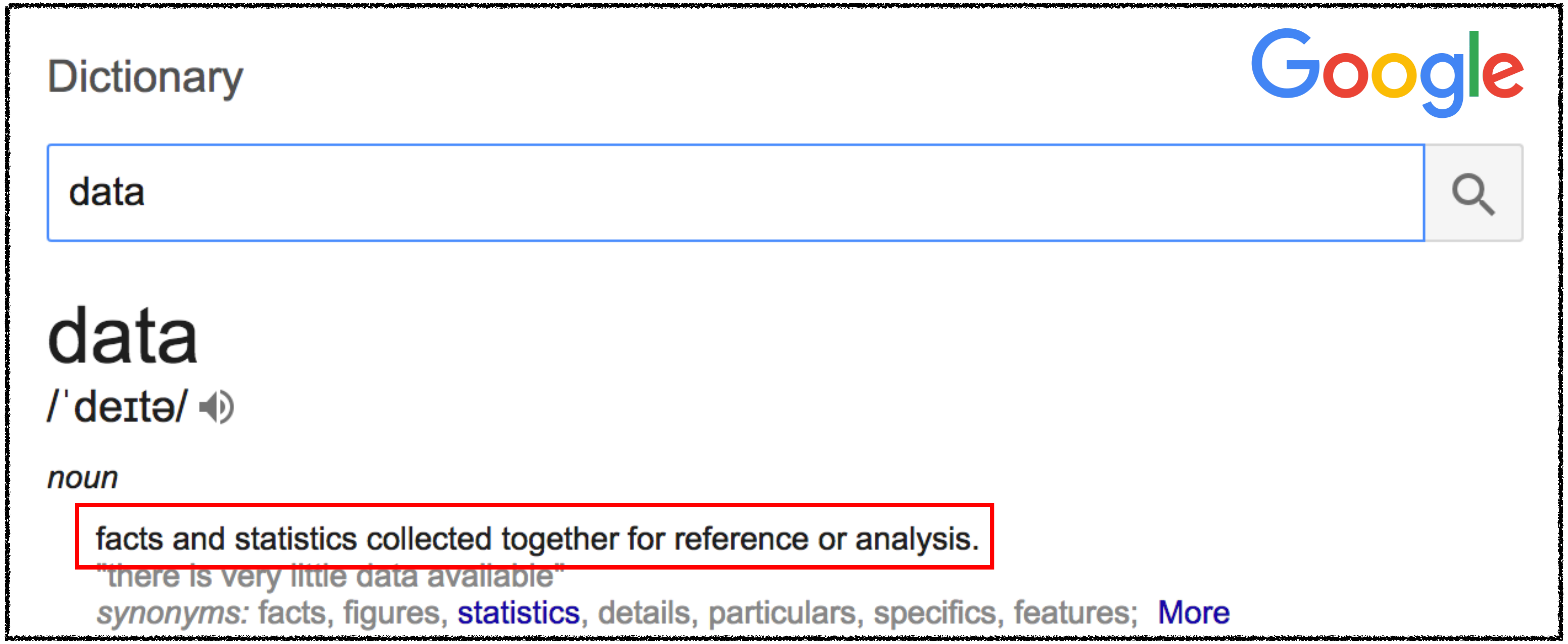
Continuous Intelligence

A design pattern in which ***real-time analytics*** *are integrated within a business operation, processing* ***current and historical data*** *to prescribe* ***actions in response to events****.*



https://www.gartner.com/en/newsroom/press-releases/2019-02-18-gartner-identifies-top-10-data-and-analytics-technolo

What we think of data



Dictionary

data

data

/ˈdeɪtə/

noun

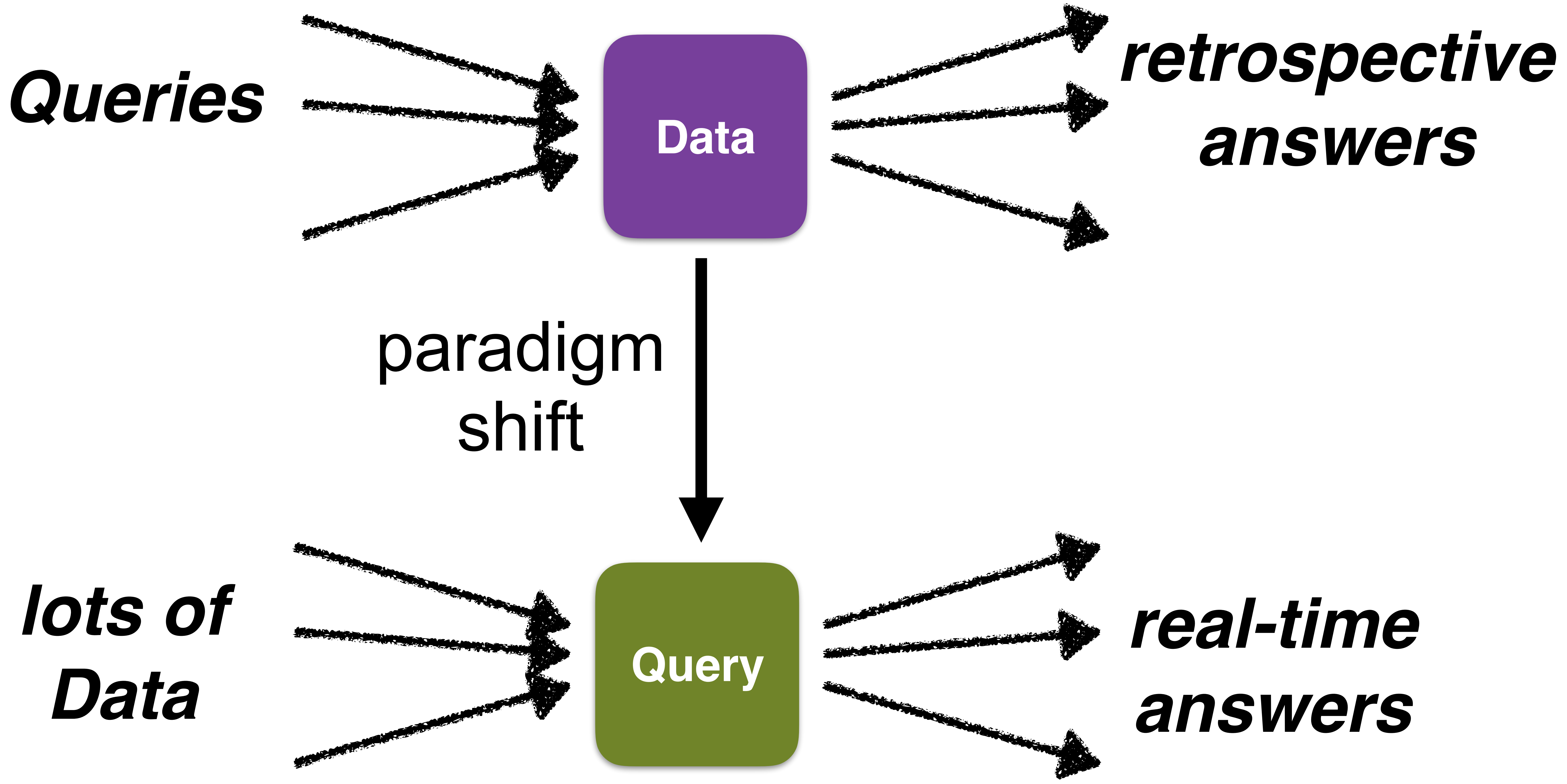
facts and statistics collected together for reference or analysis.

"there is very little data available"

synonyms: facts, figures, statistics, details, particulars, specifics, features; More

vs actual data….

The Paradigm Shift Some Missed

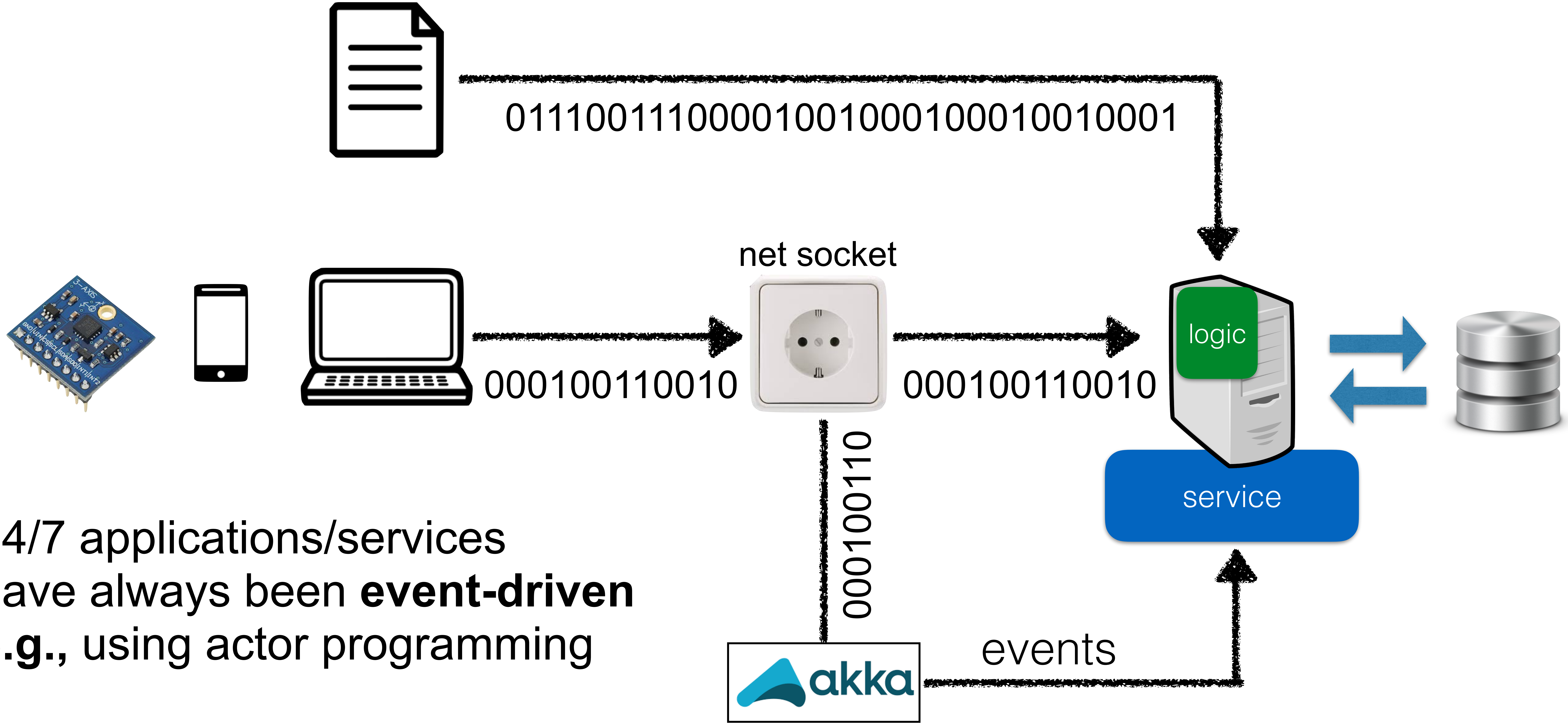


The Stream Analytics Stack

High Level Models	Stream SQL, CEP…
Compute	Flink, Beam, Kafka-Streams, Apex, Storm
Storage	Kafka, Pub/Sub, Kinesis, Pravega…

- **Data Stream Processing** as a 24/7 execution paradigm

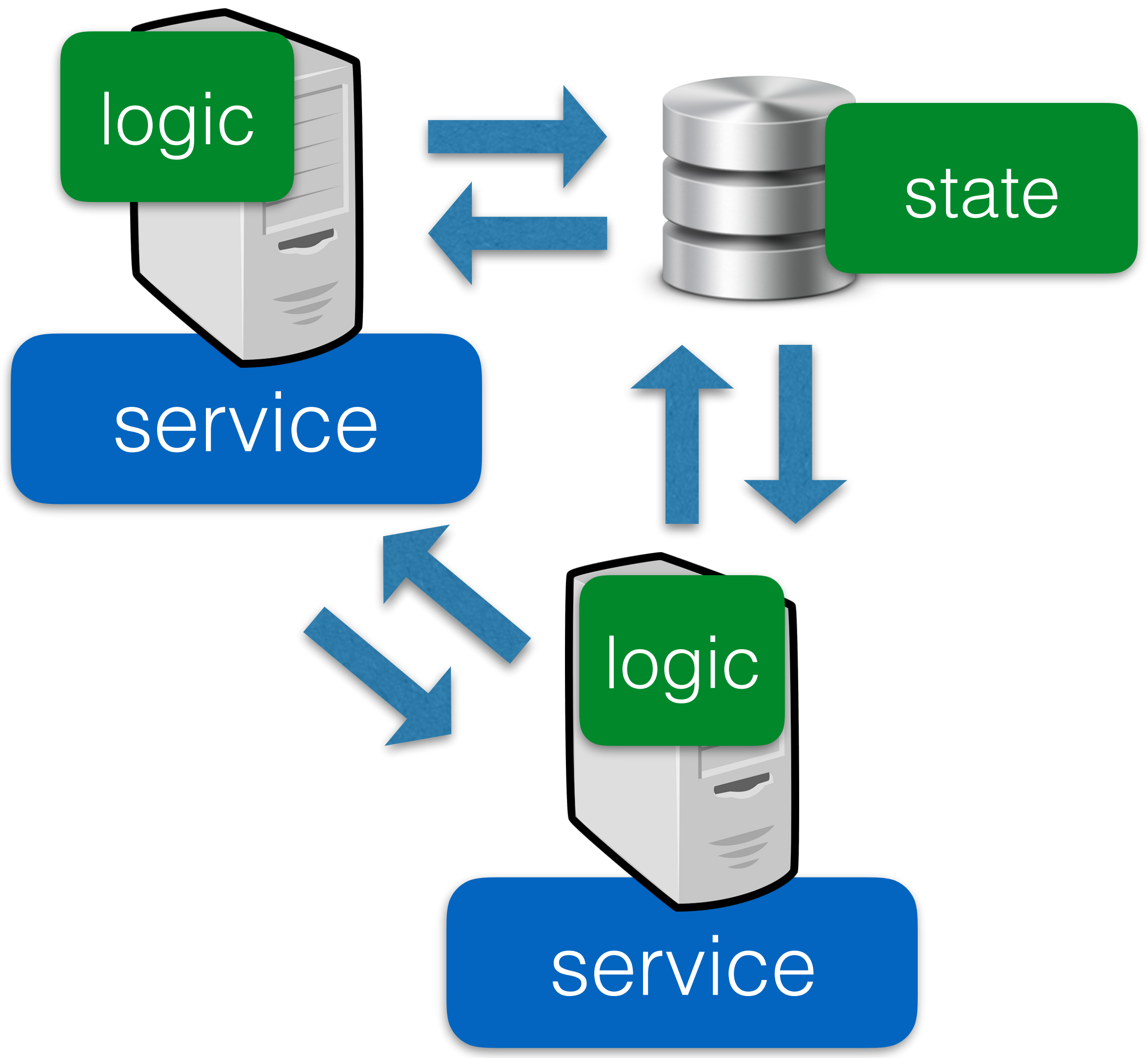
Similar Technologies



24/7 applications/services have always been **event-driven** **e.g.,** using actor programming

