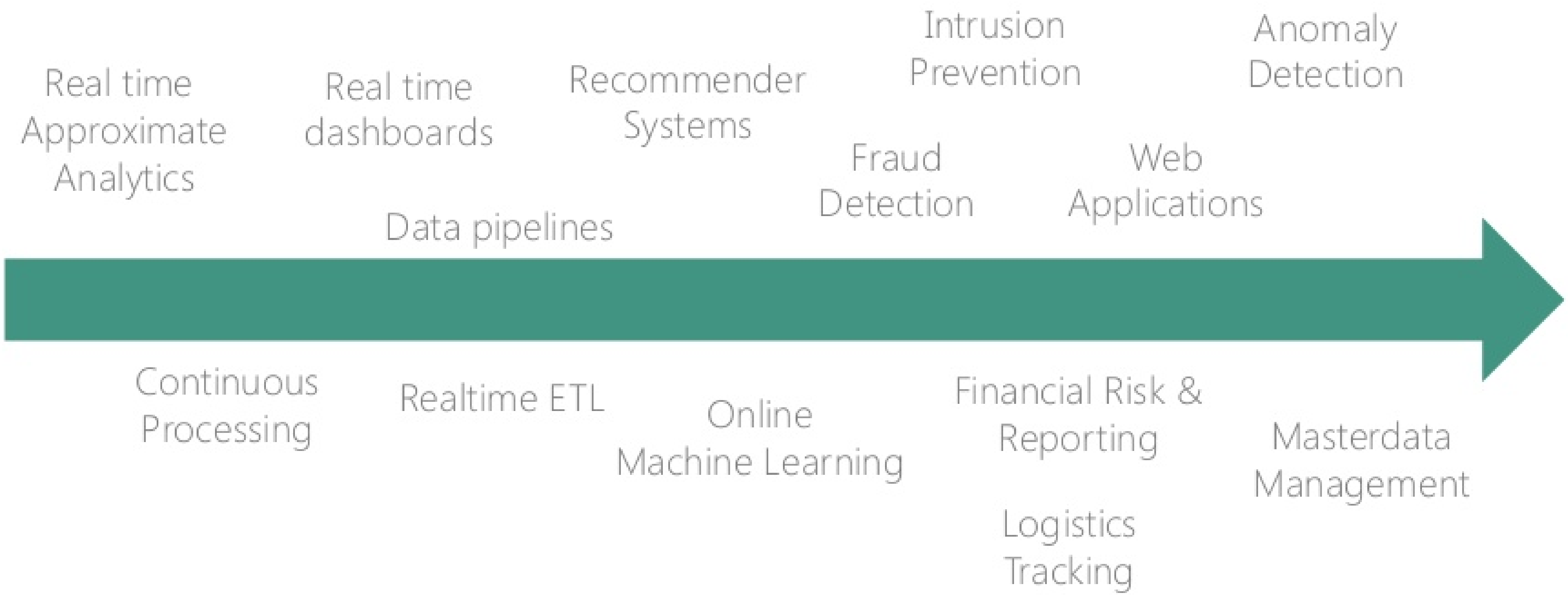
UNLOCKING THE NEXT WAVE OF STREAMING APPLICATIONS

Stephan Ewen – data Artisans CTO, Apache Flink PMC

dataArtisans

Streaming Applications over Time



Real time Approximate Analytics

Real time dashboards

Data pipelines

Recommender Systems

Intrusion Prevention

Fraud Detection

Web Applications

Anomaly Detection

Continuous Processing

Realtime ETL

Online Machine Learning

Financial Risk & Reporting

Logistics Tracking

Masterdata Management

Continuous Processing and Analytics

Data-driven Applications

Unification of Analytics and Applications

Enablers of new Applications

Abstractions, APIs

Event-time, streaming SQL, state & time, CEP

Consistency

Exactly-once, savepoints

Scalability

high parallelism, large state

Interoperability

Deployment, Connectors, Operations

Enablers of new Applications

Steam SQL,
Time-versioned Table Joins,
SQL+CEP, …

Abstractions, APIs

Event-time, streaming SQL,
state & time, CEP

Consistency

Exactly-once, savepoints

Scalable timers,
dynamic scaling,
local recovery, …

Scalability

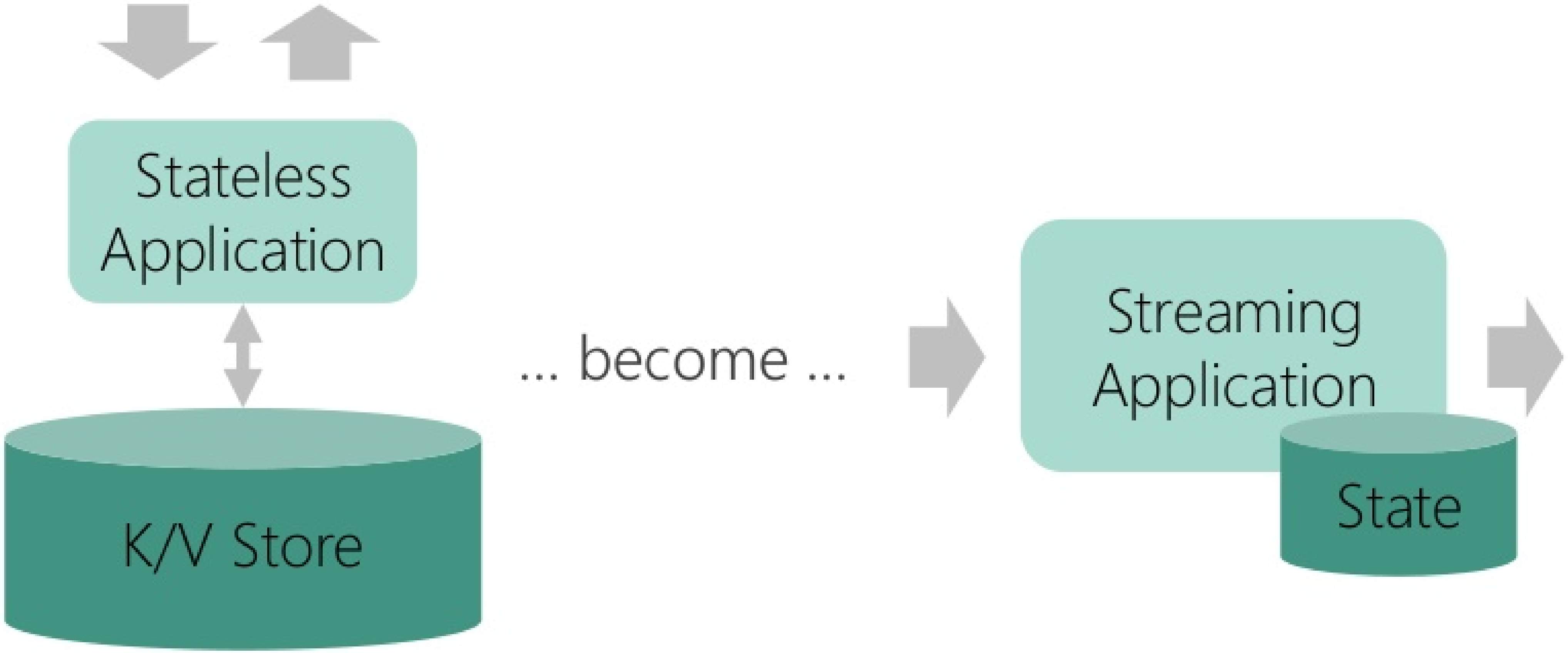
high parallelism, large state

Framework and Library,
REST-ified Flink,
SQL Client, …

Interoperability

Deployment, Connectors, Operations

Exactly-once Changed Applications



Stateless Application

K/V Store

... become ...

