Processing over a trillion events a day

CASE STUDIES IN SCALING STREAM PROCESSING AT LINKEDIN

Processing over a trillion events a day

CASE STUDIES IN SCALING STREAM PROCESSING AT LINKEDIN

Jagadish Venkatraman

Sr. Software Engineer, LinkedIn

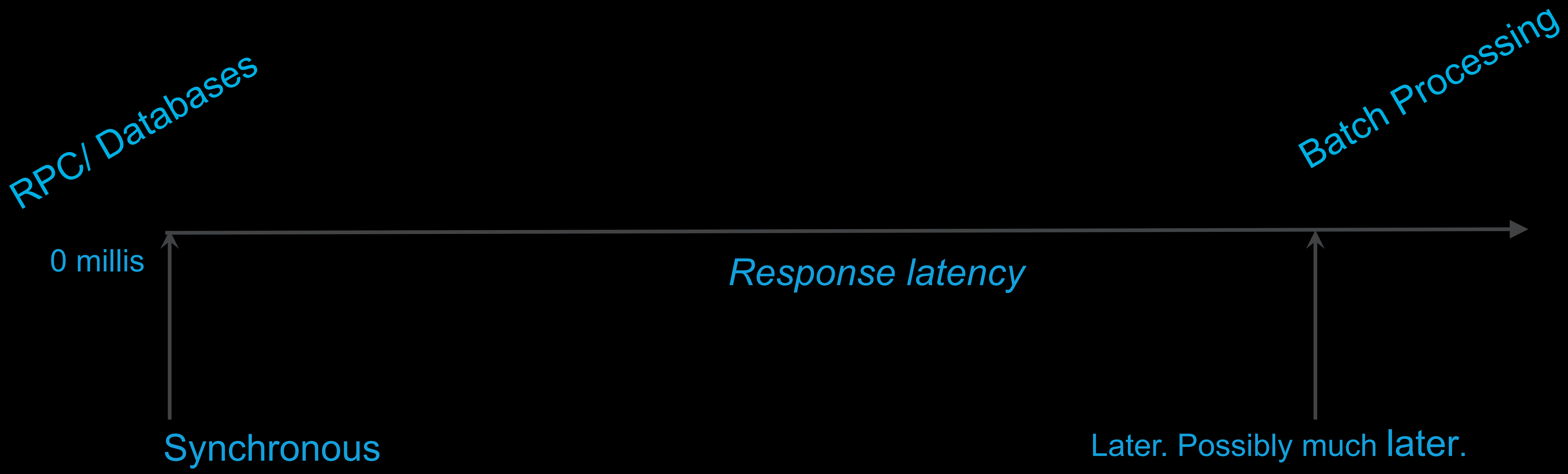
Apache Samza committer

jagadish@apache.org

Today's Agenda

1. Stream processing scenarios
2. Hard problems in stream processing
3. Case Study 1: LinkedIn's communications platform
4. Case Study 2: Activity tracking in the News feed

Data processing latencies



RPC/ Databases

Batch Processing

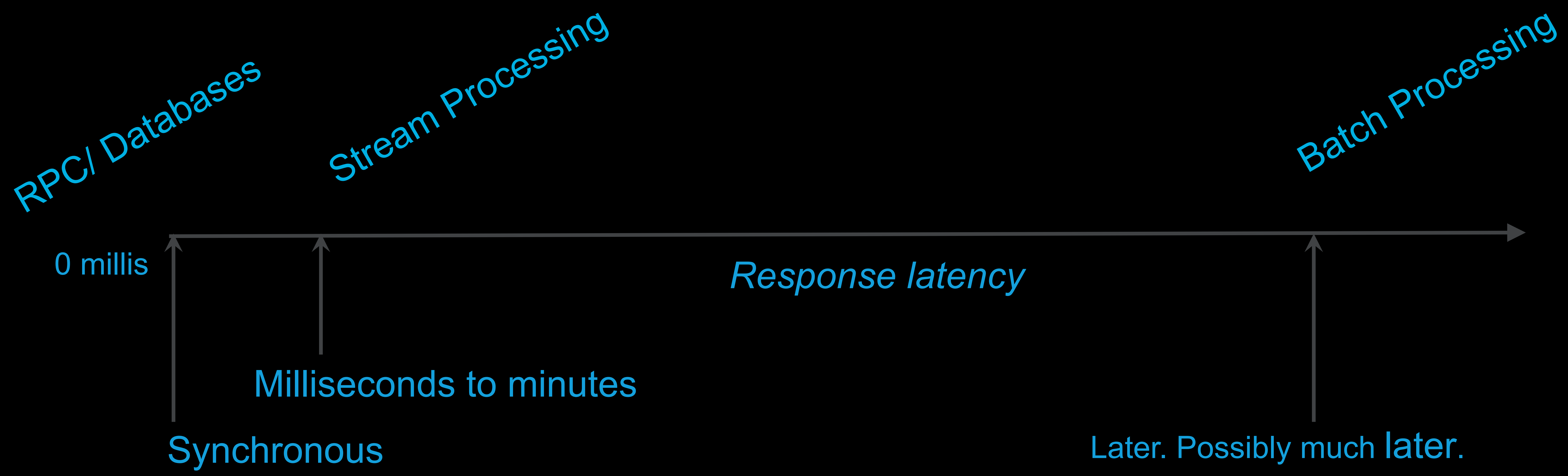
0 millis

Response latency

Synchronous

Later. Possibly much later.

Data processing latencies



Some stream processing scenarios at LinkedIn

Security

Real-time DDoS protection for members

Some stream processing scenarios at LinkedIn

Security

Real-time DDoS protection for members

Notifications

Notifications to members

Some stream processing scenarios at LinkedIn

Security

Real-time DDoS protection for members

Notifications

Notifications to members

News classification

Real-time topic tagging of articles

Some stream processing scenarios at LinkedIn

Security

Real-time DDoS protection for members

Notifications

Notifications to members

News classification

Real-time topic tagging of articles

Performance monitoring

Real-time site-speed profiling by facets

Some stream processing scenarios at LinkedIn

Security

Real-time DDoS protection for members

Notifications

Notifications to members

News classification

Real-time topic tagging of articles

Performance monitoring

Real-time site-speed profiling by facets

Call-graph computation

Analysis of service calls

Some stream processing scenarios at LinkedIn

Ad relevance

Tracking ads that were clicked

Some stream processing scenarios at LinkedIn

Ad relevance

Tracking ads that were clicked

Aggregates

Aggregated, real-time counts by dimensions

Some stream processing scenarios at LinkedIn

Ad relevance

Tracking ads that were clicked

Aggregates

Aggregated, real-time counts by dimensions

View-port tracking

Tracking session duration

Some stream processing scenarios at LinkedIn

Ad relevance

Tracking ads that were clicked

Aggregates

Aggregated, real-time counts by dimensions

View-port tracking

Tracking session duration

Profile standardization

Standardizing titles, gender, education

....

AND MANY MORE SCENARIOS IN PRODUCTION…

200+

Applications

AND MANY MORE SCENARIOS IN PRODUCTION…

200+

Applications

Powered By samza

Samza overview

- A distributed stream processing framework

- 2012: Development at LinkedIn
- Used at Netflix, Uber, Tripadvisor and several large companies.

- 2013: Apache incubator
- 2015 (Jan): Apache Top Level Project

Today’s agenda

1. Stream processing scenarios
2. Hard problems in stream processing
3. Case Study 1: LinkedIn’s communications platform
4. Case Study 2: Activity tracking in the news feed

Hard problems in stream processing

Scaling processing

- Partitioned streams
- Distribute processing among all workers

State management

- Hardware failures are inevitable
- Efficient check-pointing
- Instant recovery

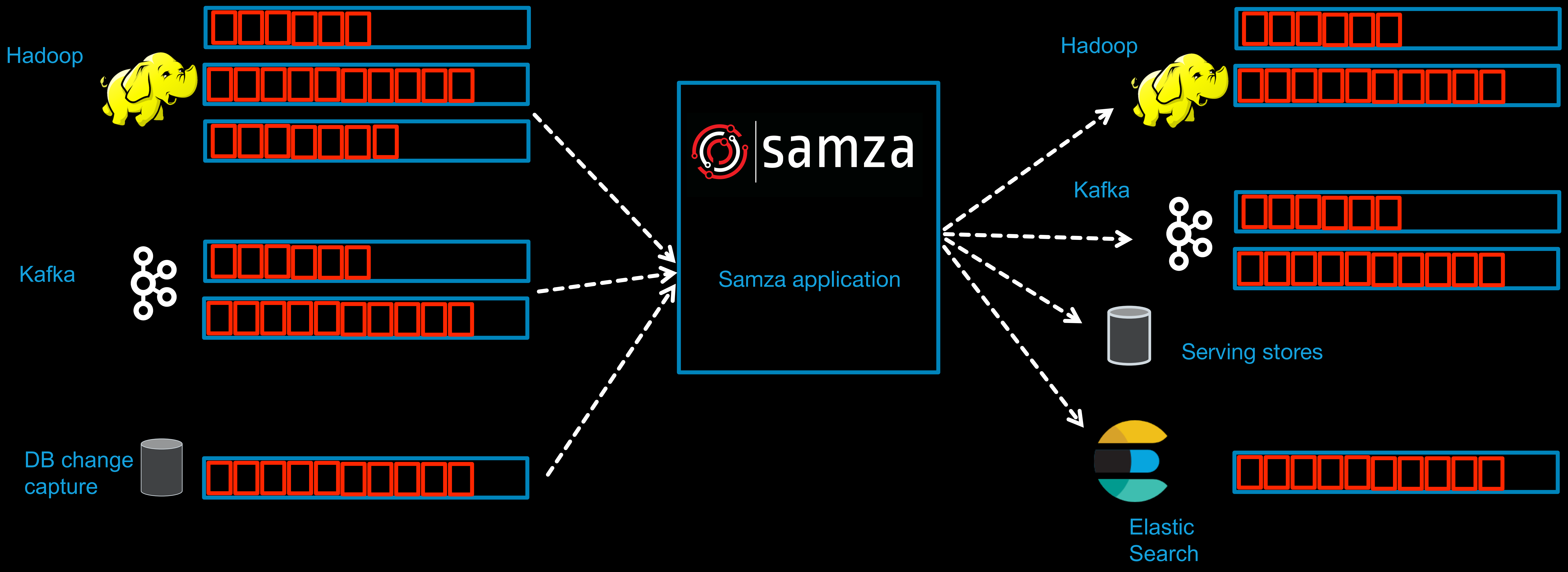
High performance remote I/O

- Need primitives for supporting remote I/O

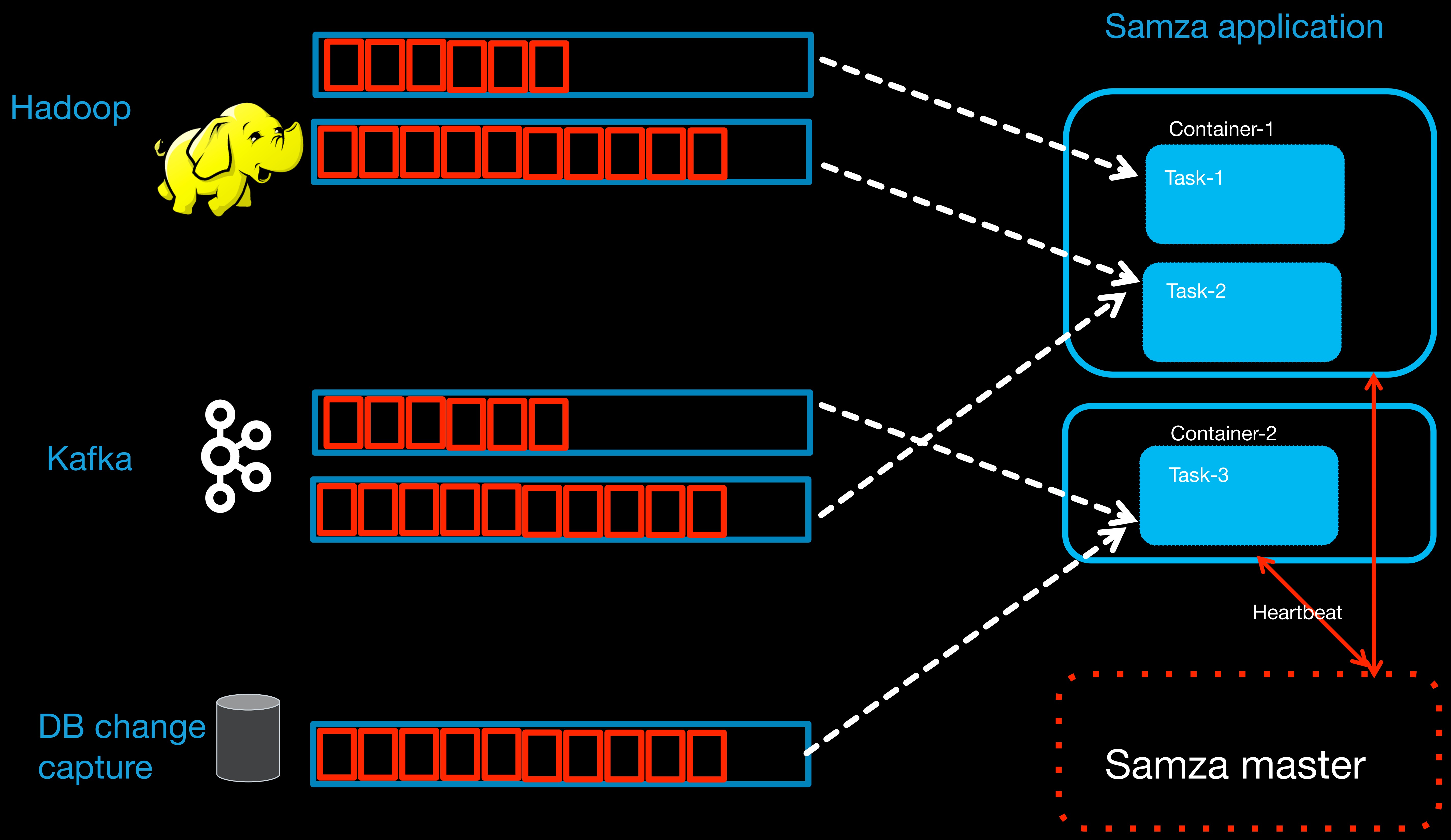
Heterogeneous deployment models

- Running on a multi-tenant cluster
- Running as an embedded library
- Unified API for batch and real-time data

Partitioned processing model



Partitioned processing model



- Each task processes one or more partitions
- The Samza master assigns partitions to tasks and monitors container liveness
- Scaling by increasing # of containers

Hard problems in stream processing

Partitioned processing

- Partitioned streams
- Distribute processing among all workers

State Management

- Hardware failures are inevitable
- Instant recovery
- Efficient Check-pointing

Efficient remote data access

- Provides primitives for supporting efficient remote I/O.

Heterogeneous deployment models

- Samza supports running on a multi-tenant cluster
- Samza can also run as a light-weight embedded library
- Unified API for batch and streaming

Why local state matters?

