Stream Processing in Java

Presented by: Vladimír Schreiner <vladimir@hazelcast.com>
@voloda

Housekeeping

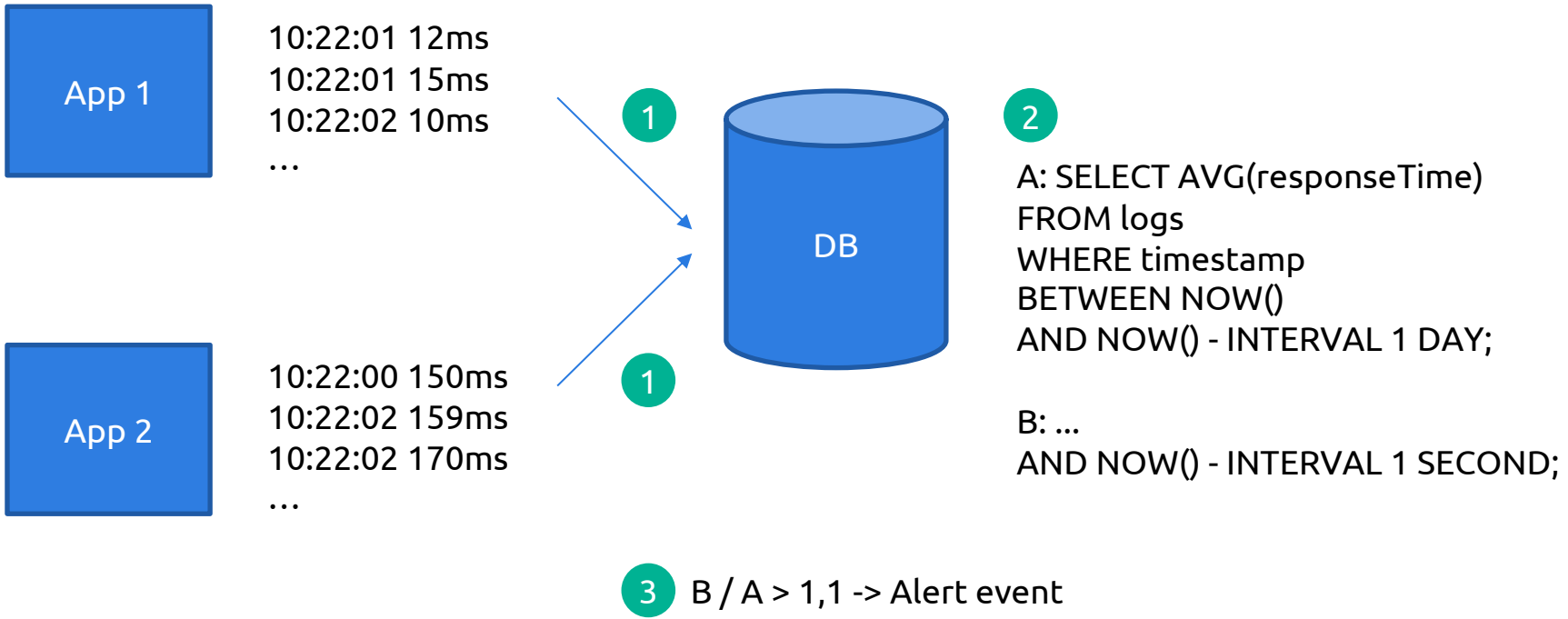
hazelcast

Agenda

- Modern DBs x Streaming
- Concern #1 - Time
- Concern #2 - Connectors
- Concern #3 – Scaling?

hazelcast

System Health Monitoring



hazelcast

DB tuned for append-only tables

- Stream: ordered sequence of immutable records
  - Log records, clicks, IoT readings, business events, …
- Stream Processing: querying the stream continuously
  - Continuously transform, join, aggregate, …

hazelcast

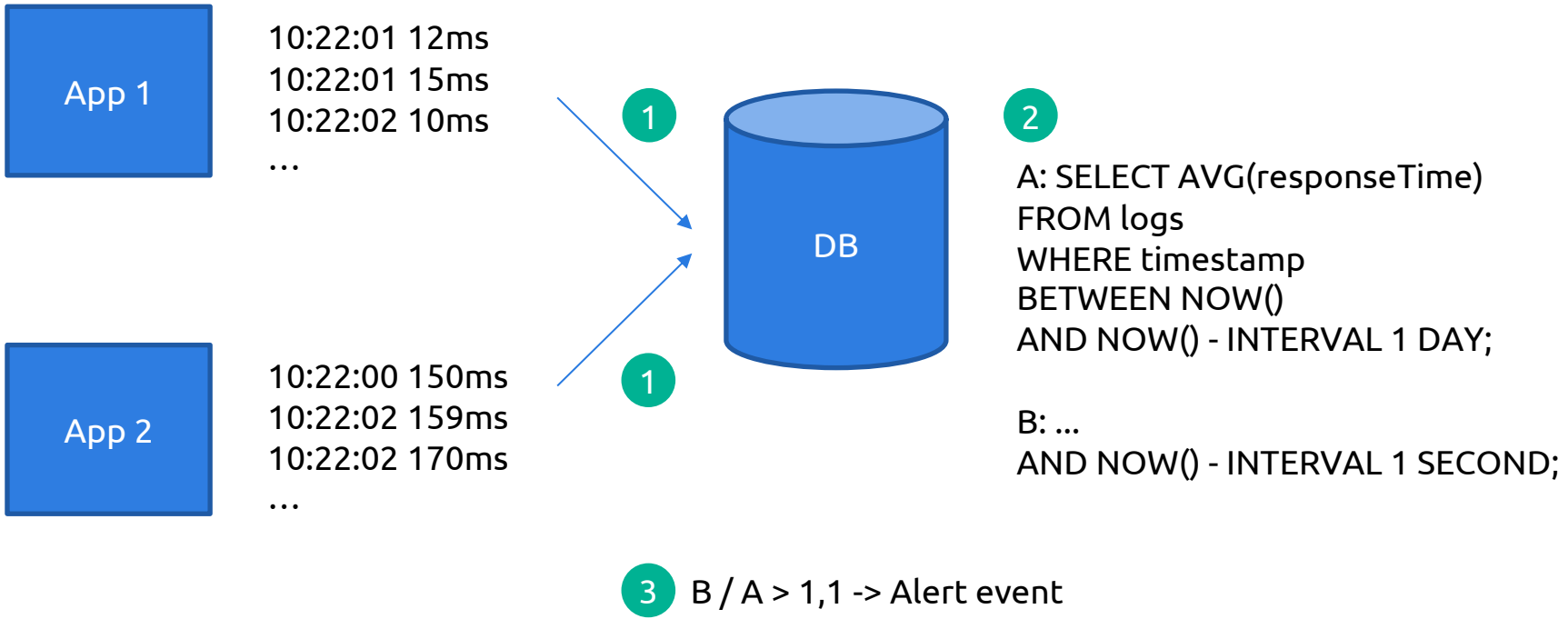
Focal areas of Streaming

1. Time as a first class citizen
2. Connectivity
3. Scale

hazelcast

Time

When should we get the results?



hazelcast

Stream Processing is ”push”

- Database allows clients to **pull** data by querying it’s state
- Streaming Engine runs a continuous query and **pushes** updates to consumers