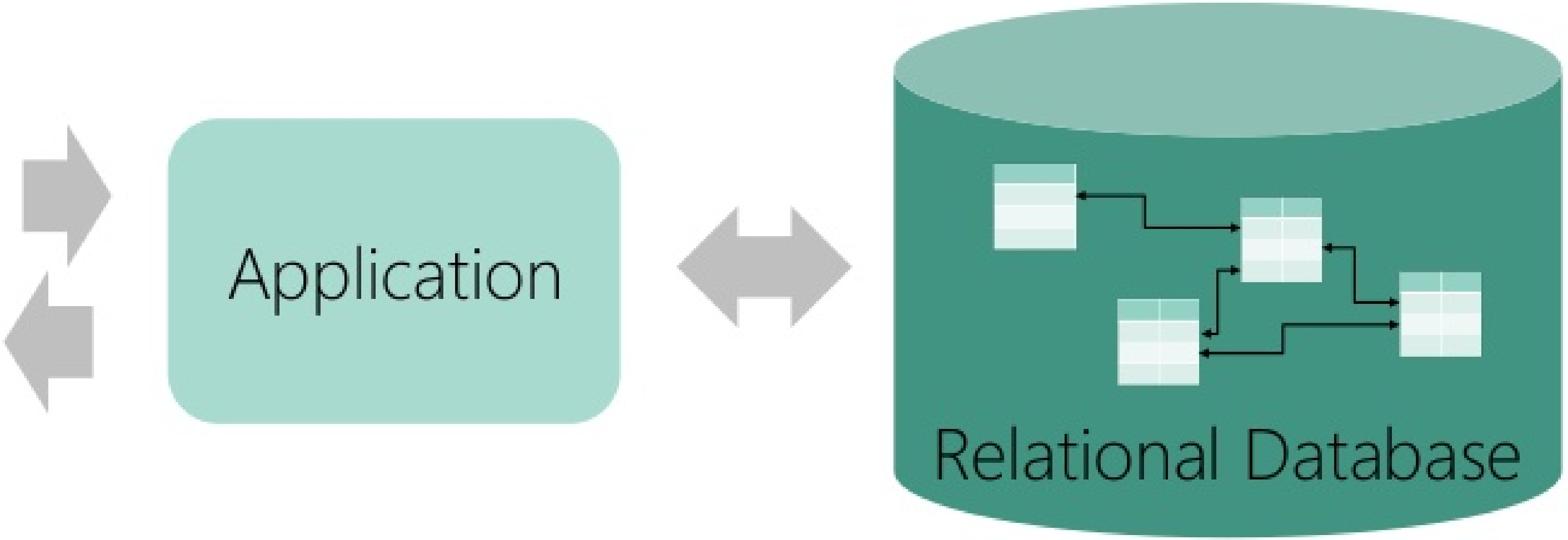
Streaming Application

State

CRUD / request/response Applications

Stateful Stream Processing Applications

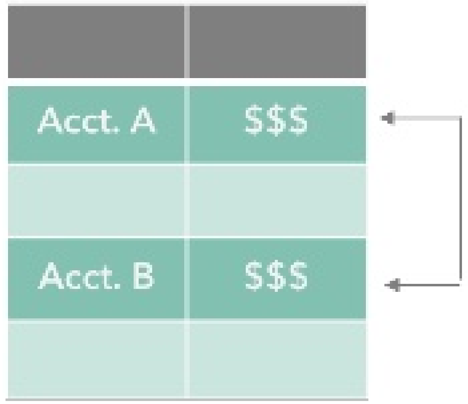
Some Applications don't move to Stream Processing



Limitation of Current Stream Processors

The Limitation

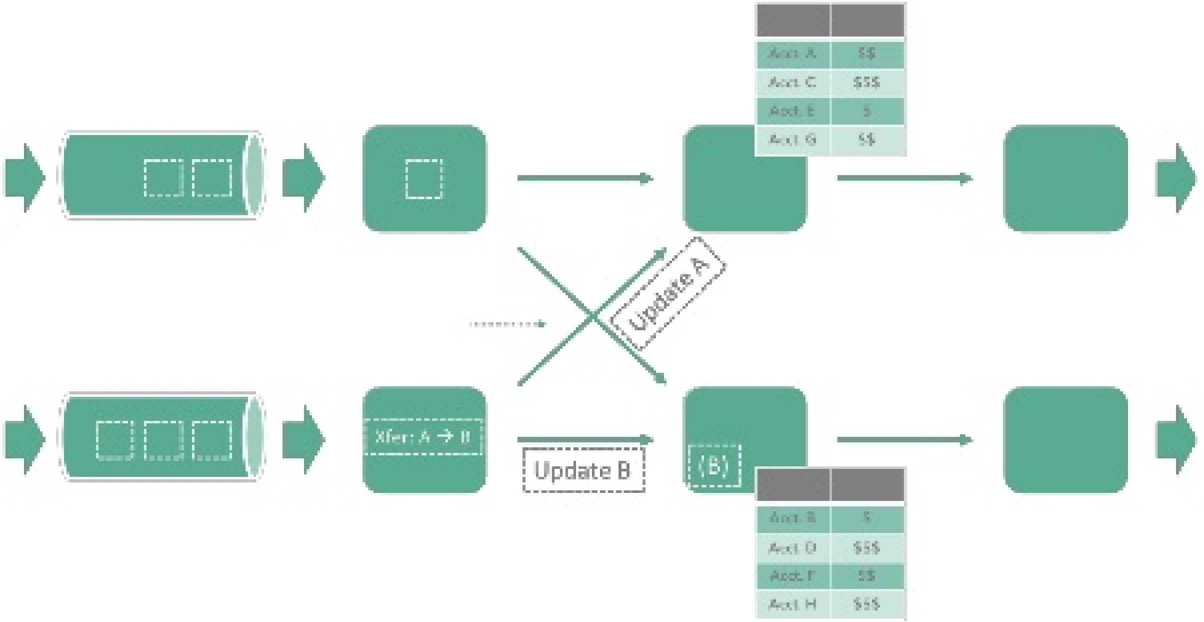
All stream processors so far can update a single key-at-a-time with correctness guarantees (exactly once)



Acct. A	$$$
Acct. B	$$$

Example

Transferring money from one account (key) to another with transactional guarantees is not feasible



Acct. A	$$
Acct. C	$$$
Acct. E	$
Acct. G	$$

Acct. B	$
Acct. D	$$$
Acct. F	$$
Acct. H	$$$

Wouldn't it be great if Stream Processors could…

- … access and update state with multiple keys at the same time
- … maintain full isolation/correctness for the multi-key operations
- … operate on multiple states at the same time
- … share the states between multiple streams