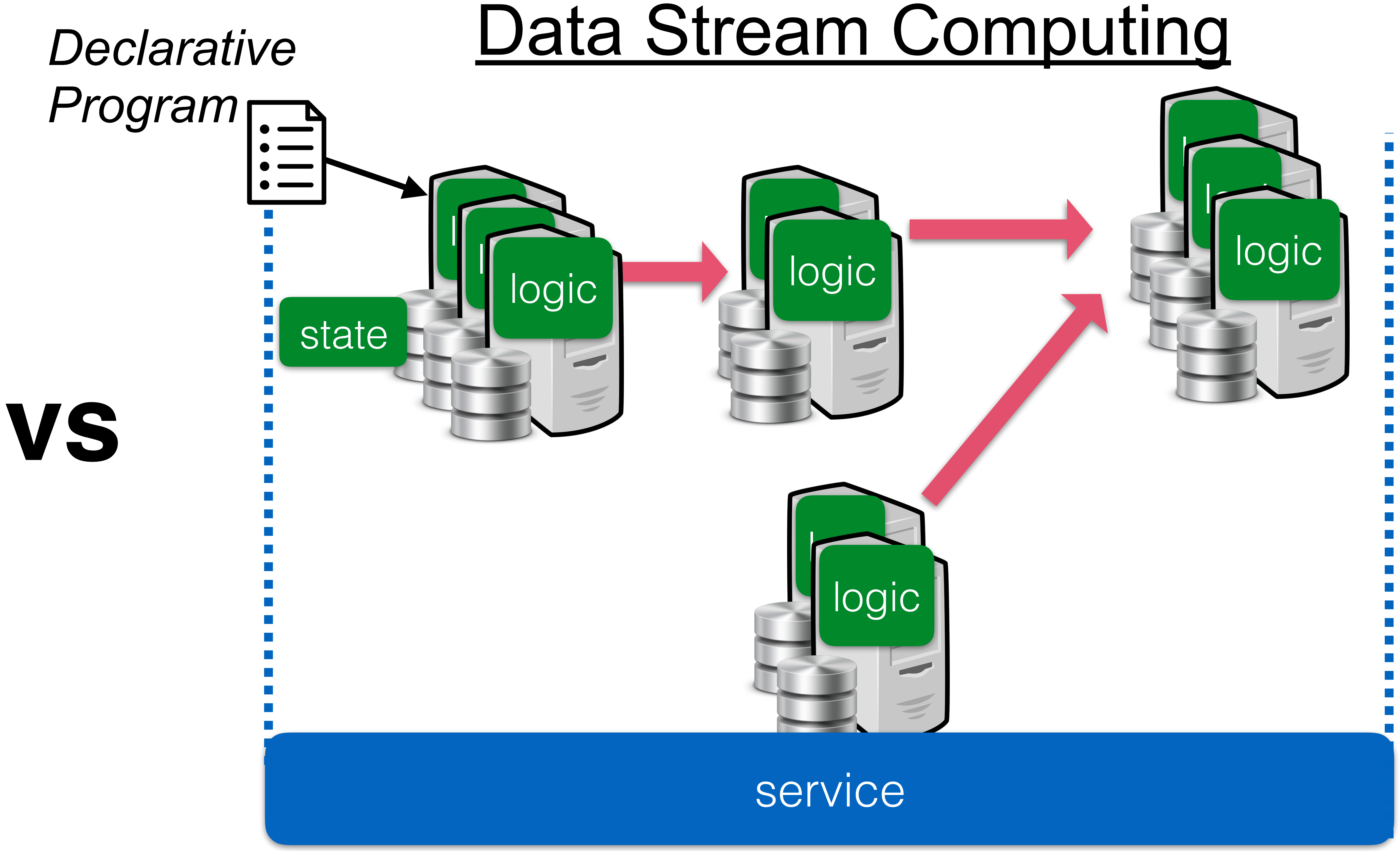
Actors vs Streams

Actor Programming



vs

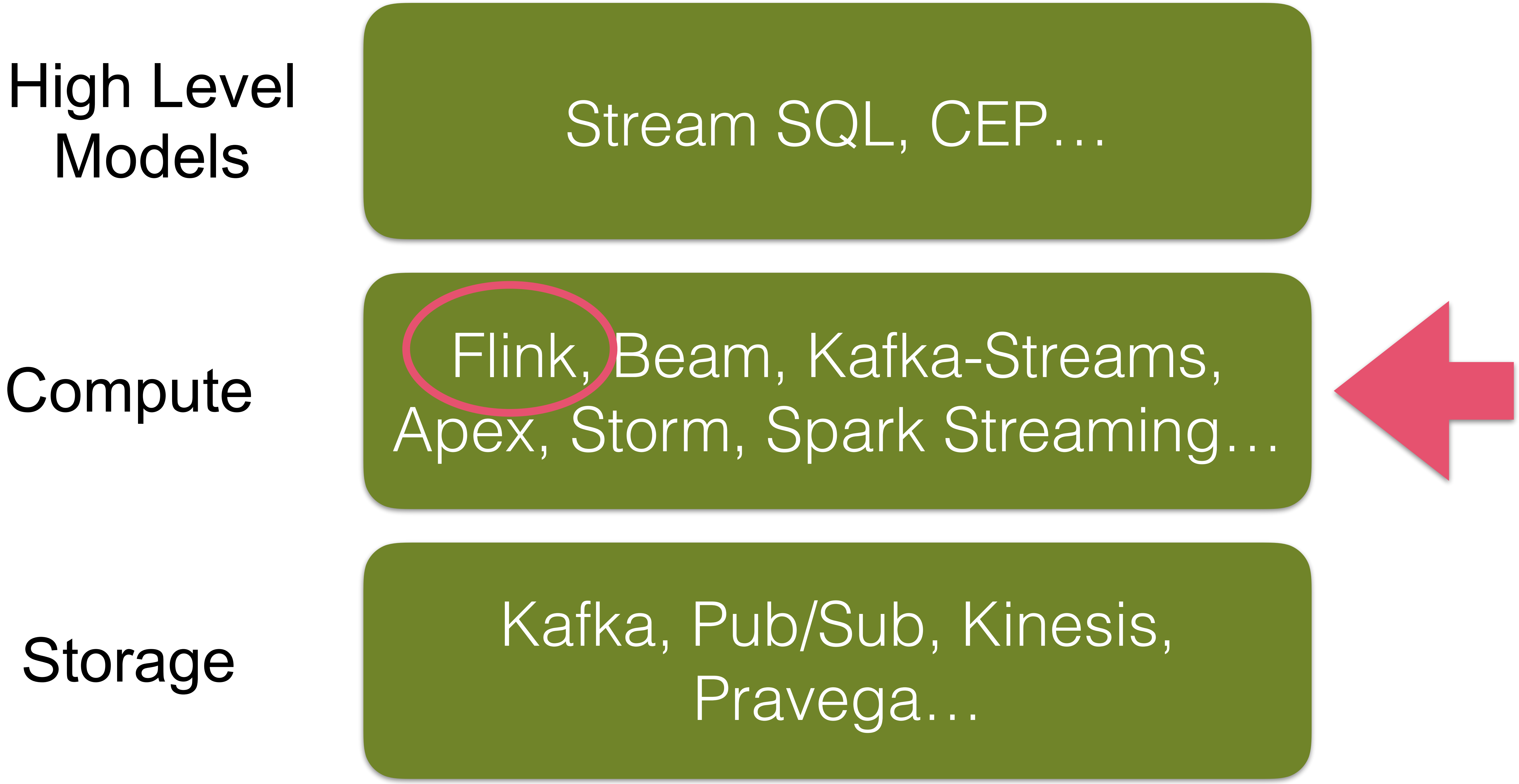
Data Stream Computing



- Low-Level Event-Based Programming
- Manual/External State
- Not Robust: Manual Fault Tolerance
- Not flexible scaling

- Declarative Programming
- State Managed by the system
- Robust: Built-in Fault Tolerance
- Scalable Deployments

The Real-Time Analytics Stack



High Level Models

Stream SQL, CEP…

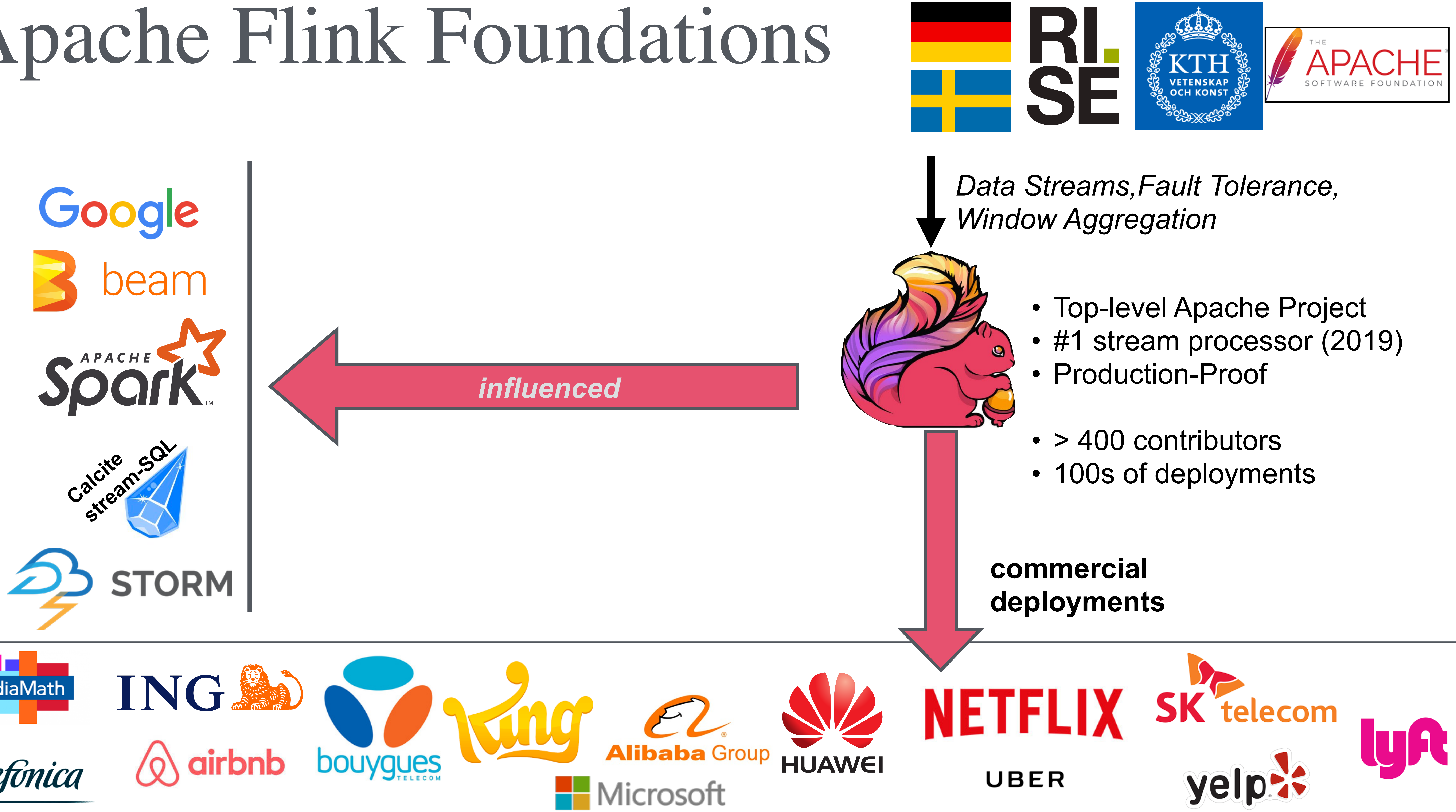
Compute

Flink, Beam, Kafka-Streams, Apex, Storm, Spark Streaming…

Storage

Kafka, Pub/Sub, Kinesis, Pravega…

Apache Flink Foundations



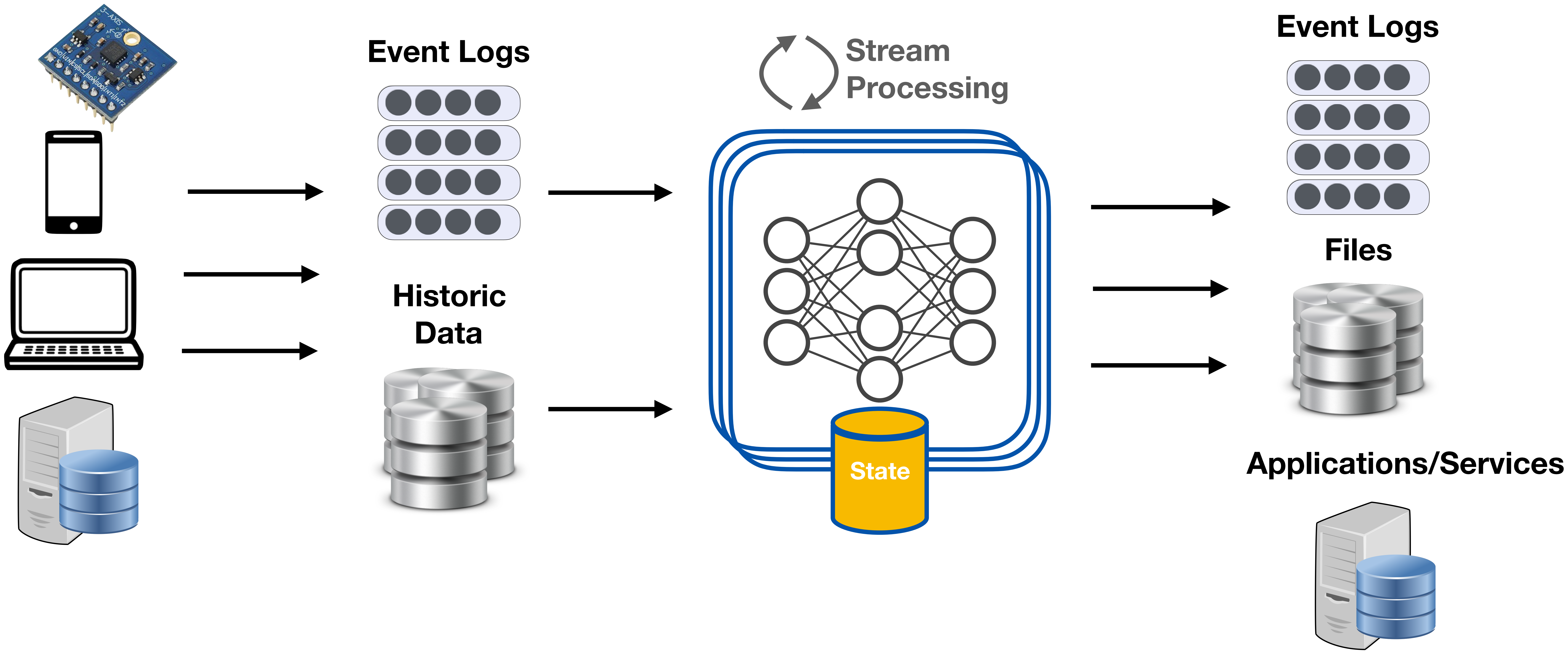
Data Streams,Fault Tolerance, Window Aggregation