- Stateless versus Stateful operations

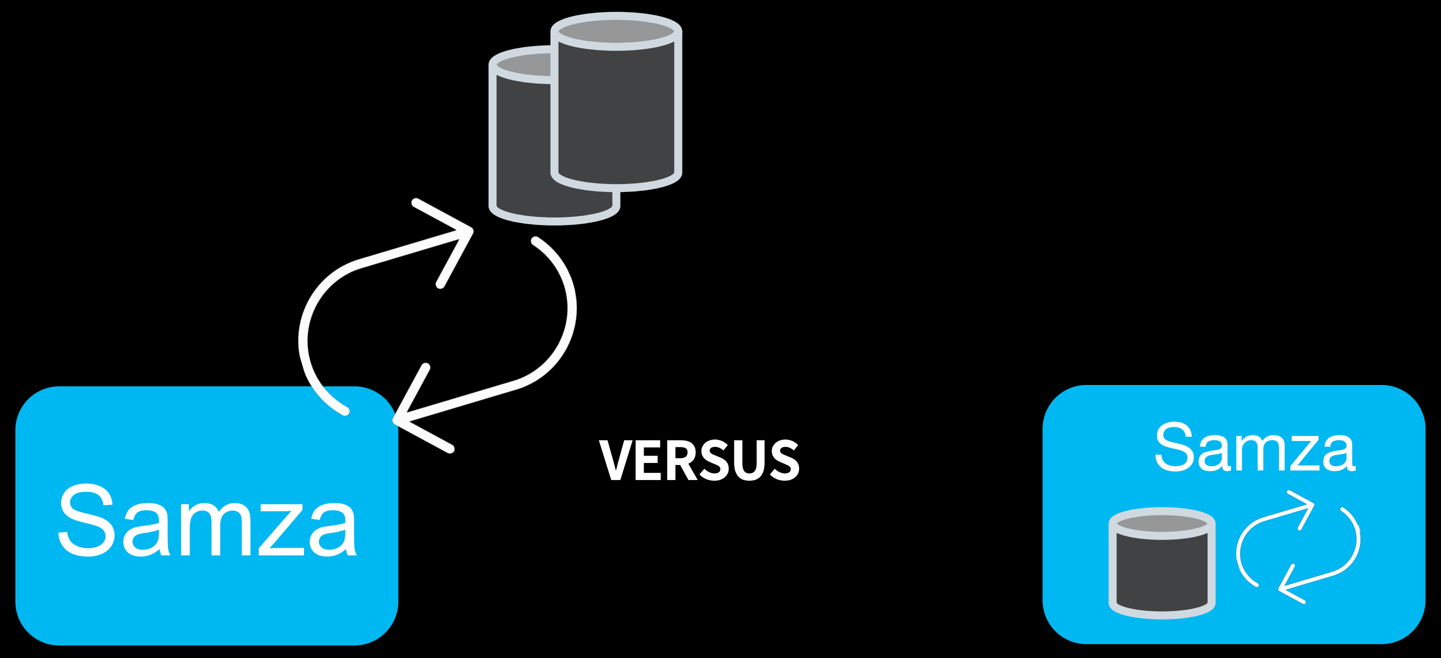
Use cases of Local state

- Temporary data storage
- Adjunct data lookups

Advantages

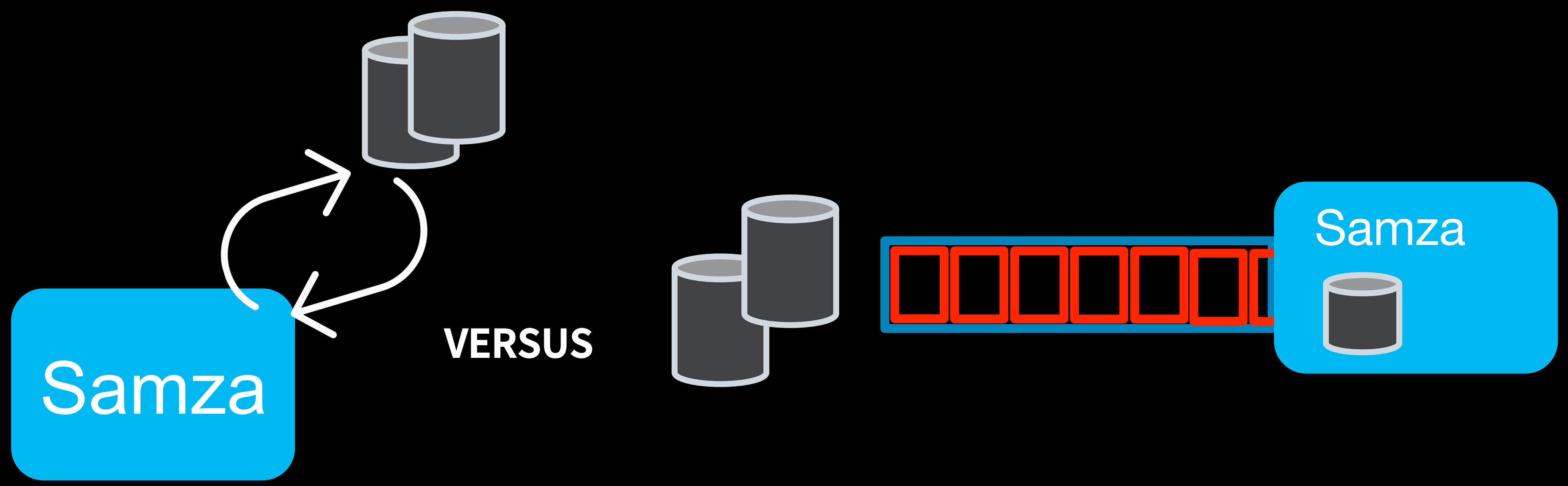
- Richer access patterns, faster than remote state

Architecture for Temporary data



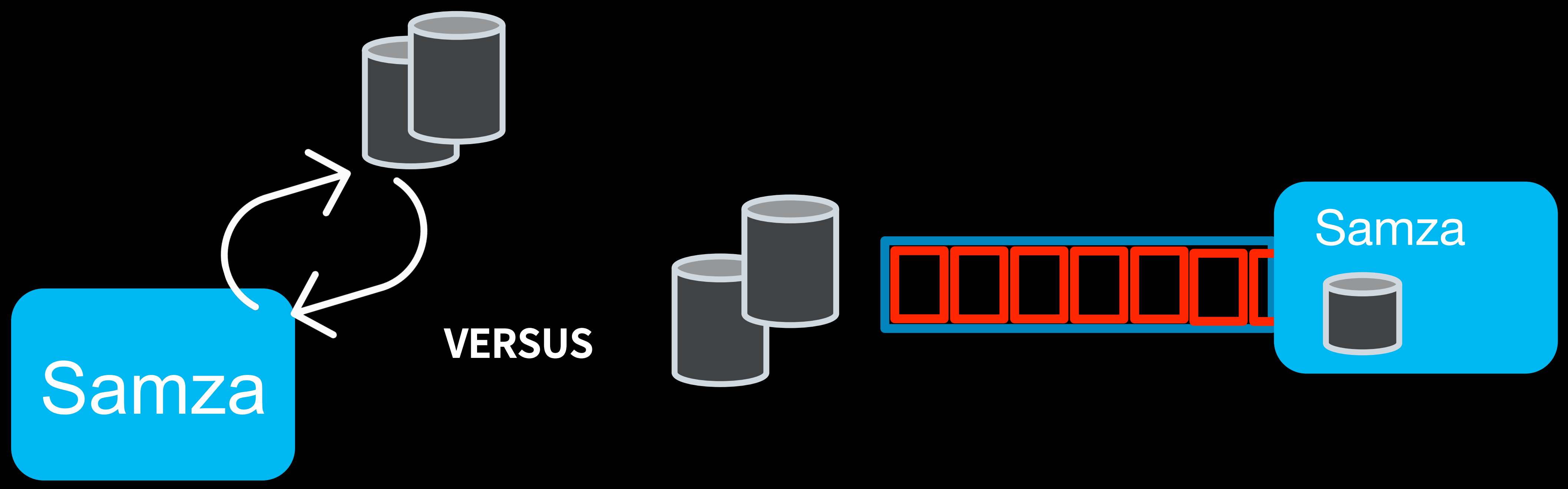
- Querying the remote DB from your app versus local embedded DB access
- Samza provides an embedded fault-tolerant, persistent database.

Architecture for Adjunct lookups



- Querying the remote DB from your app versus change capture from database.
- Turn remote I/O to local I/O
- Bootstrap streams

Architecture for Adjunct lookups

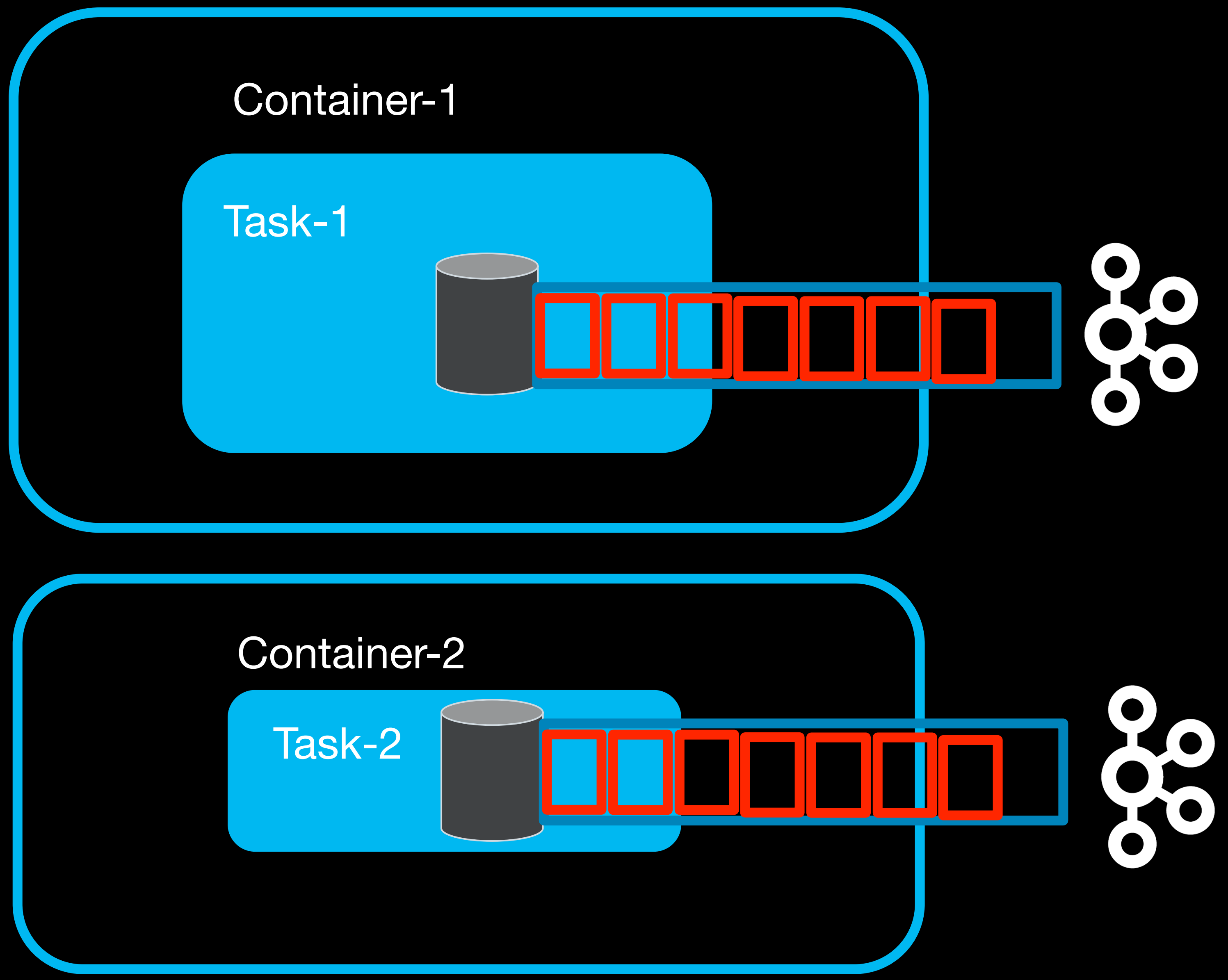


100X
Faster

1.1M
TPS

- Querying the remote DB from your app versus change capture from database.
- Bootstrap and partition the remote DB from the stream

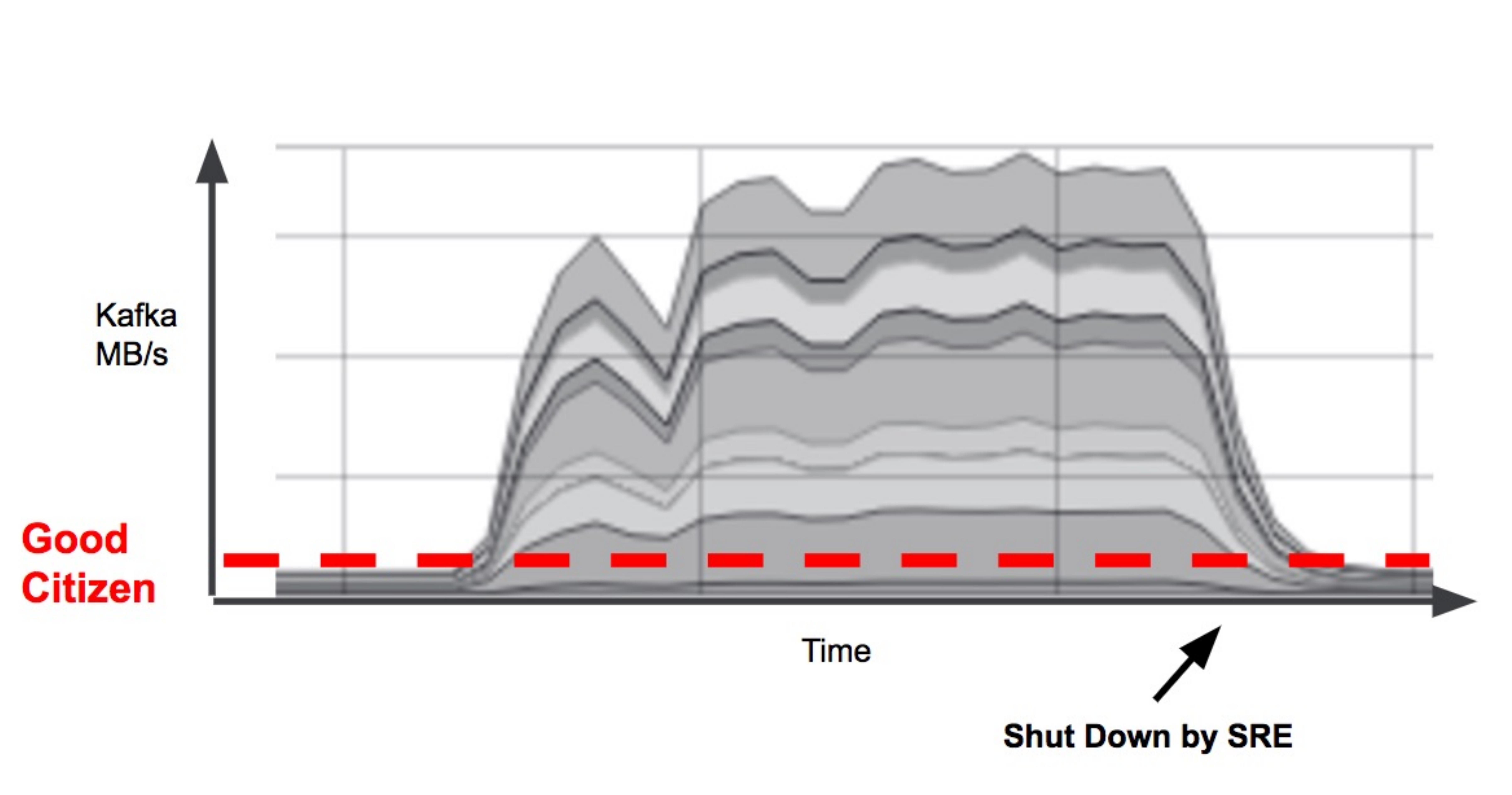
Key optimizations for scaling local state