TODAY WE ANNOUNCE

The first system for
serializable multi-row ACID transactions
on streaming data

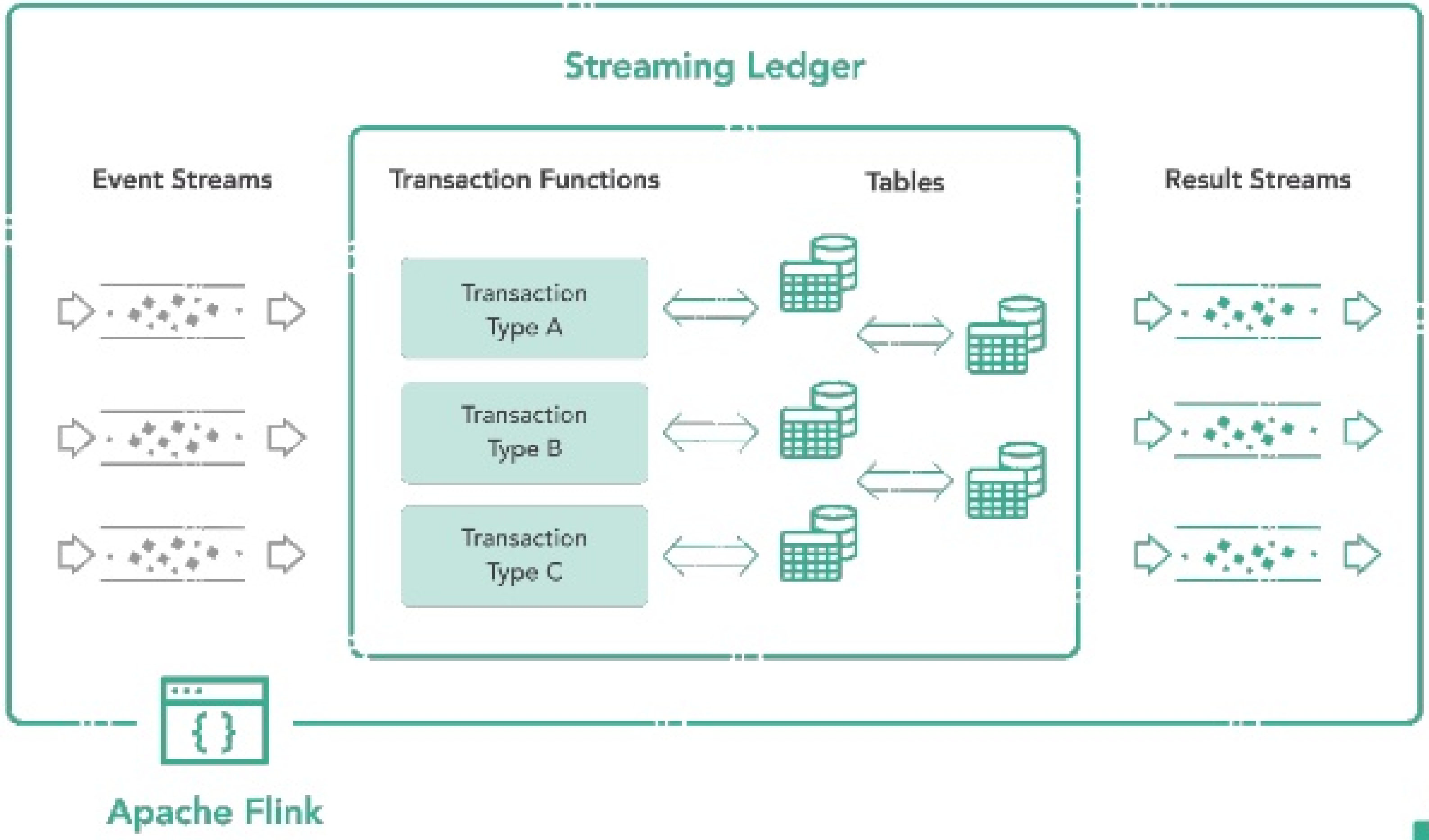
data Artisans Streaming Ledger

Serializable ACID Transactions on Streaming Data

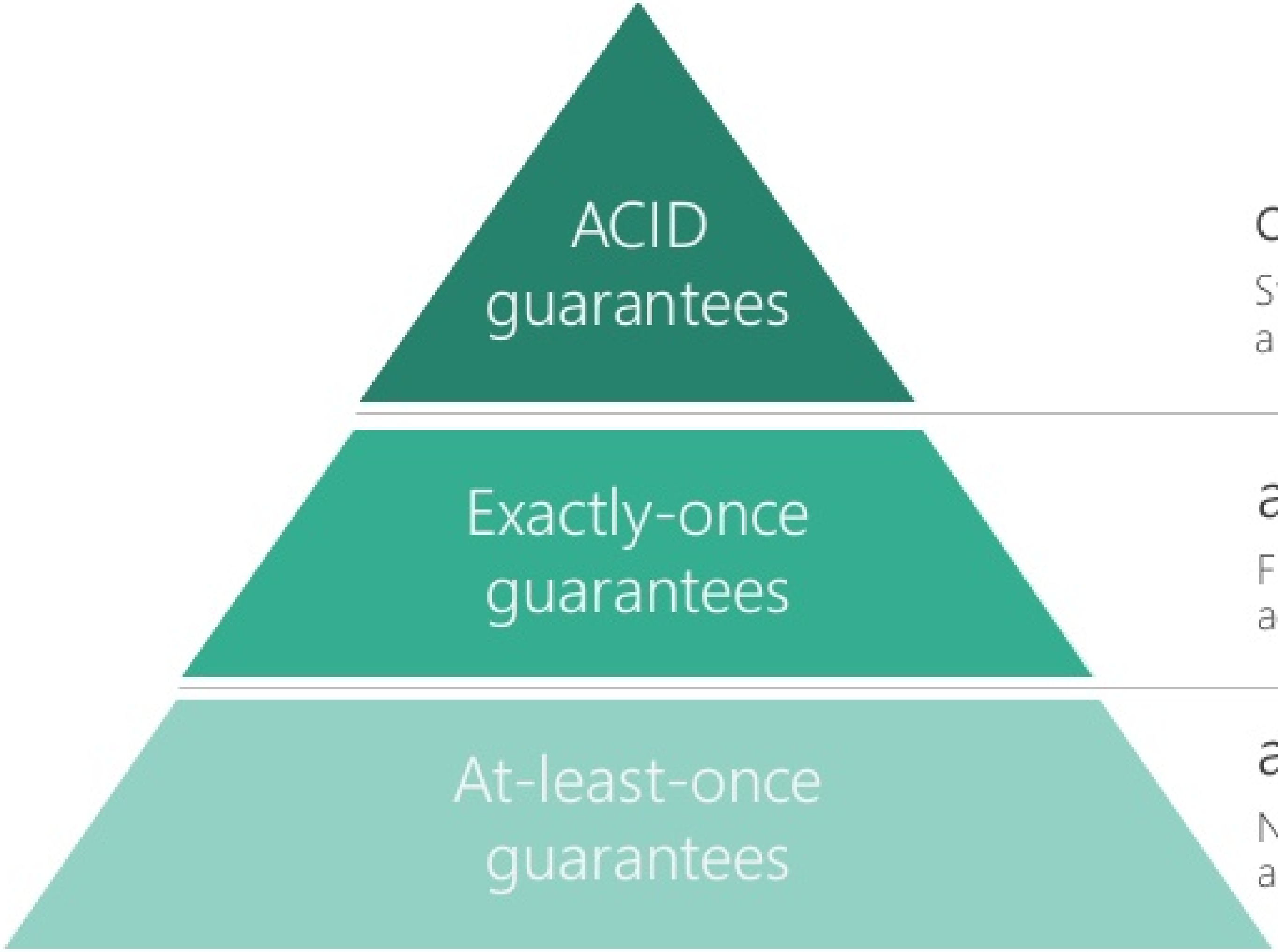
ACID Transactions for Multi-key Stream Processing

Streaming Ledger provides ACID guarantees across multiple states, rows, and streams

- **Atomicity:** the transfer affects either both accounts or none
- **Consistency:** the transfer must only happen if the account have sufficient funds
- **Isolation:** no other operation can interfere and cause an incorrect result
- **Durability:** the result of the transfer is durable



The Evolution of Stream Processing



ACID guarantees

Exactly-once guarantees

At-least-once guarantees

consistent general applications

Streaming Ledger provides ACID guarantees supporting applications that read and modify several keys

accurate single-key applications

Flink pioneered exactly once guarantees on true streaming: accurate analytics and applications, consistent single key at a time