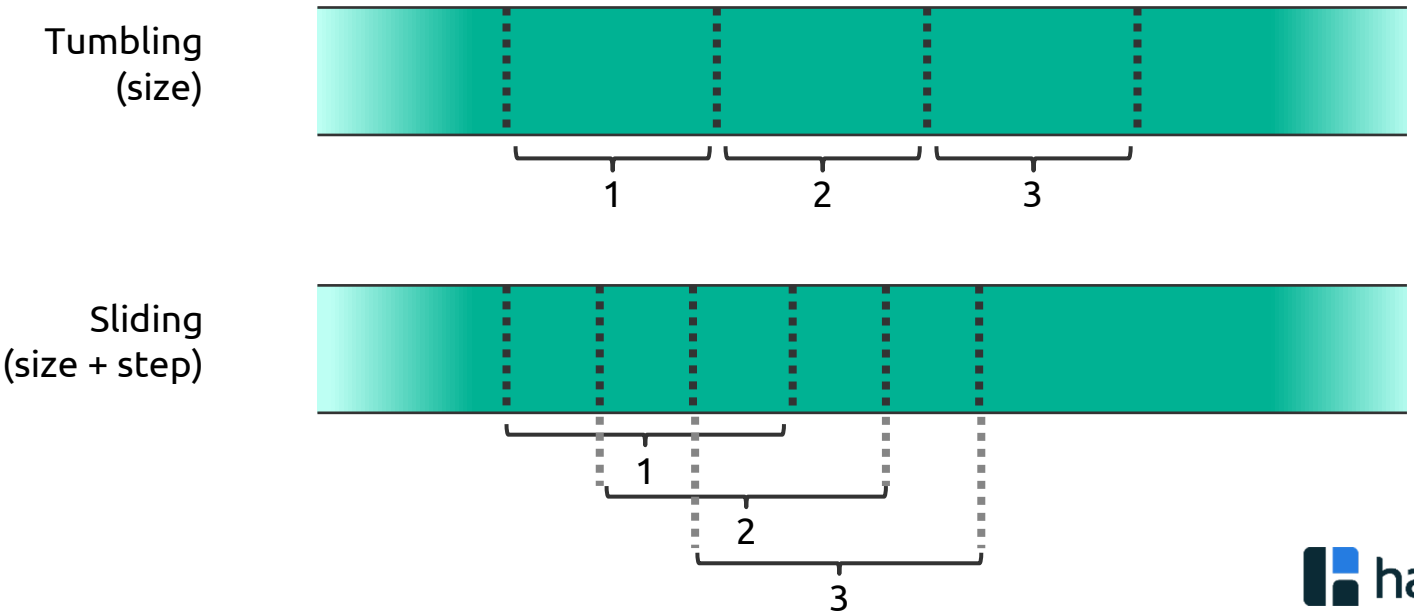
- Continuous programming model decouples query submission from result materialization!
  - Computations driven by input data, not by query submission
  - Low latency and correct results!

hazelcast

Windows to define data ranges

Windows define the ranges in the append only table

Engine runs the query as soon as it has complete data for the window



hazelcast

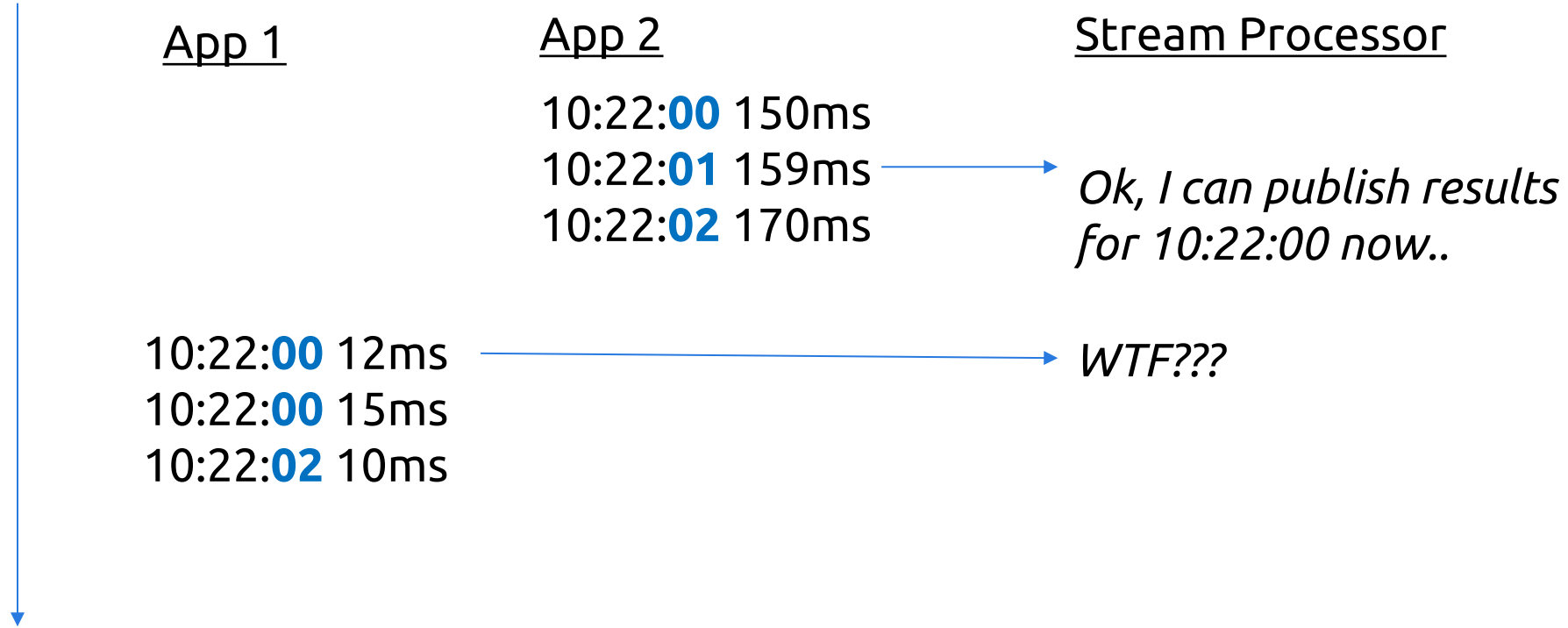
New API to define windows

```
SELECT AVG(resoponseTime)
FROM Logs
GROUP BY SLIDING(timestamp,
         INTERVAL '1‘ SECOND, INTERVAL '1' DAY)
```

Results are published after the engine recieved and processed all input data for the **time window**.

hazelcast

When the window completes?



hazelcast

Strategies for unordered data

- Waiting for stragglers

  ”Maybe somebody is late, let’s waiting another ___ before publishing results”