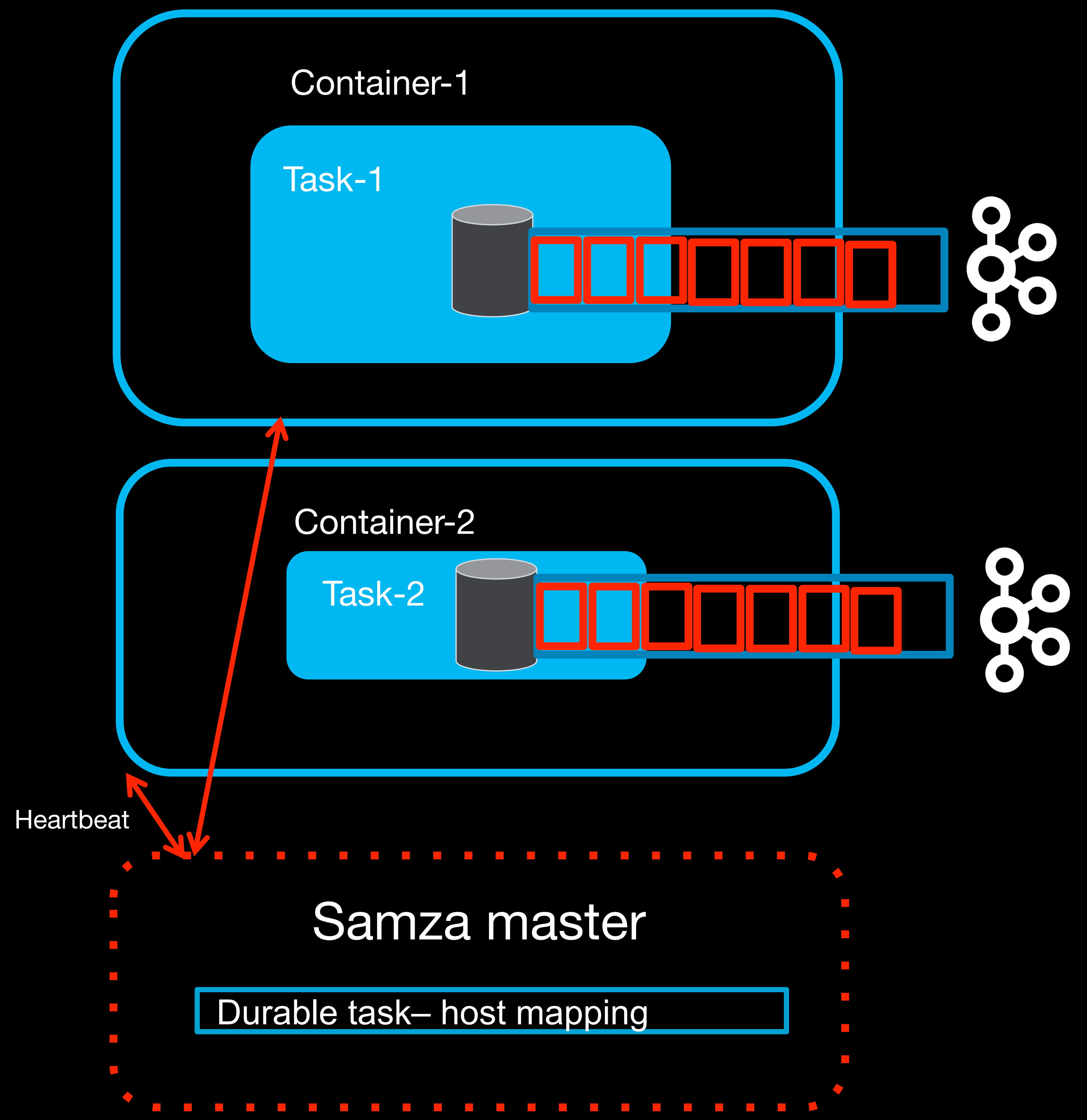
- Changelog

1. Fault tolerant, replicated into Kafka.
2. Ability to catch up from arbitrary point in the log.
3. State restore from Kafka during host failures

Speed thrills but can also KILL.



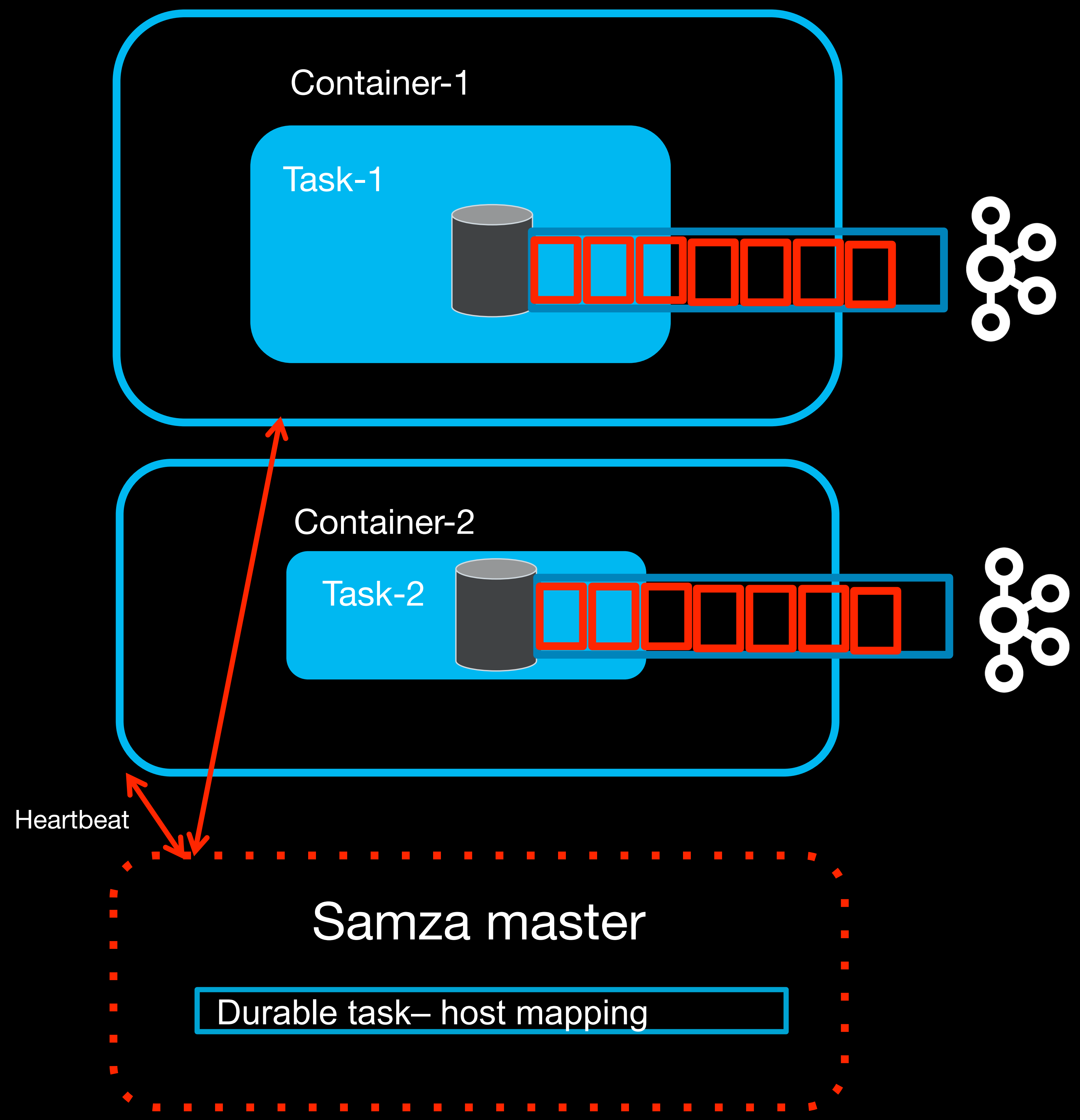
Key optimizations for scaling local state



- Incremental state check-pointing

1. Re-use on-disk state snapshot (host affinity)
2. Write on-disk file on host at checkpoint
3. Catch-up on only delta from the Kafka change-log

Key optimizations for scaling local state



- Incremental state check-pointing

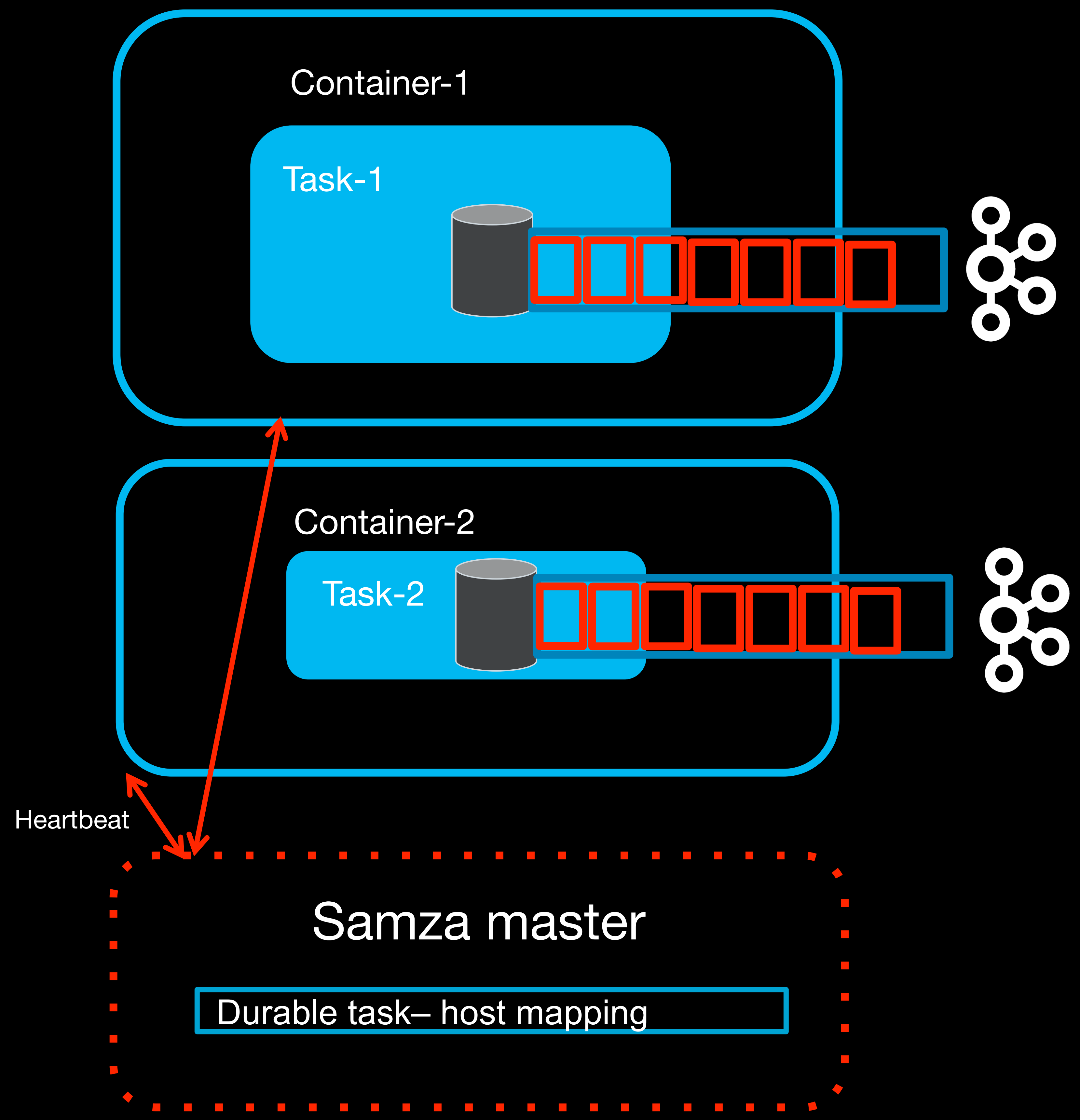
1. Re-use on-disk state snapshot (host affinity)
2. Write on-disk file on host at checkpoint
3. Catch-up on only delta from the Kafka change-log

1.2TB **State**

60x **Faster** than full checkpointing

0 **Downtime** during restarts and recovery

Key optimizations for scaling local state



Drawbacks of local state

1. Size is bounded
2. Some data is not partitionable
3. Repartitioning the stream messes up local state
4. Not useful for serving results

Hard problems in stream processing

Scaling processing

- Partitioned streams
- Distribute processing among all workers

State management

- Hardware failures are inevitable
- Efficient check-pointing
- Instant recovery

High performance remote I/O

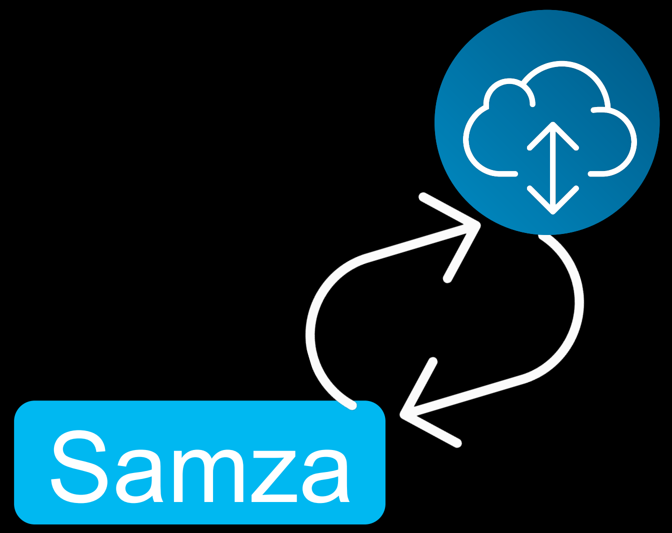
- Need primitives for supporting remote I/O

Heterogeneous deployment models

- Running on a multi-tenant cluster
- Running as an embedded library
- Unified API for batch and real-time data

Why remote I/O matters?