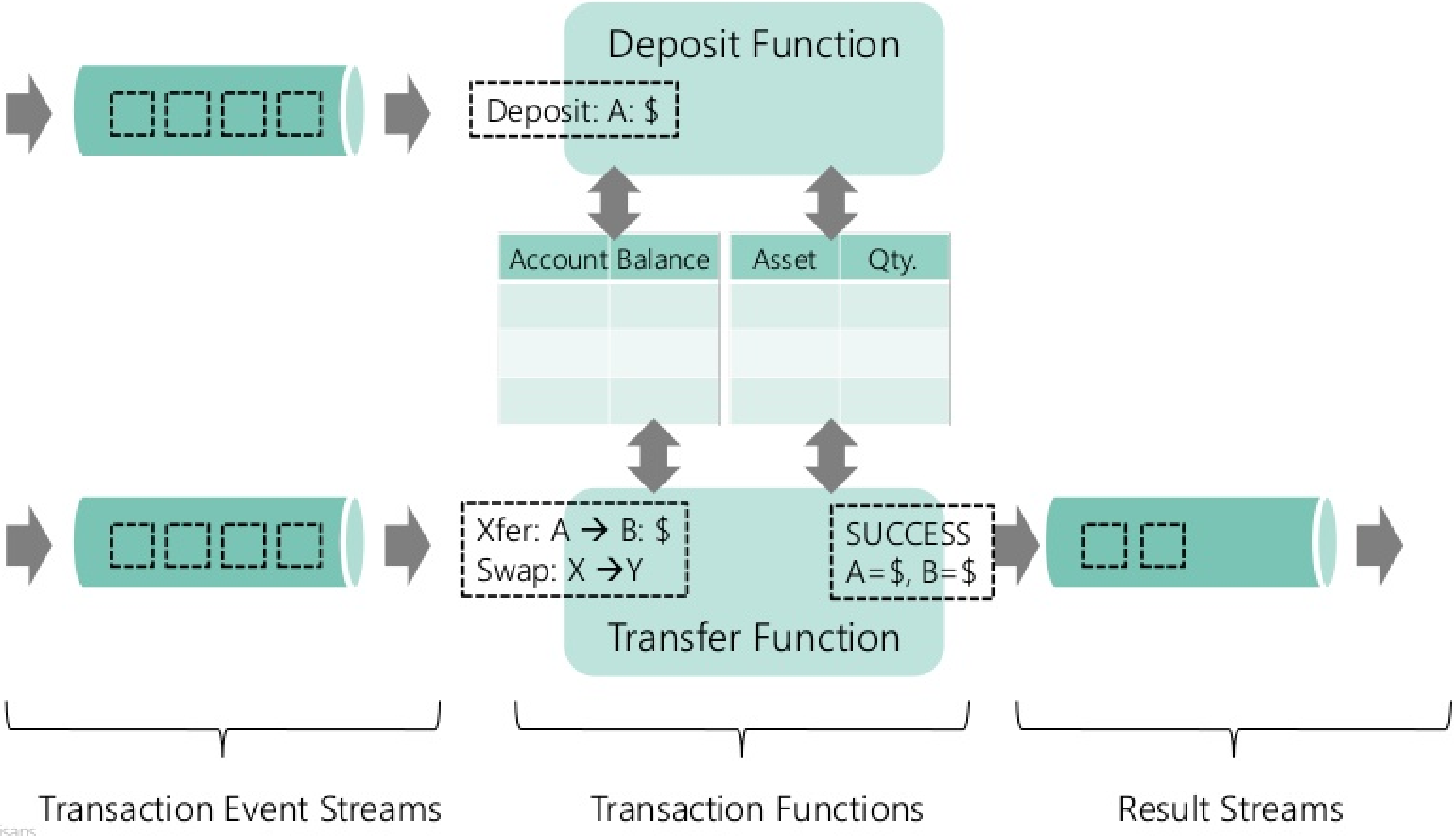
approximate real time analytics

No data loss but possible duplications to support real-time approximate analytics in a "speed layer" (lamda)

Stateful Streaming vs. Streaming Ledger

Computing	exactly-once stateful streaming	data Artisans **Streaming Ledger**
Storage	Key/Value Stores	Relational Databases
	Single datum / key at a time	Consistent total view

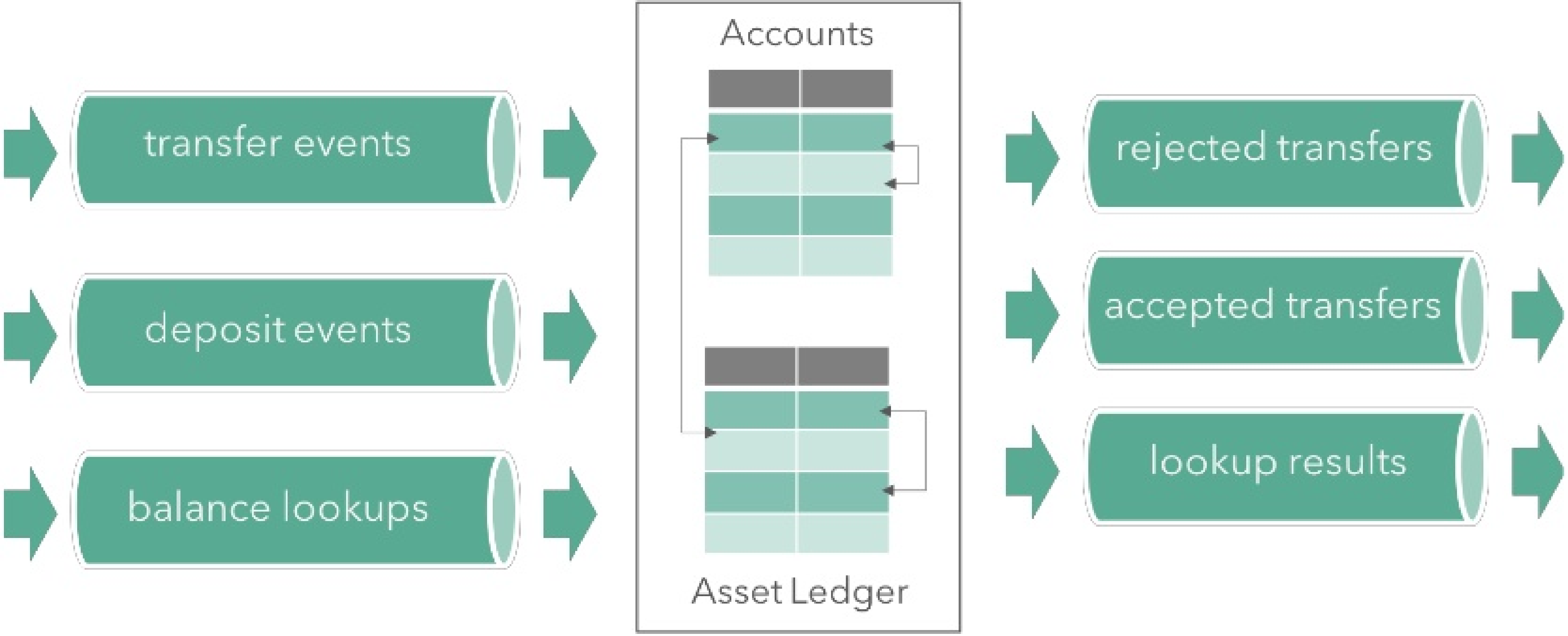
Tables, Streams, Transactions



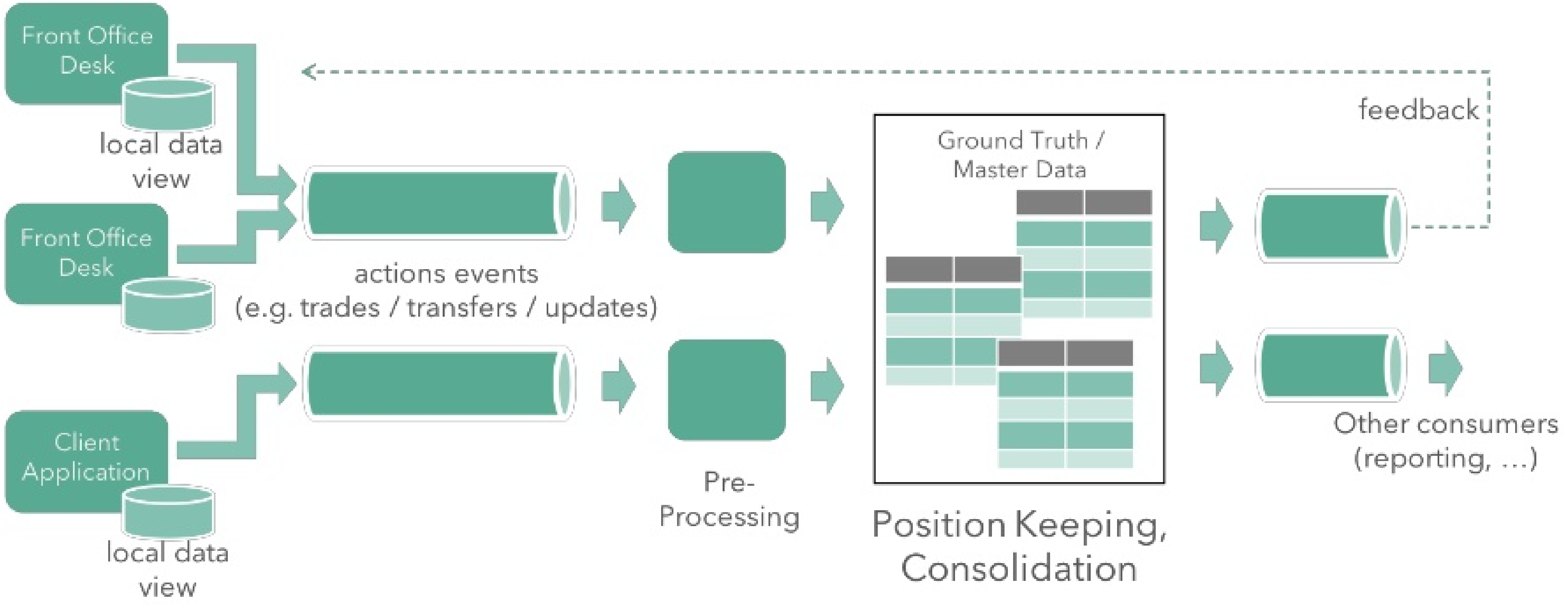
Account	Balance

Asset	Qty.