- Top-level Apache Project
- #1 stream processor (2019)
- Production-Proof

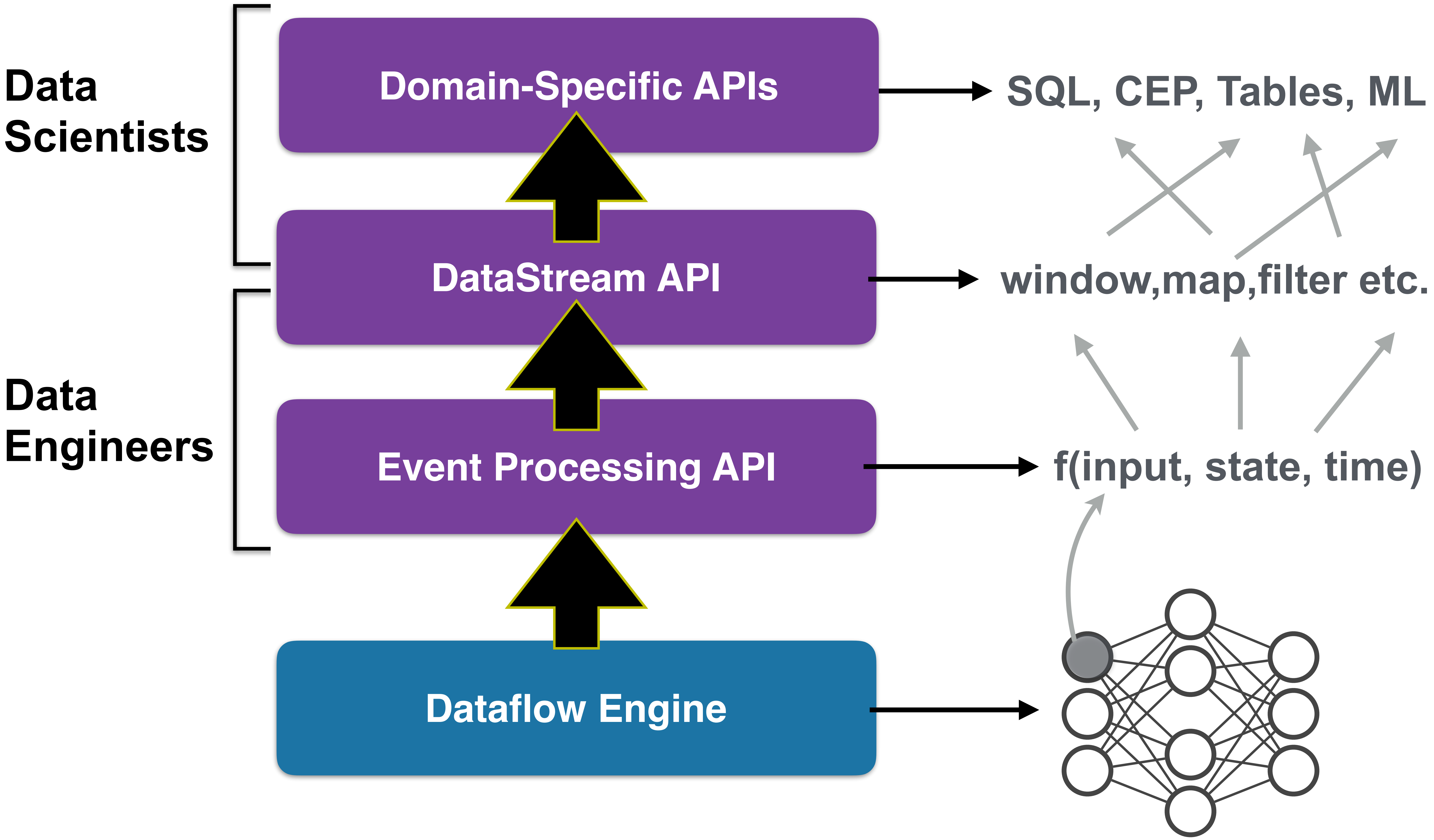
- > 400 contributors
- 100s of deployments

commercial deployments

Structure of a 24/7 Stream Application



Programming Abstractions in Flink



Automates

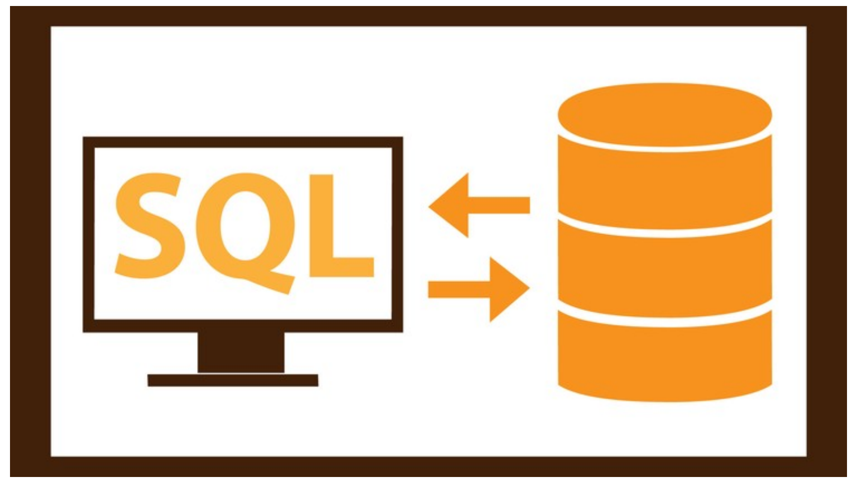
- Fully Declarative Programming
- Event Patterns, Relations etc.

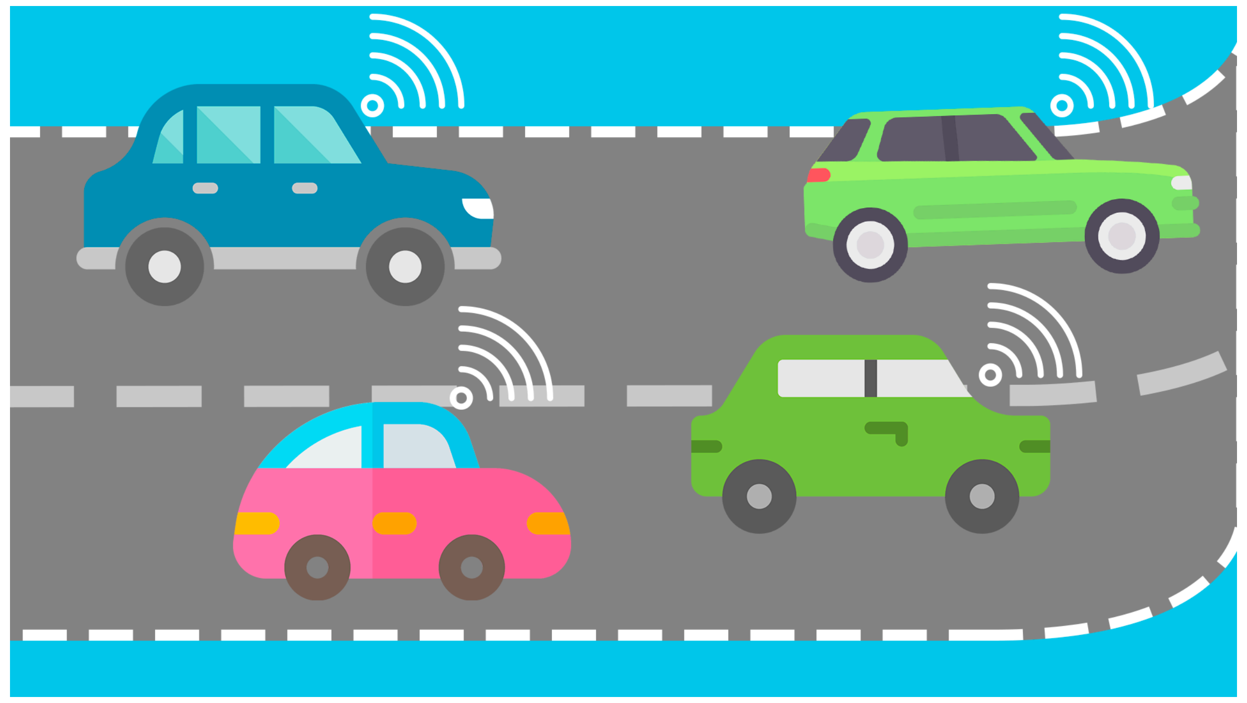
- Higher-Order Streaming Functions
- Event Windowing (sessions, time etc.)

- Dynamic program state
- Operations on out-of-order streams

- Fault Tolerance
- Scalability
- Monitoring/IO Management

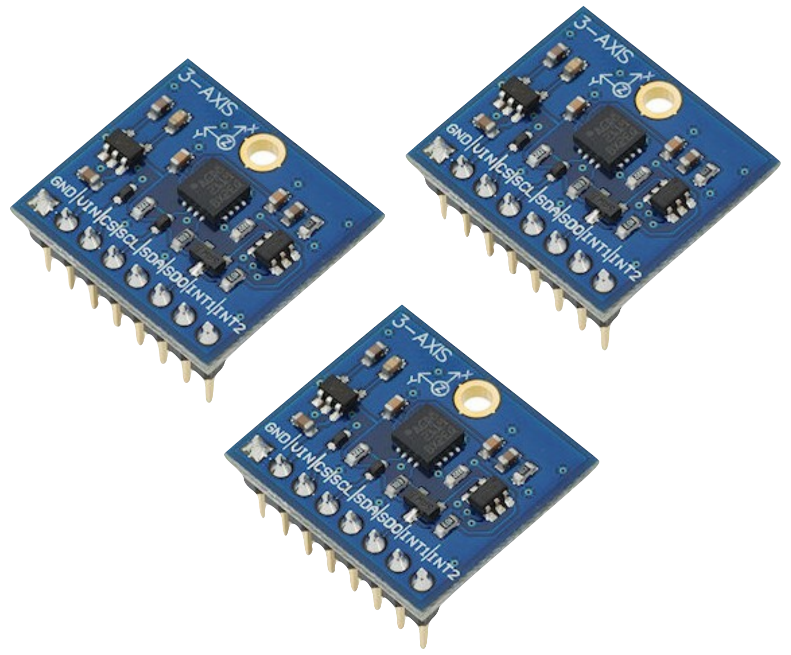
Declarative Streaming Examples





```
SELECT
  HOUR(r.rideTime) AS hourOfDay,
  AVG(f.tip) AS avgTip
FROM
  Rides r,
  Fares f
WHERE
  r.rideId = f.rideId AND
  NOT r.isStart AND
  f.payTime BETWEEN r.rideTime - INTERVAL '5' MINUTE AND r.rideTime
GROUP BY
  HOUR(r.rideTime);
```

Average Tip per Hour with Stream SQL



```
val completedRides = Pattern
 .begin[TaxiRide]("start").where(_.isStart)
 .next(“end").where(!_.isStart)

CEP.pattern[TaxiRide](allRides,
completedRides.within(Time.minutes(120)))
```

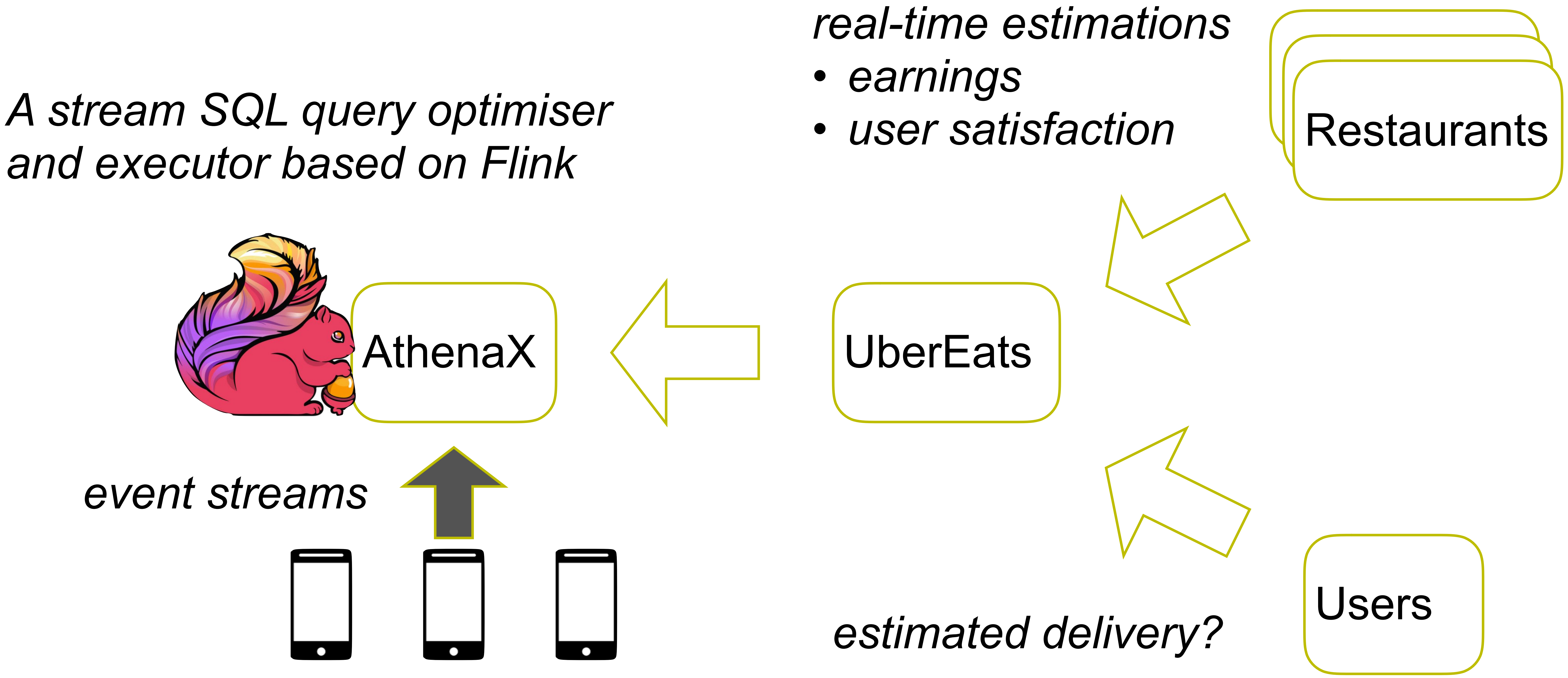
Completed Taxi Rides within 120min with Complex Event Processing

Case Study

Car Sharing

Source: FLINK FORWARD https://www.flink-forward.org/

AthenaX - An Online Warehousing Platform (2017)



A stream SQL query optimiser and executor based on Flink

real-time estimations

- *earnings*
- *user satisfaction*

Restaurants

AthenaX

UberEats

event streams

Users

estimated delivery?

AthenaX was released and open sourced by Uber Technologies. It is capable of scaling across hundreds of machines and processing hundreds of billions of real-time events daily.

https://eng.uber.com/athenax/ https://github.com/uber/AthenaX

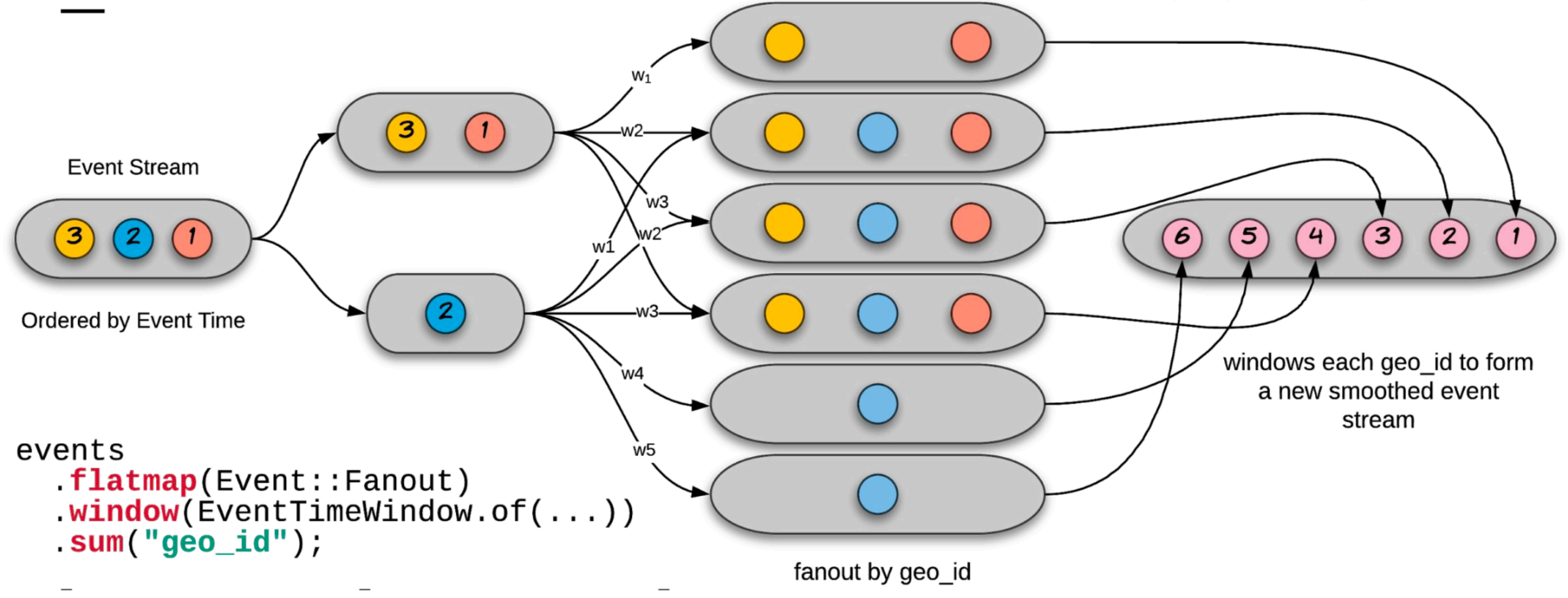
UBER

Marketplace - Dynamic Ride Pricing with Apache Flink (2018)