1. Writing to a remote data store (eg: CouchDB) for serving
2. Some data is available only in the remote store or through REST
3. Invoking down-stream services from your processor



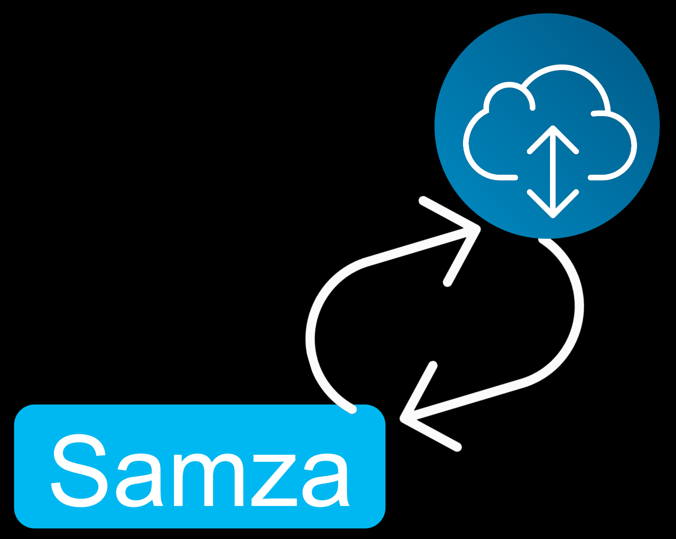
Scaling remote I/O

Hard problems

1. Parallelism need not be tied to number of partitions

2. Complexity of programming model

3. Application has to worry about synchronization and check-pointing



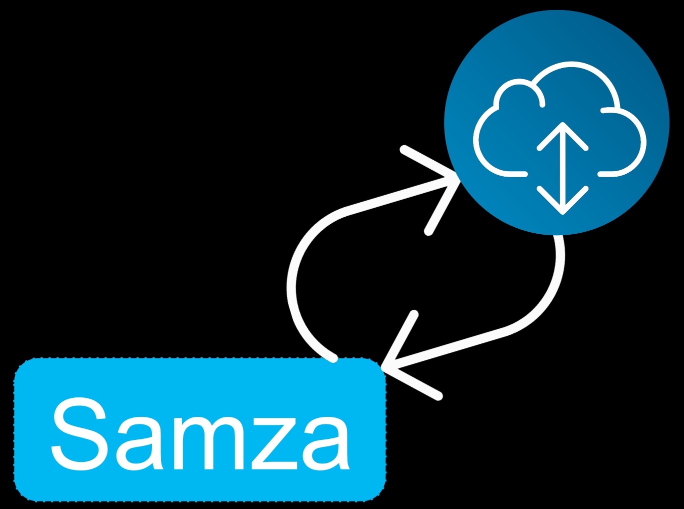
Scaling remote I/O

Hard problems

1. Parallelism need not be tied to number of partitions
2. Complexity of programming model
3. Application has to worry about synchronization and check-pointing

How we solved them?

1. Support out-of-order processing within a partition
2. Simple callback-based async API. Built-in support for both multi-threading and async processing
3. Samza handles checkpointing internally

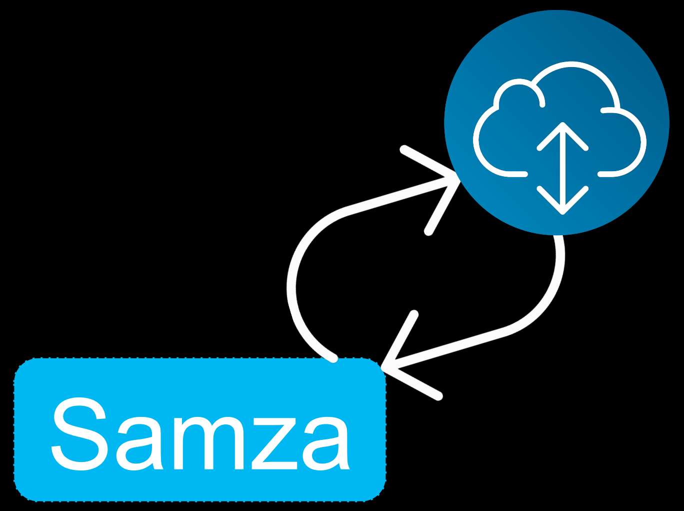


Samza

Performance results

Experiment setup

- *PageViewEvent* topic - 10 partitions
- For each event, remote lookup of the member’s inbox information (latency: 1ms to 200ms)
- Single node Yarn cluster, 1 container (1 CPU core), 1GB memory.



Performance results

Experiment setup

- *PageViewEvent* topic - 10 partitions
- For each event, remote lookup of the member's inbox information (latency: 1ms to 200ms)
- Single node Yarn cluster, 1 container (1 CPU core), 1GB memory.

10x
Faster
With In-order processing

40x
Faster
Out-of-order processing

At our scale this is HUGE!

Hard problems in stream processing

Scaling processing

- Partitioned streams
- Distribute processing among all workers

State management

- Hardware failures are inevitable
- Efficient checkpointing
- Instant recovery

High performance remote I/O

- Need primitives for supporting remote I/O

Heterogeneous deployment models

- Running on a multi-tenant cluster
- Running as an embedded library
- Unified API for batch and real-time data

Samza - Write once, Run anywhere

```
public interface StreamTask {
  void process(IncomingMessageEnvelope envelope,
               MessageCollector collector,
               TaskCoordinator coordinator) {
    // process message
  }
}
```

NEW

```
public class MyApp implements StreamApplication {
 void init(StreamGraph streamGraph,
           Config config) {
   MessageStream<PageView> pageViews = ..;
   pageViews.filter(myKeyFn)
            .map(pageView -> new ProjectedPageView())
            .window(Windows.keyedTumblingWindow(keyFn,
Duration.ofSeconds(30))
             .sink(outputTopic);
              }
}
```

Heterogeneity – Different deployment models

Samza on multi-tenant clusters

- Uses Yarn for coordination, liveness monitoring
- Better resource sharing
- Scale by increasing the number of containers

EXACT SAME CODE

Samza as a light-weight library

- Purely embedded library: No Yarn dependency
- Use zoo-keeper for coordination, liveness and partition distribution
- Seamless auto-scale by increasing the number of instances

```
StreamApplication app = new MyApp();

ApplicationRunner localRunner =

ApplicationRunner.getLocalRunner(config);

localRunner.runApplication(app);
```

Heterogeneity – Support streaming and batch jobs

Samza on Hadoop

- Supports re-processing and experimentation
- Process data on hadoop instead of copying over to Kafka ($$)
- Job automatically shuts-down when end-of-stream is reached

Samza on streaming

- World-class support for streaming inputs

EXACT SAME CODE

Today's agenda

1 Stream processing scenarios

2 Hard problems in stream processing

3 Case Study 1: LinkedIn's communications platform

4 Case Study 2: Activity tracking in the news feed

5 Conclusion

ATC GOAL:

Craft a clear, consistent conversation between our members and their professional world

Features

Channel selection

- “Don’t blast me on all channels”
- Route through Inmail, email or notification in app

Features

Channel selection

- “Don’t blast me on all channels”
- Route through Inmail, email or notification in app

Aggregation / Capping

- “Don’t flood me, Consolidate if you have too much to say!”
- “Here’s a weekly summary of who invited you to connect”

Features

Channel selection

- “Don’t blast me on all channels”
- Route through Inmail, email or notification in app

Aggregation / Capping

- “Don’t flood me, Consolidate if you have too much to say!”
- “Here’s a weekly summary of who invited you to connect”

Delivery time optimization

- “Send me when I will engage and don’t buzz me at 2AM”

Features

Channel selection

- “Don’t blast me on all channels”
- Route through Inmail, email or notification in app

Aggregation / Capping

- “Don’t flood me, Consolidate if you have too much to say!”
- “Here’s a weekly summary of who invited you to connect”

Delivery time optimization

- “Send me when I will engage and don’t buzz me at 2AM”

Filter

- “Filter out spam, duplicates, stuff I have seen or know about”

Why Apache Samza?

Requirements

1. Highly Scalable and distributed
2. Multiple sources of email
3. Fault tolerant state (notifications to be scheduled later)
4. Efficient remote calls to several services

How Samza fits in?

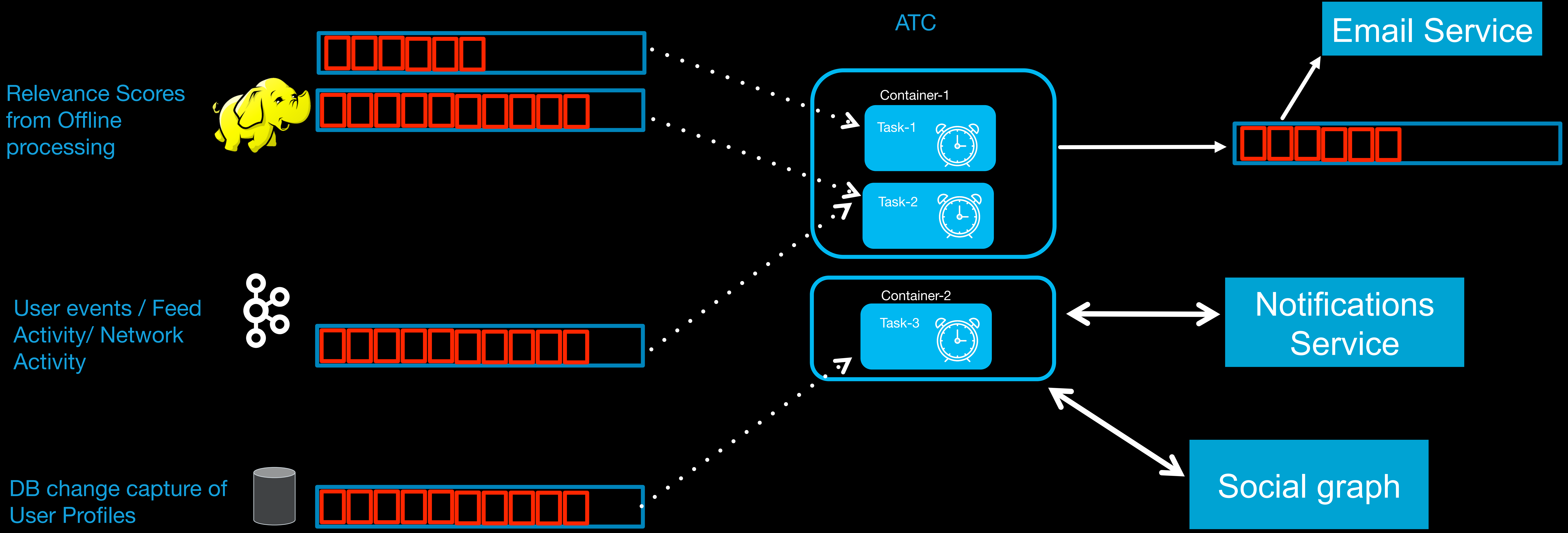
1. Samza partitions streams and provides fault-tolerant processing
2. Pluggable connector API (Kafka, Hadoop, change capture)
3. Instant recovery and incremental checkpointing
4. Async I/O support

Why Apache Samza?

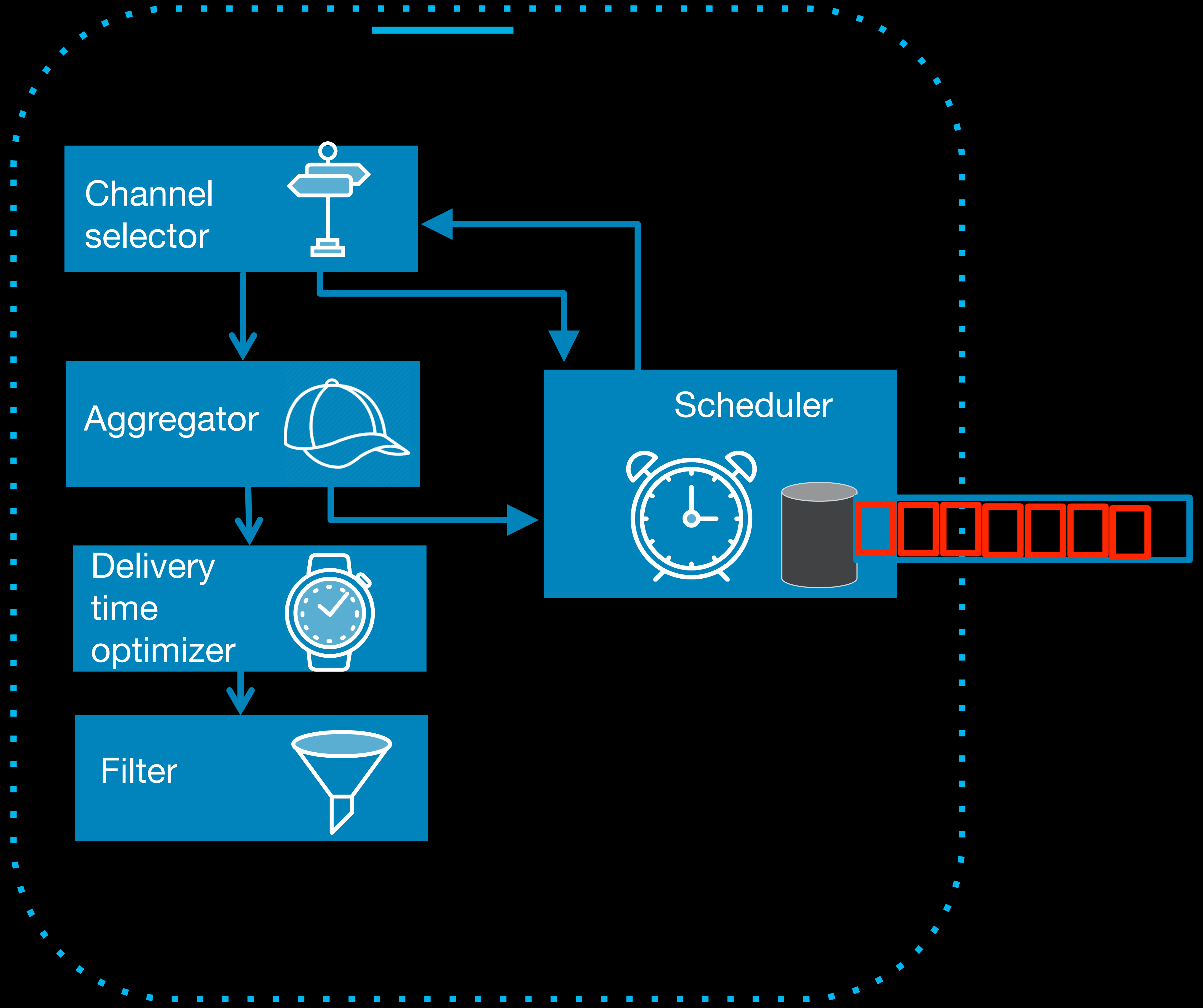
Requirements

1. Highly Scalable and distributed
2. Multiple stream joins
3. Fault tolerant state (notifications to be scheduled later)
4. Efficient remote calls to several services