Example: Transferring Cash/Assets between Accounts

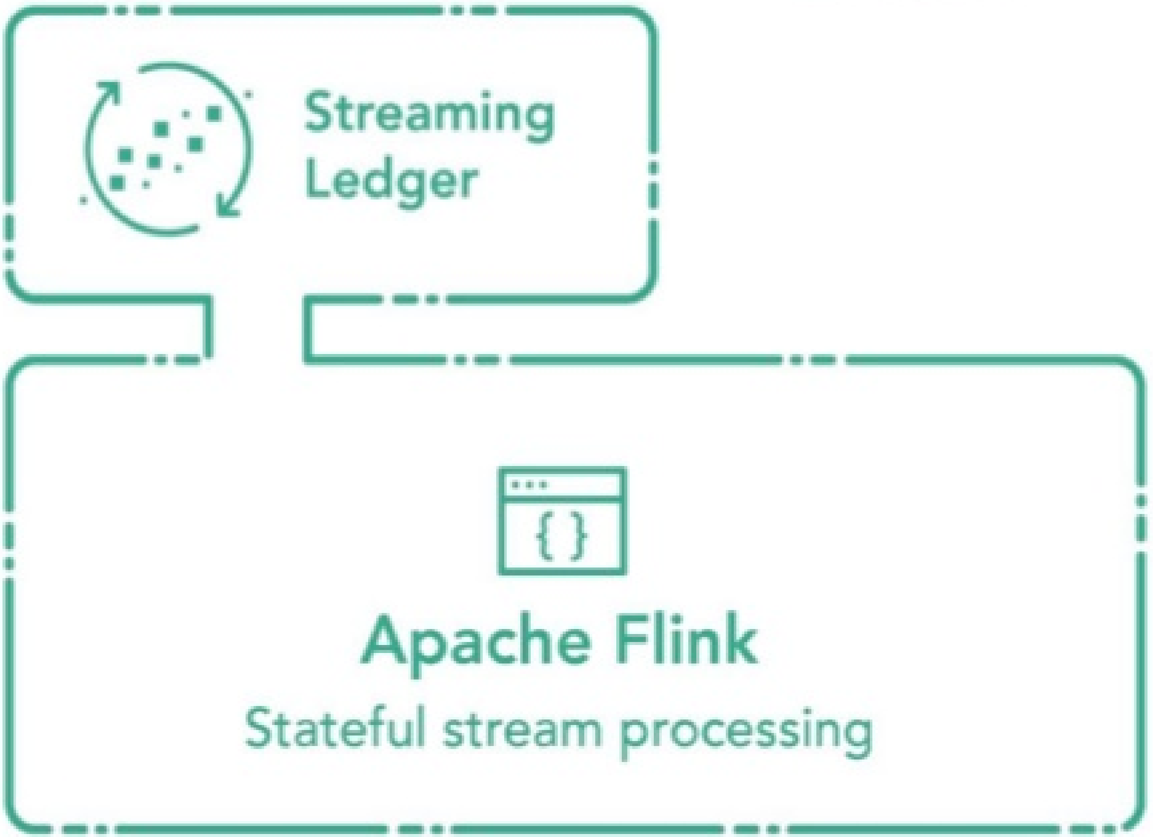


Example: Position-keeping, Reporting, Risk Management in Investment Banking



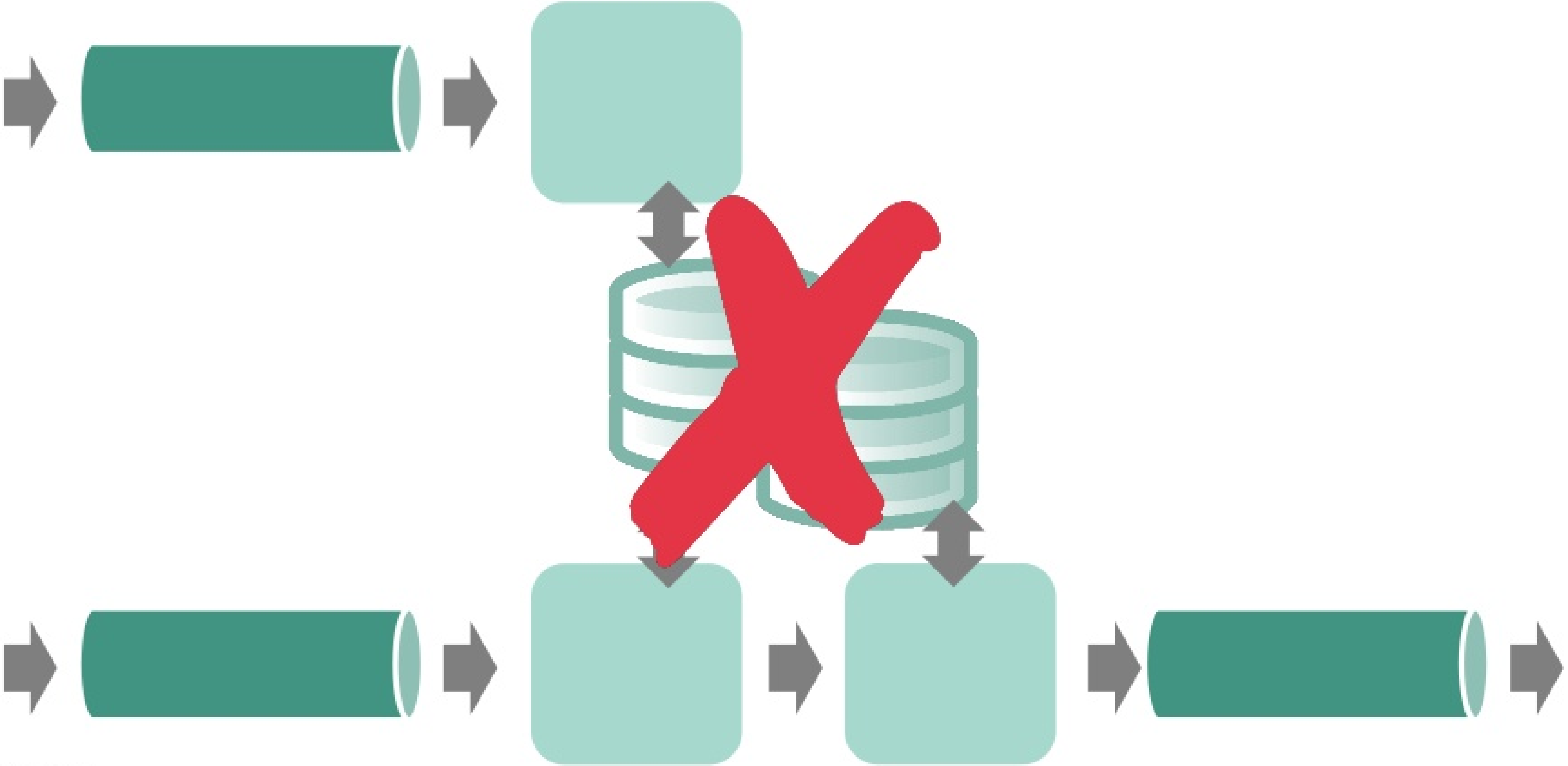
A Library on top of Apache Flink

- No additional dependencies needed
- Seamlessly integrates and composes with DataStream API and SQL
- Read from- and write to all Flink connectors
- Supports savepoints for upgrades

