RTB HOUSE =

bigdata TECHNOLOGY SUMMIT

Real-Time Data Processing at RTB House

How we have grown 10x within 2 years

Bartosz Łoś, 2019

AGENDA

- our RTB platform

AGENDA

- our RTB platform
- the previous iterations: three different architectures

AGENDA

- our RTB platform
- the previous iterations: three different architectures
- the fourth iteration: multi-dc architecture

AGENDA

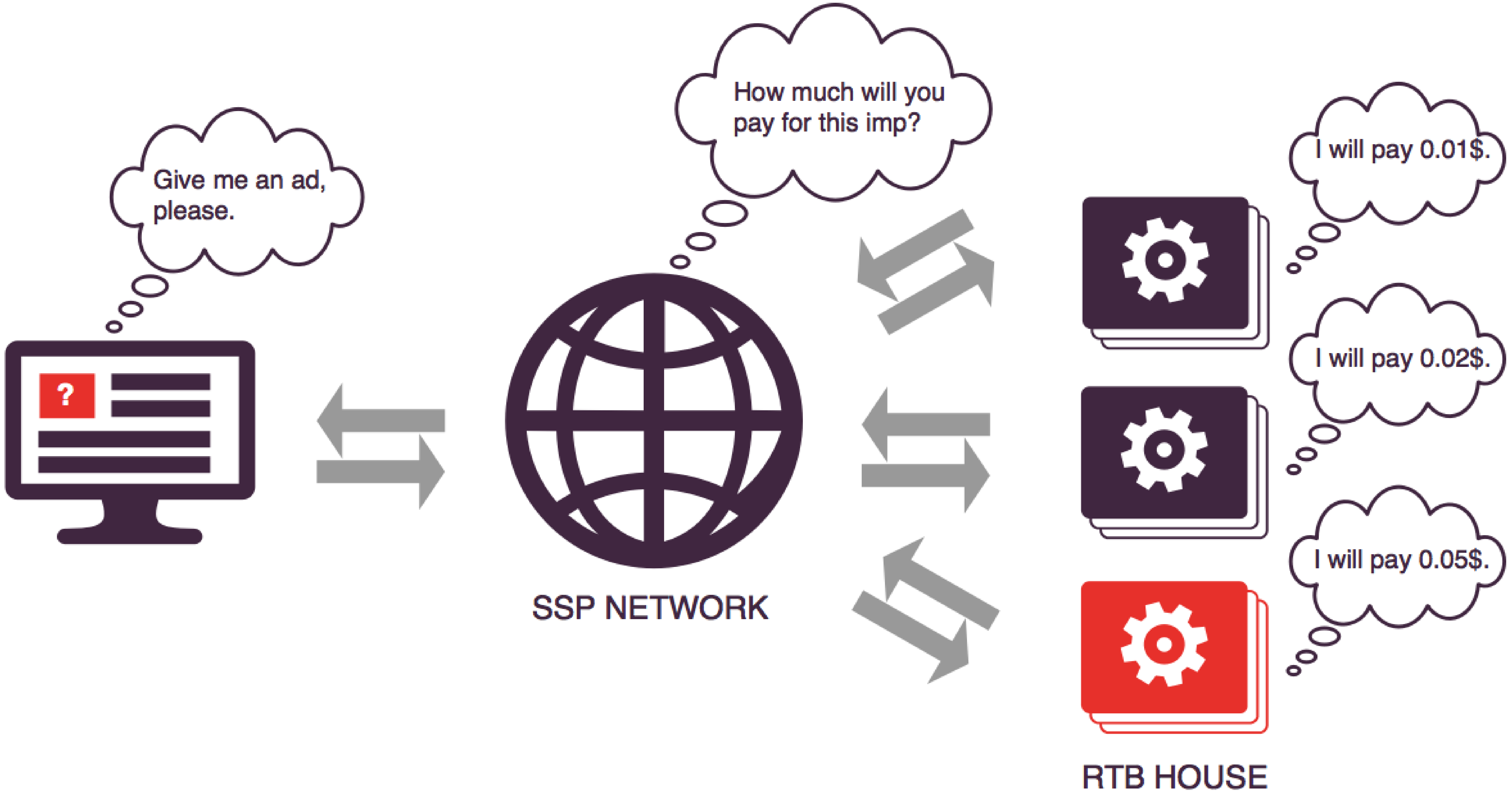
- our RTB platform
- the previous iterations: three different architectures
- the fourth iteration: multi-dc architecture
- our use cases: requirements and processing patterns

AGENDA

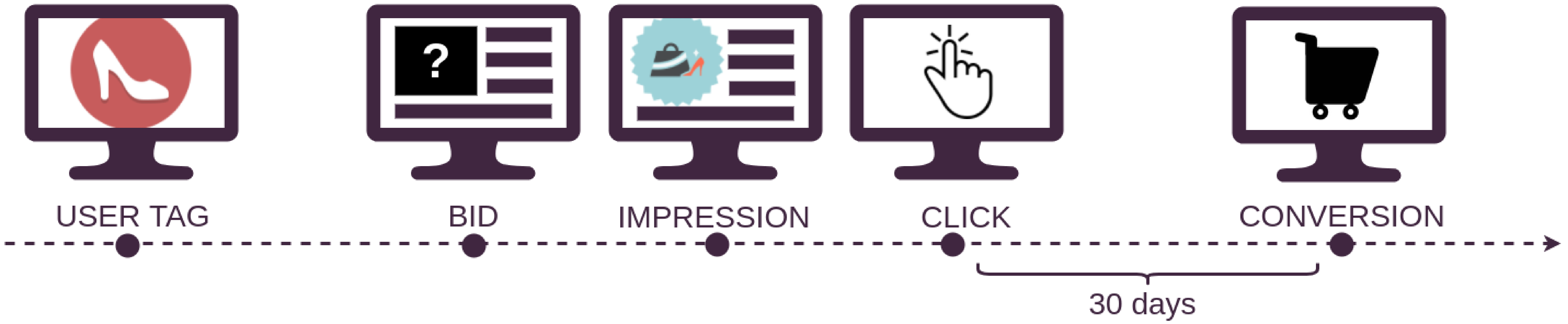
- our RTB platform
- the previous iterations: three different architectures
- the fourth iteration: multi-dc architecture
- our use cases: requirements and processing patterns
- kafka workers