Input Streams

- *supply*
- *demand (taxi orders)*
- *Trips*
- *Traffic*

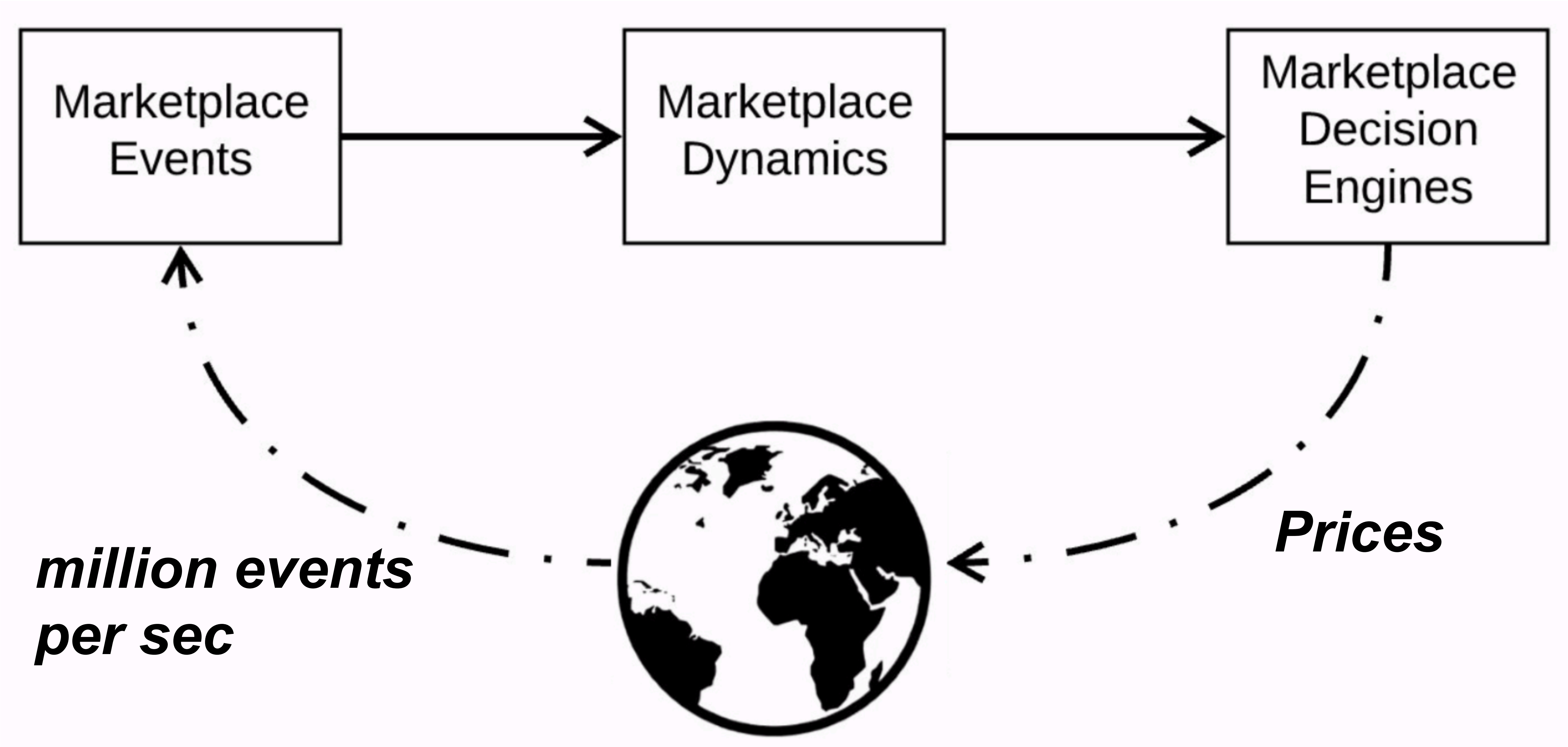
Geo-Sensitive Time-based Aggregations

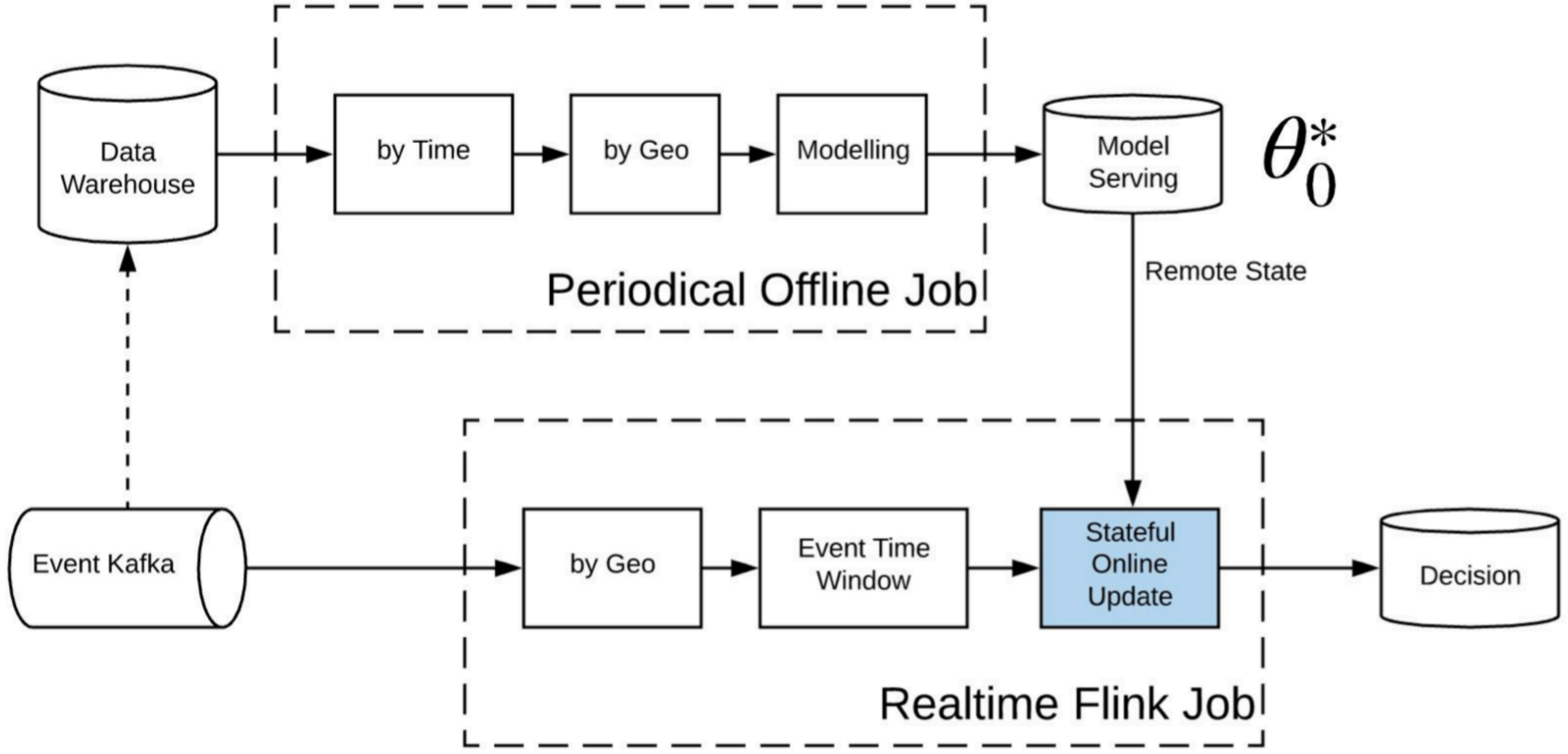


```
events
  .flatmap(Event::Fanout)
  .window(EventTimeWindow.of(...))
  .sum("geo_id");
```

Output Decisions

- *Pricing*
- *Dispatch*
- *Promotions*
- *Driver Positioning*


Marketplace Events
Marketplace Decision Dynamics
Marketplace Decision Engines
million events per sec
Prices



Compute Location-Sensitive Trends in Rider Demand and Driver Availability

https://marketplace.uber.com/

Flink Forward 2018

Dynamic Pricing - A Data Stream-Powered Standard

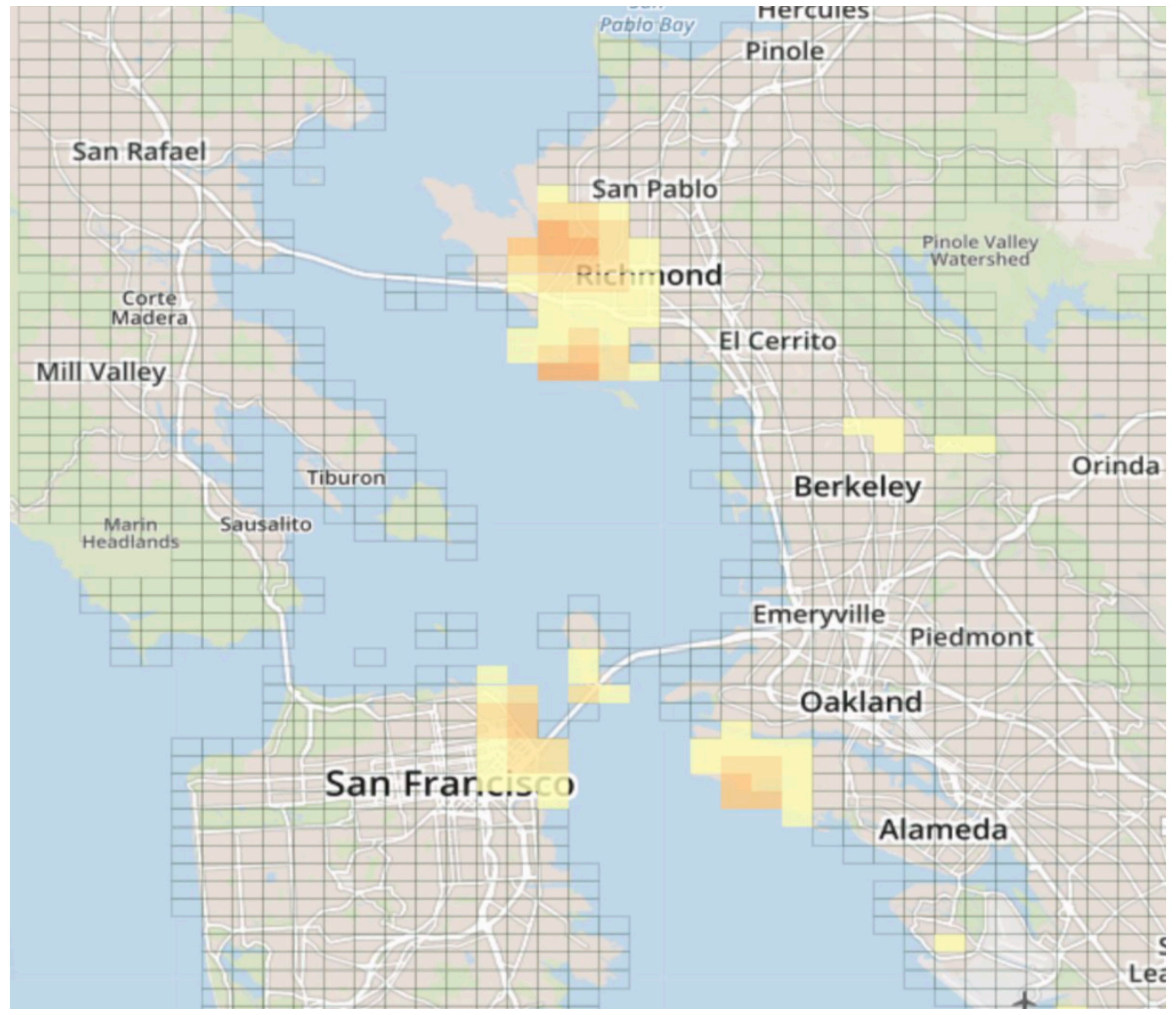
- *Dynamic Pricing*
  - *more profitable*
  - *best deals for users*

- *competition had to adapt*

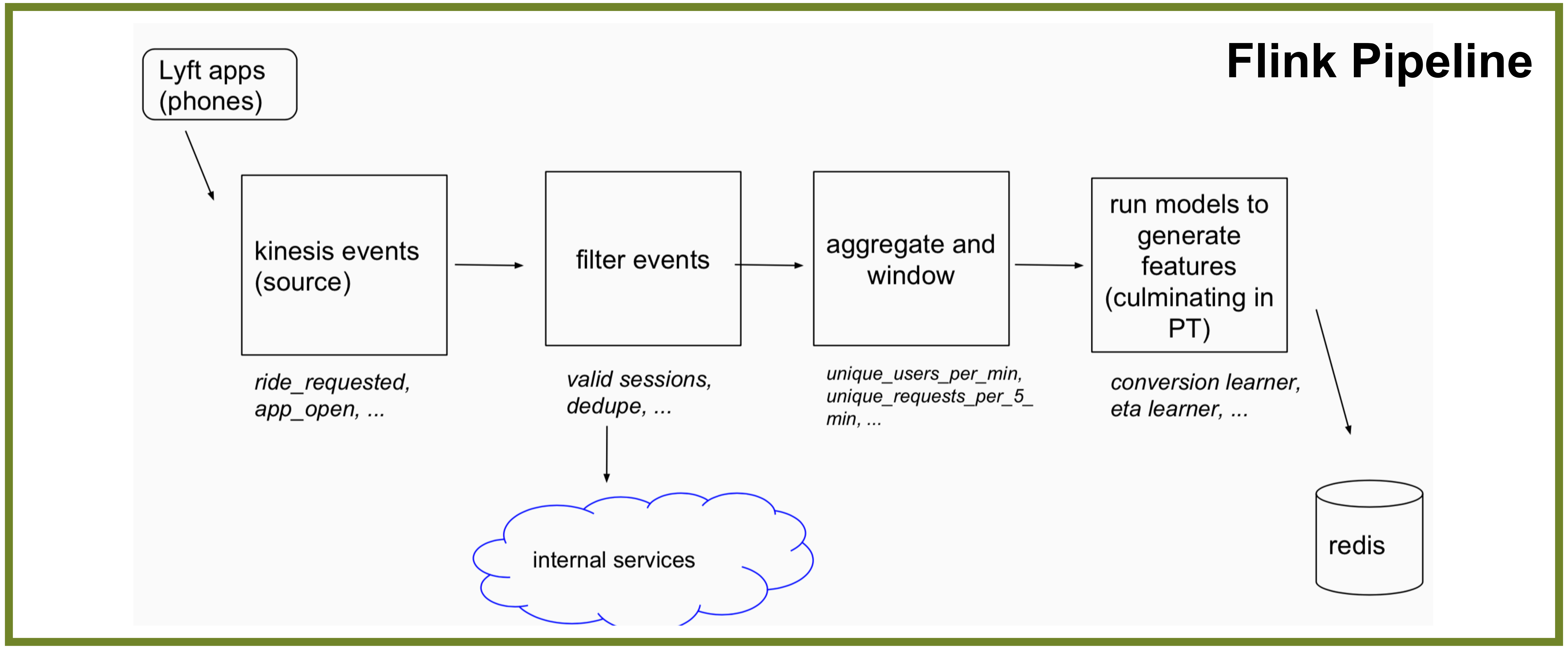
Dynamic Pricing (2019)

too many
San Francisco

too few
San Francisco



- PrimeTime Real-Time Service
- Price Multiplier per geog. cell
- 3M Geohashes/min



Flink Pipeline

Lyft apps (phones)

kinesis events (source)

ride_requested, app_open, ...

filter events

valid sessions, dedupe, ...

aggregate and window

unique_users_per_min, unique_requests_per_5_min, ...