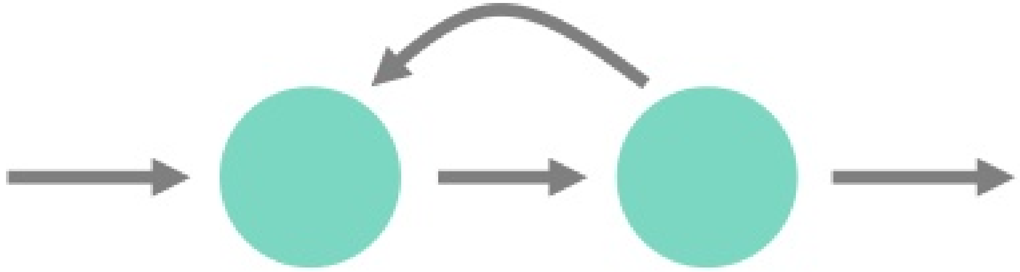


How does it work?

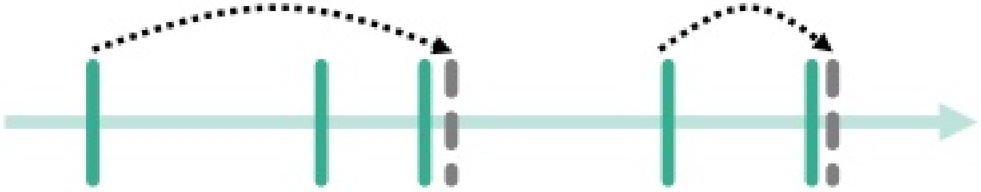
Relational Database embedded? No!



First-of-a-kind - Unique Approach



Iterative streaming dataflows



event re-ordering & out-of-order processing

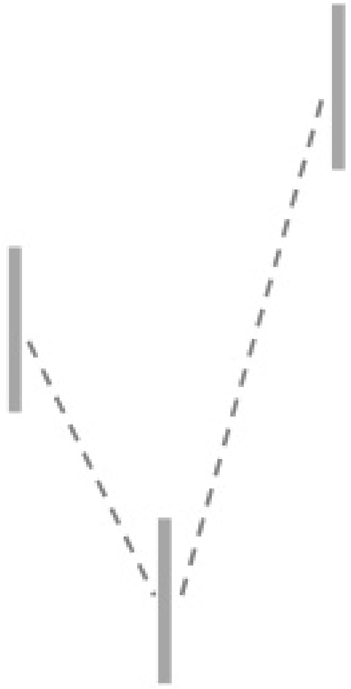
(0xf876ab78|0x0b7cc7a3)

Logical Clocks to define schedule

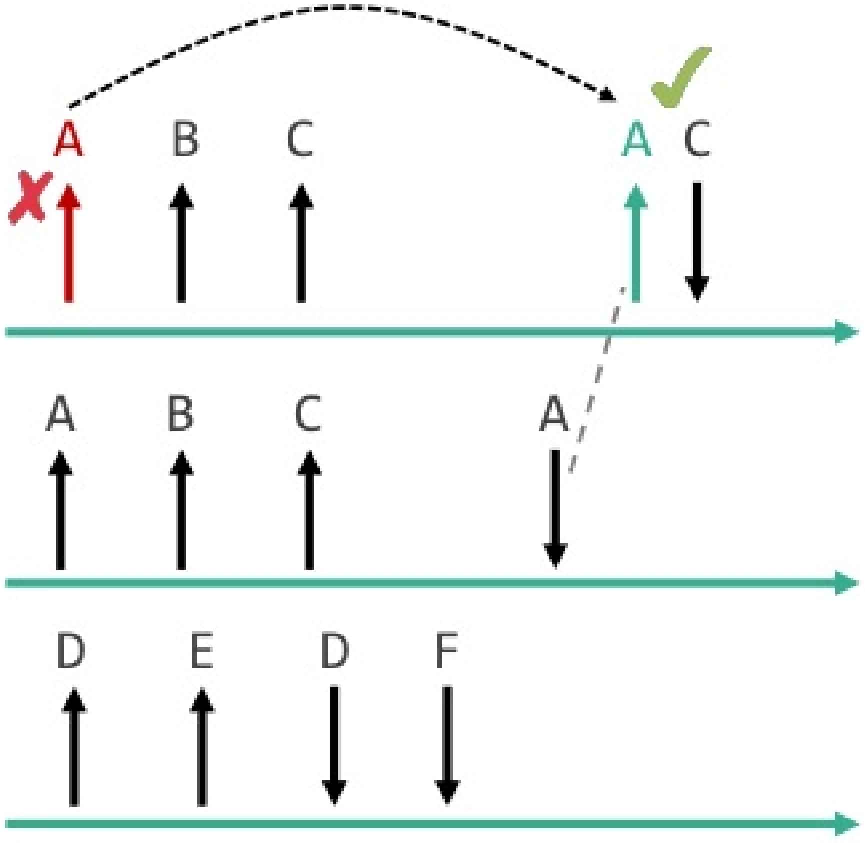
Conflict-free Schedule

```
Txn: (A,B,C) -> C

Txn: (A,B,C) -> A

Txn: (D,E) -> (D,F)
```