OUR RTB PLATFORM

OUR RTB PLATFORM: THE CONTEXT

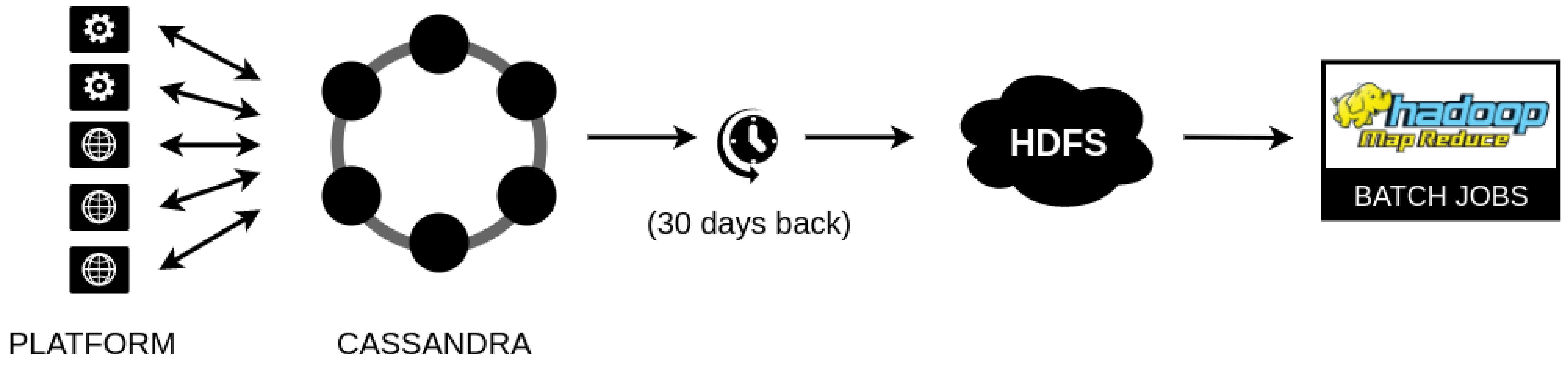


OUR RTB PLATFORM: THE CONTEXT

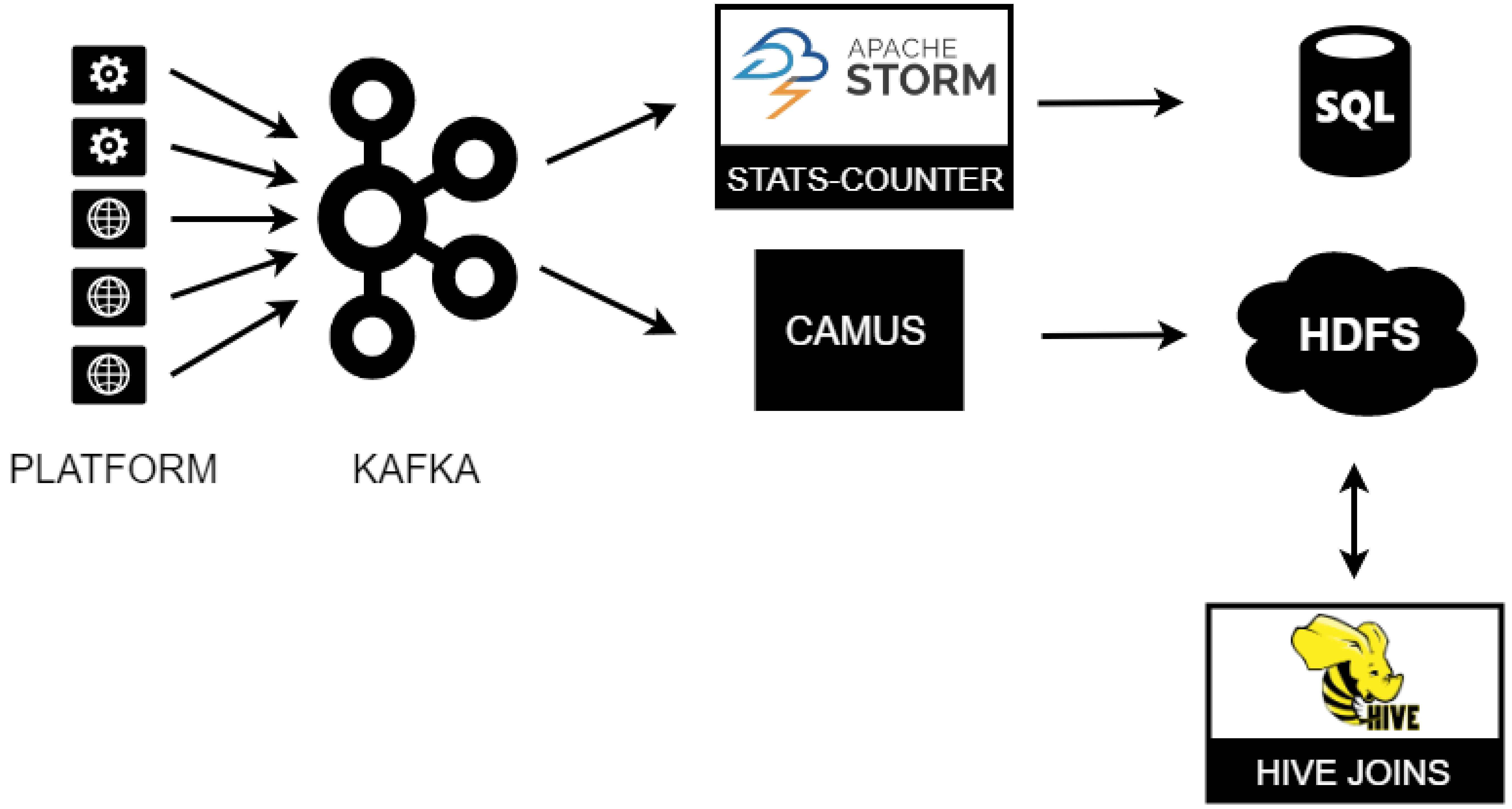


THE PREVIOUS ITERATIONS

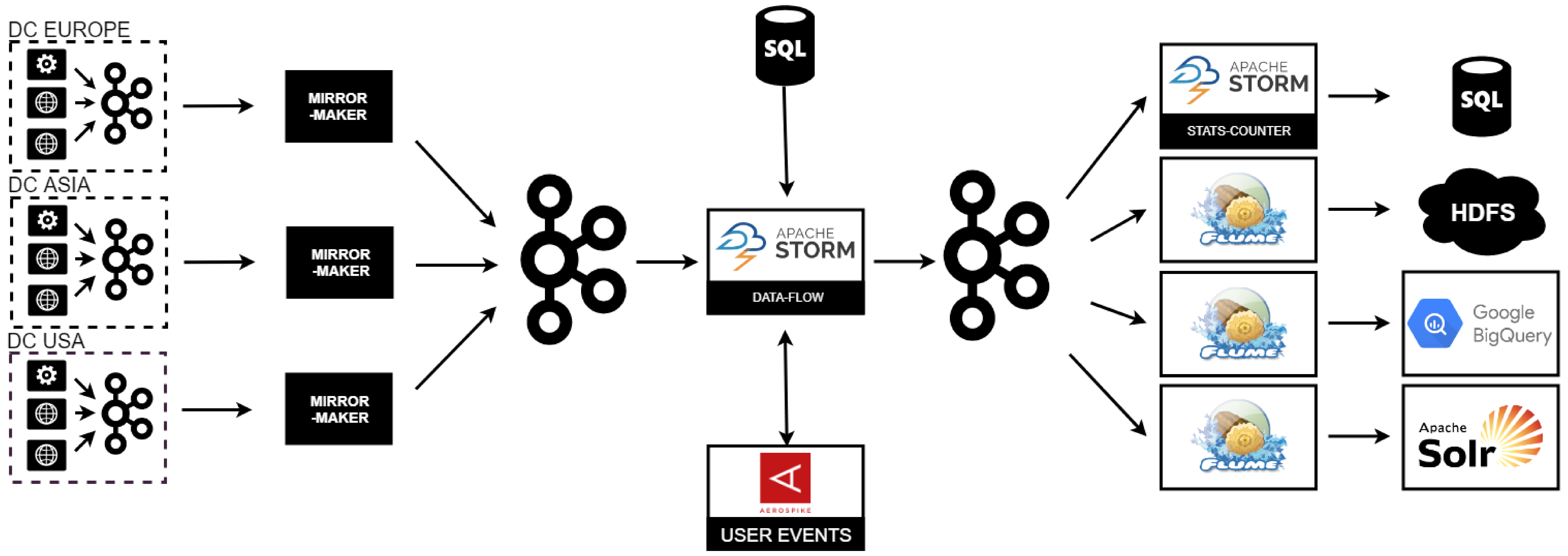
THE 1ST ITERATION: MUTABLE IMPRESSIONS



THE 2ND ITERATION: LAMBDA ARCHITECTURE



THE 3RD ITERATION: IMMUTABLE STREAMS OF EVENTS



THE FOURTH ITERATION: MULTI-DC

THE 4TH ITERATION: MAIN CHANGES

- 10x larger scale:
  - from 350K to 3.5M bid requests/s within 2 years

THE 4TH ITERATION: MAIN CHANGES

- 10x larger scale:
  - from 350K to 3.5M bid requests/s within 2 years
- full multi-dc architecture:
  - synchronization of user profiles
  - merging streams of events

THE 4TH ITERATION: MAIN CHANGES

- 10x larger scale:
  - from 350K to 3.5M bid requests/s within 2 years
- full multi-dc architecture:
  - synchronization of user profiles
  - merging streams of events
- fixed partitioning in all DCs:
  - parallelism, merging, end-to-end lag

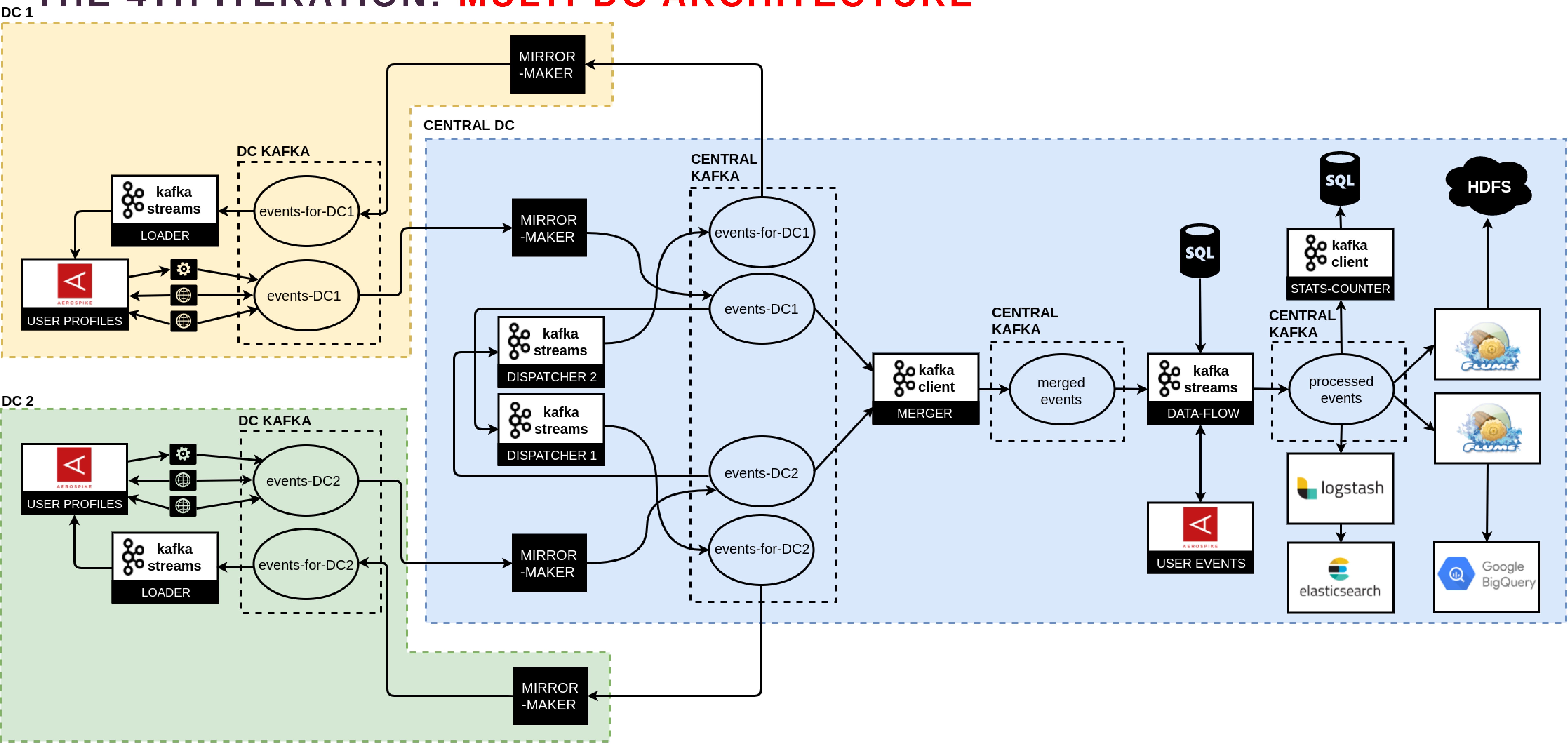
THE 4TH ITERATION: MAIN CHANGES

- 10x larger scale:
  - from 350K to 3.5M bid requests/s within 2 years
- full multi-dc architecture:
  - synchronization of user profiles
  - merging streams of events
- fixed partitioning in all DCs:
  - parallelism, merging, end-to-end lag
- end-to-end exactly-once processing:
  - at-least-once output semantics & deduplication

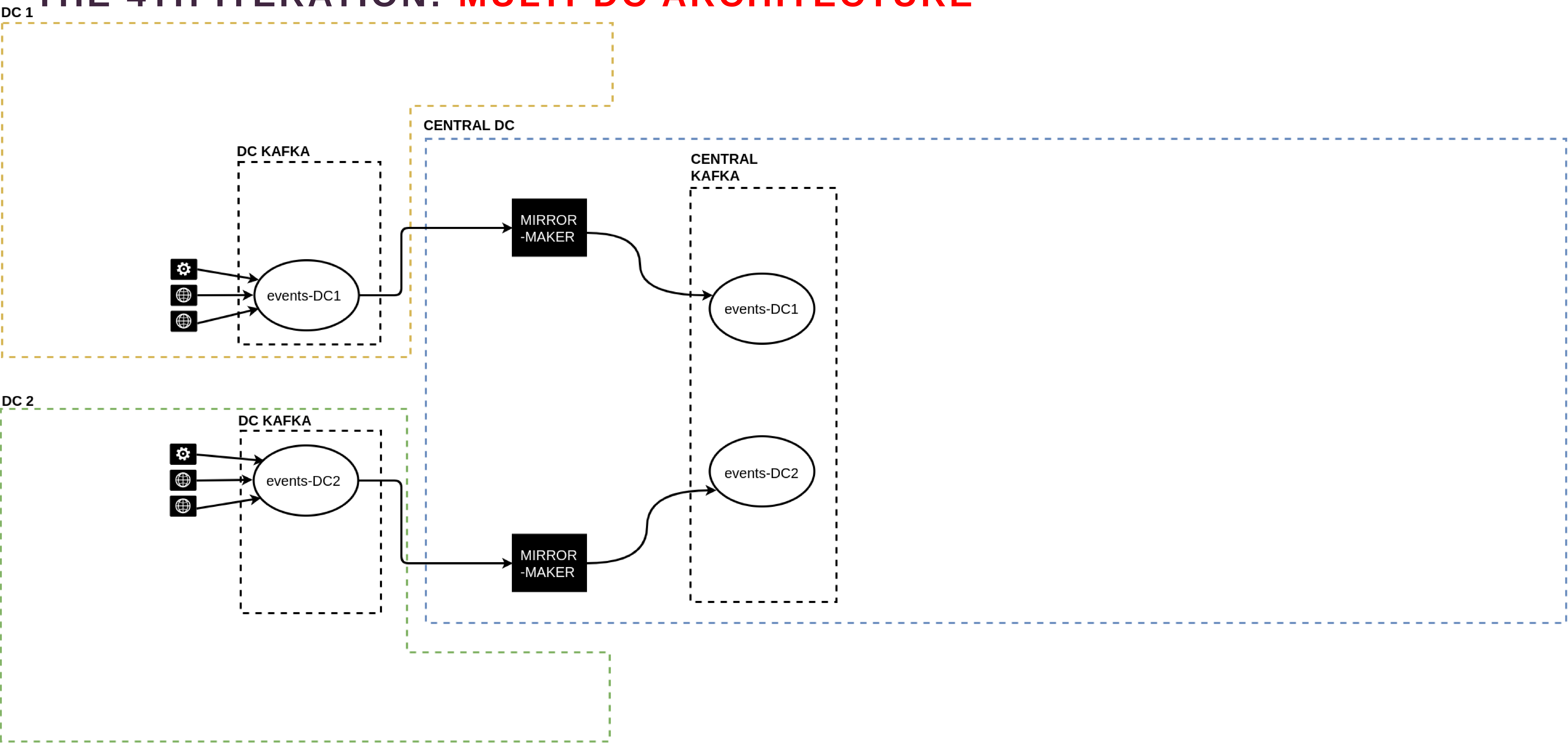
THE 4TH ITERATION: MAIN CHANGES

- 10x larger scale:
  - from 350K to 3.5M bid requests/s within 2 years
- full multi-dc architecture:
  - synchronization of user profiles
  - merging streams of events
- fixed partitioning in all DCs:
  - parallelism, merging, end-to-end lag
- end-to-end exactly-once processing:
  - at-least-once output semantics & deduplication
- a few better components:
  - new stats-counter, new data-flow
  - logstash
  - merger, dispatcher & loader