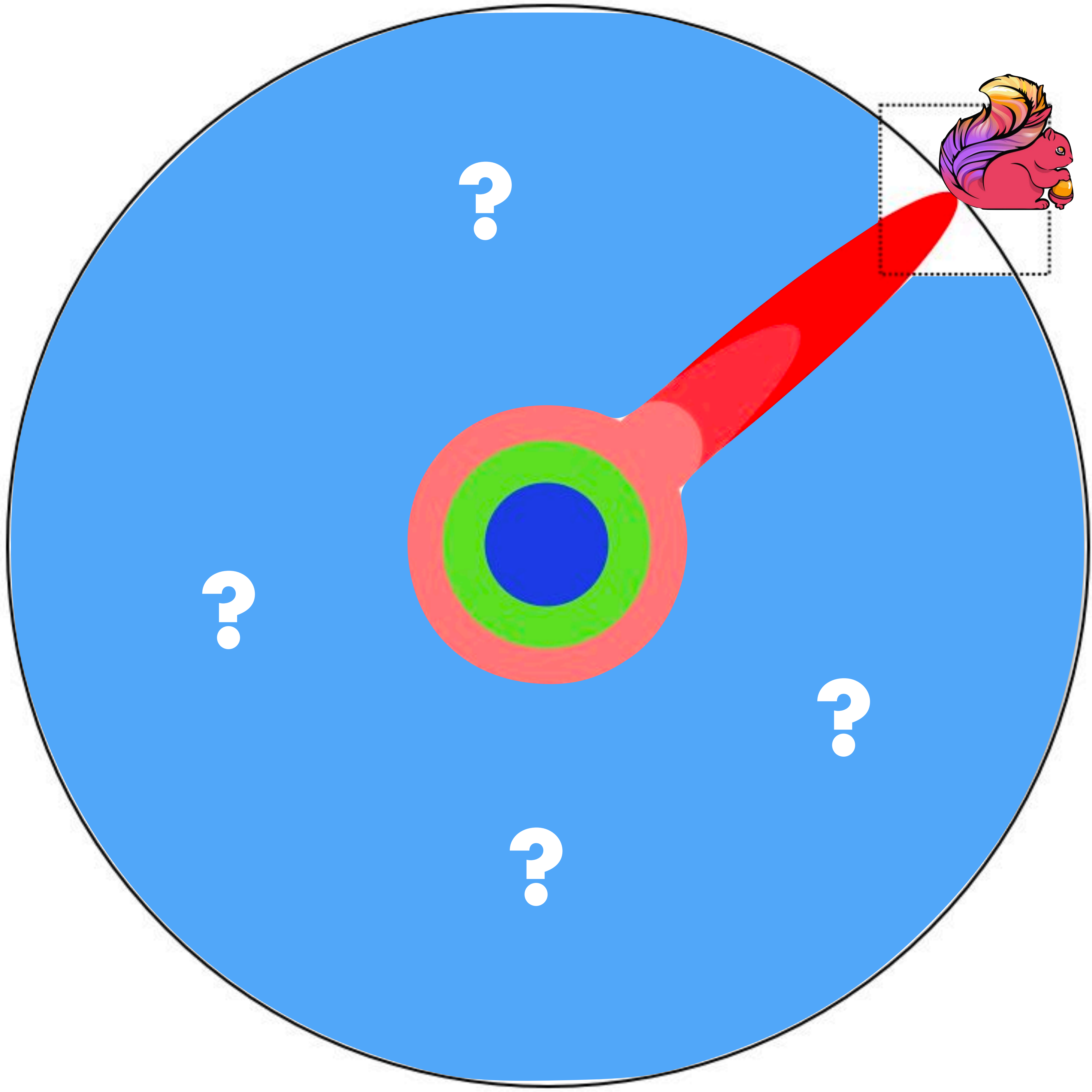
run models to generate features (culminating in PT)

conversion learner, eta learner, ...

internal services

redis

The Bigger Picture



Data Processing

Data Streams

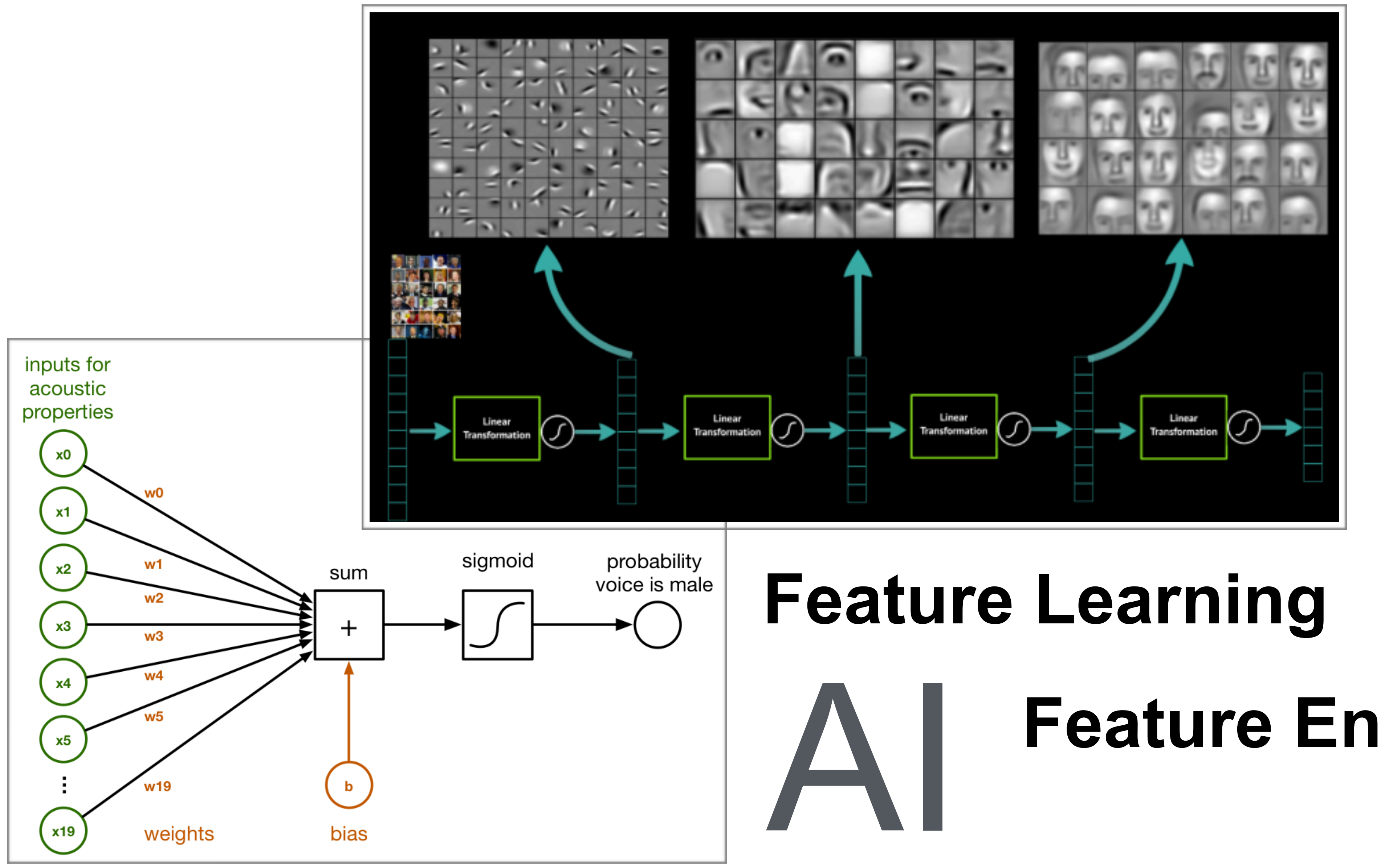
- scalable, fault tolerant analytics
- event-based business logic
- out-of-order computation
- dynamic relational tables (SQL)
- event pattern-matching (CEP)

but what about deeper analytics…

- tensors
- graph algorithms
- deep learning
- feature learning
- reinforcement learning
- ….

Data Pipelines Today

- Many Frameworks/Frontends for different needs
- (ML Training & Serving, SQL, Streams, Tensors, Graphs)



Feature Learning

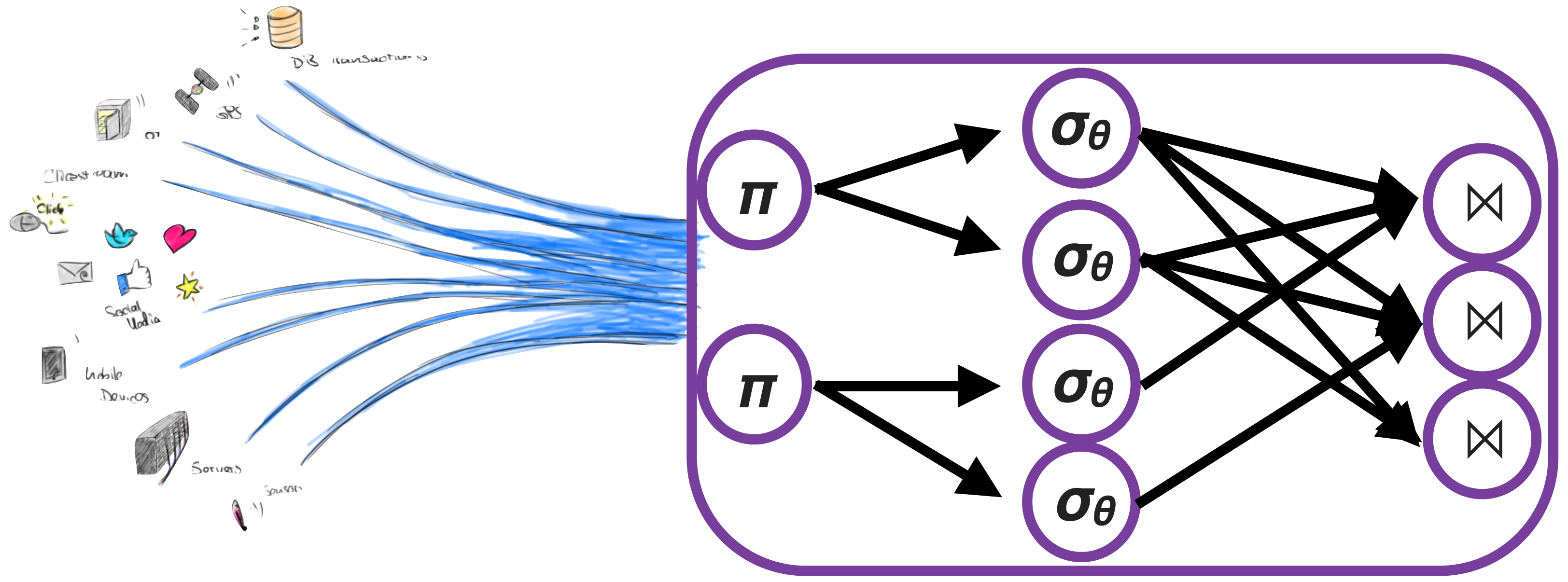
AI

Feature Engineering

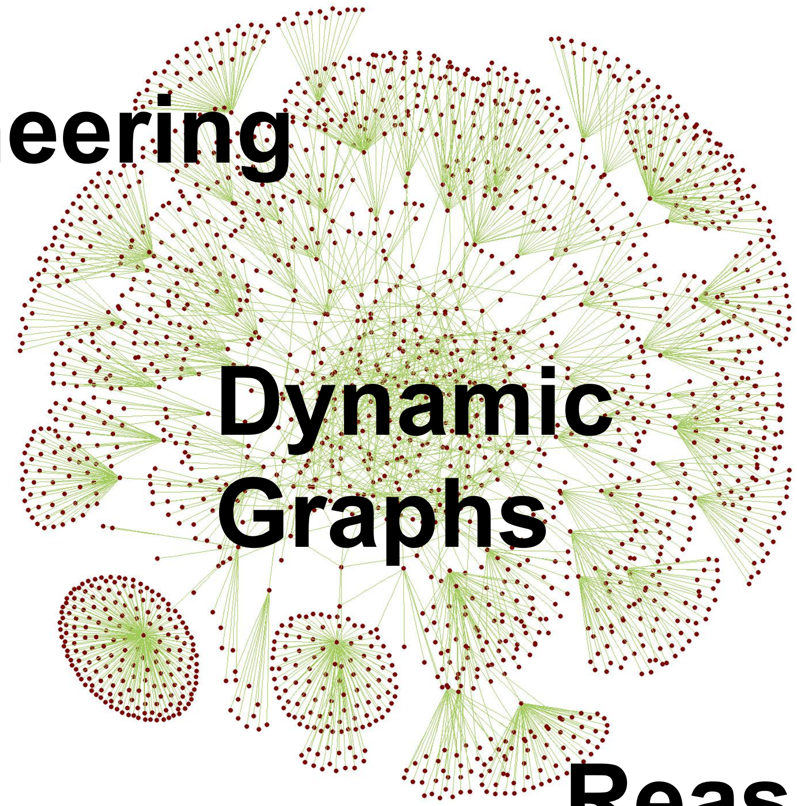
Tensor Programming

RL

Simulation tasks



Streams



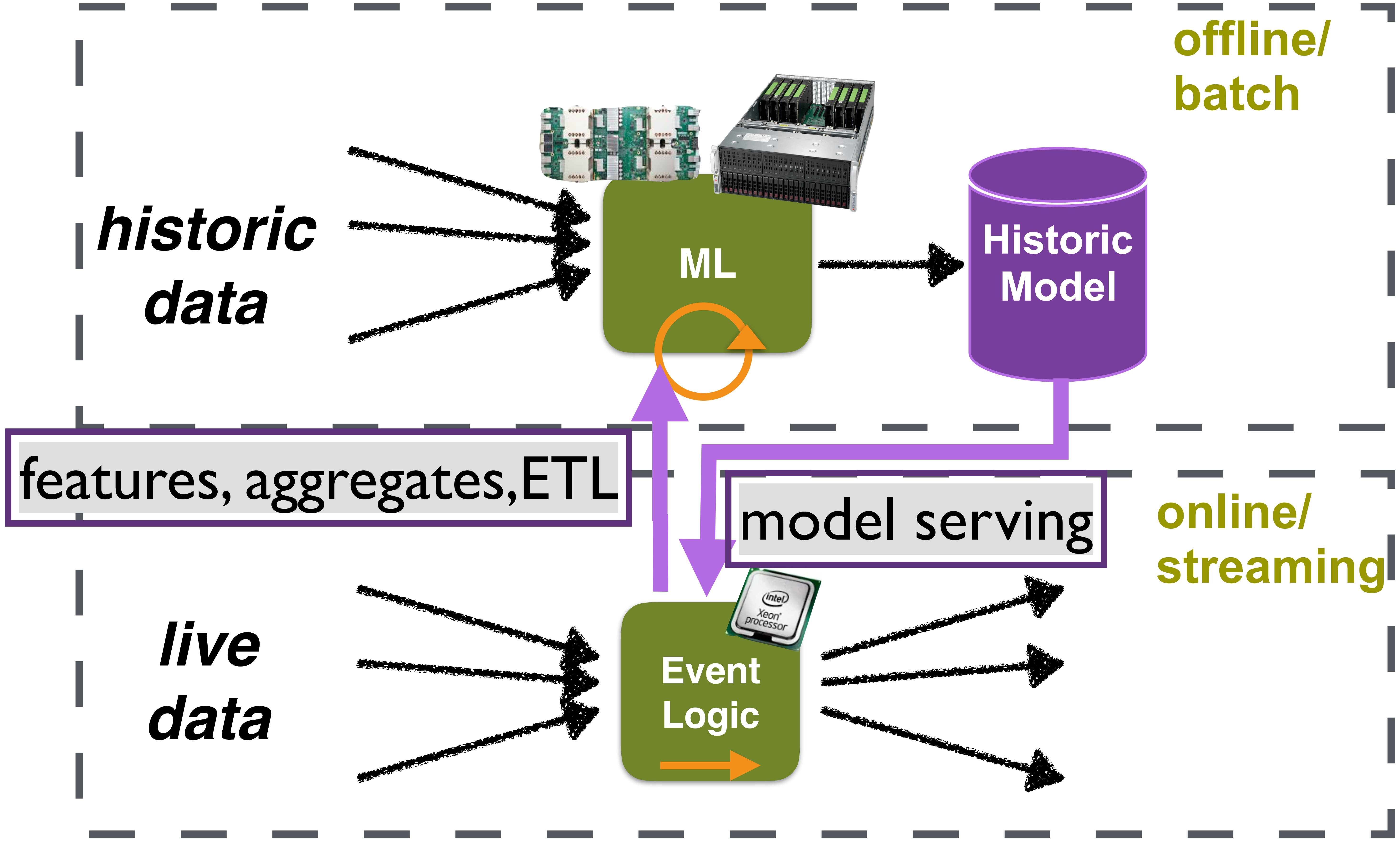
Dynamic Graphs

ML

Model Serving

Reasoning

Fundamental Problems



Framework/Library Silos

↓

Fragmented Codebases/Runtimes

↓

Unshared Hardware

↓

Over-materialization of results

↓

Ridiculously Unoptimised Programs

↓

No continuous intelligence

Next paradigm shift?

historic data
ML
Historic Model
offline/ batch
features, aggregates,ETL
model serving
online/ streaming
live data
Event Logic
Live Model
?
critical decision making

Secret Sauce?