transaction events

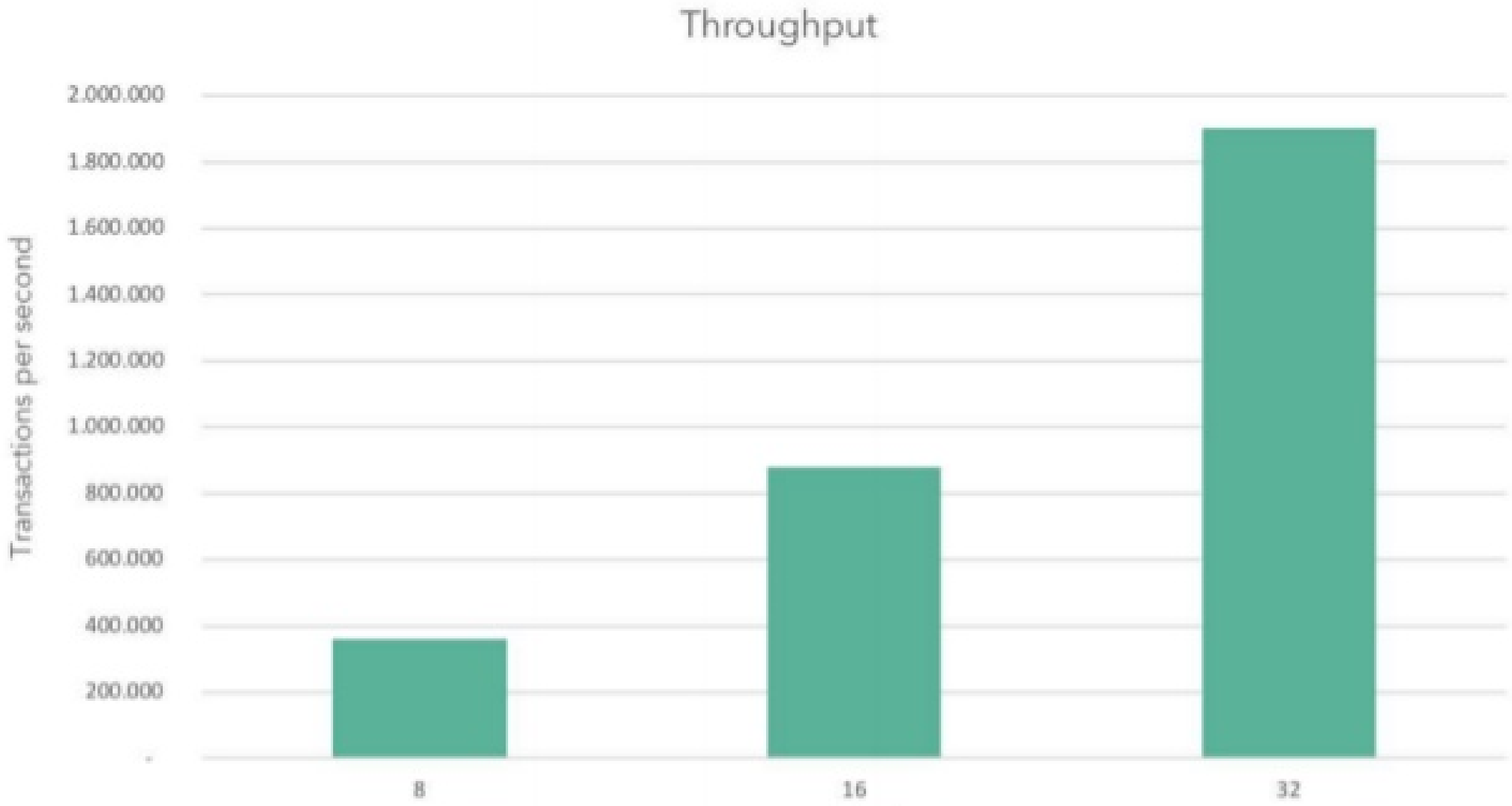


define ordering for schedule



reorder events to obey schedule

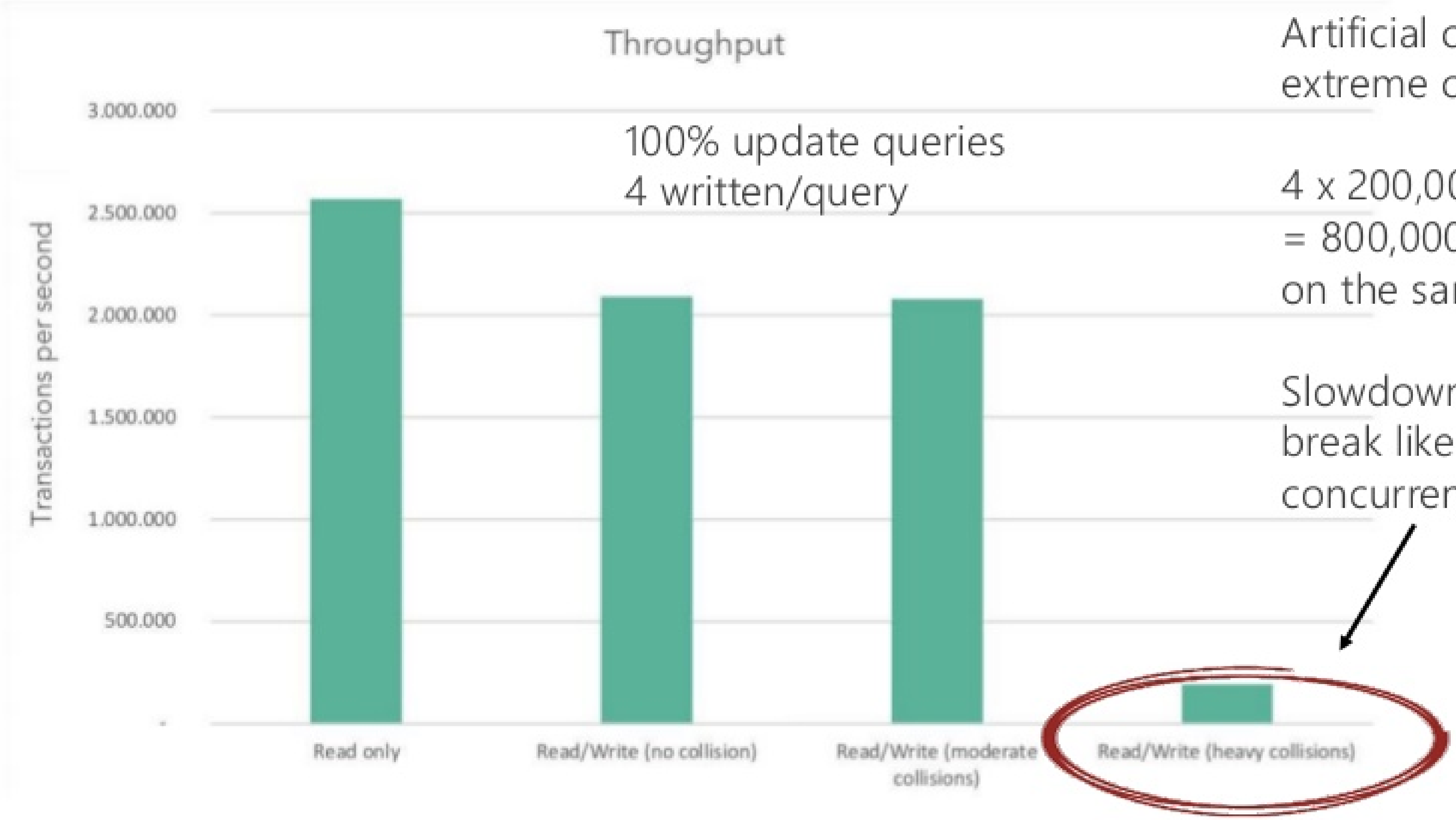
Performance (early results) - Scalability



(parallelism)

200 million rows
100% update queries
4 rows written/query

Performance (early results) – Key Contention



100% update queries
4 written/query

Artificial case of extreme contention:

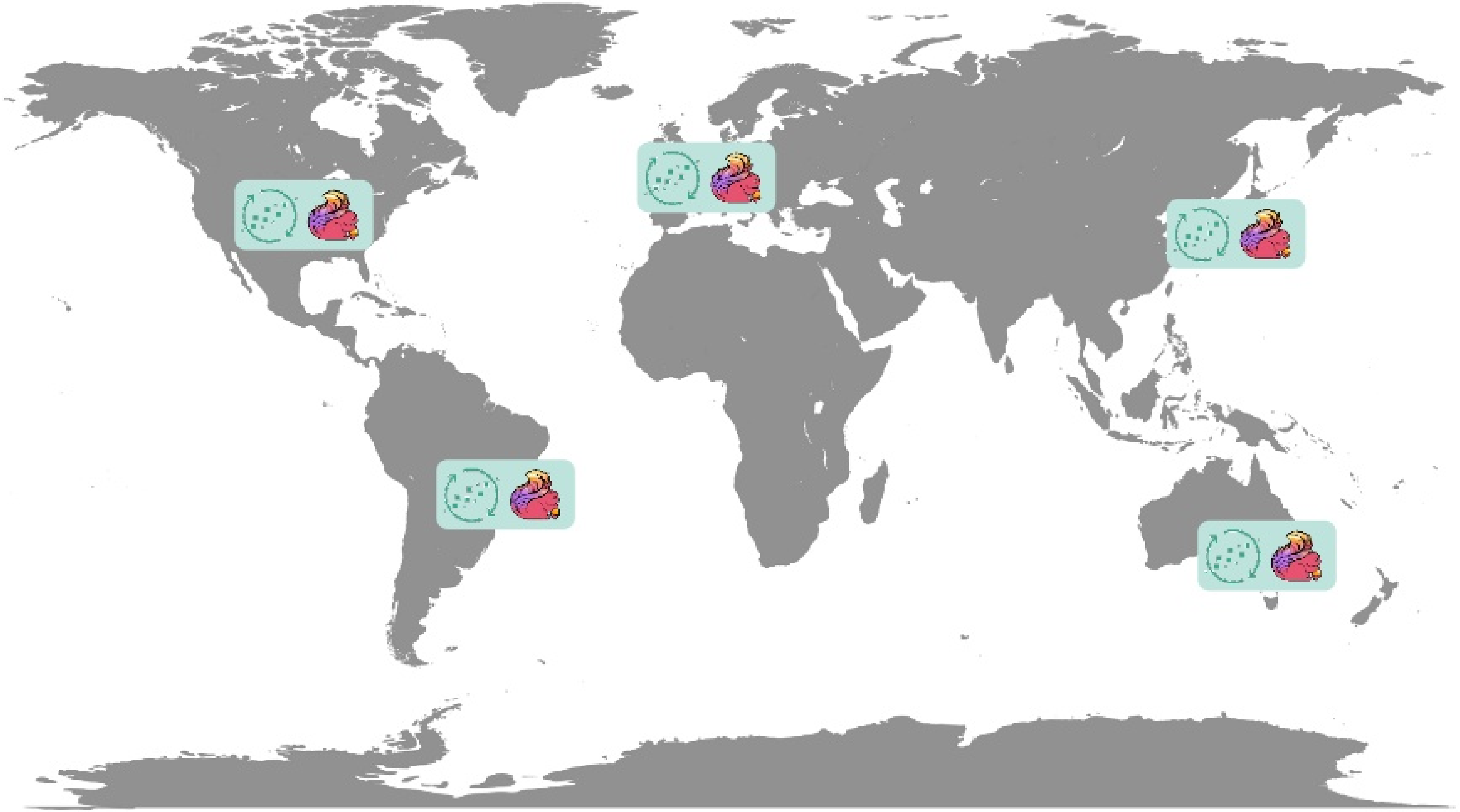
4 x 200,000
= 800,000 updates/sec
on the same 1,000 keys