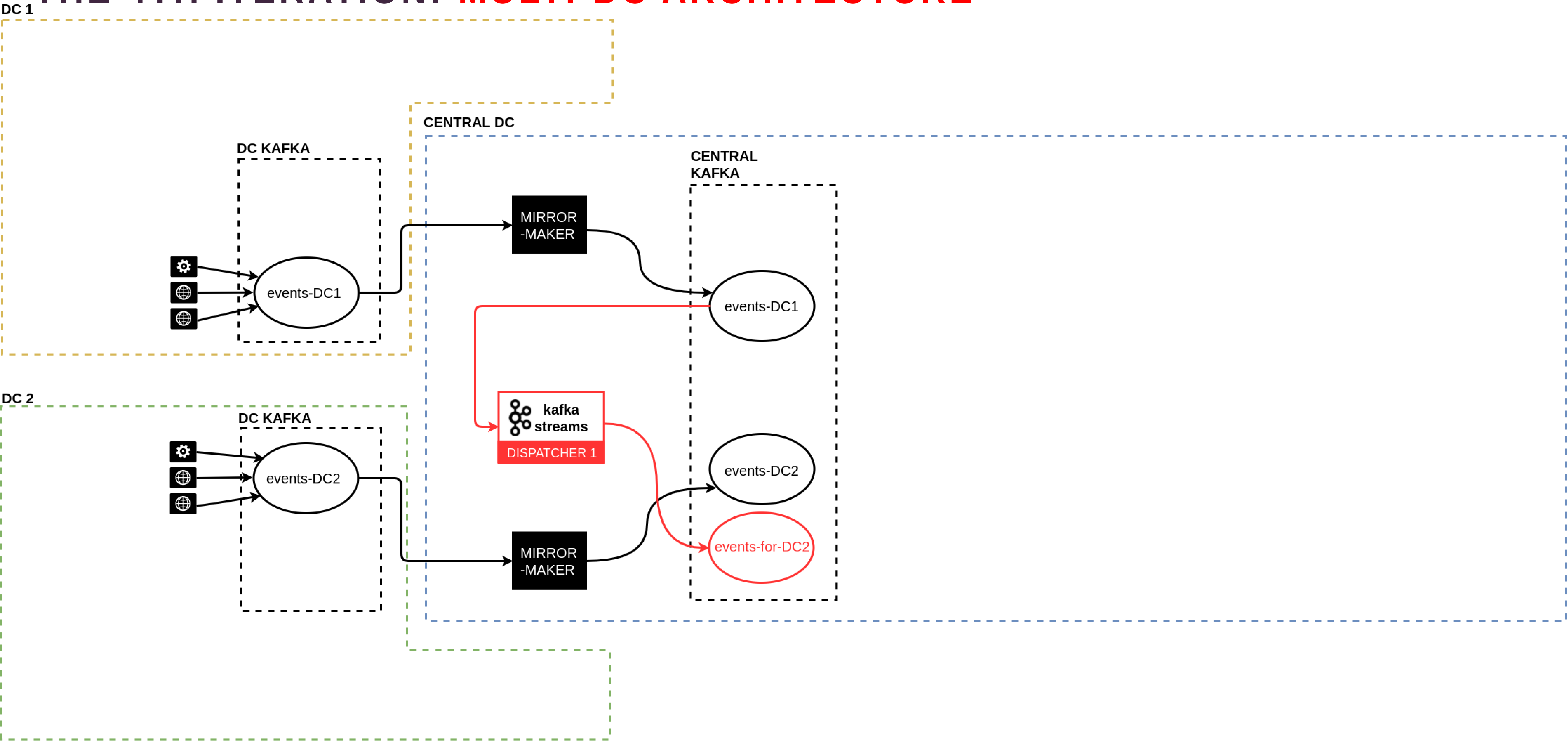
THE 4TH ITERATION: MULTI-DC ARCHITECTURE



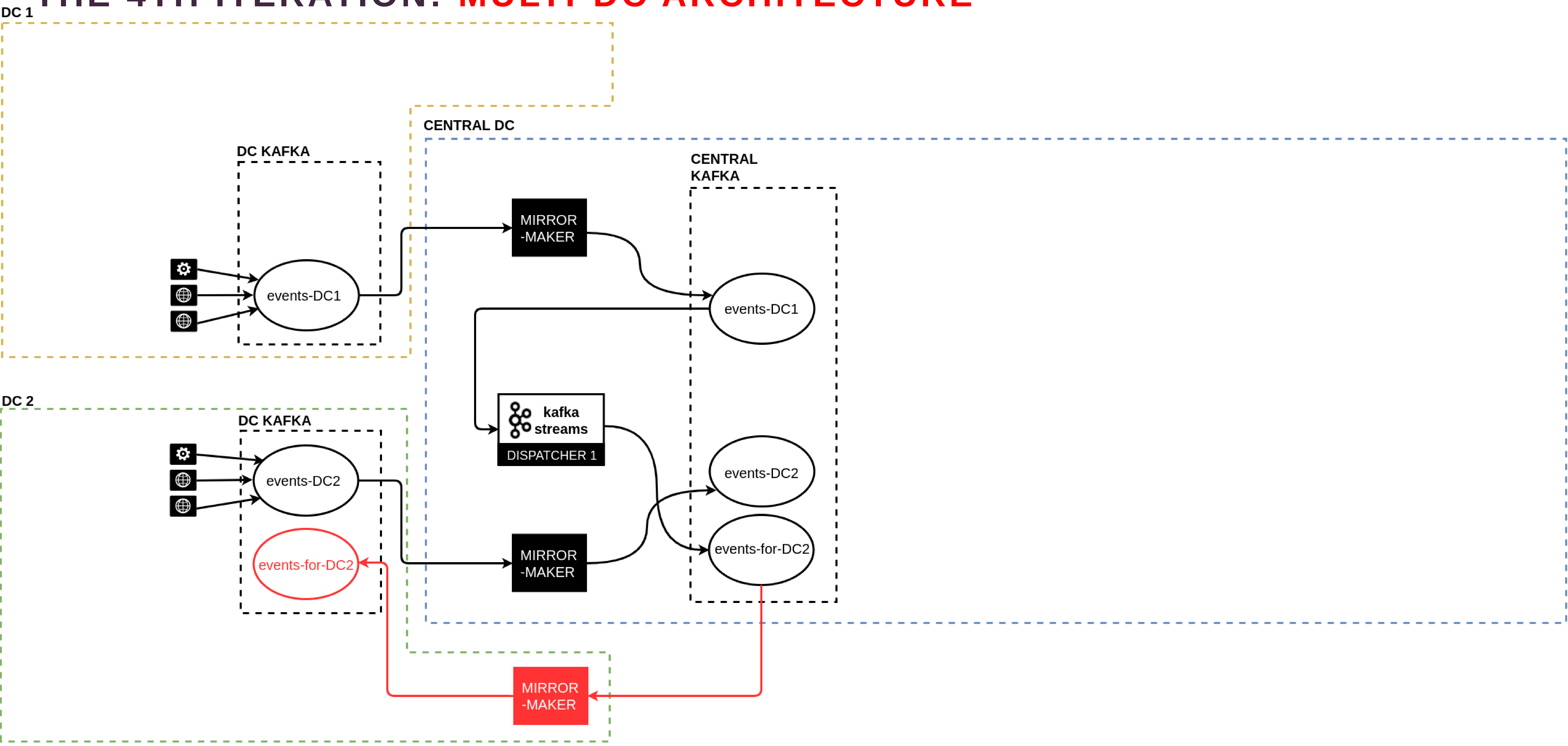
THE 4TH ITERATION: MULTI-DC ARCHITECTURE



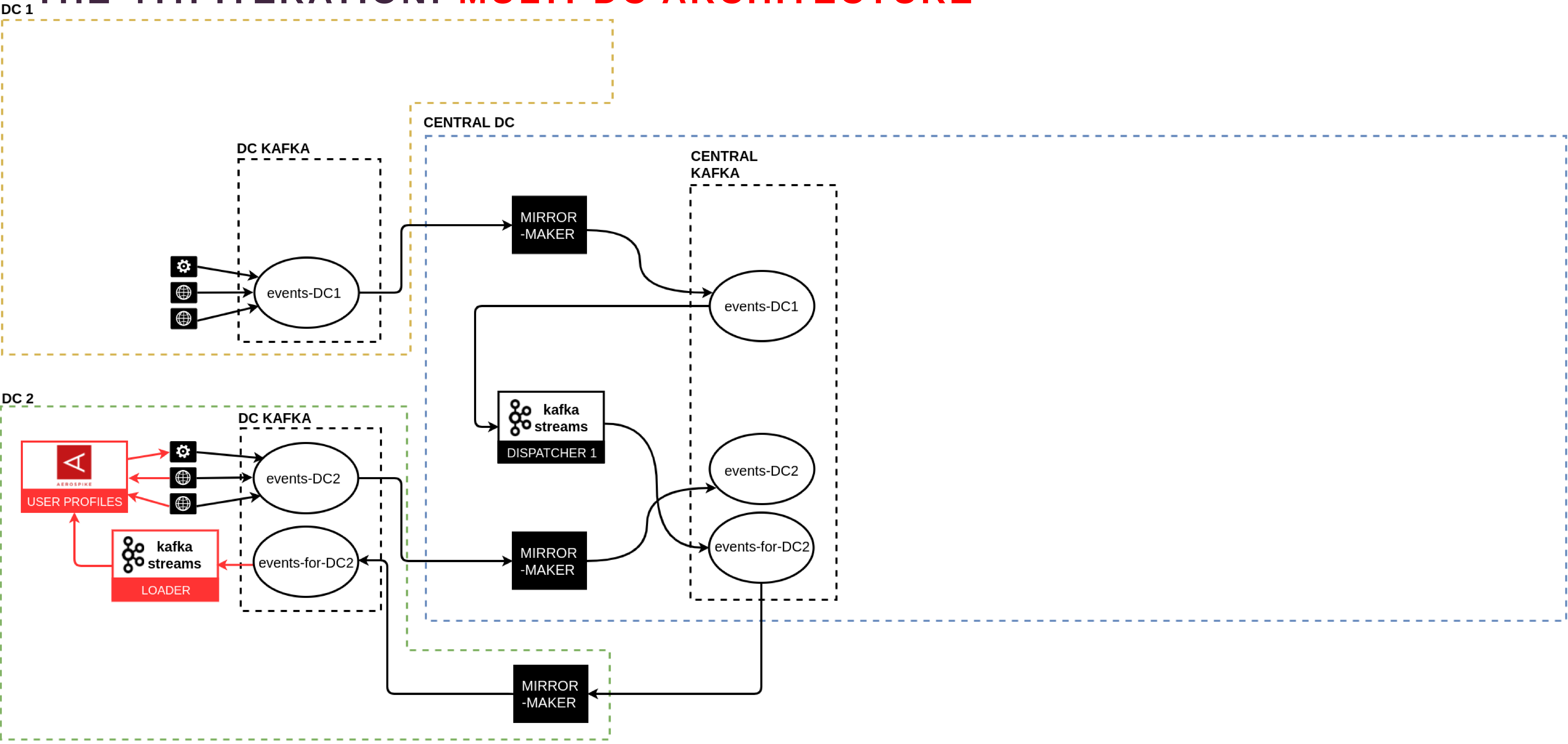
THE 4TH ITERATION: MULTI-DC ARCHITECTURE



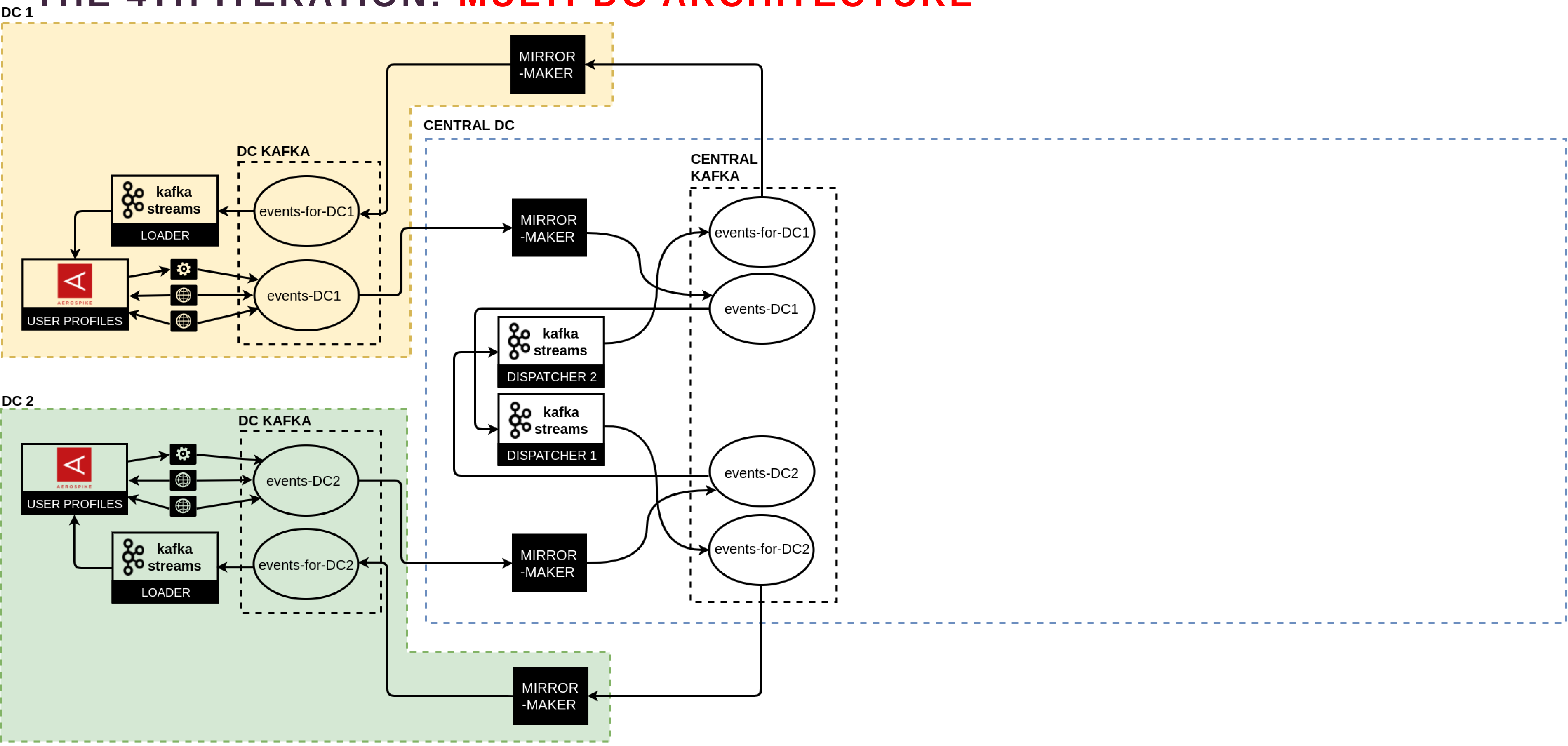