ATC Architecture


Relevance Scores from Offline processing
User events / Feed Activity/ Network Activity
DB change capture of User Profiles
ATC
Container-1
Task-1
Task-2
Container-2
Task-3
Email Service
Notifications Service
Social graph

Inside each ATC Task



Today's agenda

1 Stream Processing Scenarios

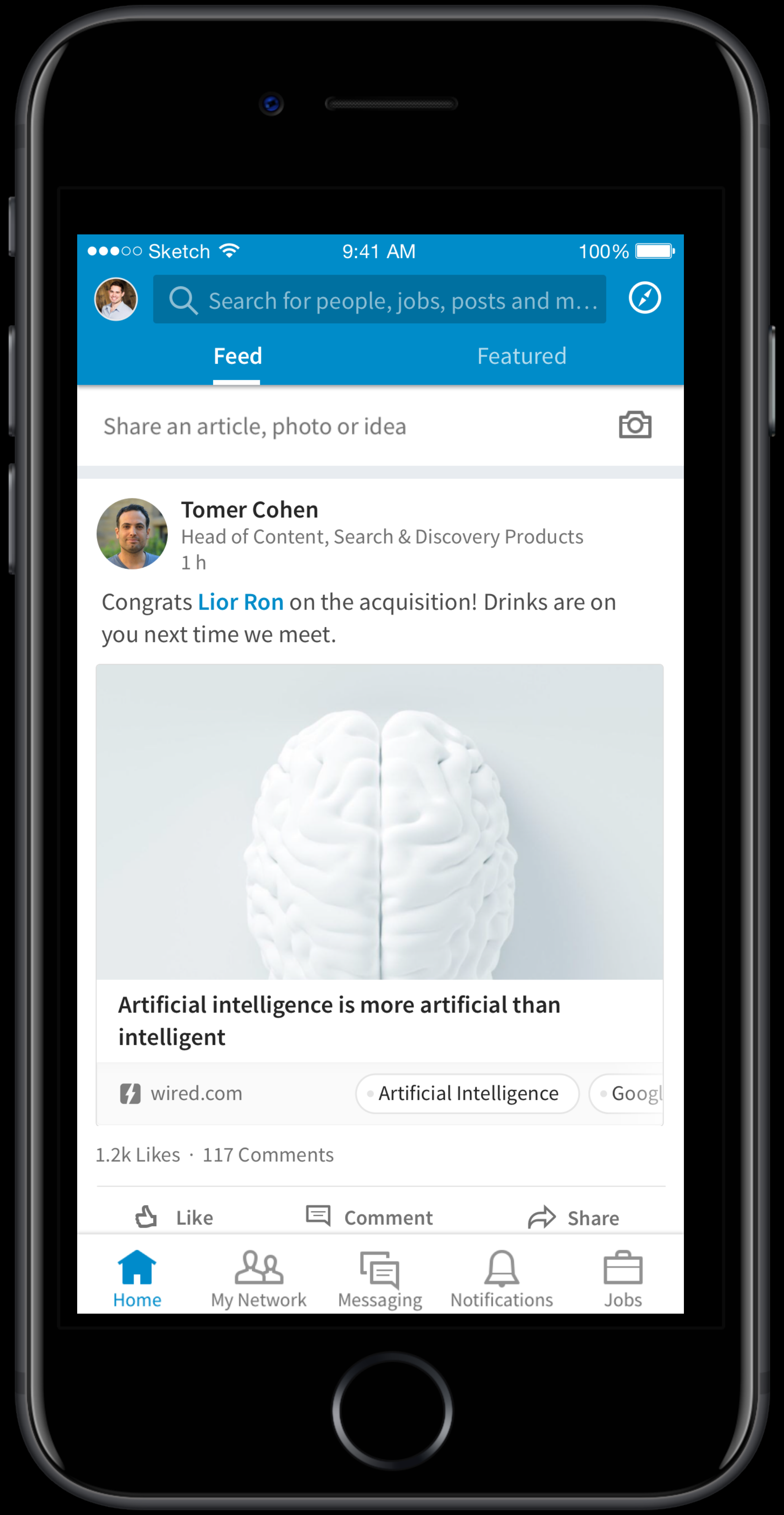
2 Hard Problems in Stream Processing

3 Case Study 1: LinkedIn's communications platform

4 Case Study 2: Activity tracking in the news feed

Activity tracking in News feed

HOW WE USE SAMZA TO IMPROVE YOUR NEWS FEED..


●●●○○ Sketch
9:41 AM
100%
Search for people, jobs, posts and m…
Feed
Featured
Share an article, photo or idea

Tomer Cohen
Head of Content, Search & Discovery Products
1 h

Congrats Lior Ron on the acquisition! Drinks are on you next time we meet.

Artificial intelligence is more artificial than intelligent

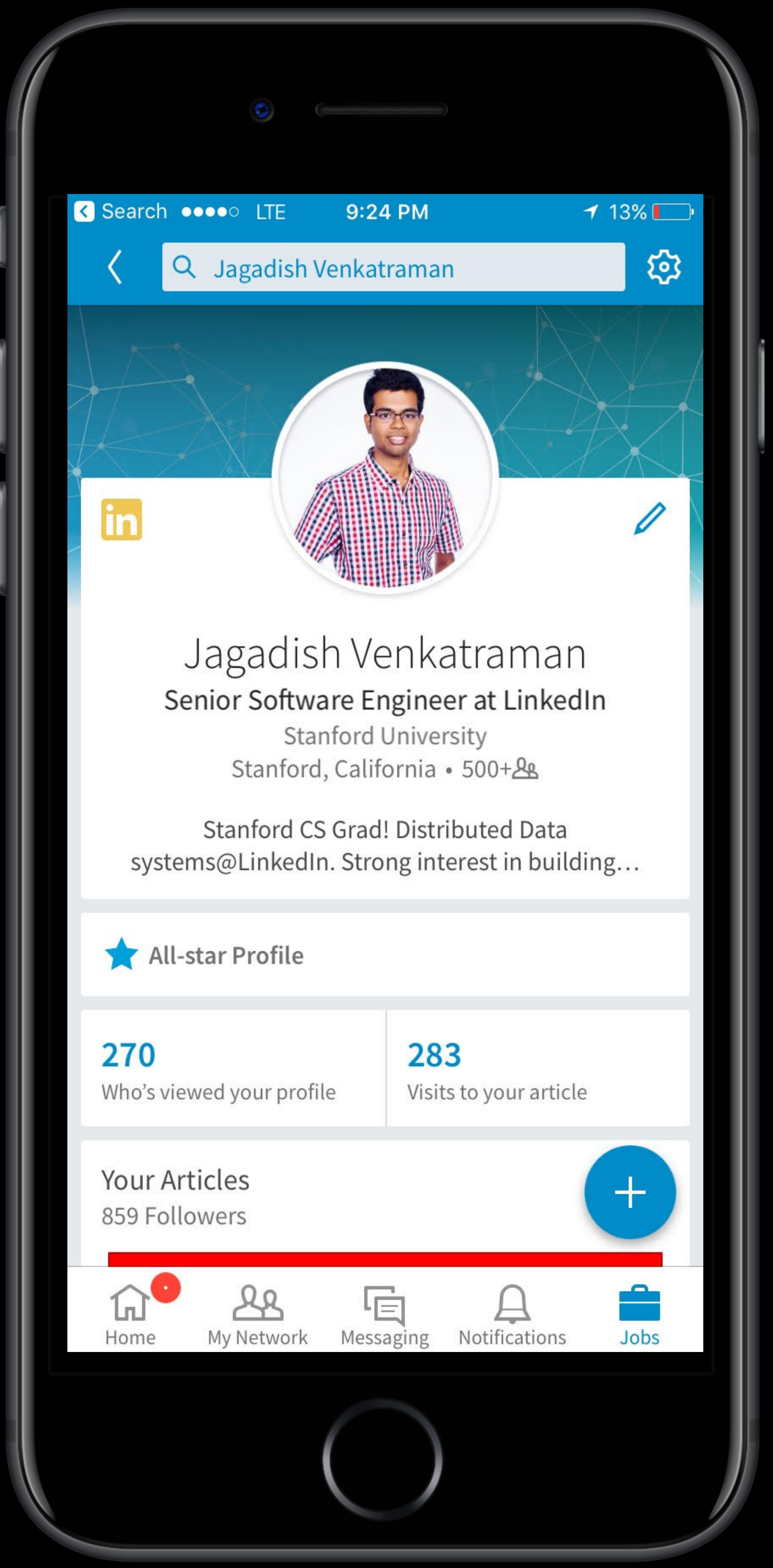
wired.com · Artificial Intelligence · Googl

1.2k Likes · 117 Comments

ACTIVITY TRACKING: GOAL

Power relevant, fresh content for the LinkedIn Feed

Server-side tracking event



Jagadish Venkatraman

Jagadish Venkatraman

Senior Software Engineer at LinkedIn

Stanford University

Stanford, California • 500+

Stanford CS Grad! Distributed Data systems@LinkedIn. Strong interest in building…

All-star Profile

270 Who's viewed your profile

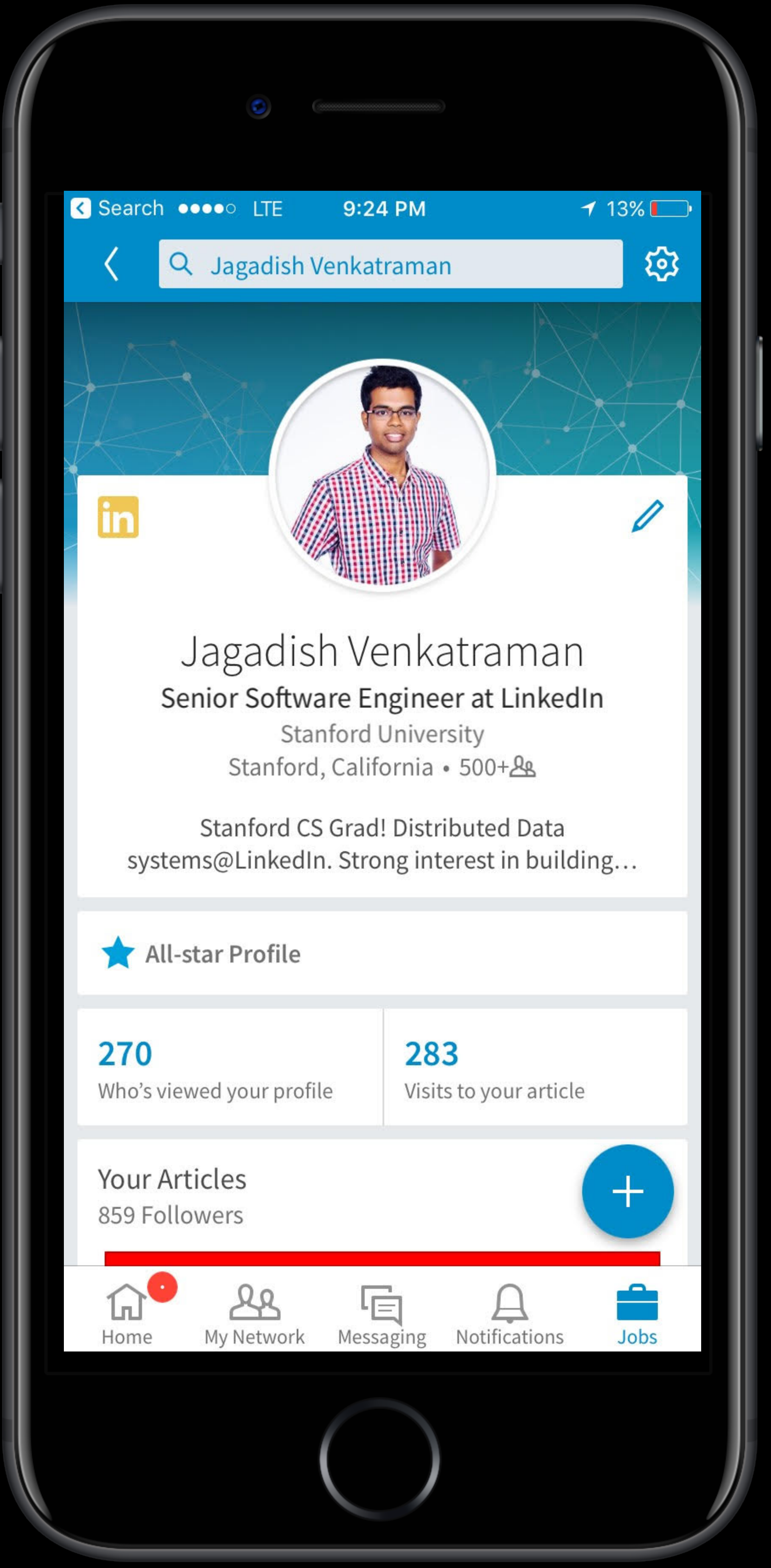
283 Visits to your article

Your Articles

859 Followers

Feed Server

Server-side tracking event



Jagadish Venkatraman
Senior Software Engineer at LinkedIn
Stanford University
Stanford, California • 500+

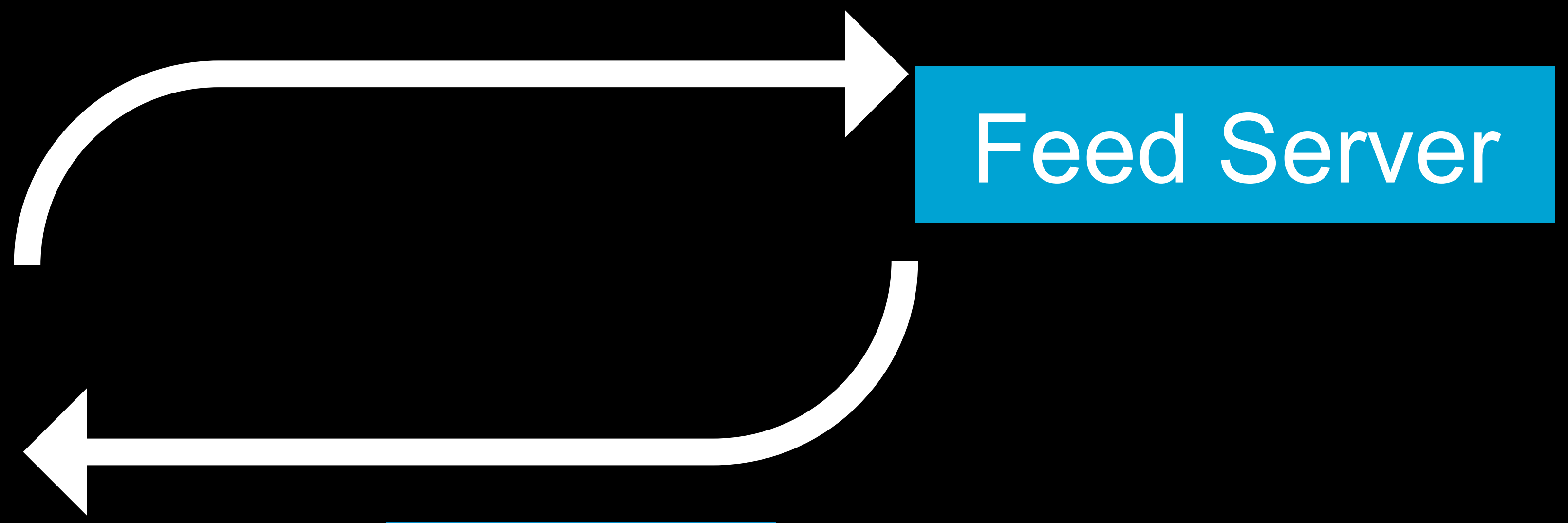
Stanford CS Grad! Distributed Data systems@LinkedIn. Strong interest in building…