Slowdown, but does not break like optimistic concurrency approaches

Running a distributed setup

... and because we can...

...we run a globally distributed setup



Apache Flink: The heavy lifter

This technology is possible, because Apache Flink offers such powerful building blocks

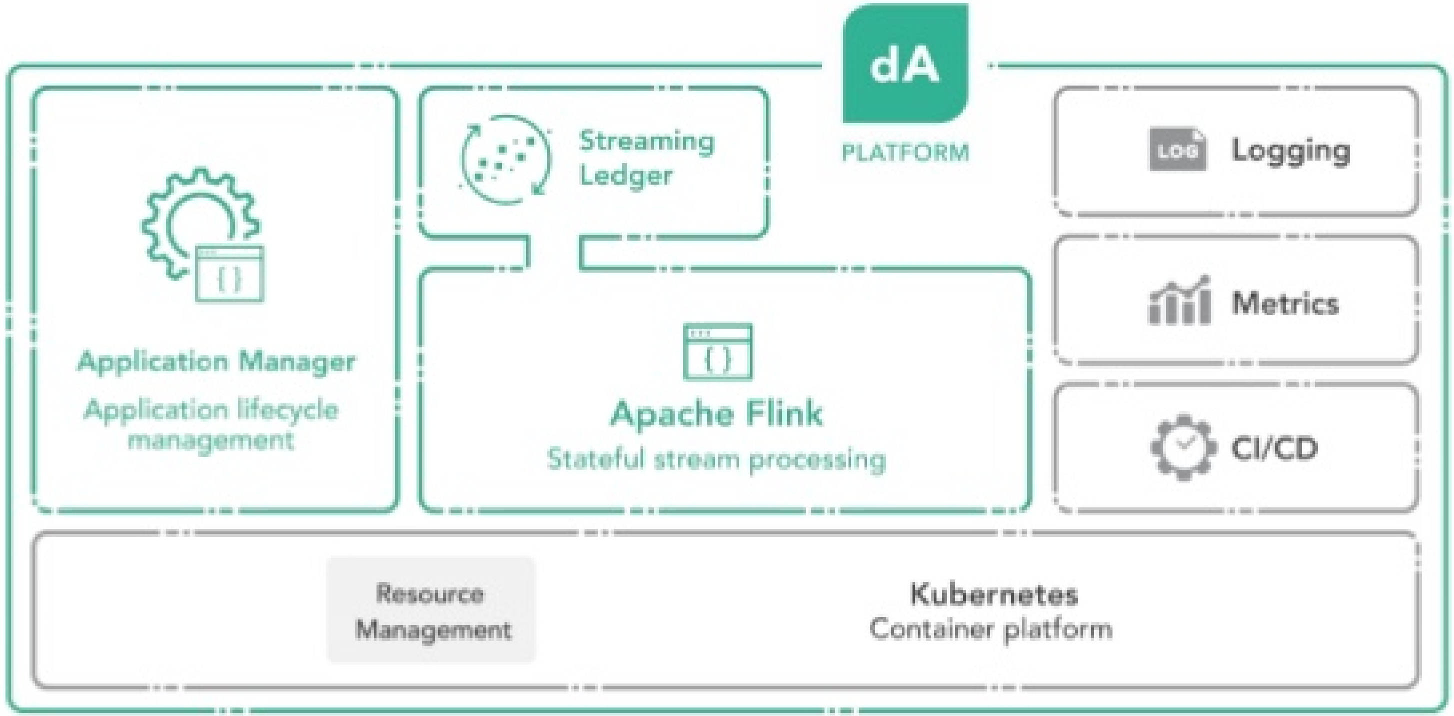
- Continuous processing
- Iterative flows
- Flexible state abstraction
- Asynchronous checkpoints
- Sophisticated event-time/watermarks

Part of data Artisans Platform River Edition

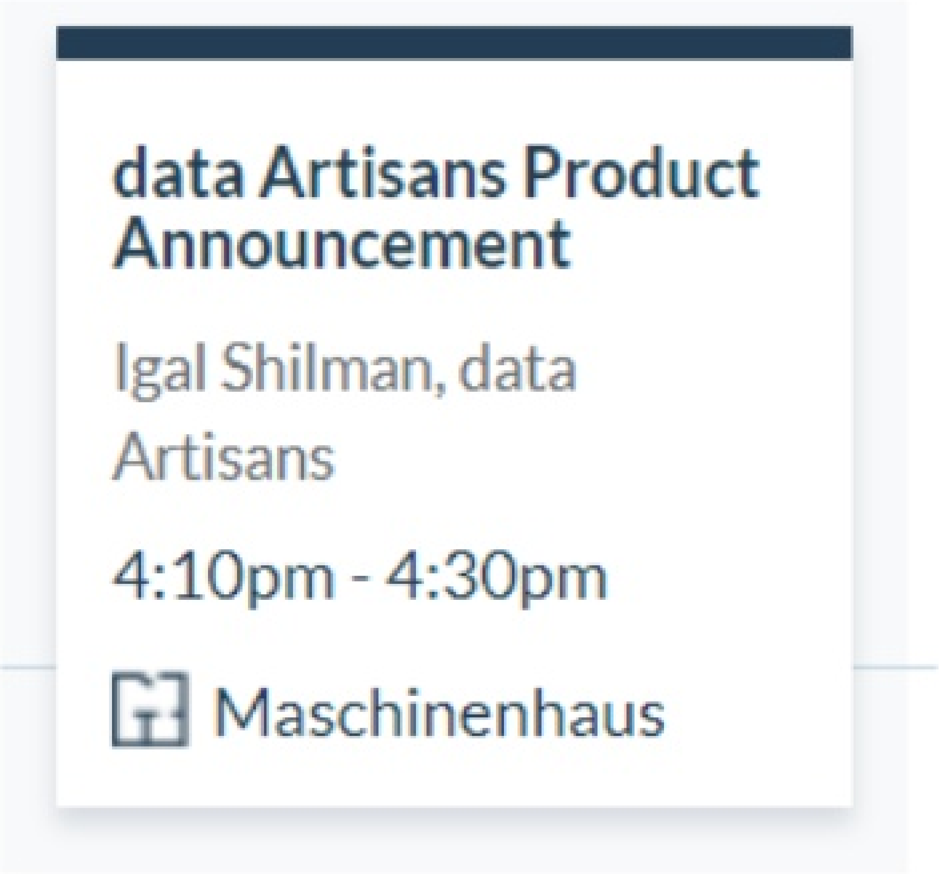
Streaming Ledger is part of data Artisans Platform

- **Stream Edition:** Apache Flink, Application Manager
- **River Edition:** Apache Flink, Application Manager, Streaming Ledger

API and single-node implementation under Apache 2.0 license



Learn more at



data Artisans Product Announcement

Igal Shilman, data Artisans

4:10pm - 4:30pm

Maschinenhaus

At the booth

dataArtisans

Thank you!
Enjoy the conference!

dataArtisans