THE 4TH ITERATION: MULTI-DC ARCHITECTURE



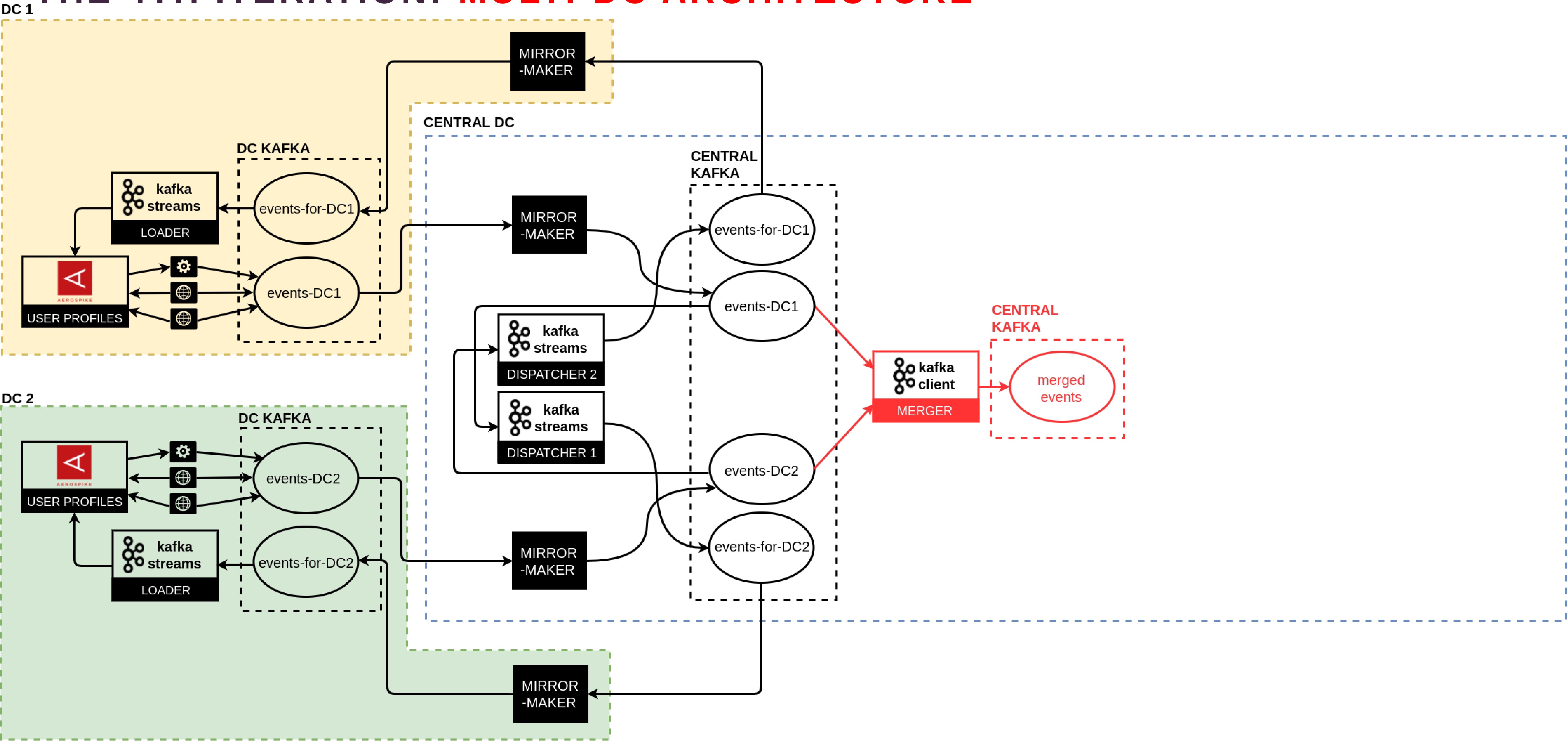
THE 4TH ITERATION: MULTI-DC ARCHITECTURE



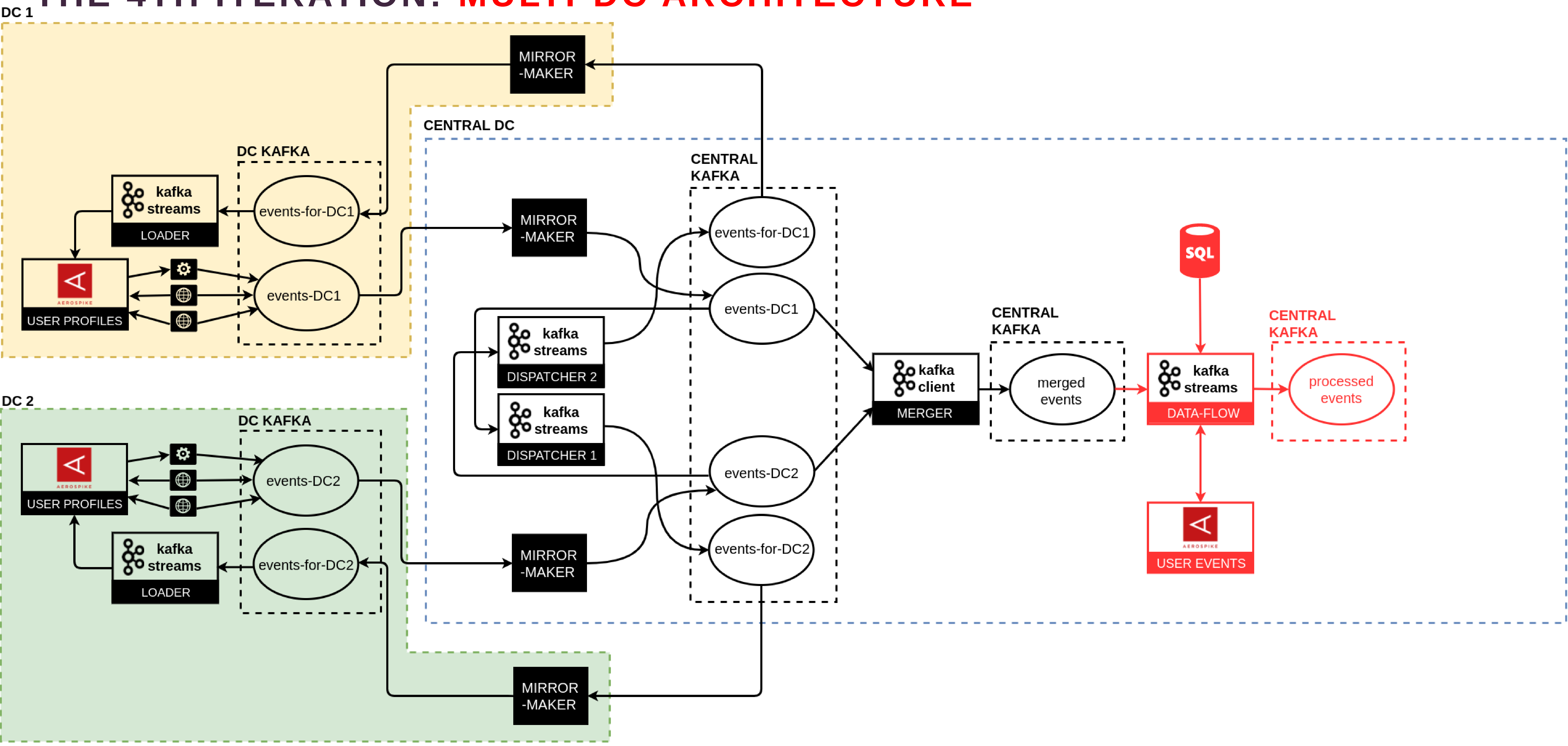
THE 4TH ITERATION: MULTI-DC ARCHITECTURE



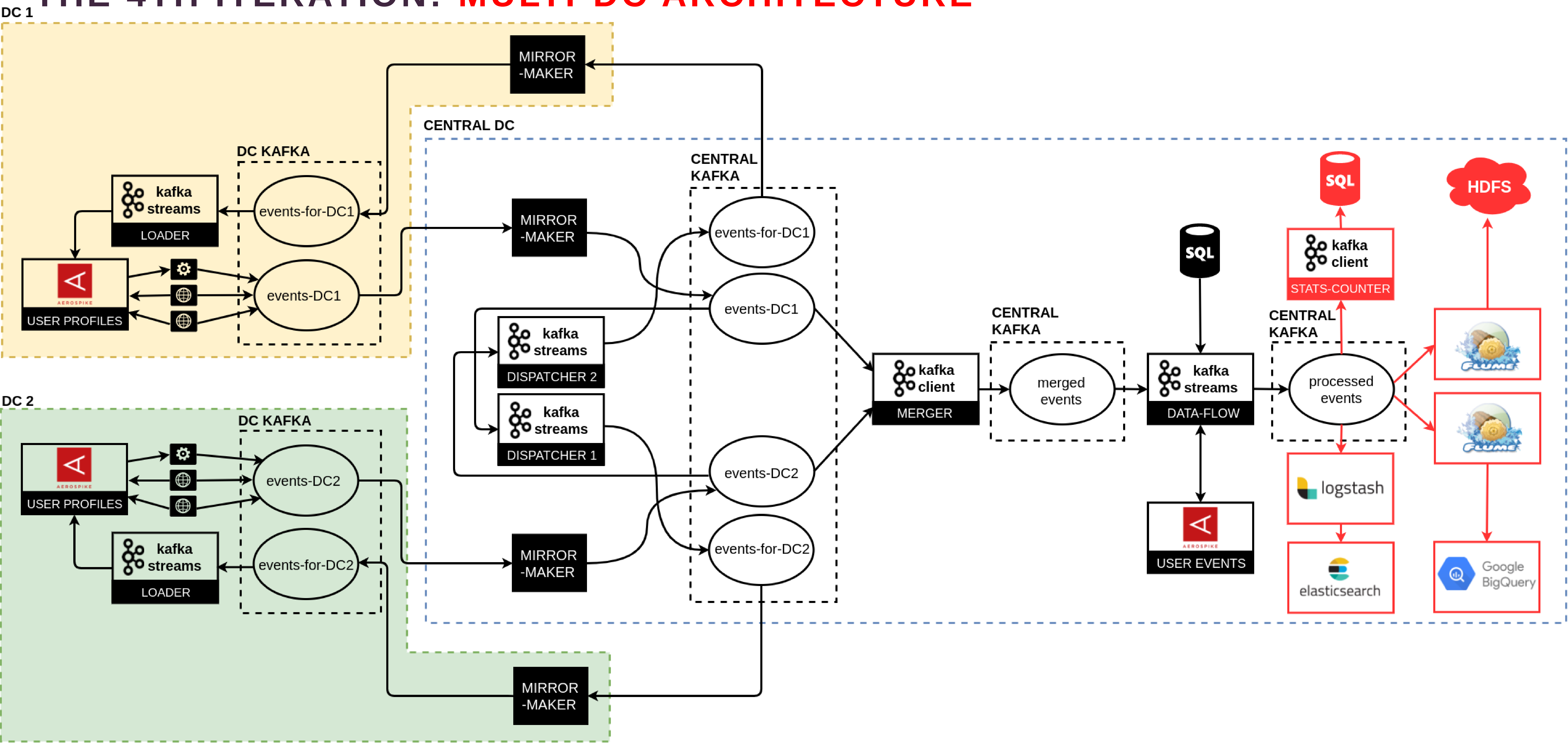
THE 4TH ITERATION: MULTI-DC ARCHITECTURE