- Publish early results

  “Based on the data seen so far, the result is ___ ”

hazelcast

Time - Summary

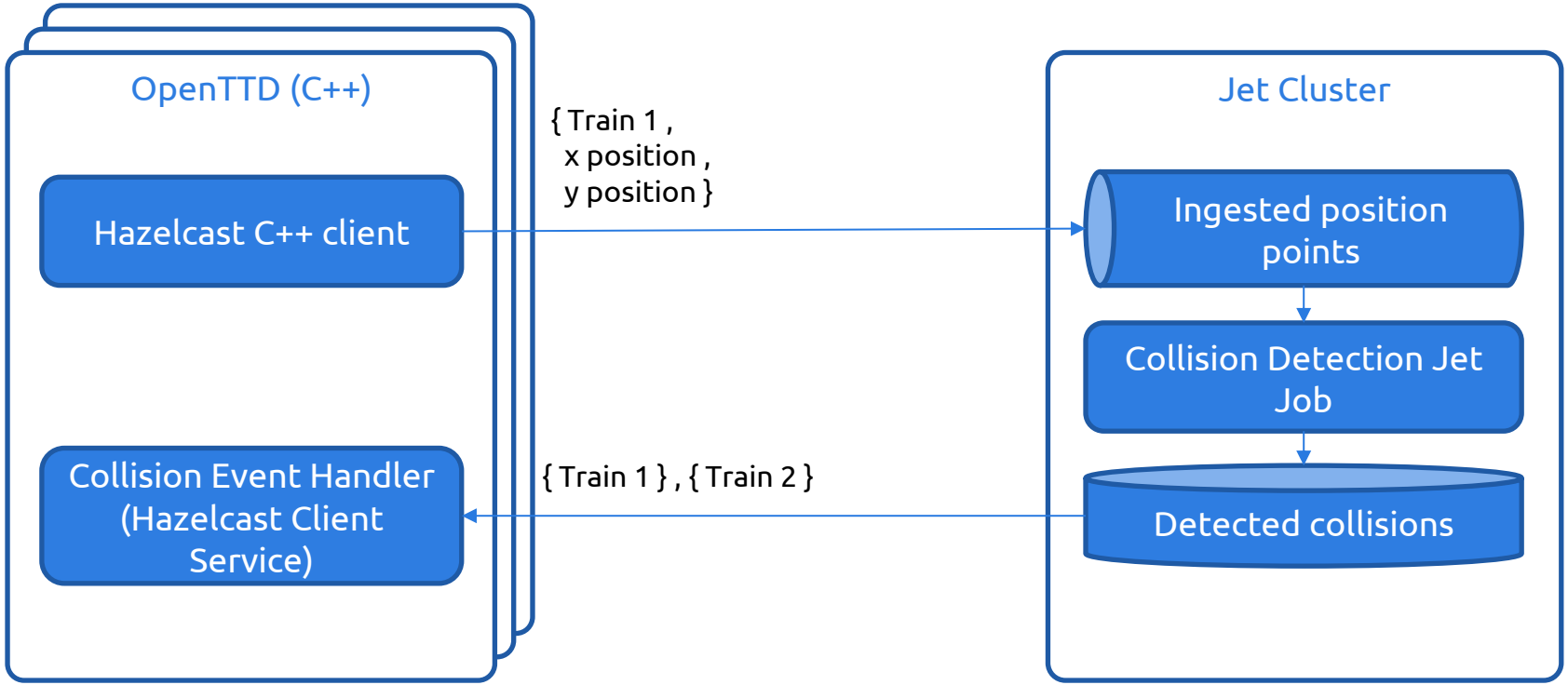
- Streaming queries run continuously and push results to subscribers.
  - **Event-driven querying for lower latency**
- Declarative API to for data driven "triggers”
  - Windowing - which data are required for the computation
  - When to publish the results

hazelcast

Use Case: Analytics and Decision Making

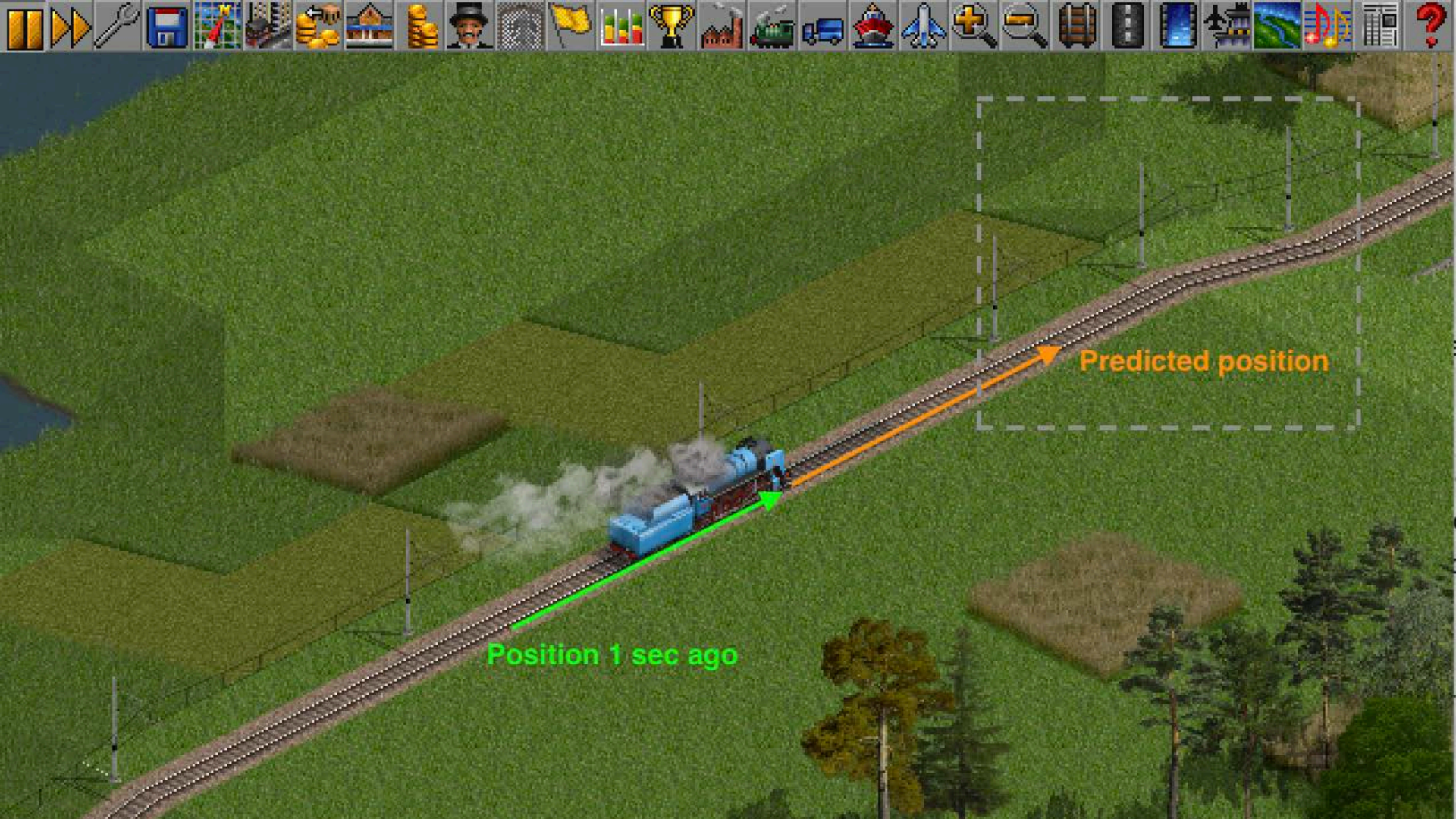
- Real-time dashboards
- Stats (gaming, infrastructure monitoring)
- Decision making
- Recommendations
- Prediction - often based on algorithmic prediction (push stream through ML model)
- Complex Event Processing
- Moving average

Train Demo!