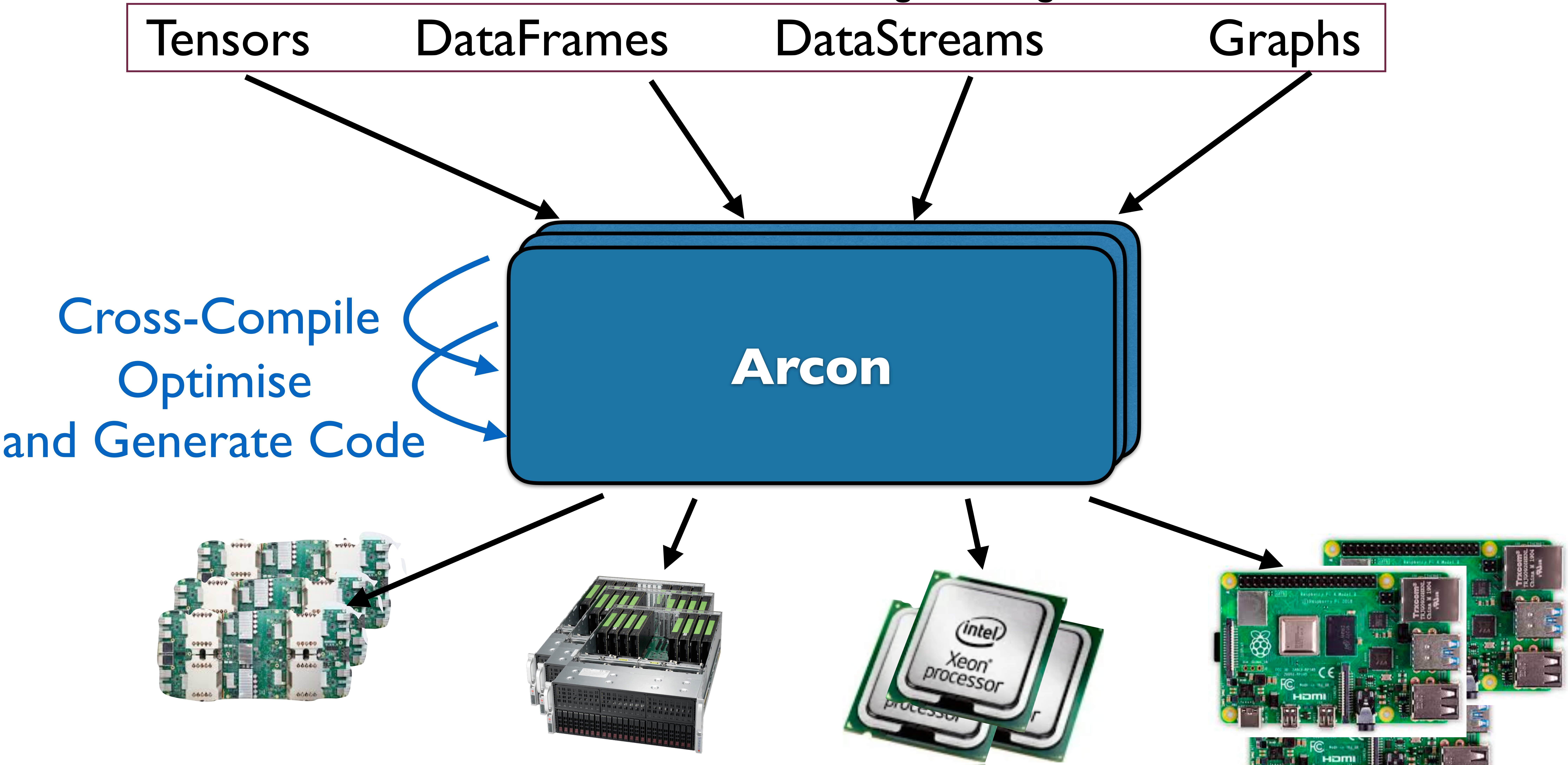
“A revolutionary technology
*that does **NOT** require you to throw **tons** of data*
to your problem to be able to solve it”

The Compiler

- Instead, compilers can understand **instructions…**
- **explained** by **humans** in a **high-level declarative language**
- and then **optimise** them
- and translate to primitive machines to **execute** them **reliably**

The Arcon Vision

Unified Declarative Programming



Tensors

DataFrames

DataStreams

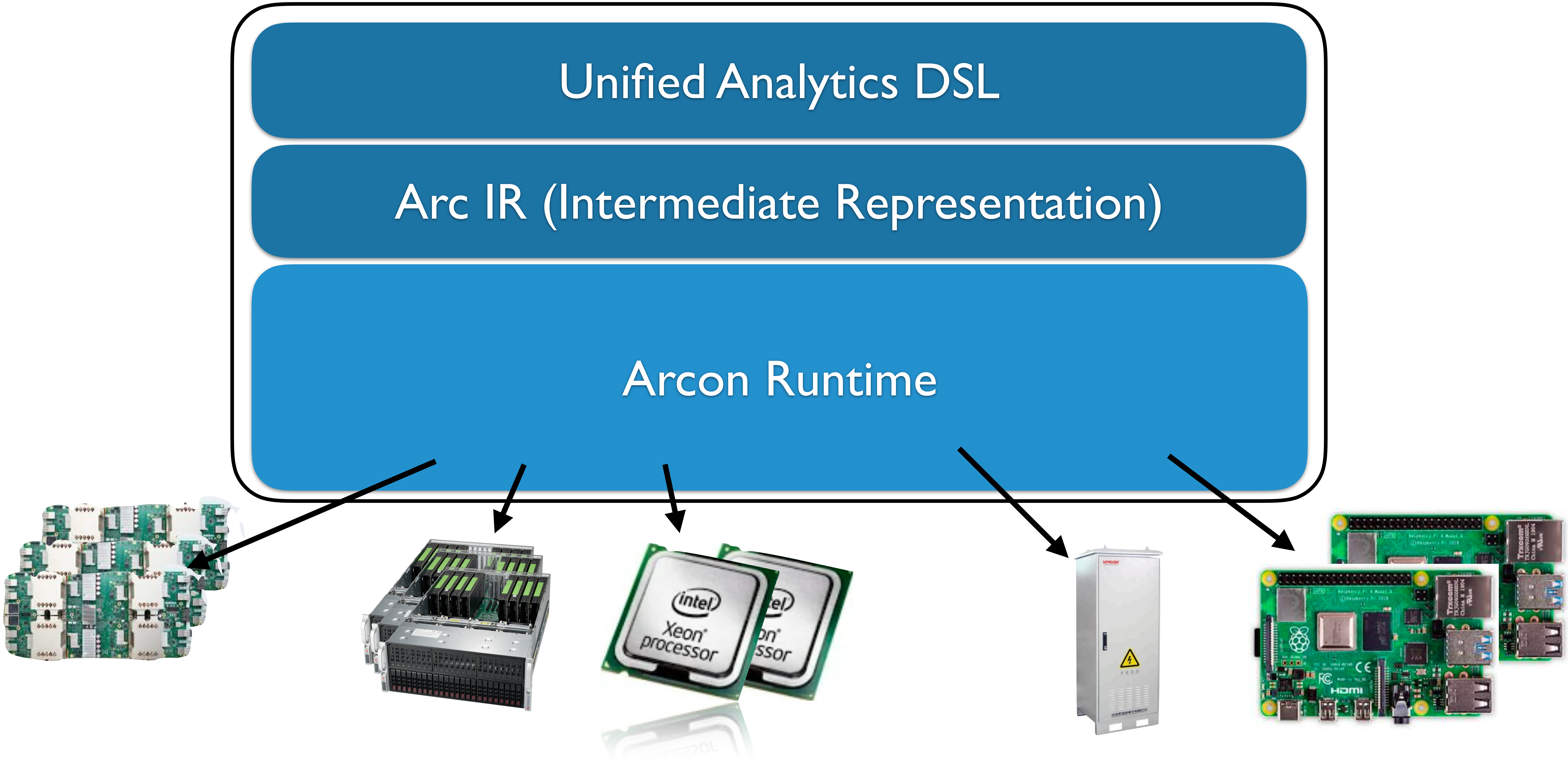
Graphs

Cross-Compile
Optimise
and Generate Code

Arcon

Shared Native Execution

The Arcon Architecture



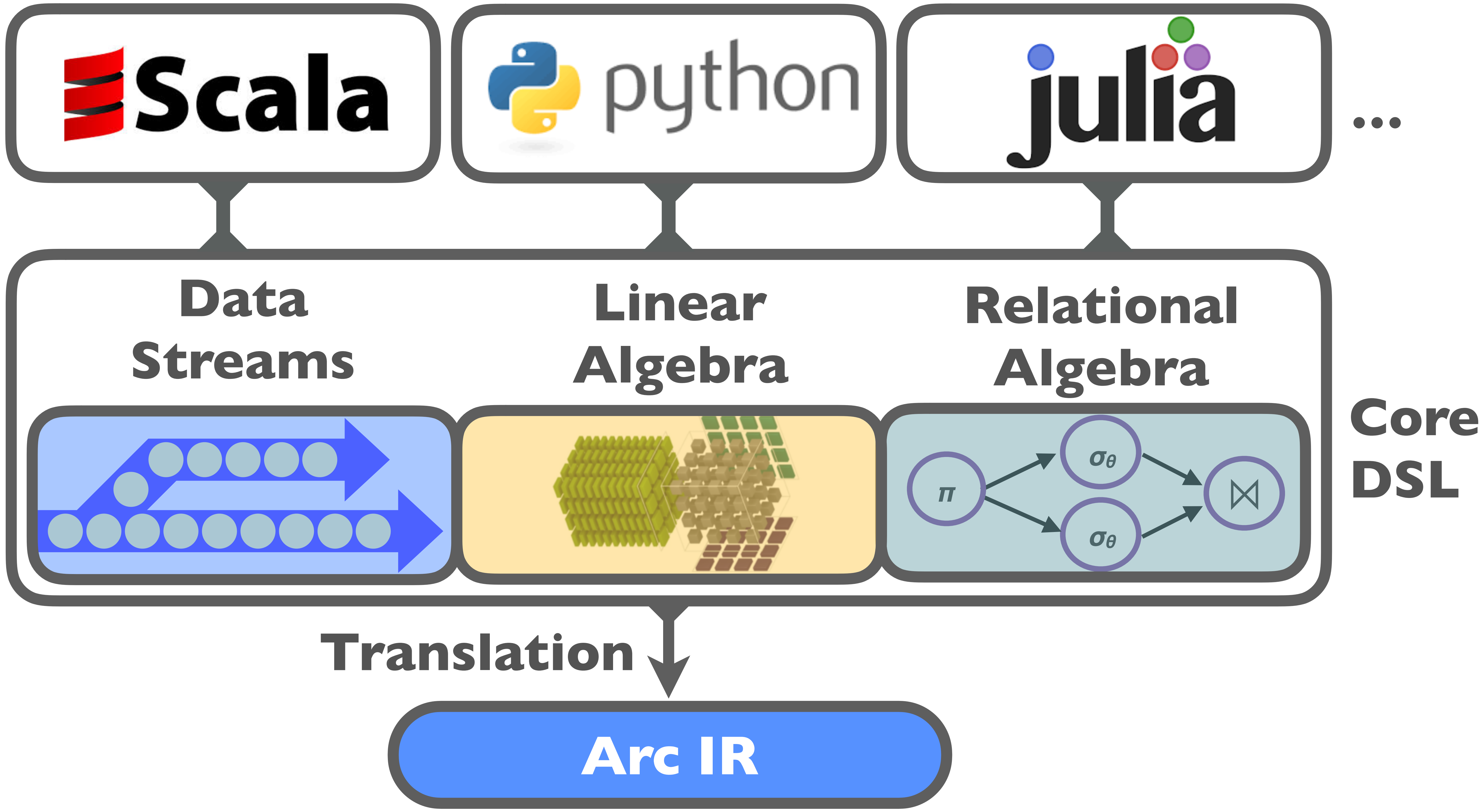
Unified Analytics DSL

Arc IR (Intermediate Representation)