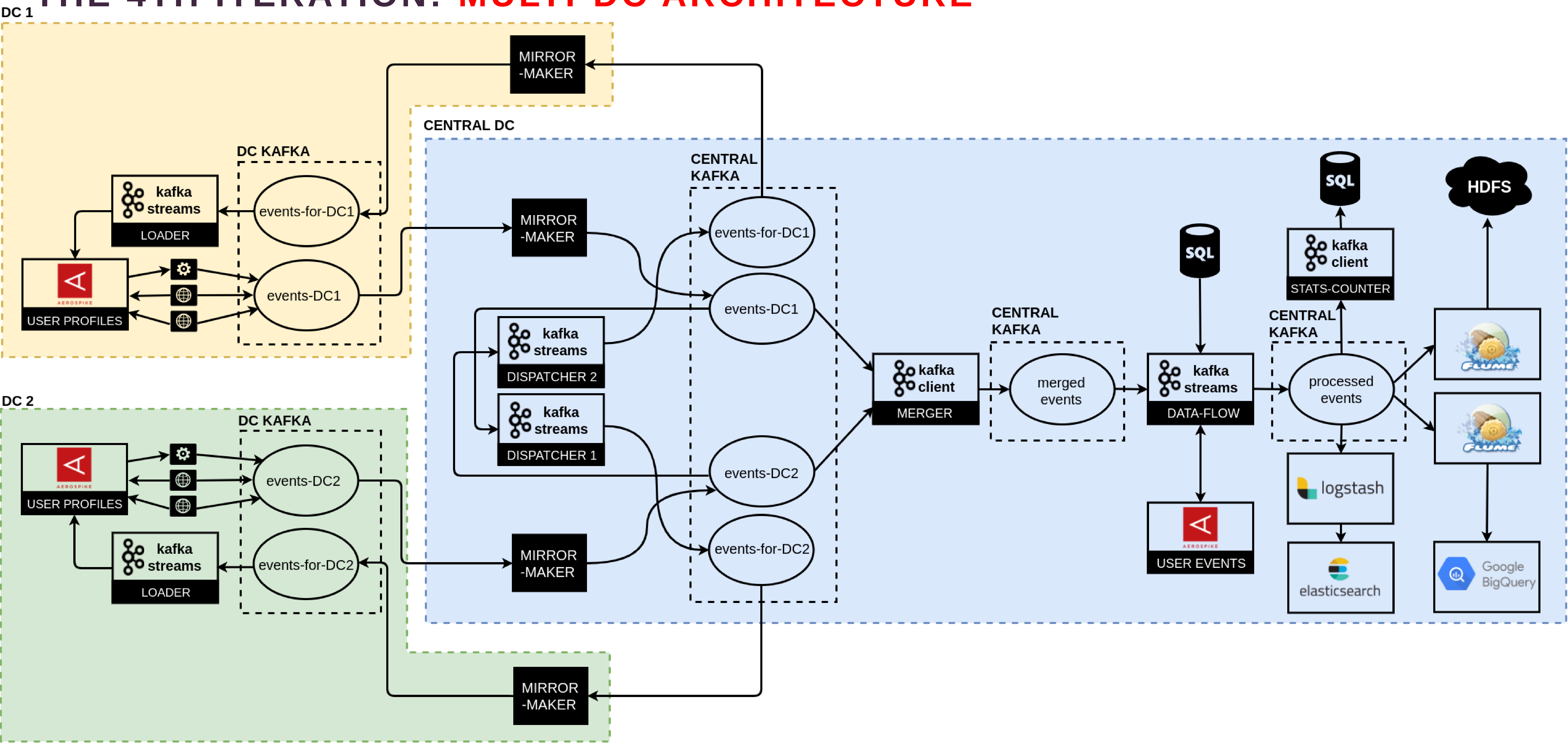
THE 4TH ITERATION: MULTI-DC ARCHITECTURE



THE 4TH ITERATION: MULTI-DC ARCHITECTURE

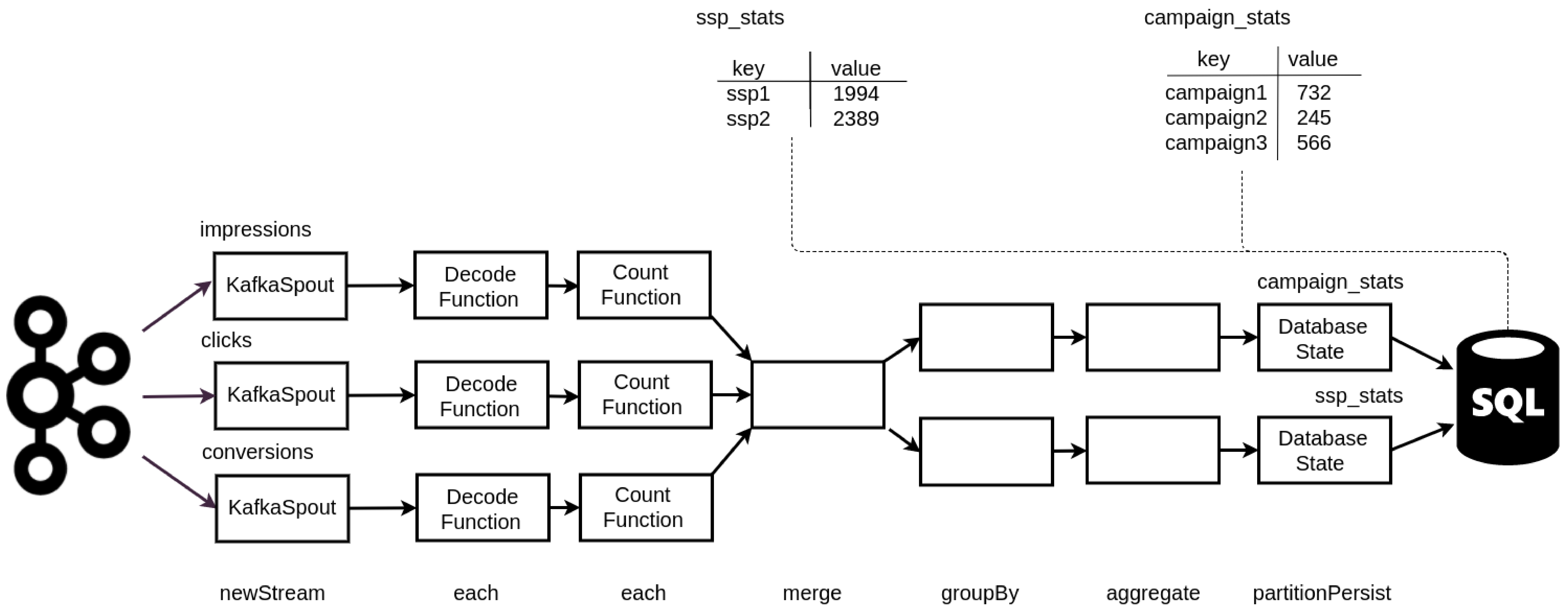


THE 4TH ITERATION: MULTI-DC ARCHITECTURE



OUR USE CASES

STATS-COUNTER: STORM TOPOLOGY (THE 2ND ITERATION)