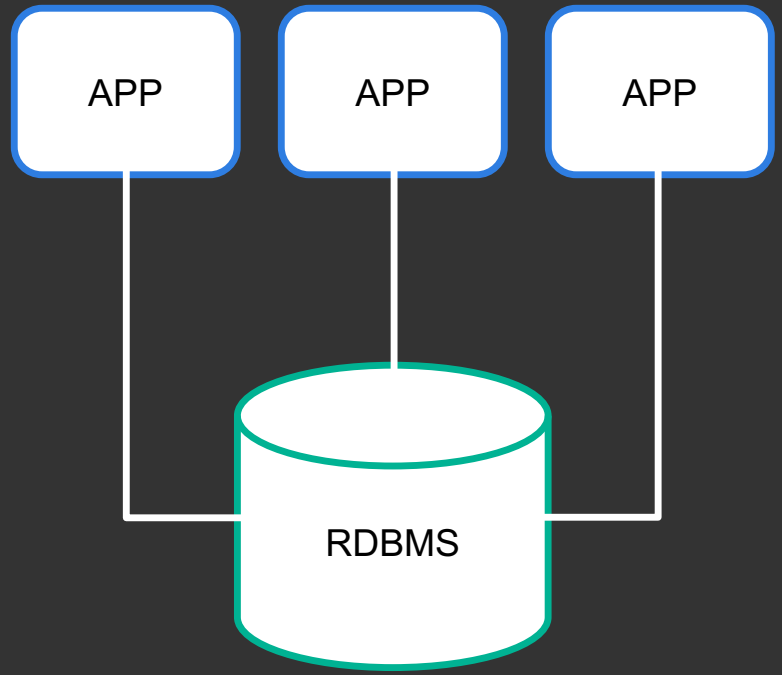
https://github.com/vladoschreiner/transport-tycoon-demo

hazelcast


Predicted position
Position 1 sec ago

Connectors

Typical Legacy Architecture



One needs stream of data for streaming!

- Stream is a sequence of immutable events,
  the append-only table

- To enable streaming, we need applications that:
  - Produces events
  - Deliver events to a streaming engine
  - E.g.: using client (agent), publish changes to a message broker