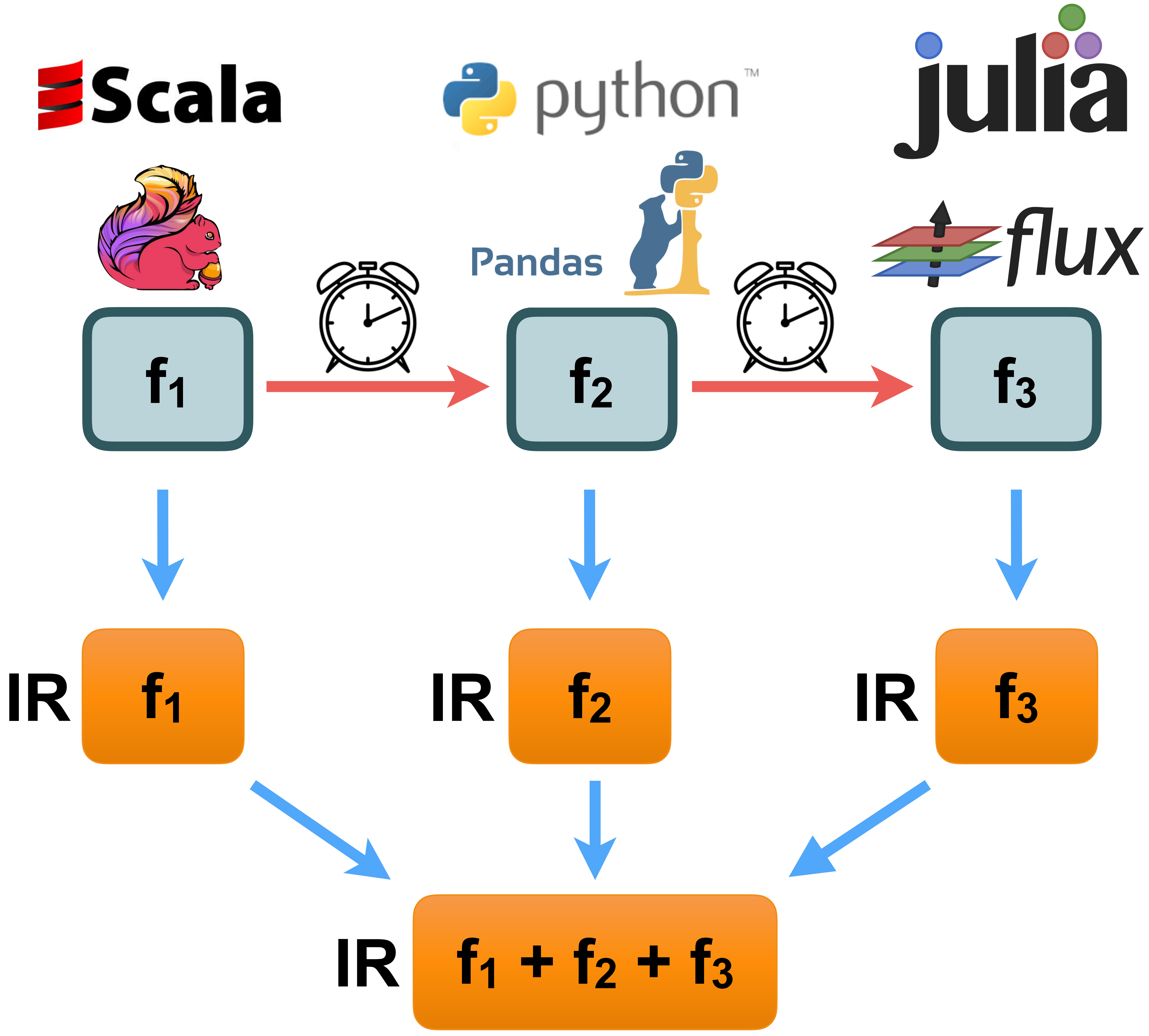
Arcon Runtime

Unified Analytics DSL

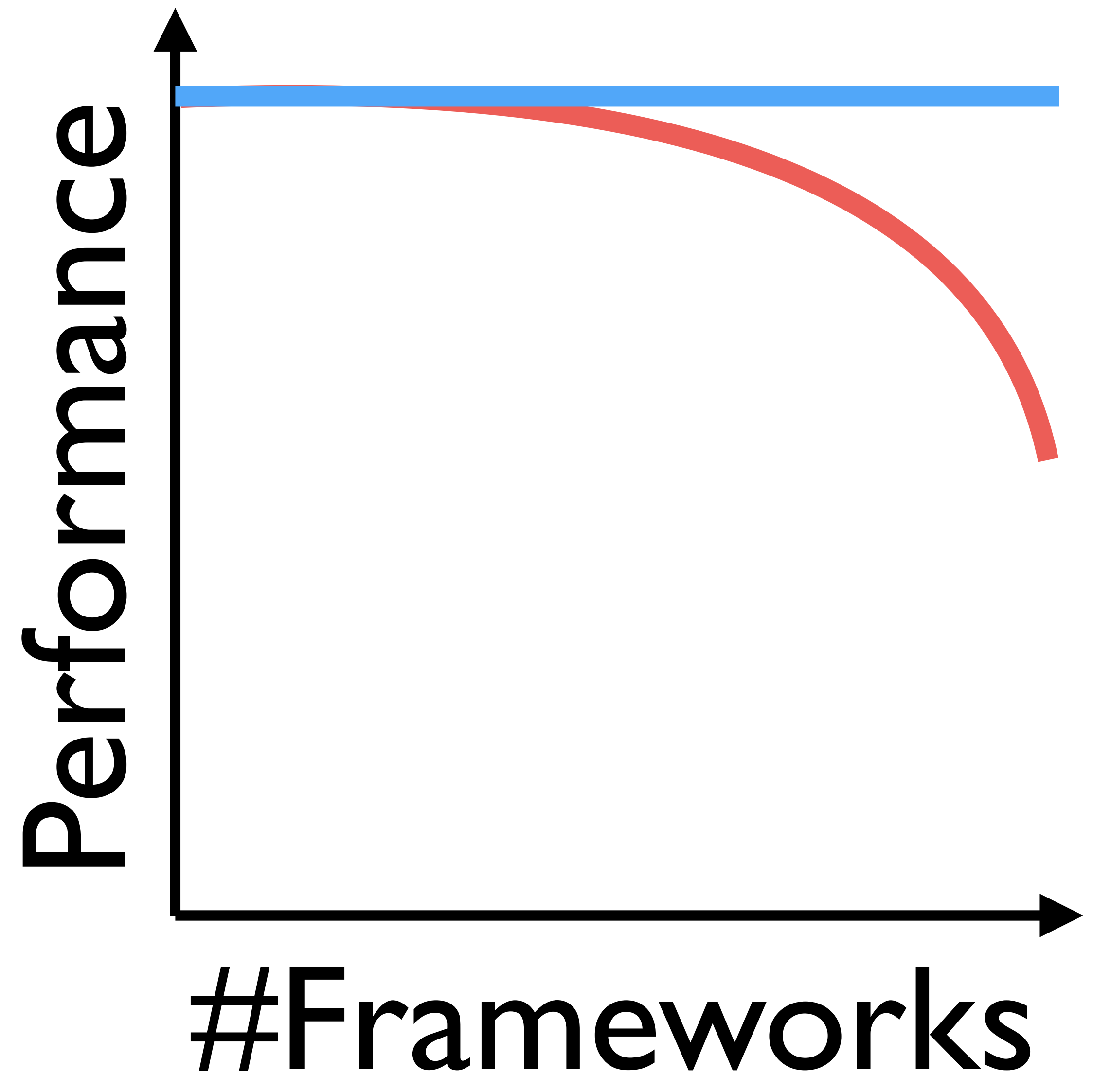
- Host language-agnostic core
- Compositional
- First-class citizen support for:
  - *streams, tensors, relations*



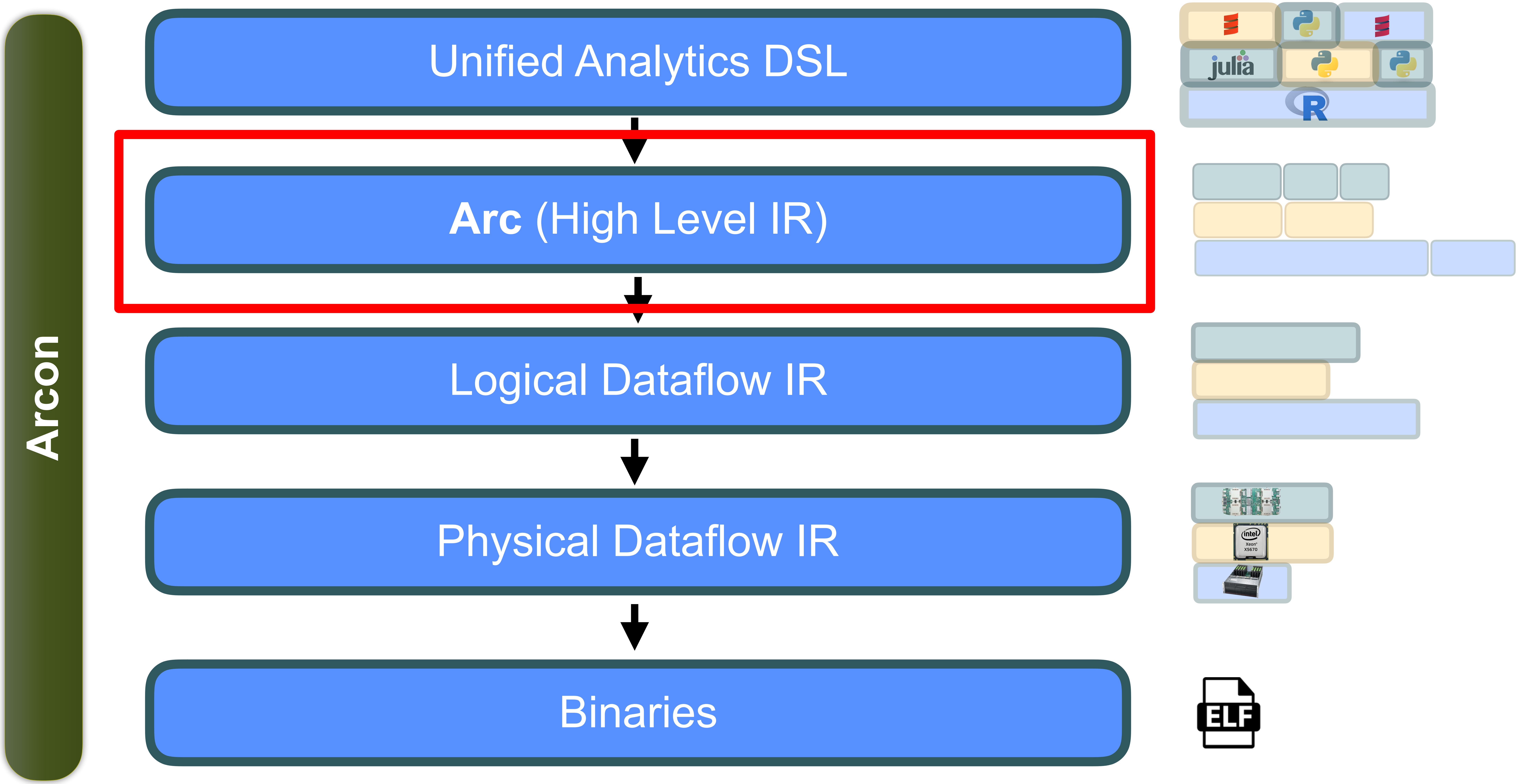
IR Intuition



- No cross-optimisation is possible, e.g. resource sharing
- Data movement costs (→)



Arcon Compiler Pipeline



Arcon

Unified Analytics DSL

Arc (High Level IR)

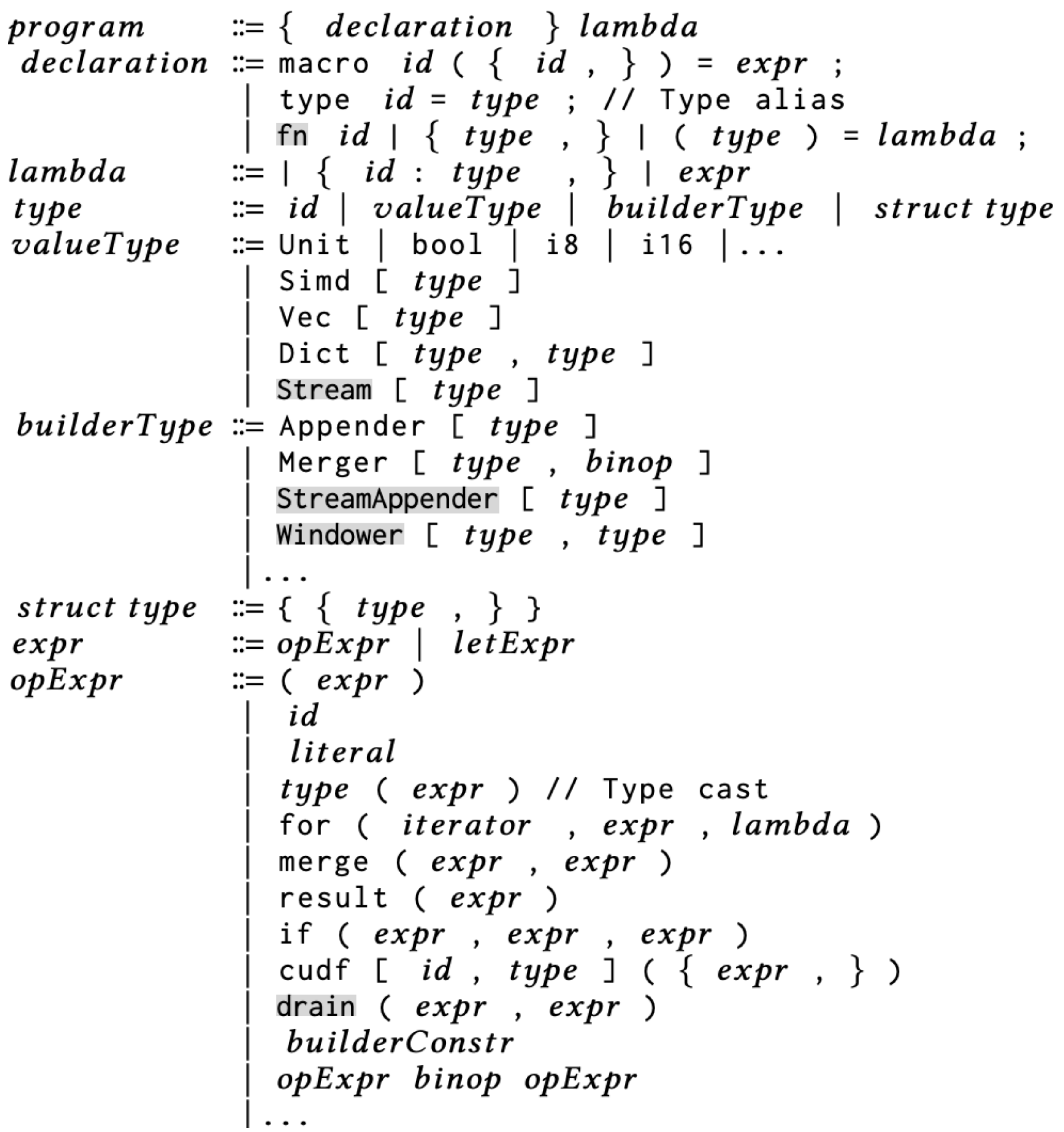
Logical Dataflow IR

Physical Dataflow IR

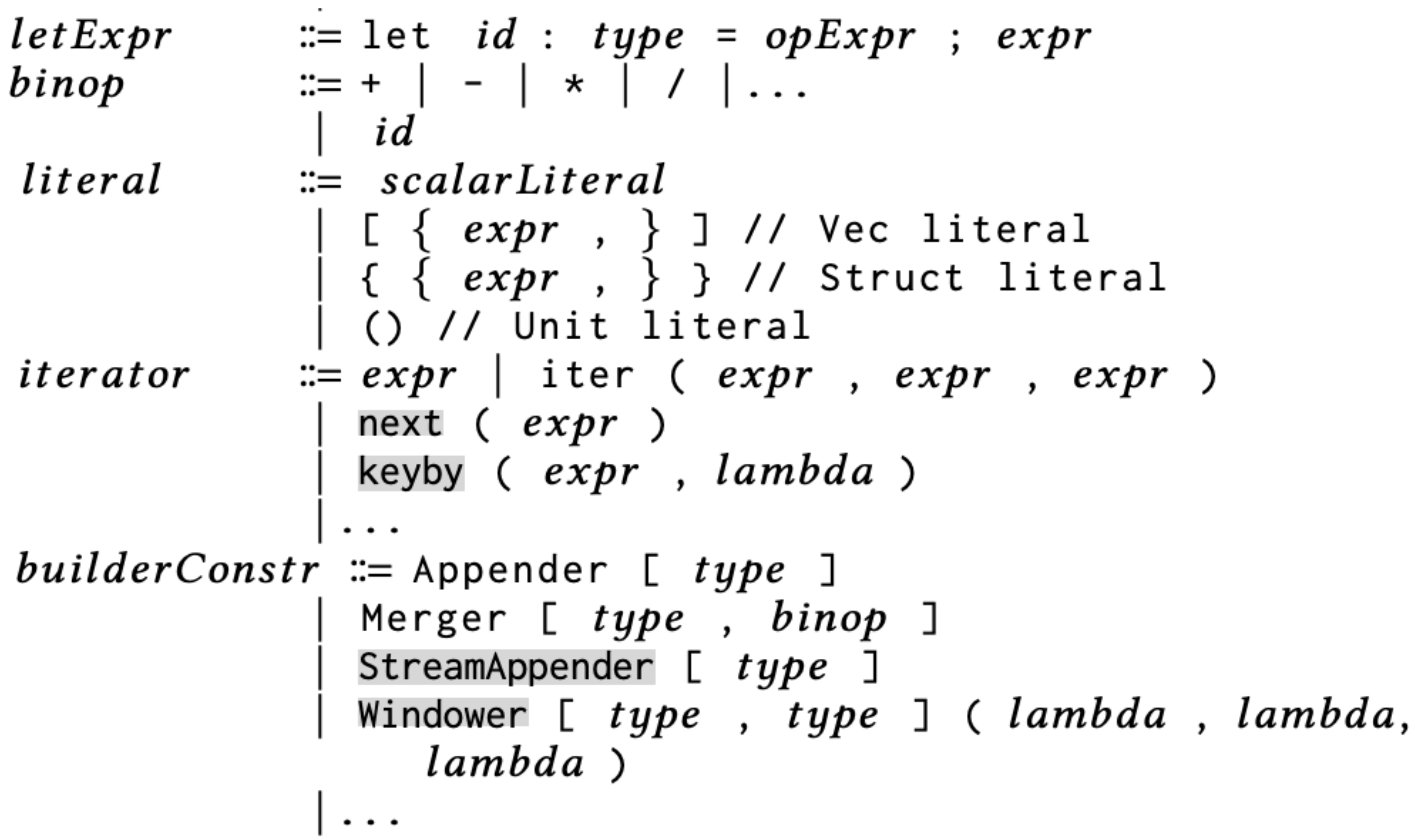
Binaries

Arc IR

- A minimal yet feature-complete set of read/write-only types and expressions



```
program      ::= {  declaration  } lambda
 declaration ::= macro  id ( {  id , } ) = expr ;
                | type  id = type  ; // Type alias
                | fn  id | { type  , } | ( type  ) = lambda ;
lambda       ::= | {  id :  type   ,  } | expr
type         ::=  id | valueType | builderType |  struct type
valueType    ::= Unit  | bool  | i8  | i16  |...
                | Simd [ type ]
                | Vec [ type ]
                | Dict [ type , type ]
                | Stream [ type ]
builderType  ::= Appender [ type ]
                | Merger [ type , binop ]
                | StreamAppender [ type ]
                | Windower [ type , type ]
                |...
 struct type ::= { { type  , } }
expr         ::= opExpr |  letExpr
opExpr       ::= ( expr  )
                |  id
                |  literal
                | type  ( expr  ) // Type cast
                | for (  iterator  , expr , lambda )
                | merge ( expr , expr )
                | result ( expr )
                | if ( expr  , expr  , expr  )
                | cudf [  id , type ] ( { expr  , } )
                | drain ( expr  , expr  )
                |  builderConstr
                | opExpr  binop  opExpr
                |...
```



```
letExpr       ::= let  id :  type  = opExpr  ;  expr
binop         ::= + | - | * | / |...
                | id
 literal      ::=  scalarLiteral
                | [ { expr  , } ] // Vec literal
                | { { expr  , } } // Struct literal
                | () // Unit literal
 iterator     ::= expr | iter ( expr , expr , expr )
                | next ( expr )
                | keyby ( expr  , lambda )
                |...
 builderConstr ::= Appender [ type ]
                | Merger [ type , binop ]
                | StreamAppender [ type ]
                | Windower [ type , type ] ( lambda , lambda,
                    lambda )
                |...
```