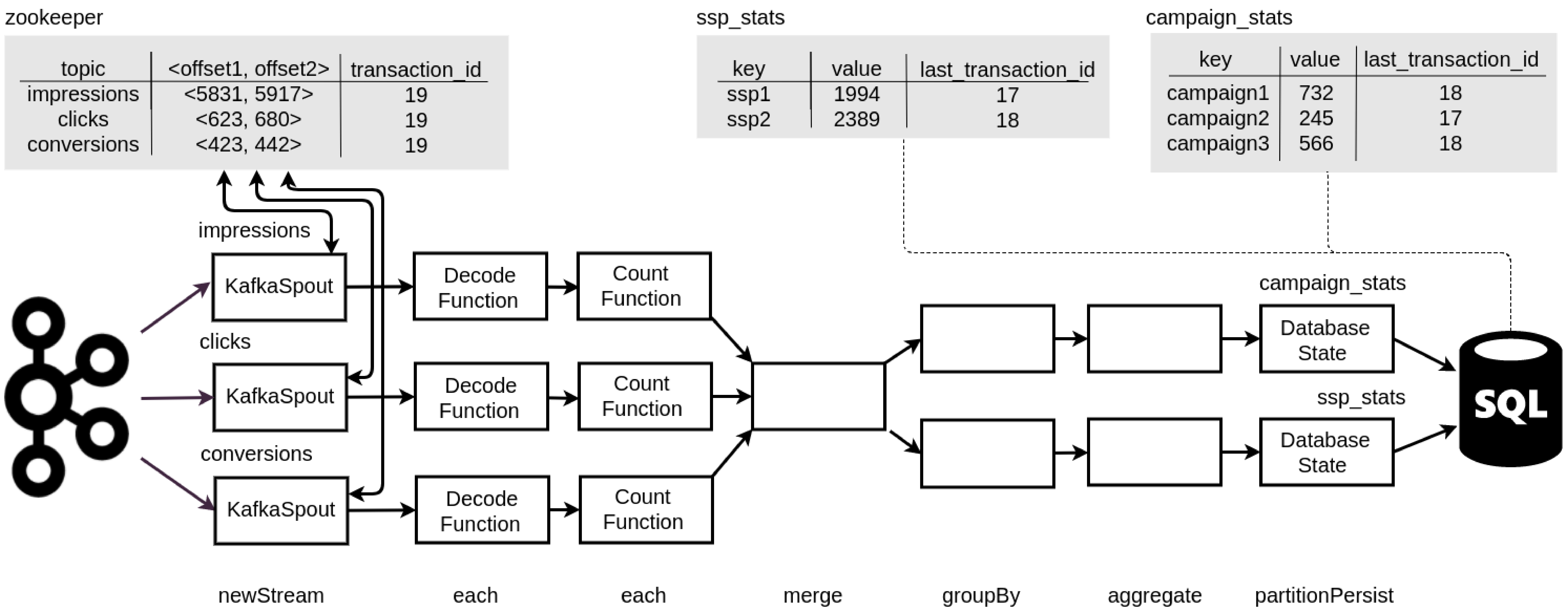
ssp_stats

key	value
ssp1	1994
ssp2	2389

campaign_stats

key	value
campaign1	732
campaign2	245
campaign3	566

APACHE STORM: TRIDENT + EXACTLY-ONCE STATE



zookeeper

topic	<offset1, offset2>	transaction_id
impressions	<5831, 5917>	19
clicks	<623, 680>	19
conversions	<423, 442>	19