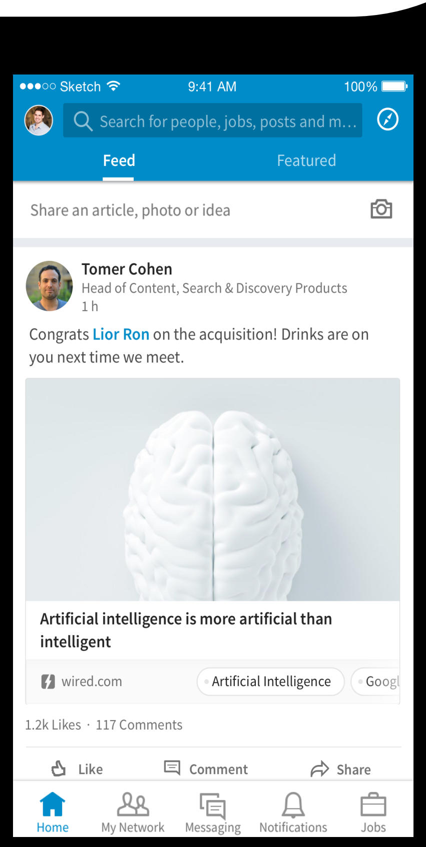
All-star Profile

270 Who's viewed your profile

283 Visits to your article

Your Articles
859 Followers





Tomer Cohen
Head of Content, Search & Discovery Products
1 h

Congrats Lior Ron on the acquisition! Drinks are on you next time we meet.

Artificial intelligence is more artificial than intelligent

wired.com · Artificial Intelligence · Googl

1.2k Likes · 117 Comments

Server-side tracking event

Search ●●●●○ LTE 9:24 PM 13%

Jagadish Venkatraman

Jagadish Venkatraman

Senior Software Engineer at LinkedIn

Stanford University

Stanford, California • 500+

Stanford CS Grad! Distributed Data systems@LinkedIn. Strong interest in building…

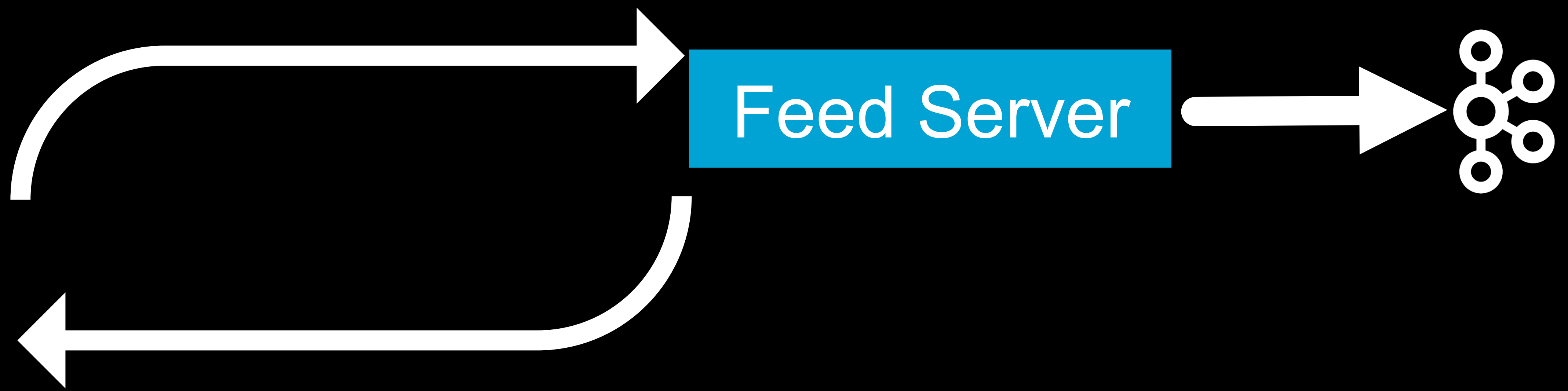
All-star Profile

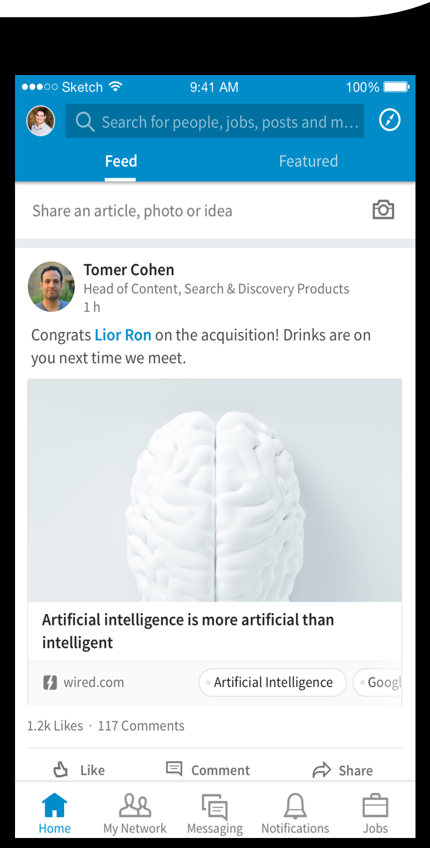
270 Who's viewed your profile

283 Visits to your article

Your Articles

859 Followers





Sketch 9:41 AM 100%

Search for people, jobs, posts and m...

Share an article, photo or idea

Tomer Cohen

Head of Content, Search & Discovery Products

1 h

Congrats Lior Ron on the acquisition! Drinks are on you next time we meet.

Artificial intelligence is more artificial than intelligent

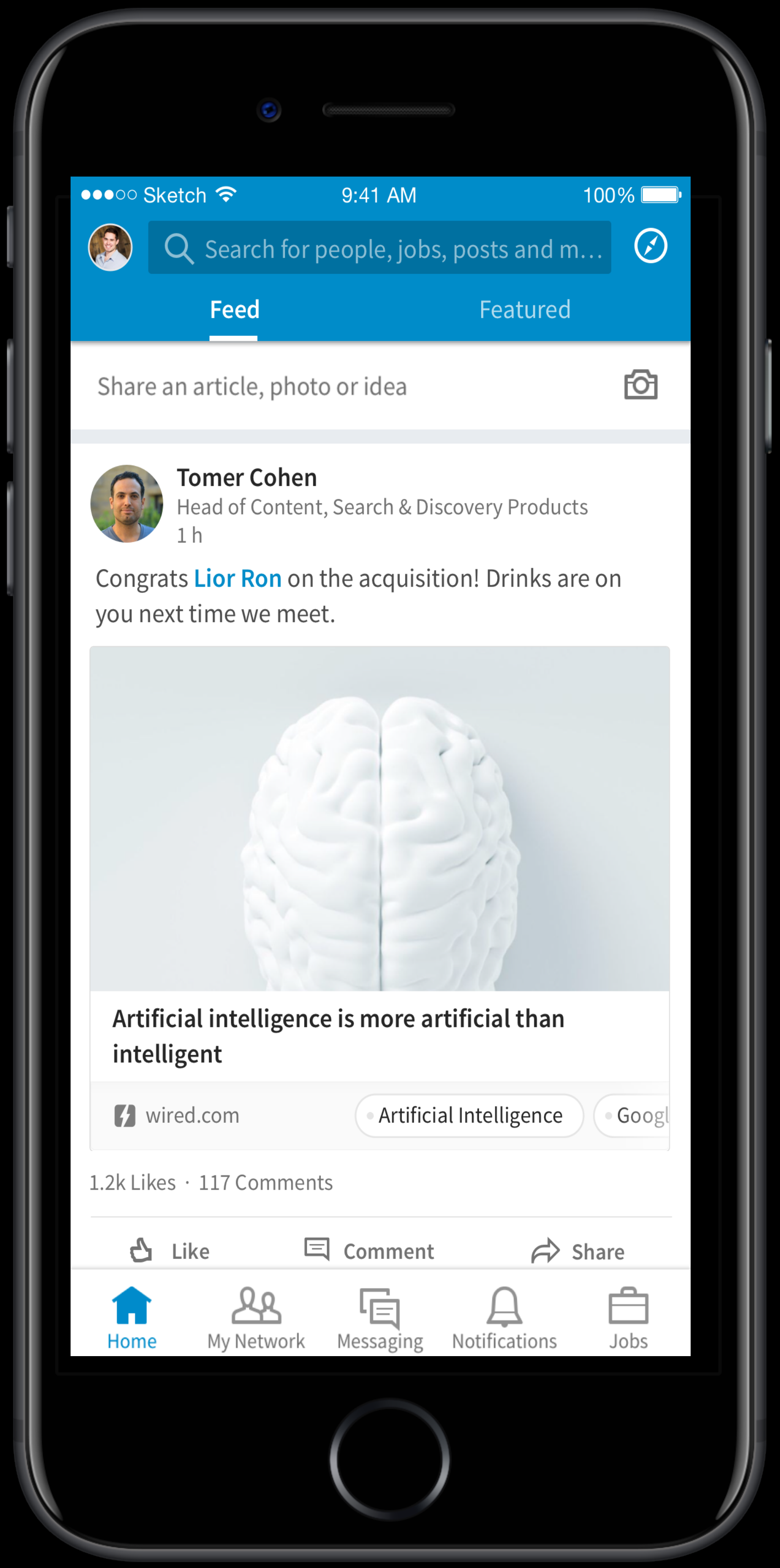
wired.com

1.2k Likes · 117 Comments

Rich payload

```
{
 “paginationId”:
 “feedUpdates”: [{
   “updateUrn”: “update1”
   “trackingId”:
   “position”:
   “creationTime”:
   “numLikes”:
   “numComments”:
   “comments”: [
       {“commentId”: }
    ]
   }, {
   “updateUrn”:“update2”
   “trackingId”:
 ..
}
```

Client-side tracking event



●●●○○ Sketch
9:41 AM
100%
Search for people, jobs, posts and m…
Feed
Featured
Share an article, photo or idea

Tomer Cohen
Head of Content, Search & Discovery Products
1 h

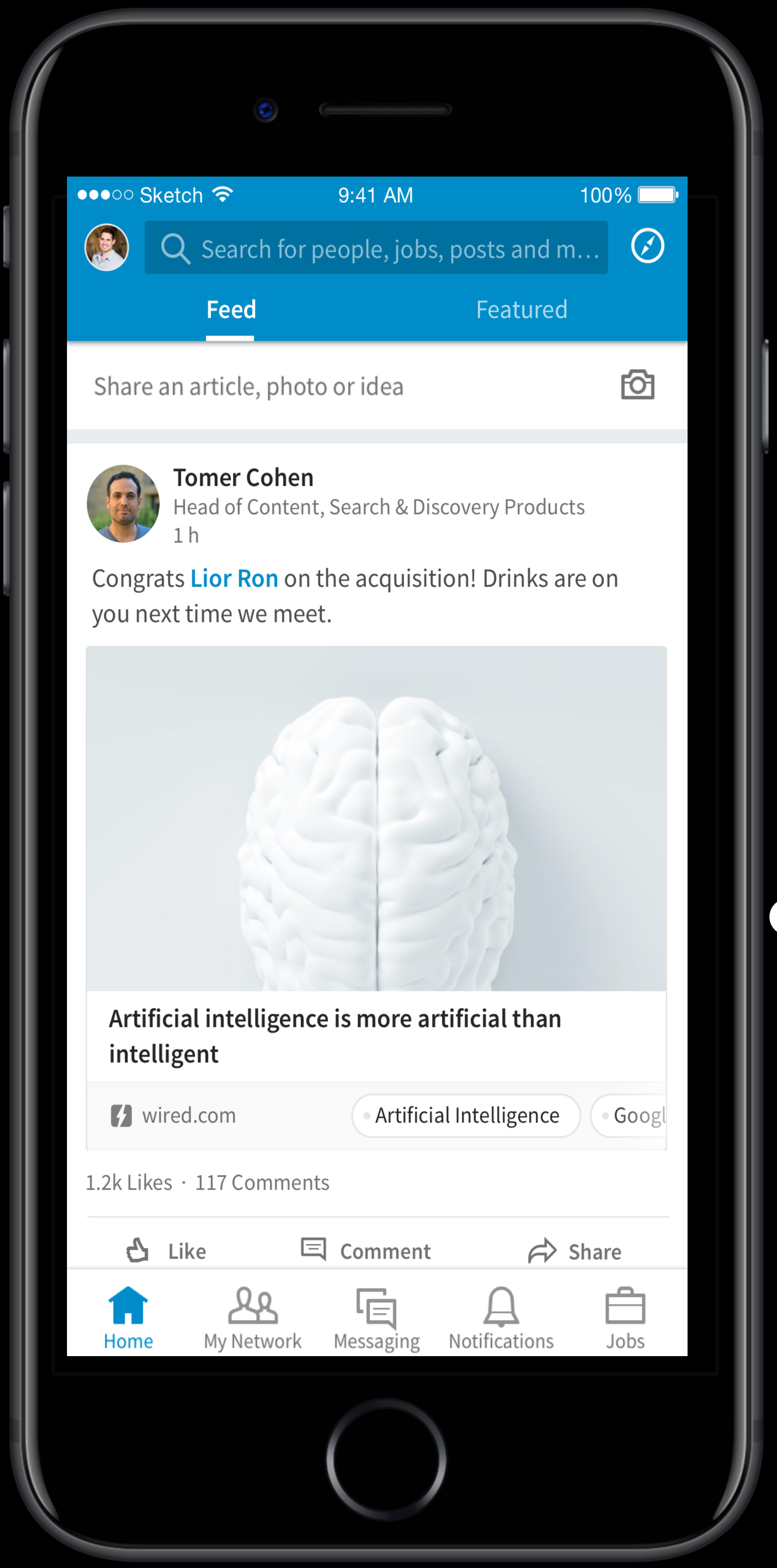
Congrats Lior Ron on the acquisition! Drinks are on you next time we meet.

Artificial intelligence is more artificial than intelligent

wired.com

1.2k Likes · 117 Comments

Client-side tracking event



●●●○○ Sketch 9:41 AM 100%

Search for people, jobs, posts and m...