Read More

[Paper] Arc: An IR for Batch and Stream Programming @ DBPL19

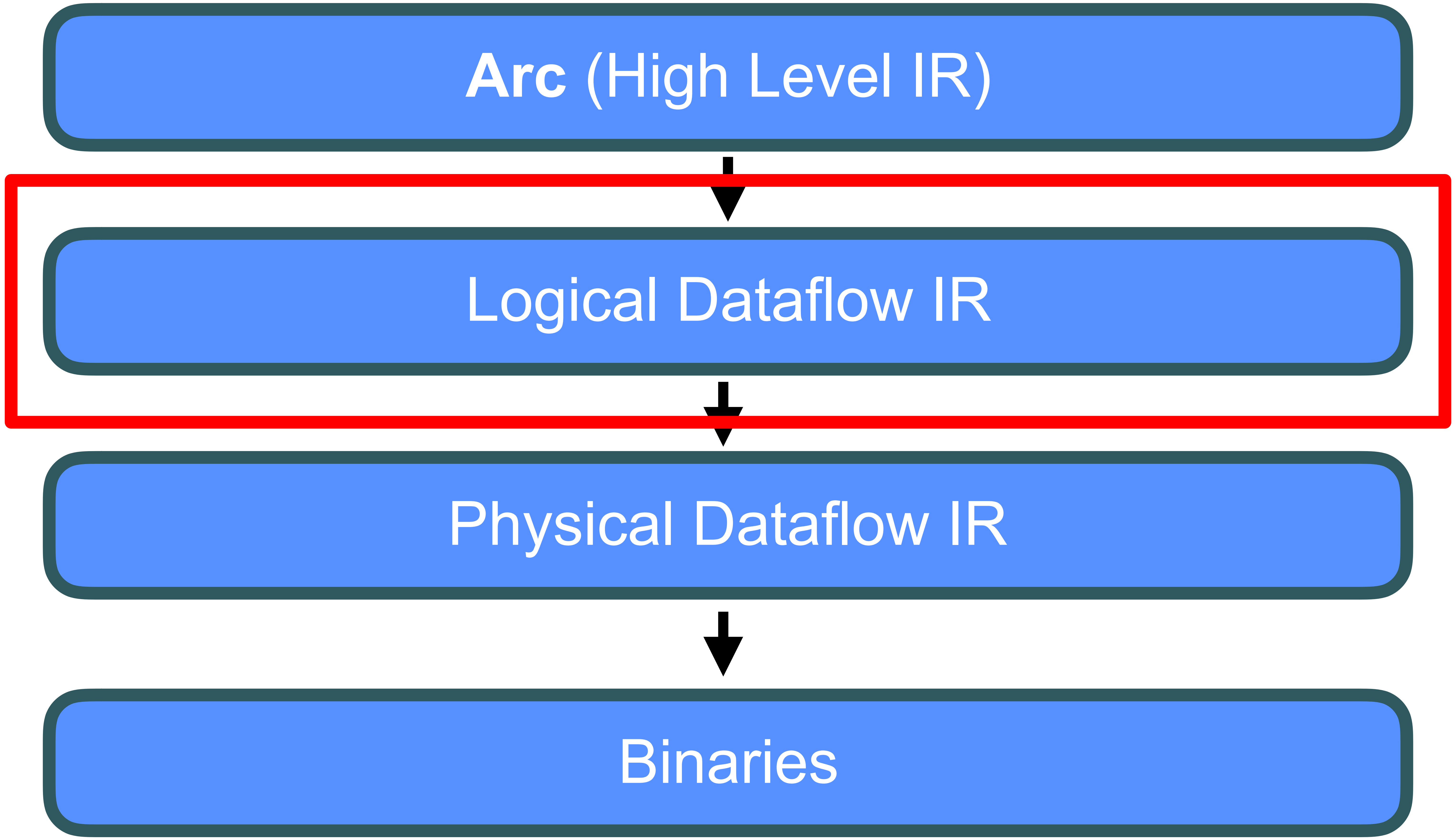
[Code] https://github.com/cda-group/arc

Arc Optimisations

- Arc supports **both** compiler and dataflow optimisations
  - **Compiler**: Loop unrolling, partial evaluation,
  - **Dataflow**: Operator fusion, fission, reordering, predicate pushdown, specialisation, ...

Unlocking Speed

Arc can boost even existing frameworks



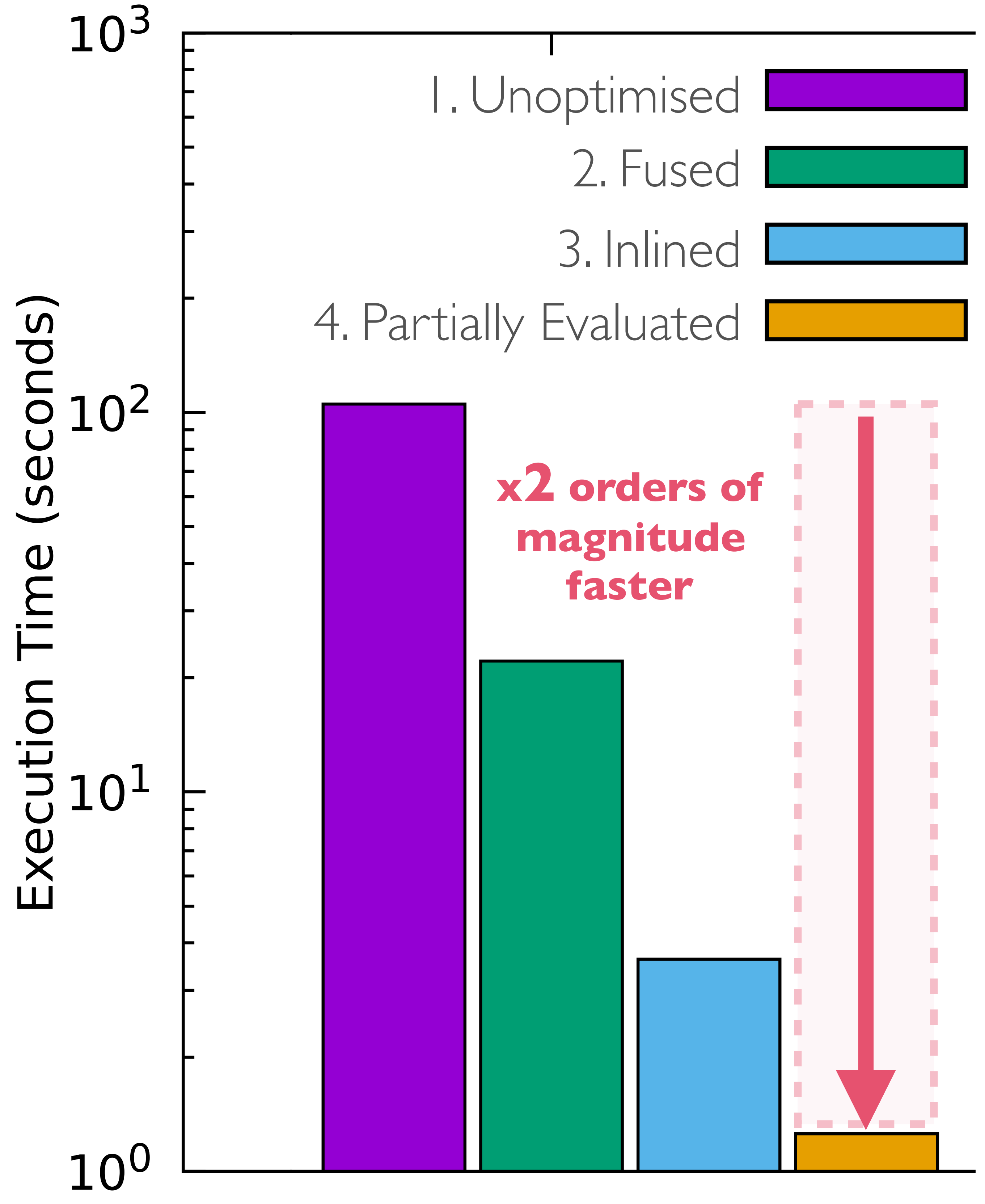
Arc (High Level IR)

Logical Dataflow IR

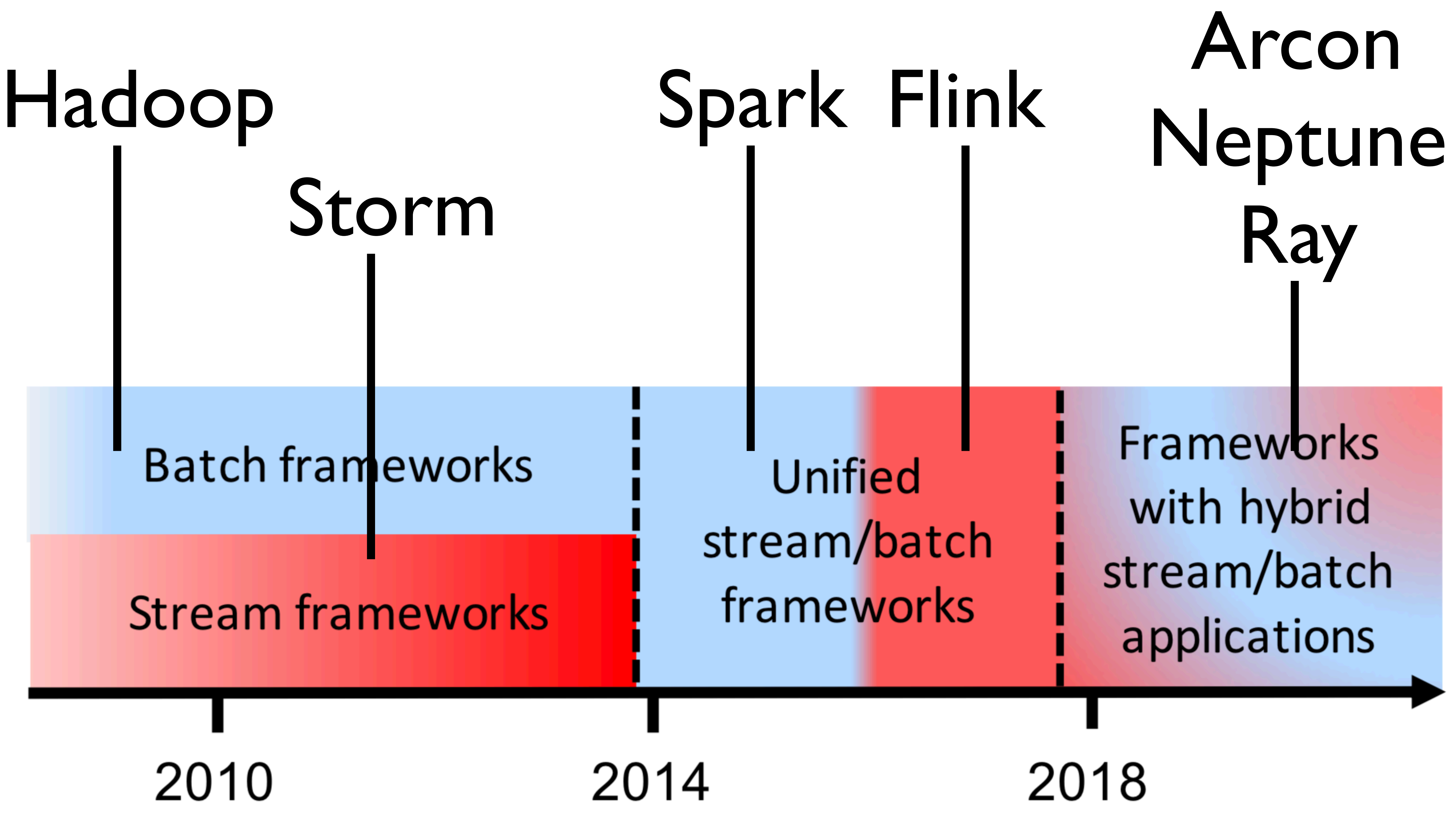
Physical Dataflow IR

Binaries

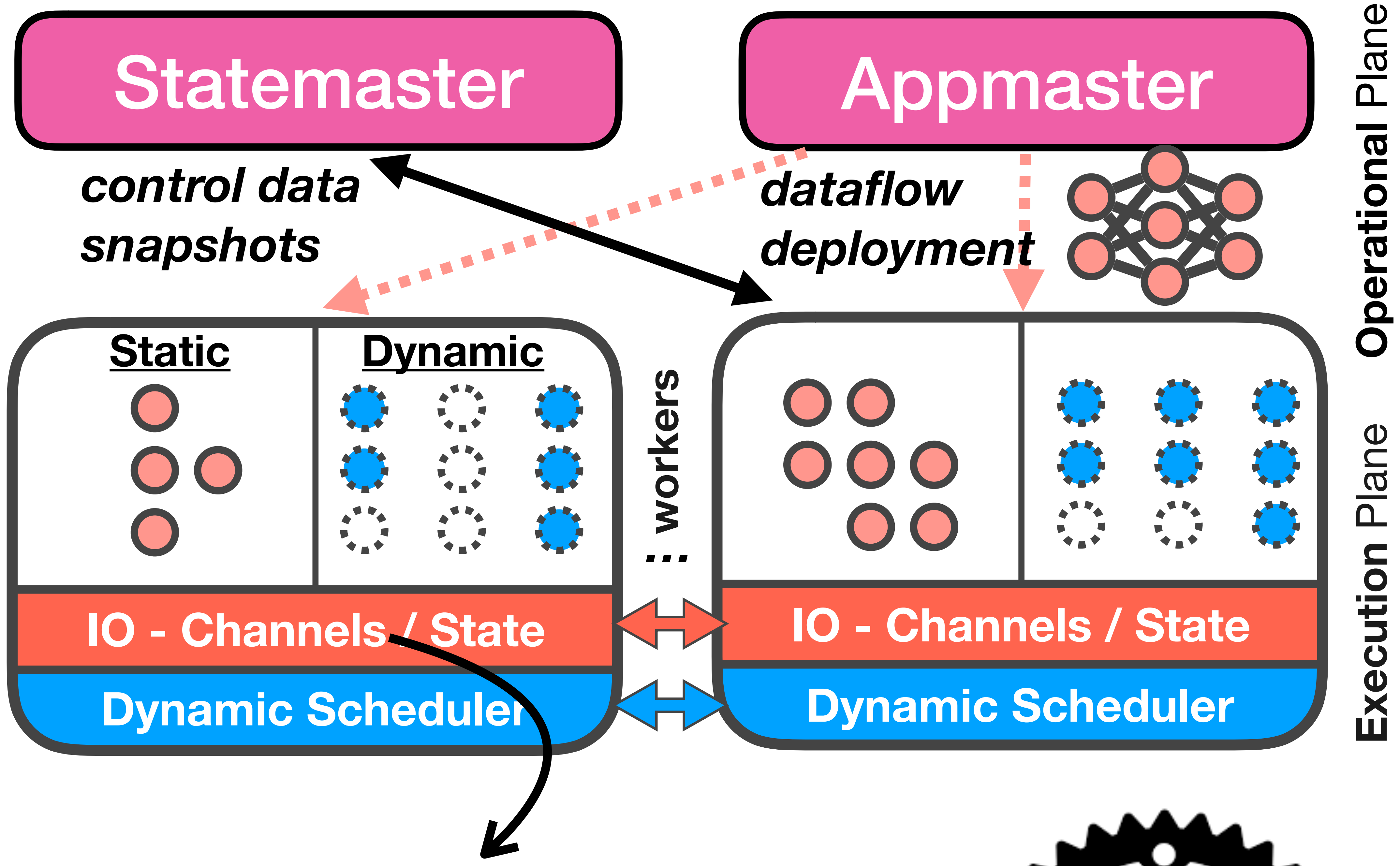
10M elements
50 map operations
on **Apache Flink**



A Runtime Capable for Unified Analytics



Neptune: Scheduling Suspendable Tasks for Unified Stream/Batch Applications SOCC 2019
Garefalakis, Karanasos, Pietzuch



Flexible State Backends
(external/shared, embedded)

Performance Matters

- Arc Optimiser : **~10x Speedup**
- Shared Hardware Acceleration : **~10^{2}x Speedup**
- Data Parallel Execution : **~10^{3}x Speedup**

Learn More

Code: https://github.com/cda-group/arc

https://github.com/cda-group/arcon

Project: https://cda-group.github.io

https://twitter.com/SenorCarbone