hazelcast

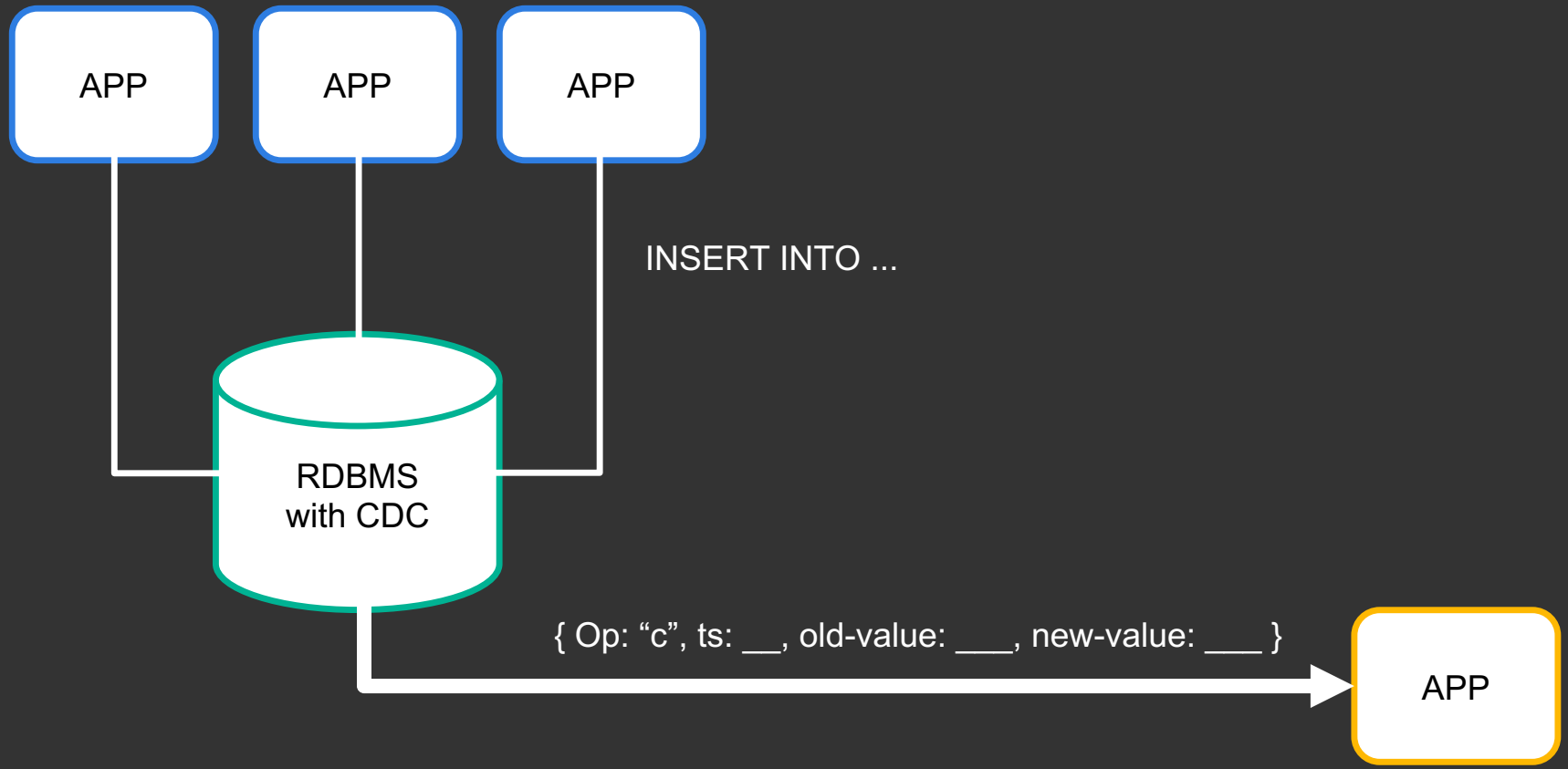
In the meanwhile in a real world

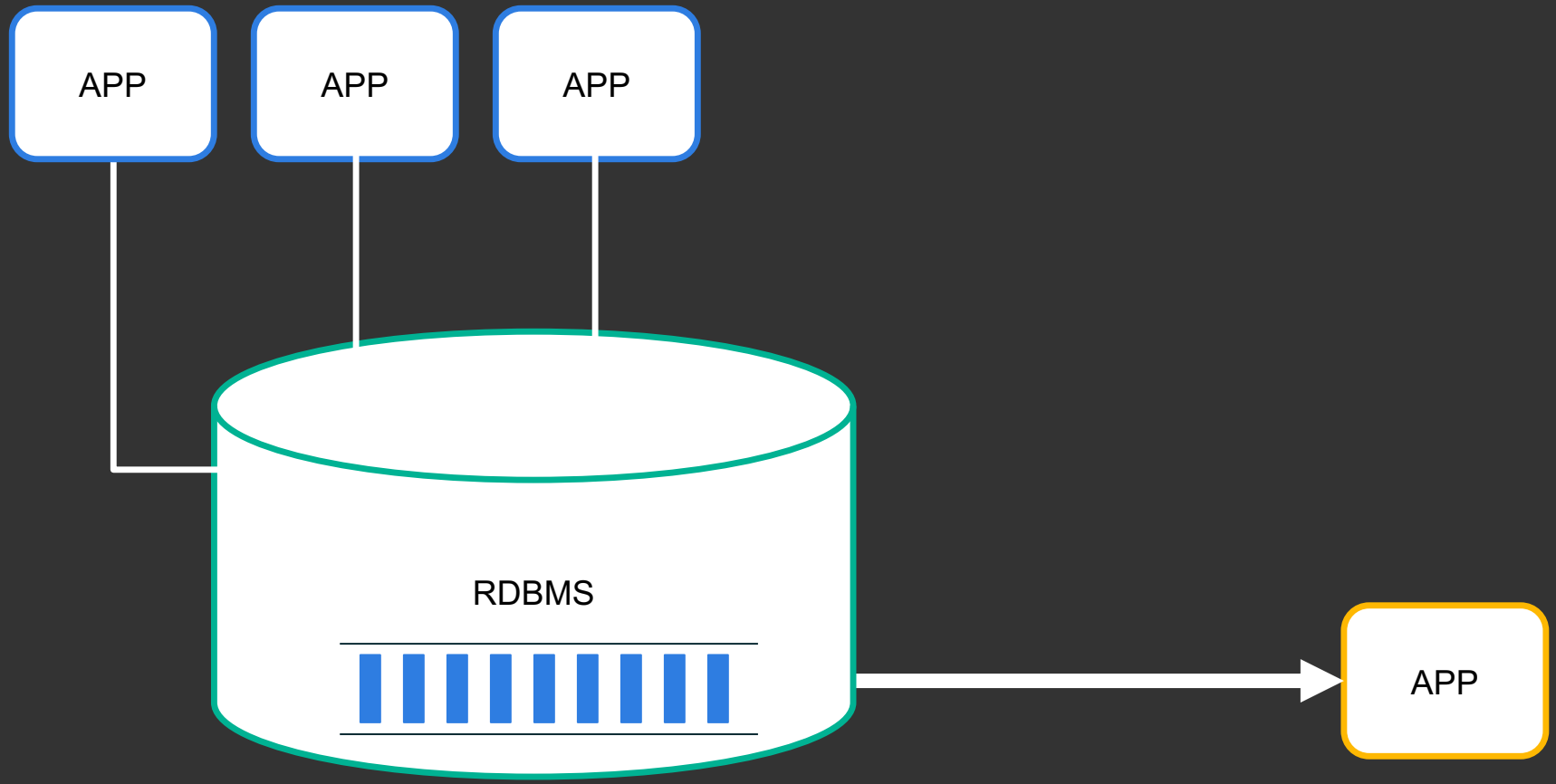
Rank			DBMS	Database Model	Score		
Jan 2020	Dec 2019	Jan 2019			Jan 2020	Dec 2019	Jan 2019
1.	1.	1.	Oracle	Relational, Multi-model	1346.68	+0.29	+77.85
2.	2.	2.	MySQL	Relational, Multi-model	1274.65	-1.01	+120.39
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1098.55	+2.35	+58.29
4.	4.	4.	PostgreSQL	Relational, Multi-model	507.19	+3.82	+41.08
5.	5.	5.	MongoDB	Document, Multi-model	426.97	+5.85	+39.78
6.	6.	6.	IBM Db2	Relational, Multi-model	168.70	-2.65	-11.15
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	151.44	+1.19	+8.00
8.	8.	↓ 7.	Redis	Key-value, Multi-model	148.75	+2.51	-0.27
9.	9.	9.	Microsoft Access	Relational	128.58	-0.89	-13.04
10.	↑ 11.	10.	SQLite	Relational	122.14	+1.78	-4.66

hazelcast

IMPEDANCE MISMATCH

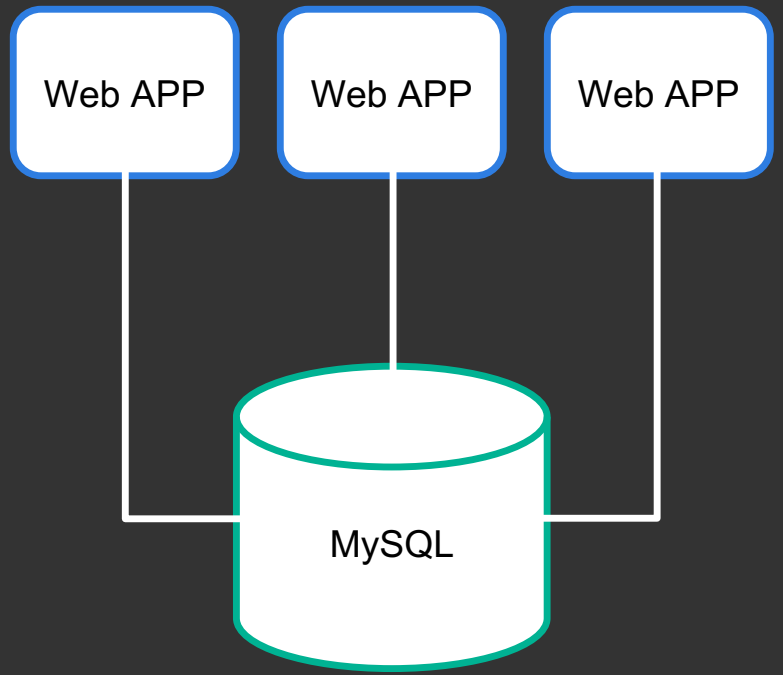
Change Data Capture (CDC) turns legacy database to a source of event stream.


APP
APP
APP
INSERT INTO ...
RDBMS
with CDC
{ Op: “c”, ts: __, old-value: ___, new-value: ___ }
APP