Feed Featured

Share an article, photo or idea

Tomer Cohen
Head of Content, Search & Discovery Products
1 h

Congrats Lior Ron on the acquisition! Drinks are on you next time we meet.

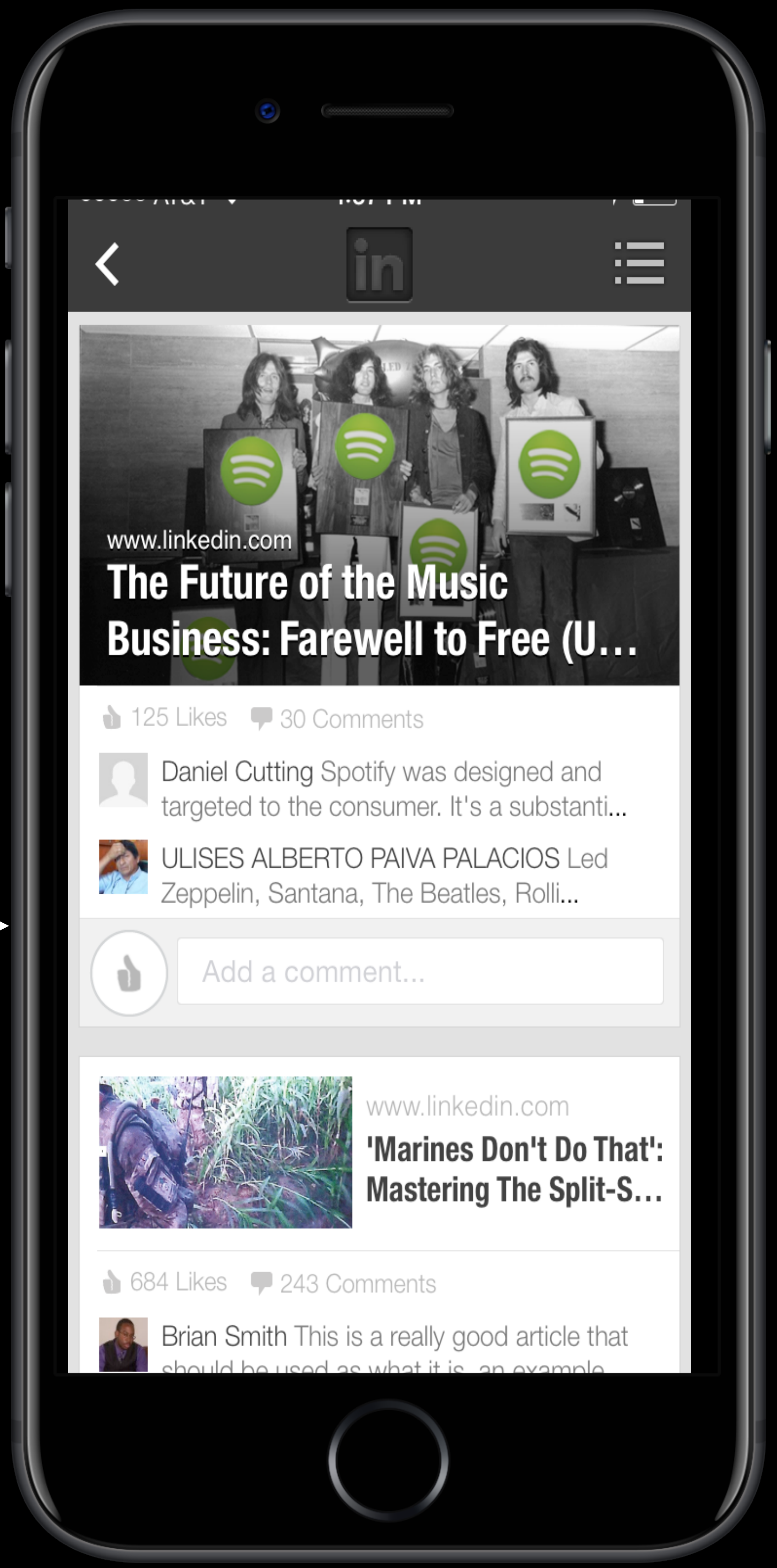
Artificial intelligence is more artificial than intelligent

wired.com · Artificial Intelligence · Googl

1.2k Likes · 117 Comments

Like Comment Share

Home My Network Messaging Notifications Jobs



www.linkedin.com
The Future of the Music Business: Farewell to Free (U...

125 Likes 30 Comments

Daniel Cutting Spotify was designed and targeted to the consumer. It's a substanti...

ULISES ALBERTO PAIVA PALACIOS Led Zeppelin, Santana, The Beatles, Rolli...

Add a comment...

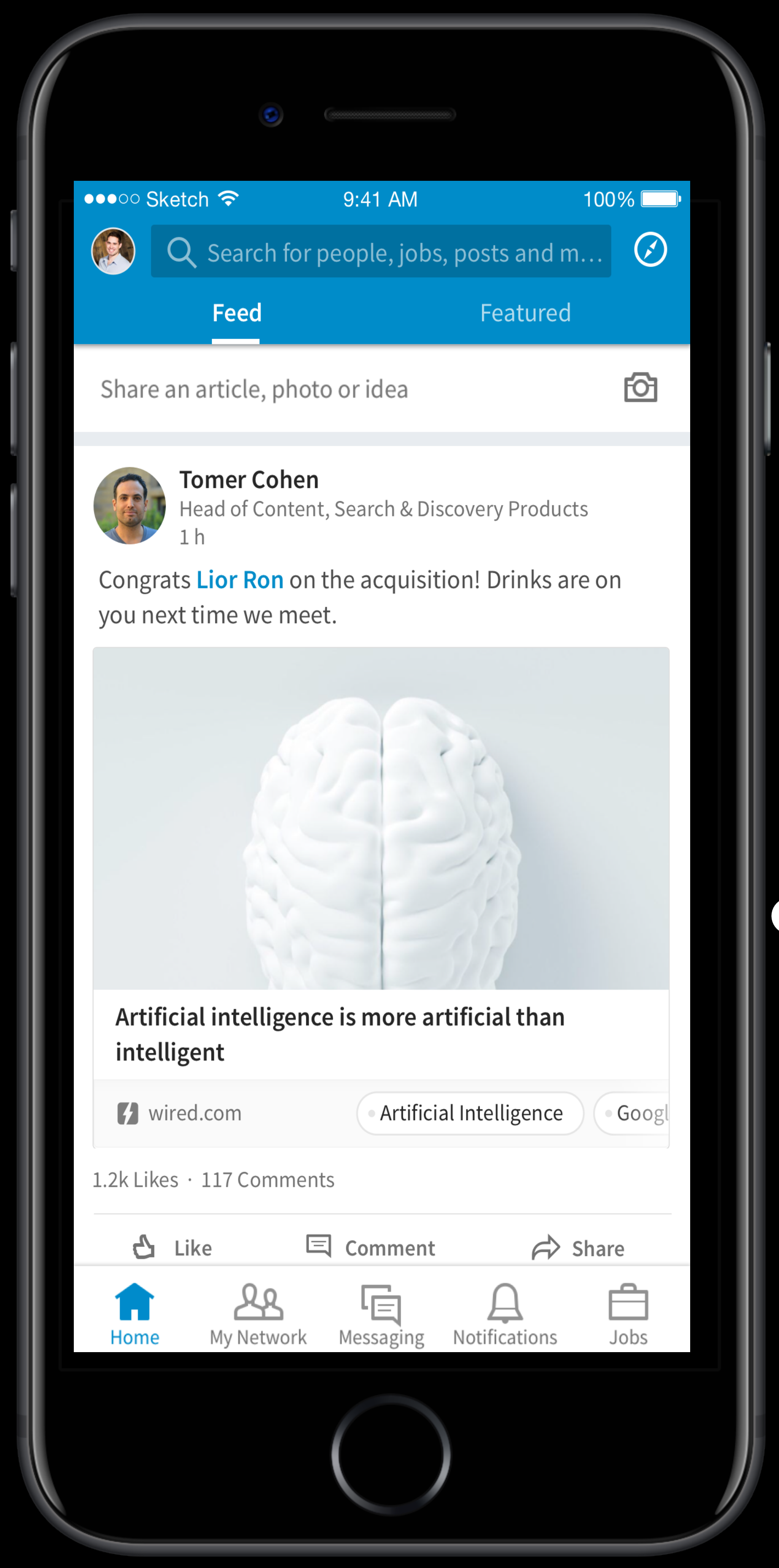
www.linkedin.com
'Marines Don't Do That': Mastering The Split-S...

684 Likes 243 Comments

Brian Smith This is a really good article that should be used as what it is, an example

- Track user activity in the app
- Fire timer during a scroll or a view-port change

Client-side tracking event



●●●○○ Sketch 9:41 AM 100%

Search for people, jobs, posts and m…

Feed Featured

Share an article, photo or idea

Tomer Cohen
Head of Content, Search & Discovery Products
1 h

Congrats Lior Ron on the acquisition! Drinks are on you next time we meet.

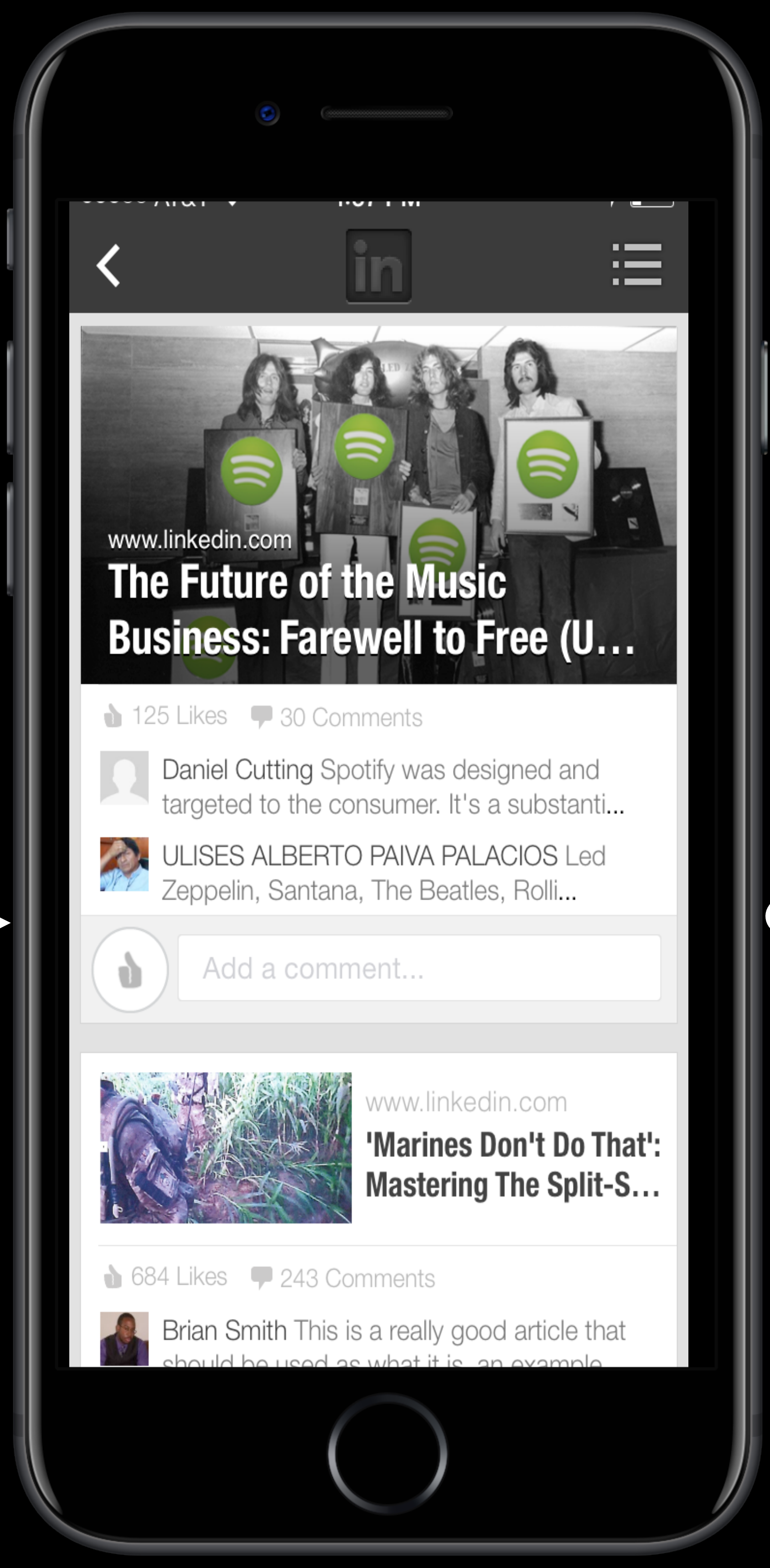
Artificial intelligence is more artificial than intelligent

wired.com · Artificial Intelligence · Googl

1.2k Likes · 117 Comments

Like · Comment · Share

Home · My Network · Messaging · Notifications · Jobs



www.linkedin.com
The Future of the Music Business: Farewell to Free (U…

125 Likes · 30 Comments

Daniel Cutting Spotify was designed and targeted to the consumer. It's a substanti...

ULISES ALBERTO PAIVA PALACIOS Led Zeppelin, Santana, The Beatles, Rolli...