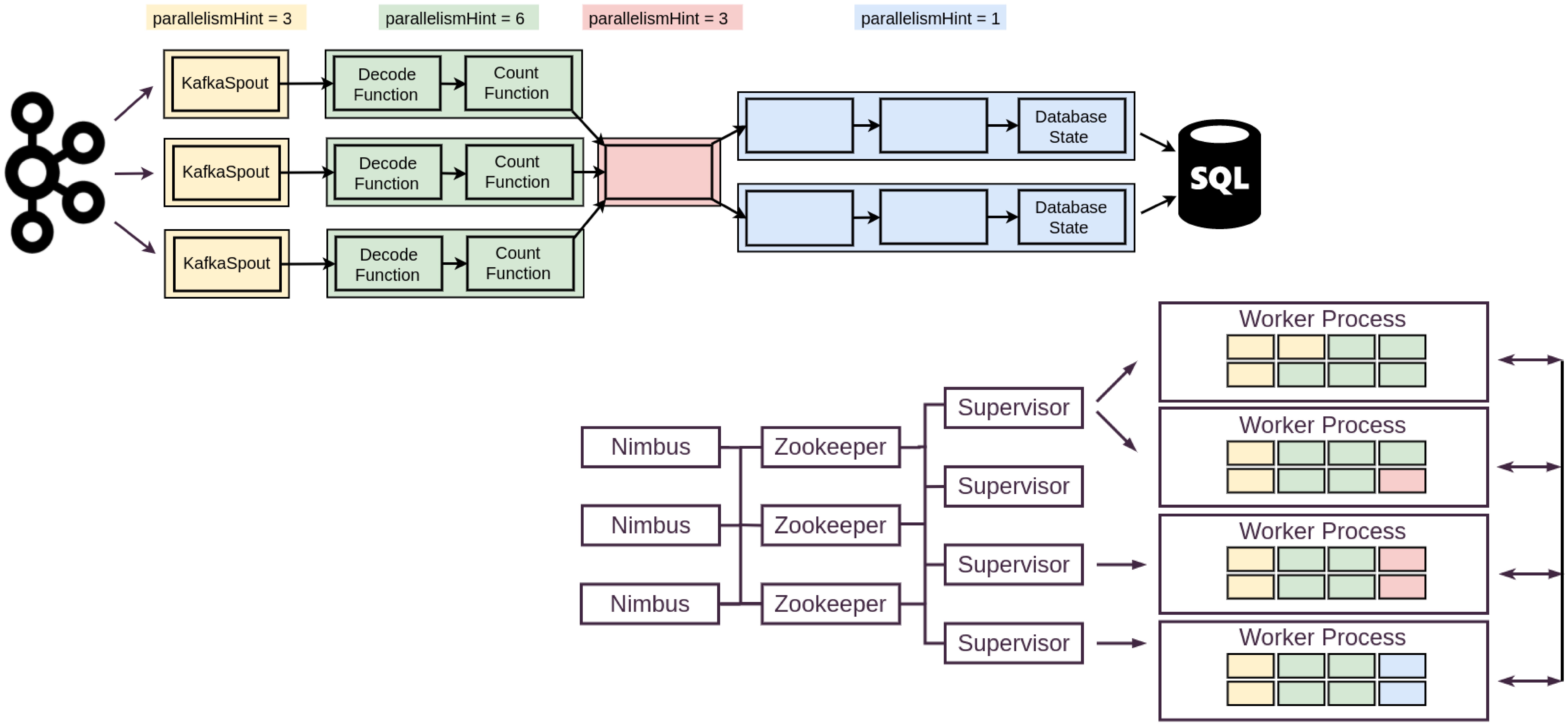
ssp_stats

key	value	last_transaction_id
ssp1	1994	17
ssp2	2389	18

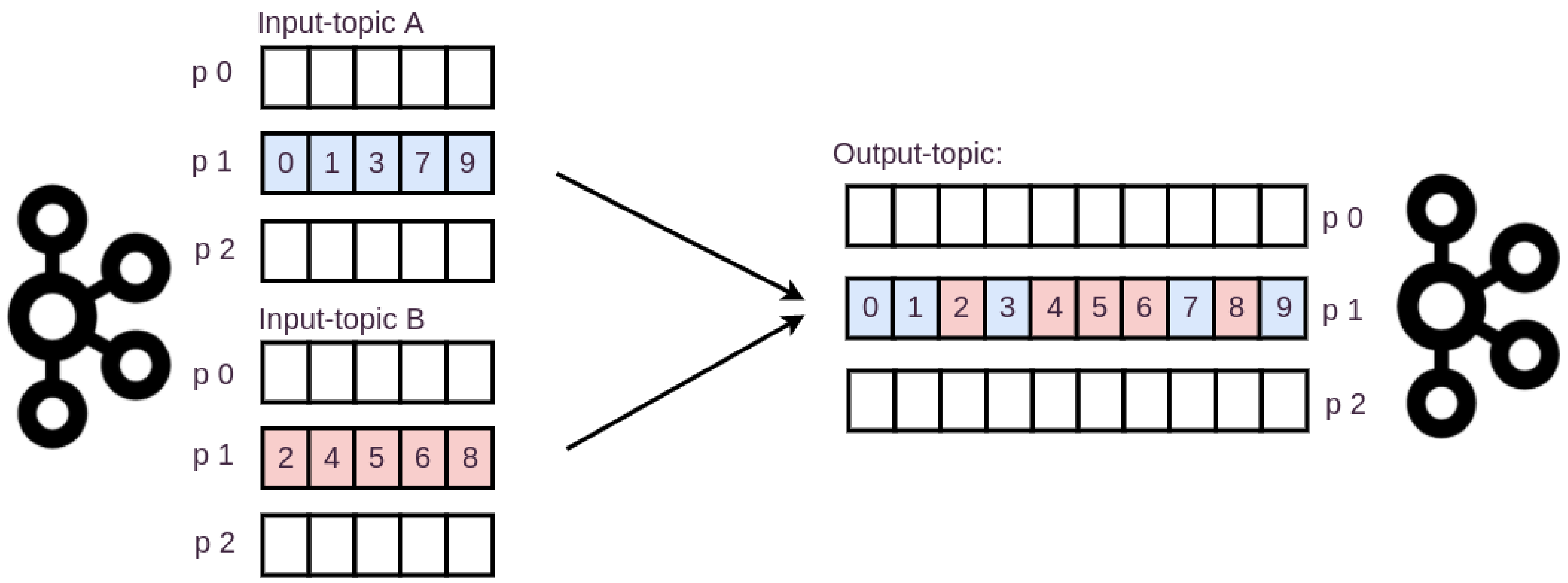
campaign_stats

key	value	last_transaction_id
campaign1	732	18
campaign2	245	17
campaign3	566	18

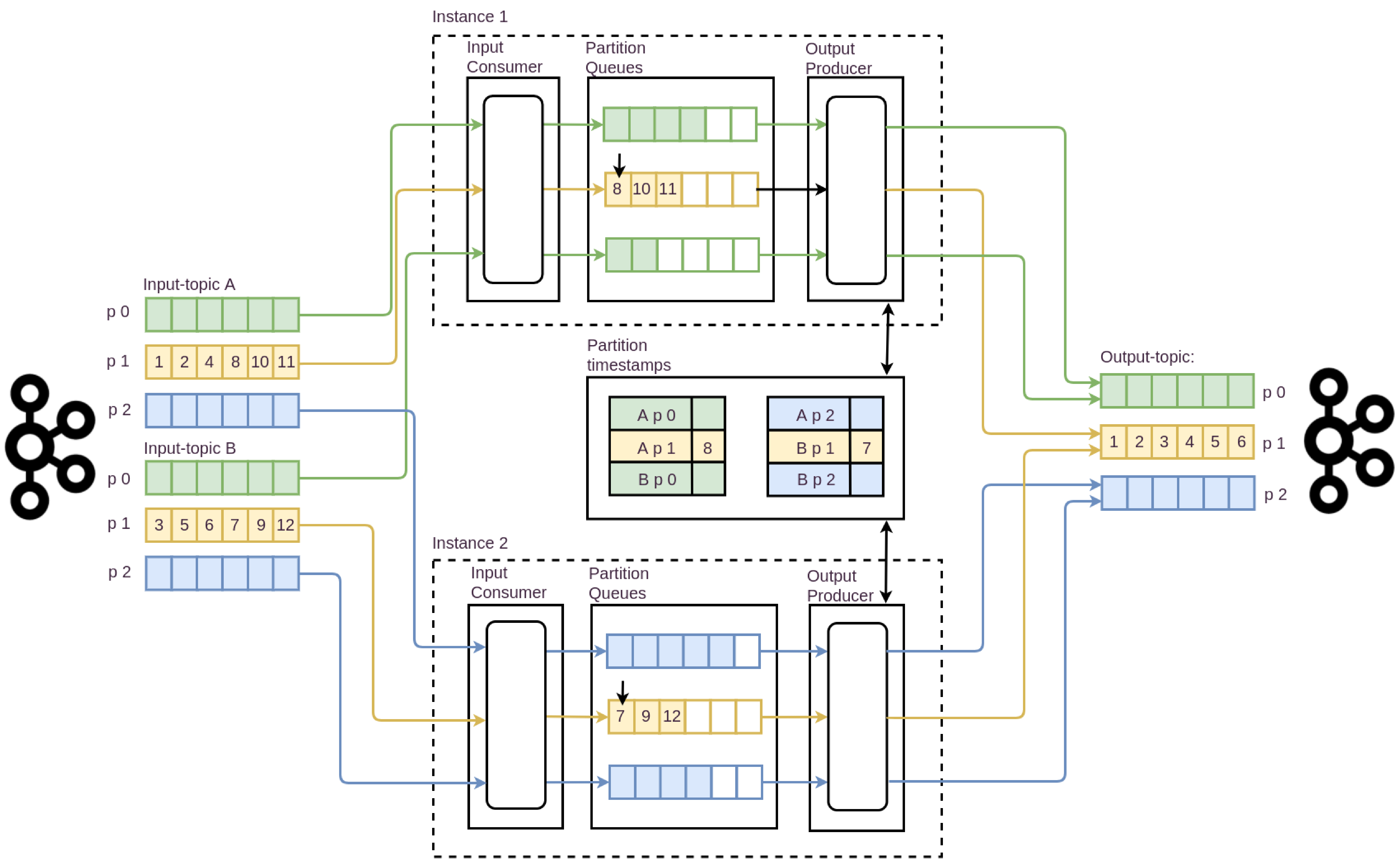
APACHE STORM: PARALLELISM MODEL



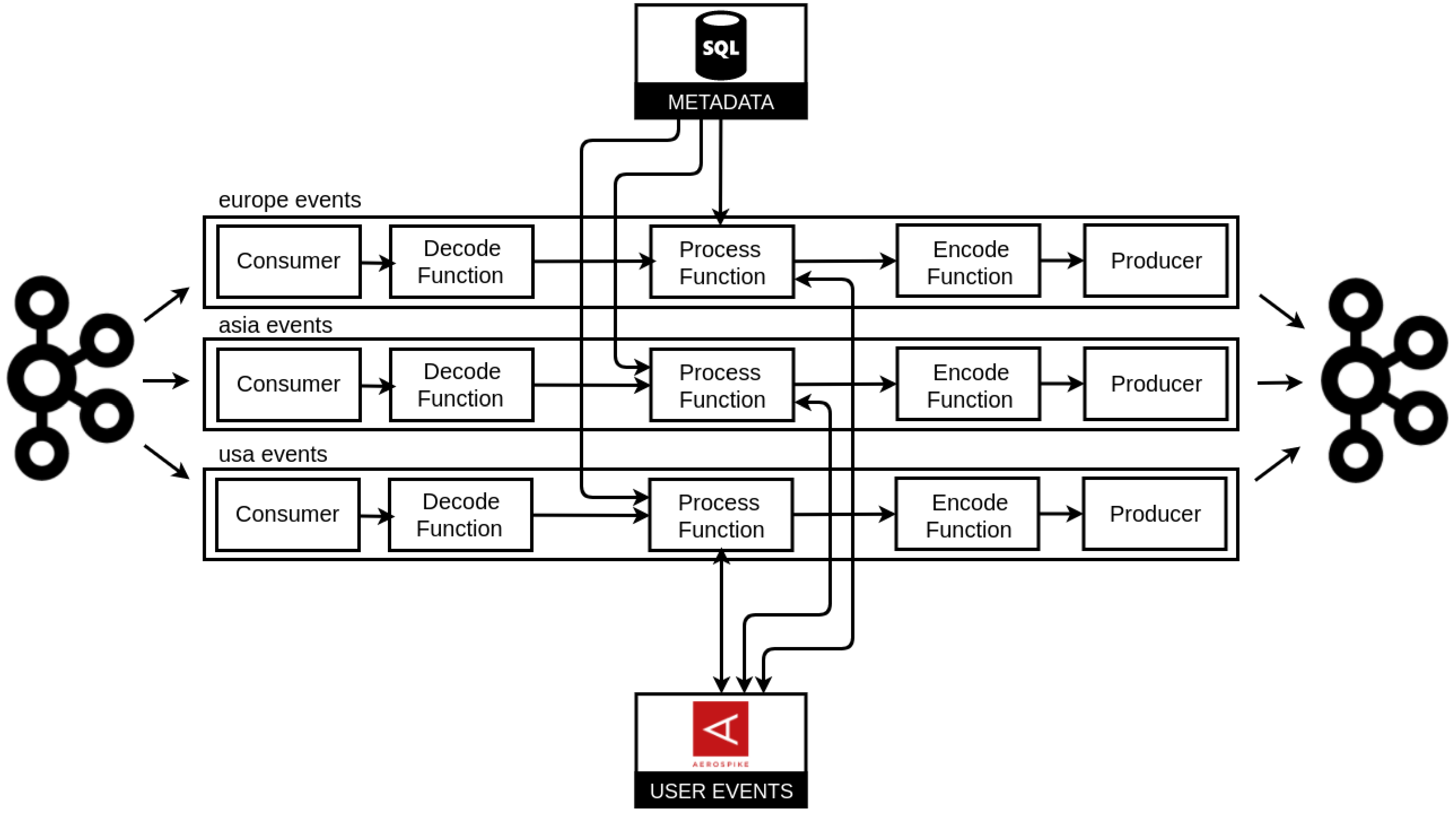
MERGER (THE 4TH ITERATION)