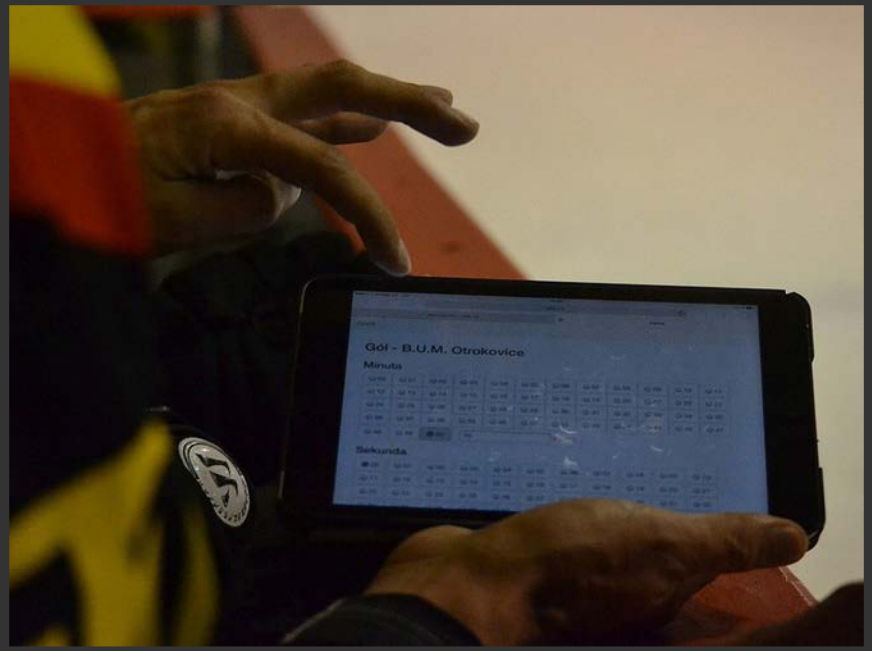

APP
APP
APP
RDBMS
APP

Travelling Back in Time

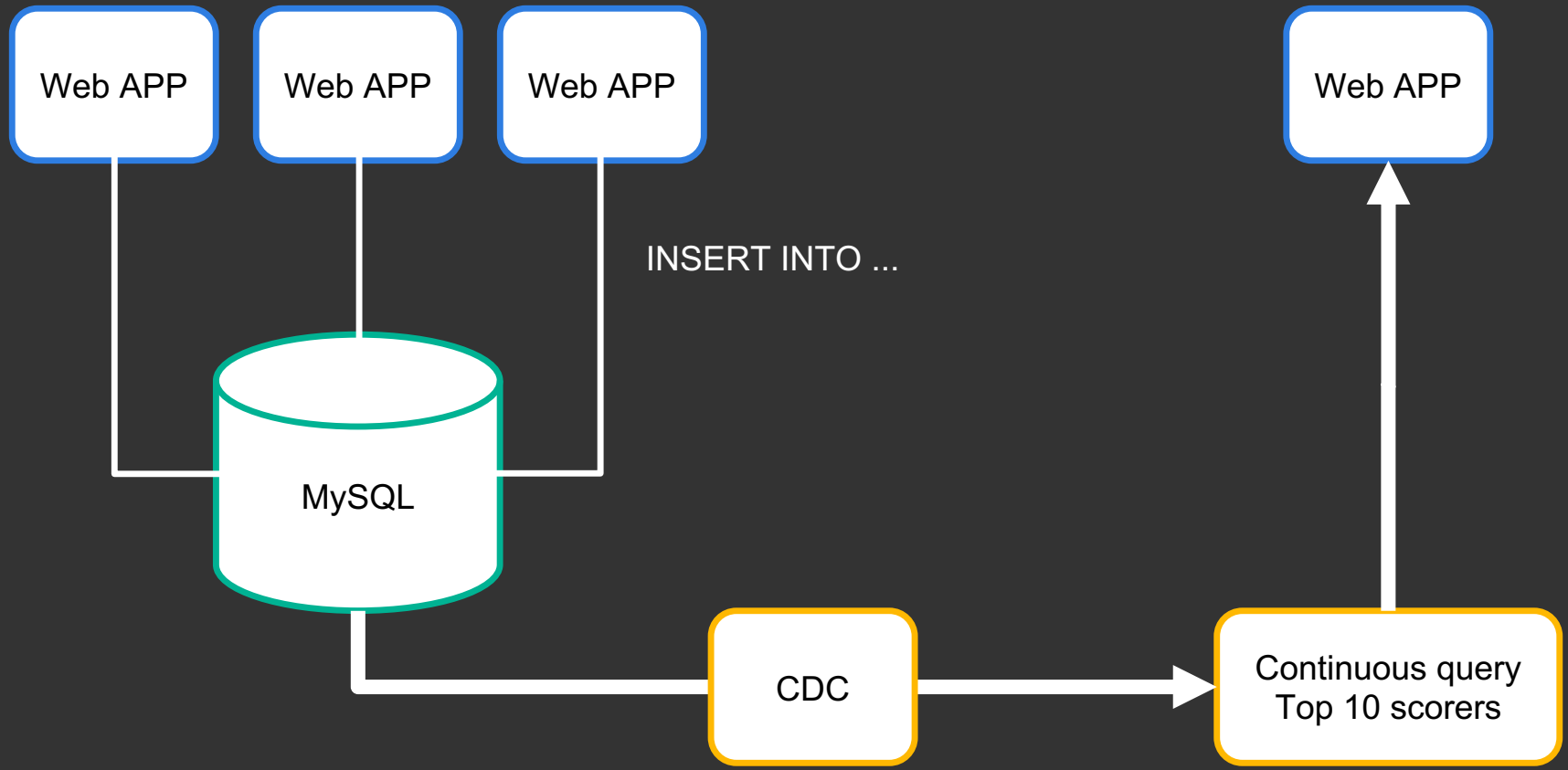
https://github.com/hazelcast/hazelcast-jet-demos/tree/master/debezium-cdc-without-kafka



LAMP stack
GBs of data
Mostly OLTP, OLAP after hours



Real-time updates = Real-time expectations

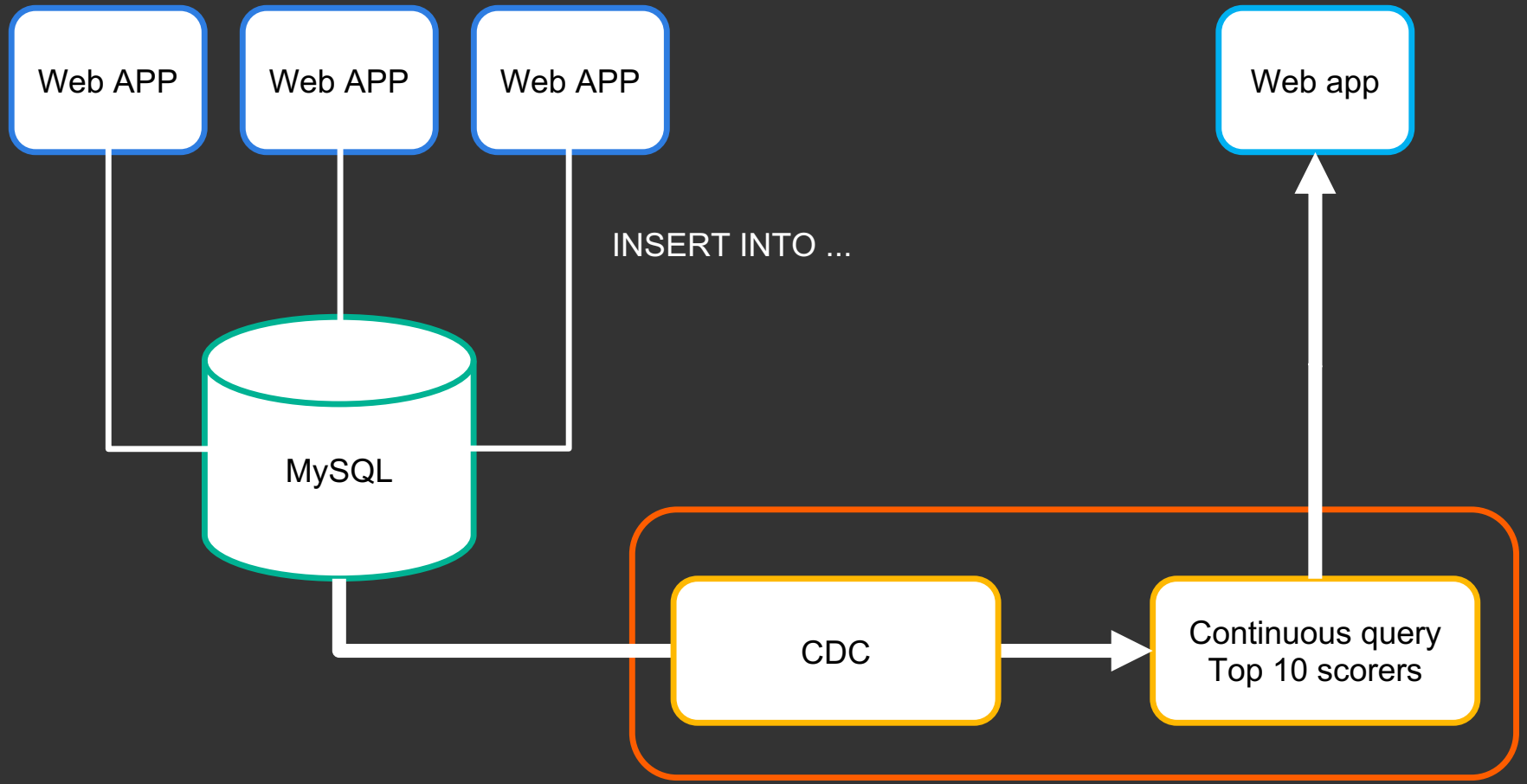

Web APP
Web APP
Web APP
INSERT INTO ...
MySQL
CDC
Continuous query
Top 10 scorers
Web APP