Add a comment...

www.linkedin.com
'Marines Don't Do That': Mastering The Split-S…

684 Likes · 243 Comments

Brian Smith This is a really good article that

```
“feedImpression”: {
  “urn”:
  “trackingId”:
  “duration”:
  “height”:
}, {
  “urn”:
  “trackingId”:
} ,]
```

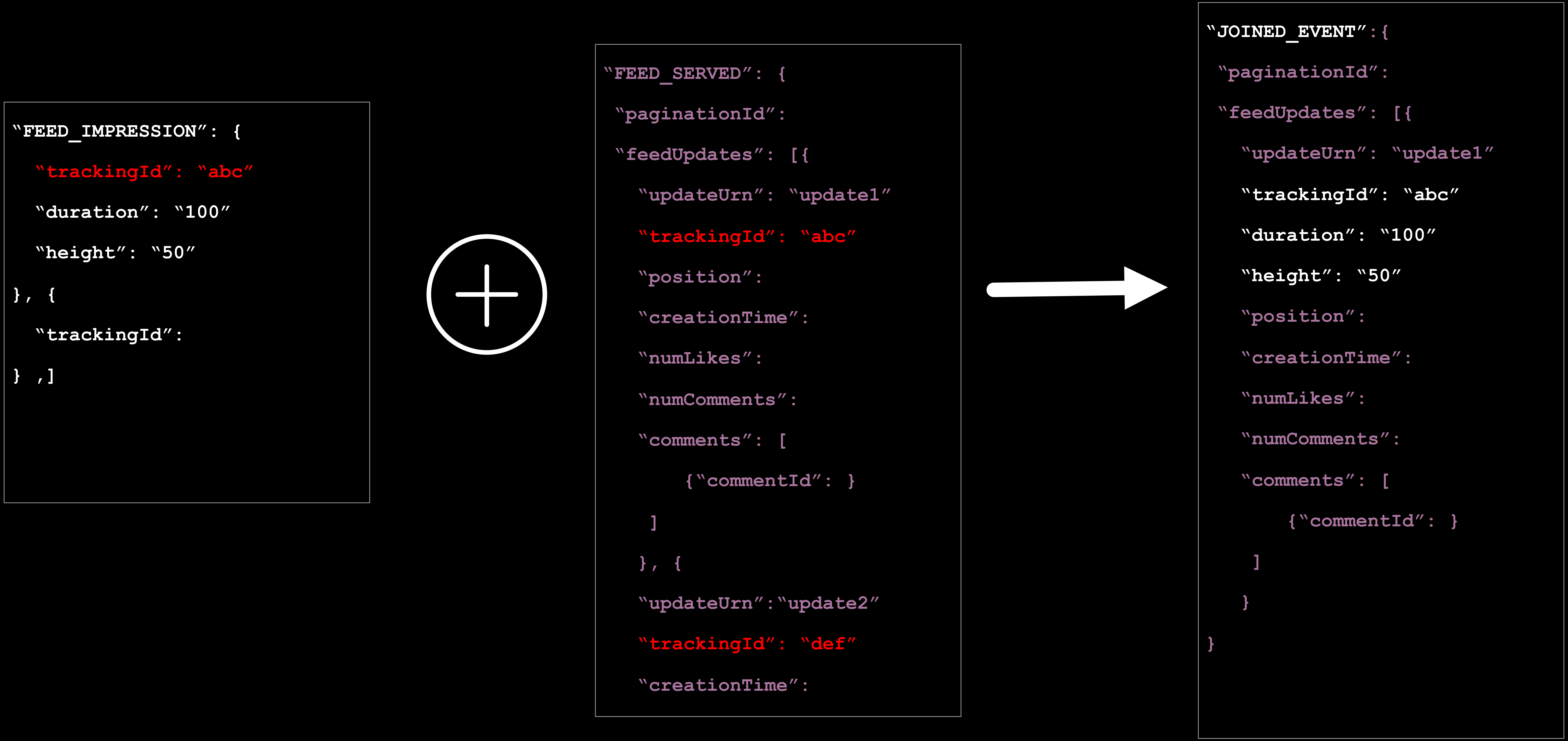
- Light pay load
- Bandwidth friendly
- Battery friendly

Join client-side event with server-side event

```
“FEED_IMPRESSION”: {
  “trackingId”: “abc”
  “duration”: “100”
  “height”: “50”
}, {
  “trackingId”:
} ,]
```

```
“FEED_SERVED”: {
 “paginationId”:
 “feedUpdates”: [{
   “updateUrn”: “update1”
   “trackingId”: “abc”
   “position”:
   “creationTime”:
   “numLikes”:
   “numComments”:
   “comments”: [
        {“commentId”: }
    ]
   }, {
   “updateUrn”:“update2”
   “trackingId”: “def”
   “creationTime”:
```

Samza Joins client-side event with server-side event

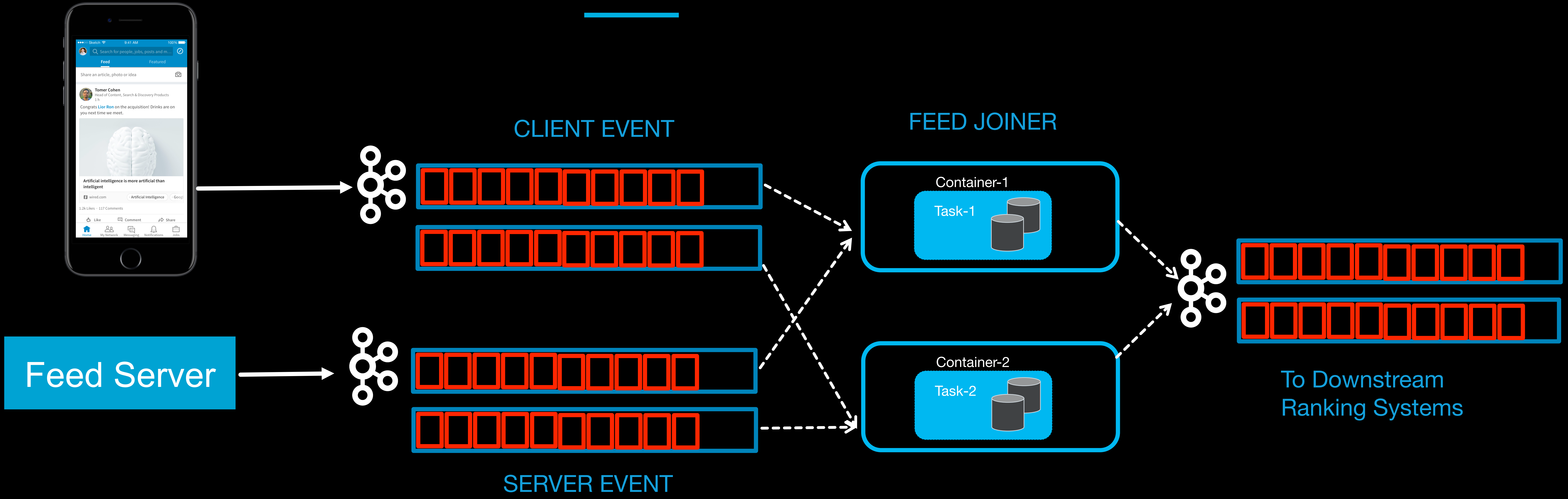


```
“FEED_IMPRESSION”: {
  “trackingId”: “abc”
  “duration”: “100”
  “height”: “50”
}, {
  “trackingId”:
} ,]
```

```
“FEED_SERVED”: {
 “paginationId”:
 “feedUpdates”: [{
   “updateUrn”: “update1”
   “trackingId”: “abc”
   “position”:
   “creationTime”:
   “numLikes”:
   “numComments”:
   “comments”: [
       {“commentId”: }
   ]
   }, {
   “updateUrn”:“update2”
   “trackingId”: “def”
   “creationTime”:
```

```
“JOINED_EVENT”:{
 “paginationId”:
 “feedUpdates”: [{
   “updateUrn”: “update1”
   “trackingId”: “abc”
   “duration”: “100”
   “height”: “50”
   “position”:
   “creationTime”:
   “numLikes”:
   “numComments”:
   “comments”: [
       {“commentId”: }
    ]
   }
}
```

Architecture



1.2

Billion

Processed per-day

90

Containers

Distributed, Partitioned

Recap

1 Stream processing scenarios

2 Hard problems in stream processing

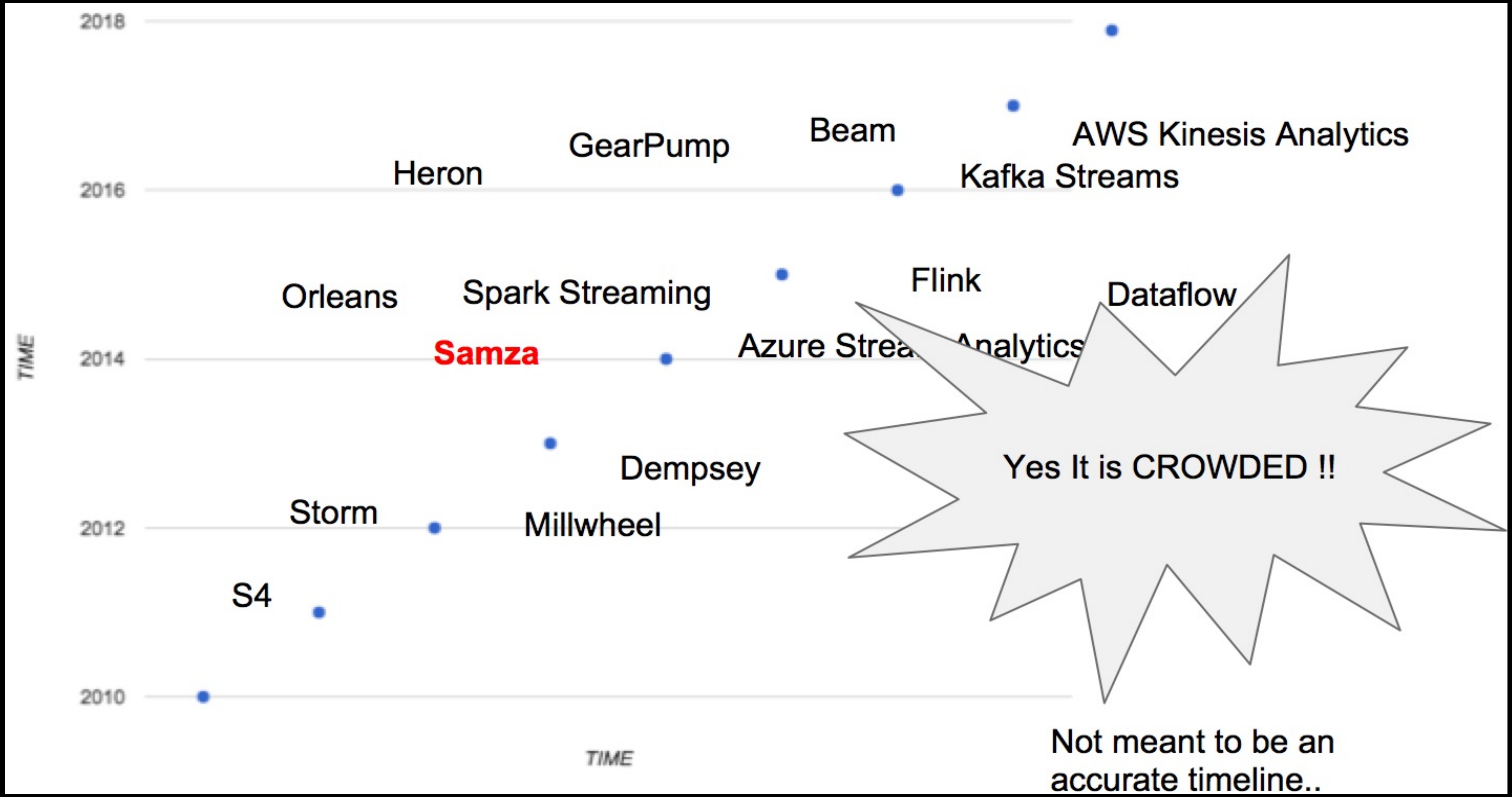
3 Case Study 1: LinkedIn’s communications platform

4 Case Study 2: Activity tracking in the news feed

Recap

SUMMARY OF WHAT WE LEARNT IN THIS TALK

Key differentiators for Apache Samza



Not meant to be an accurate timeline..

Key differentiators for Apache Samza

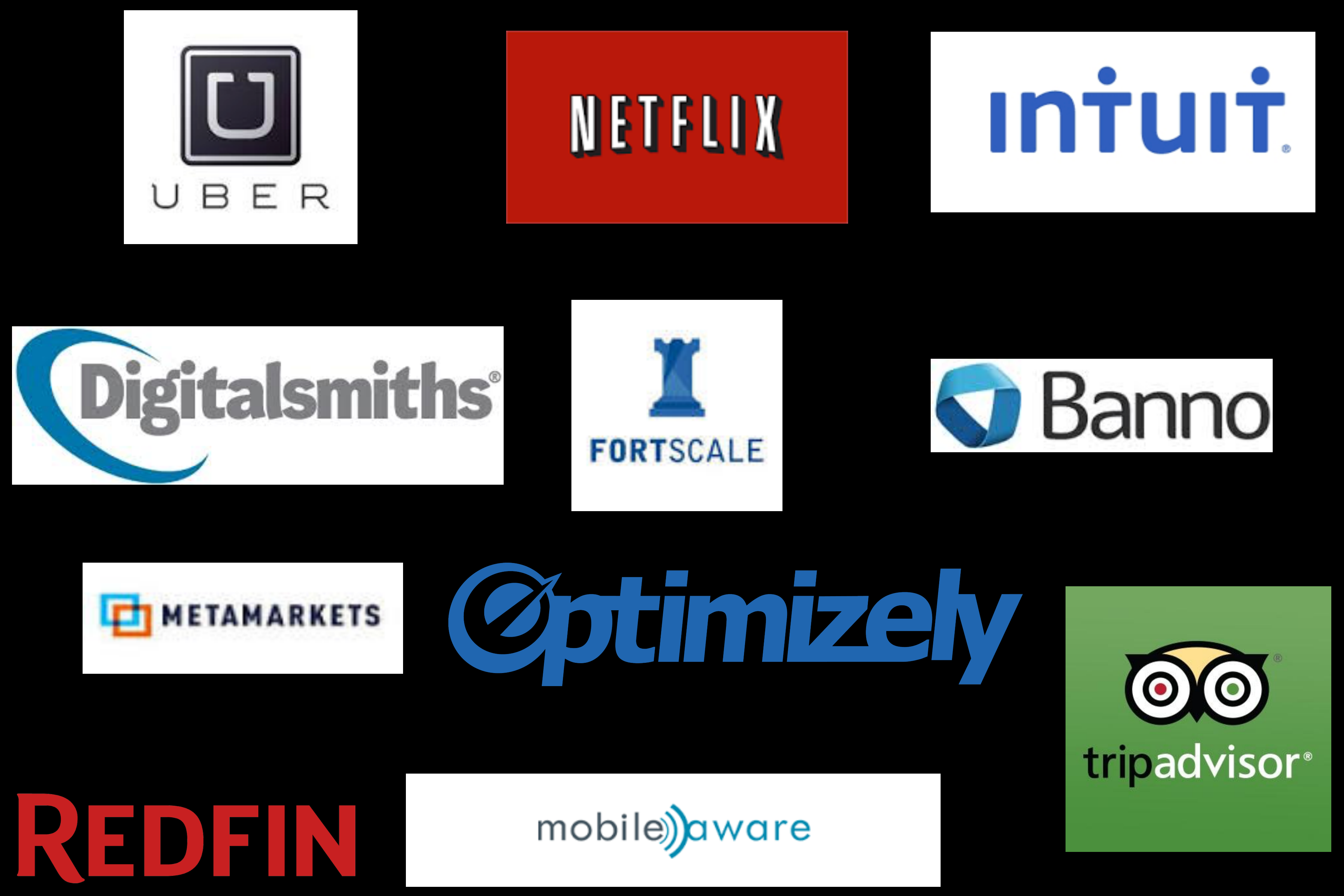
- Stream Processing both as a multi-tenant service and a light-weight embedded library
- No micro batching (first class streaming)
- Unified processing of batch and streaming data
- Efficient remote calls I/O using built-in async mode
- World-class support for scalable local state
  - Incremental check-pointing
  - Instant restore with zero down-time
- Low level API and Stream based high level API (DSL)

Coming soon

samza

- Event time, and out of order arrival handling
- Beam API integration
- SQL on streams

Powered by samza



STABLE, MATURE STREAM PROCESSING FRAMEWORK

200+

Applications at LinkedIn

Q & A

http://samza.apache.org