Hazelcast Jet

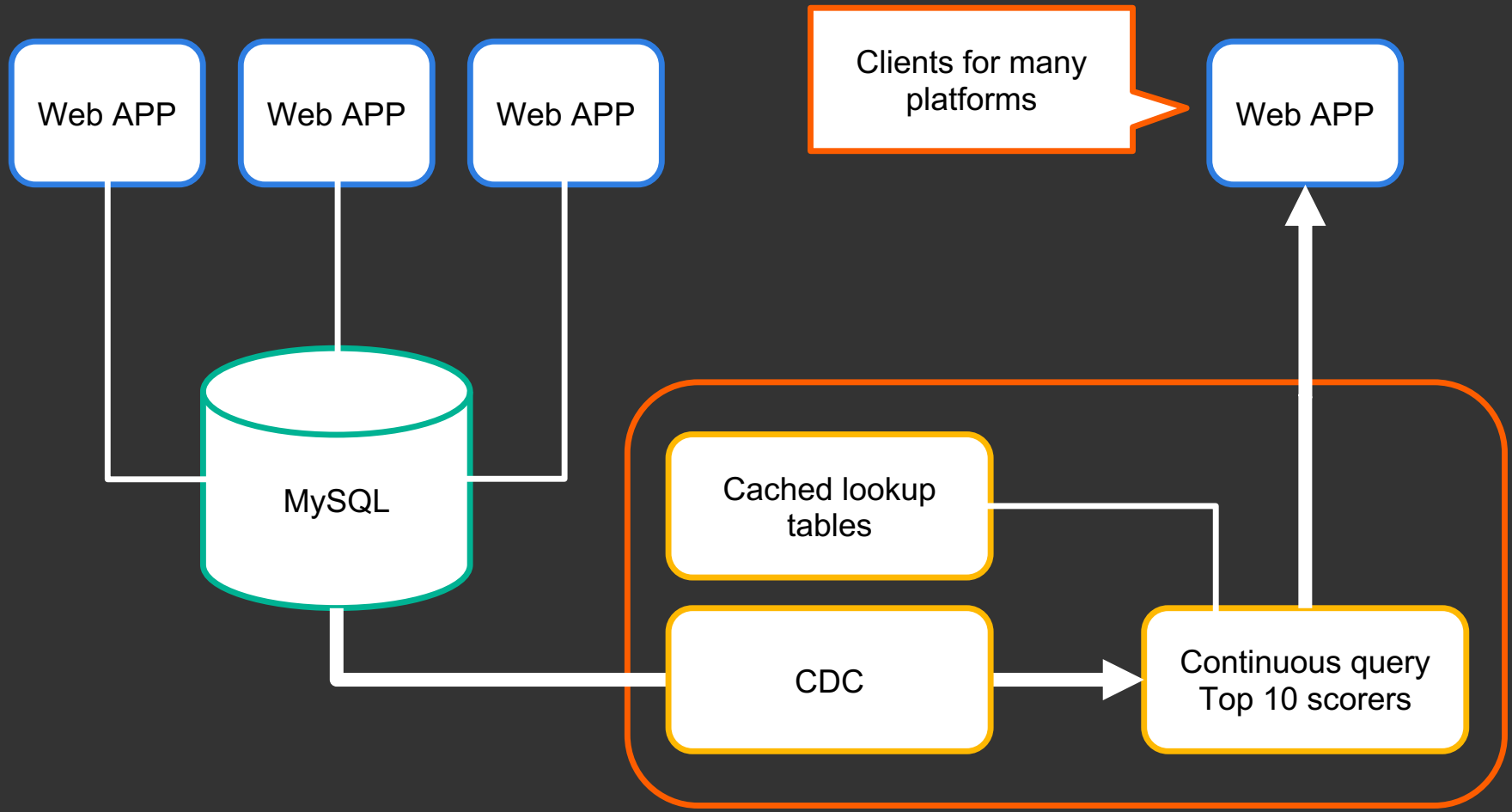
Open-source library with stream processor, connectors including CDC and a key-value store.

Single Java Binary
Elastic Clustering
No ZooKeeper, HDFS..
Just Java 8 and above

Hazelcast Jet runs Debezium for CDC


Web APP
Web APP
Web APP
Web app
INSERT INTO ...
MySQL
CDC
Continuous query
Top 10 scorers

```
# The following can be used as easy to replay
# backup logs or for replication.
server-id               = 1
log_bin                 = /var/log/mysql/mysql-bin.log
binlog_format     = row
binlog_row_image  = full
expire_logs_days        = 2
```

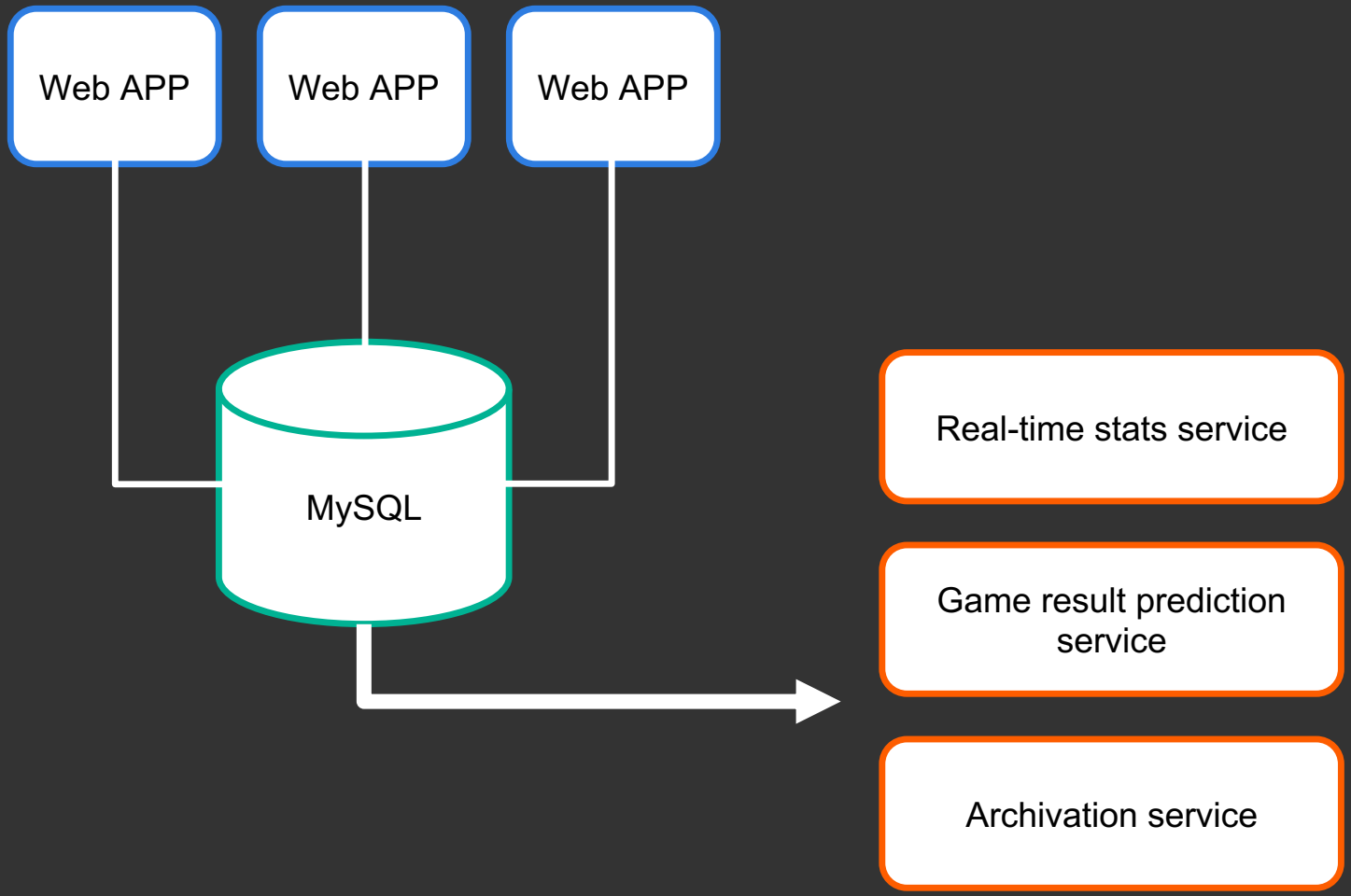

Web APP
Web APP
Web APP
Clients for many platforms
Web APP
MySQL
Cached lookup tables
CDC
Continuous query Top 10 scorers

Relational databases usually keep shorter history, compared to dedicated log-based storages.

CDC + SPE + CACHE =

Materialized Views Offloaded from the DB

Modularization Microservices


Web APP
Web APP
Web APP
MySQL
Real-time stats service
Game result prediction service
Archivation service

Connectors - summary

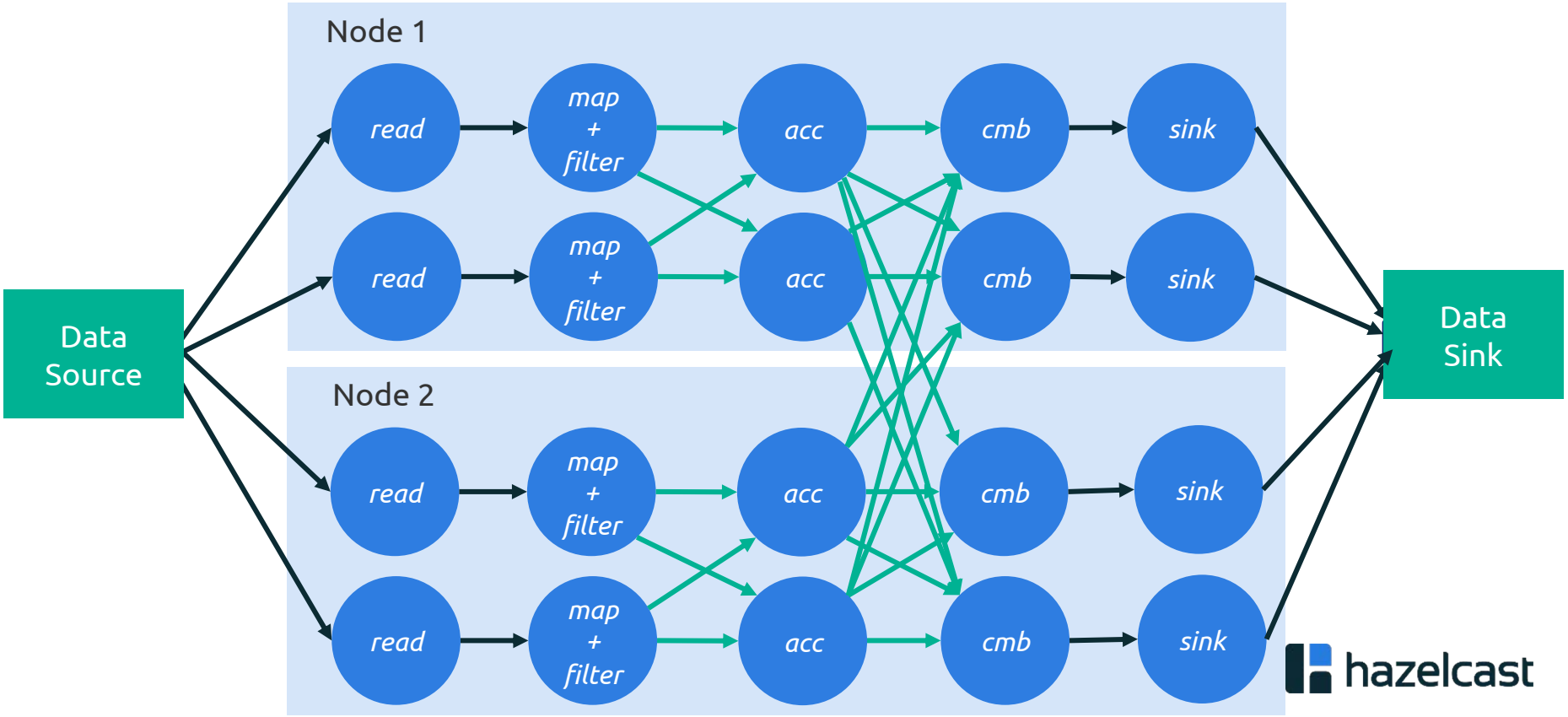
- Stream processing requires input data presented as streams - sequences of immutable records (”append-only table”)
- Streaming frameworks come with clients, agents and connectors (messaging, Kafka, …)
- CDC - a streaming API to a database.
  - Extracts database changes
  - No standard, many CDC vendors.

- Consider Jet for materialized views in Java

hazelcast

Scaling

Modern SPEs are build to scale



Do I need a streaming cluster?

- A single node can handle 1 million events / s
- Fault-tolerance for instant failover
- Elasticity for performance spikes

hazelcast

Fault Tolerance Using Replication

- State of the computation replicated across the cluster
- The tasks of the failed member recovered on other members using the backup replicas
- Regular snapshots for a light-weigh F-T
  - Regularly snapshot cluster state and store it reliably
  - Restart computation from last snapshot if it fails
  - Replay a short history of the stream

hazelcast

Summary

- Streaming is database ”optimised for append-only tables”
- Main UC: Event-driven querying
- Connect streaming to legacy applications using CDC
- Create materialized views outside the database to reduce the database load and modularize your architecture.

https://jet.hazelcast.org/

https://hazelcast.com/blog/how-hazelcast-jet-compares-to-apache-spark/

hazelcast


Thank You
hazelcast