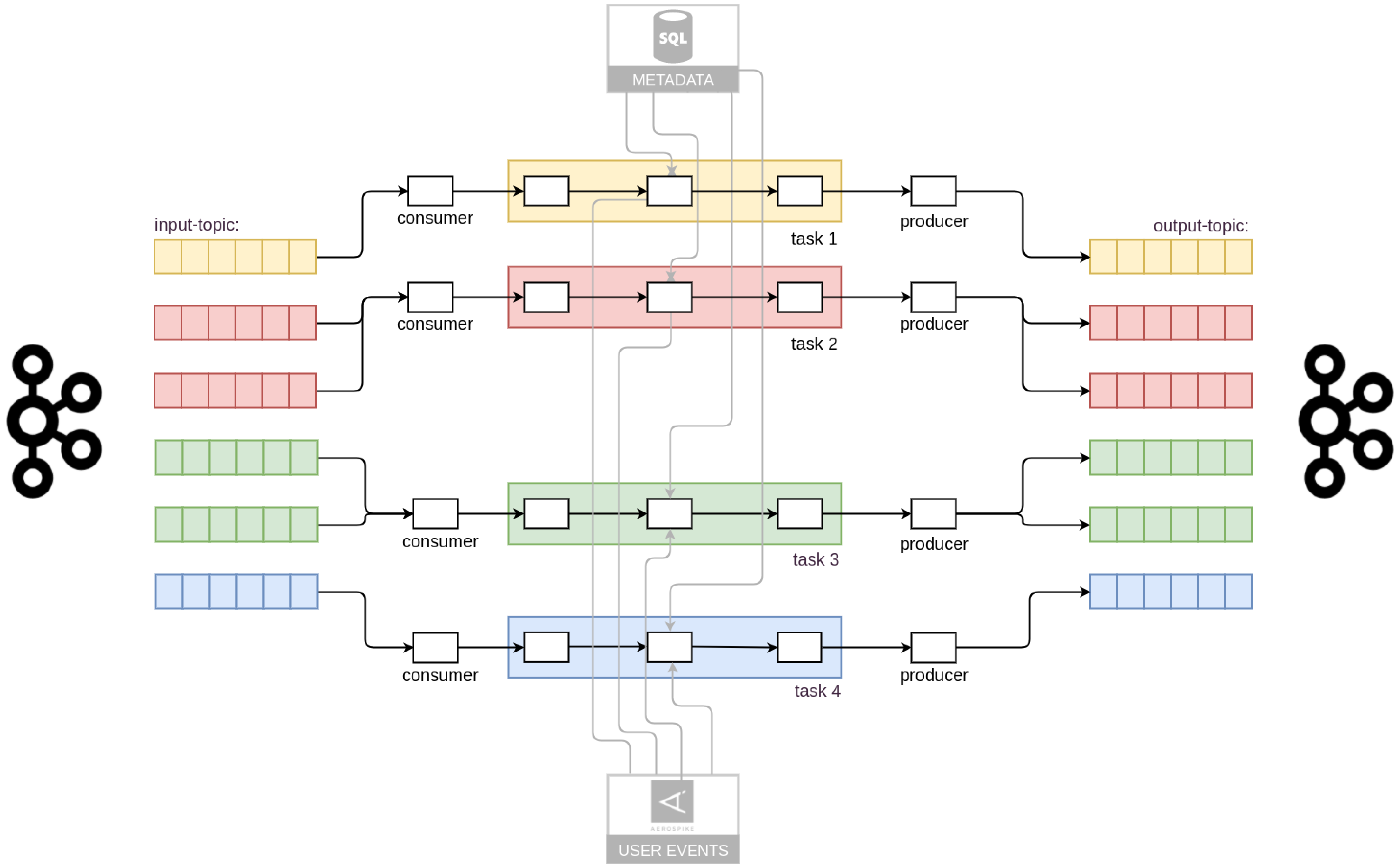
MERGER: KAFKA CONSUMER API



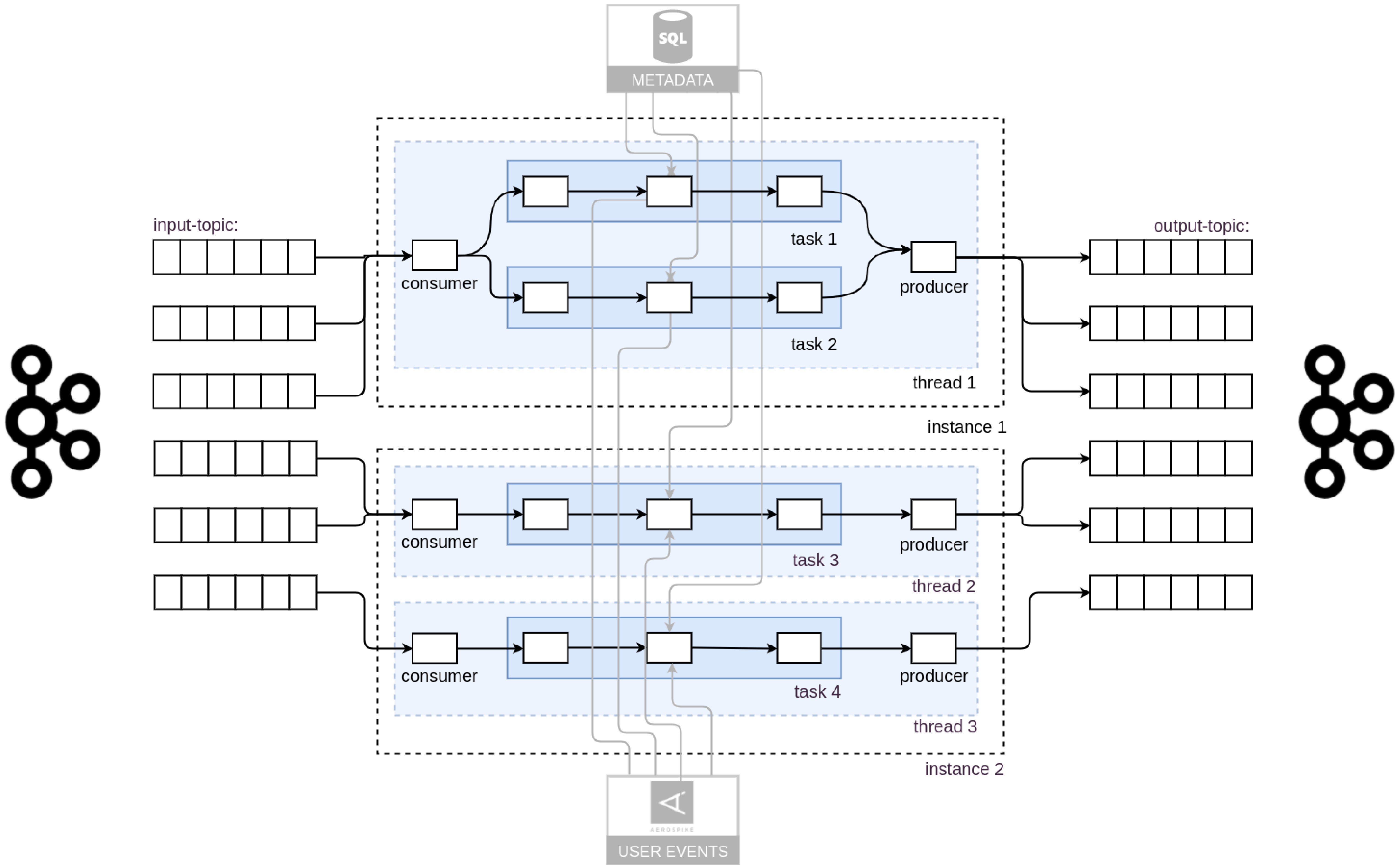
DATA-FLOW: KAFKA STREAMS (THE 4TH ITERATION)



KAFKA STREAMS: PARALLELISM MODEL



KAFKA STREAMS: PARALLELISM MODEL



KAFKA STREAMS: EXACTLY-ONCE DELIVERY

Kafka Streams:

- processing.guarantee = exactly-once

KAFKA STREAMS: EXACTLY-ONCE DELIVERY

Kafka Streams:

- processing.guarantee = exactly-once

Producer:

- transactions
- enable.idempotence = true

KAFKA STREAMS: EXACTLY-ONCE DELIVERY

Kafka Streams:

- processing.guarantee = exactly-once

Producer:

- transactions
- enable.idempotence = true

Consumer:

- isolation.level = read_committed

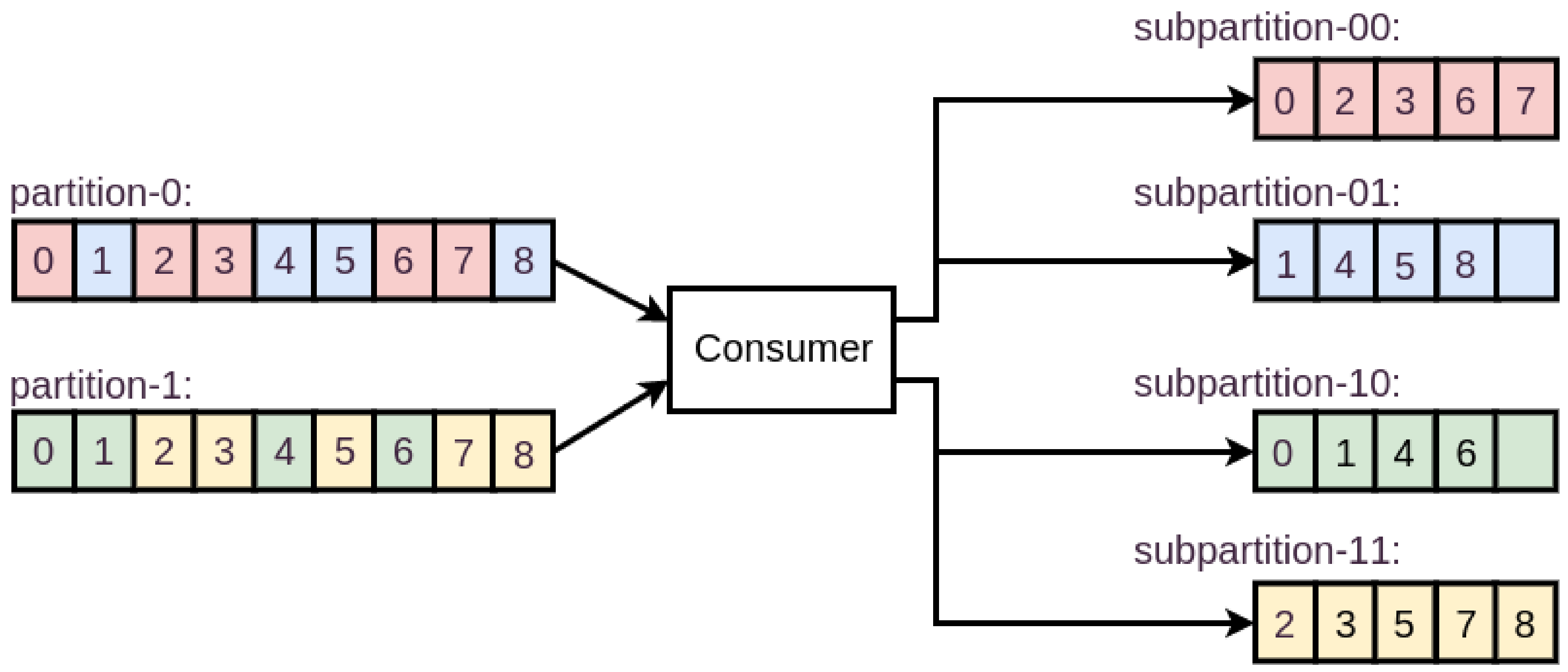
KAFKA WORKERS

KAFKA WORKERS: MAIN FEATURES

- higher level of distribution

KAFKA WORKERS: MAIN FEATURES

- higher level of distribution



KAFKA WORKERS: MAIN FEATURES

- higher level of distribution

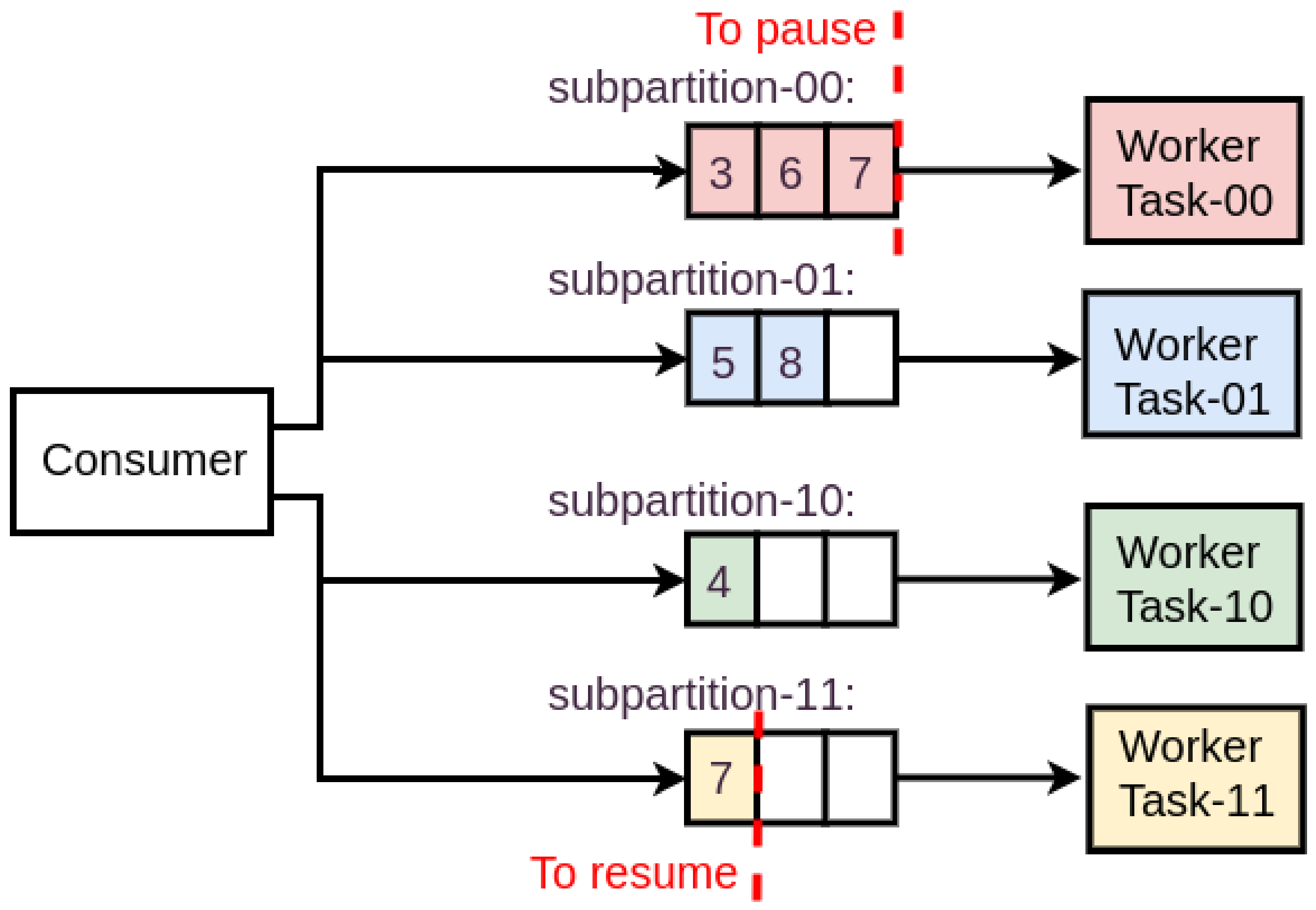
```
public interface WorkerPartitioner<K, V> {

    int subpartition(ConsumerRecord<K, V> consumerRecord);

}
```

KAFKA WORKERS: MAIN FEATURES

- higher level of distribution
- possibility to pause and resume processing for given partition

KAFKA WORKERS: MAIN FEATURES

- higher level of distribution
- possibility to pause and resume processing for given partition



KAFKA WORKERS: MAIN FEATURES

- higher level of distribution
- possibility to pause and resume processing for given partition

```
public interface WorkerTask<K, V> {

    boolean accept(WorkerRecord<K, V> record);

    void process(WorkerRecord<K, V> record, RecordStatusObserver observer);

}
```

KAFKA WORKERS: MAIN FEATURES

- higher level of distribution
- possibility to pause and resume processing for given partition
- asynchronous processing
  - tighter control of offsets commits
  - backpressure
  - processing timeouts

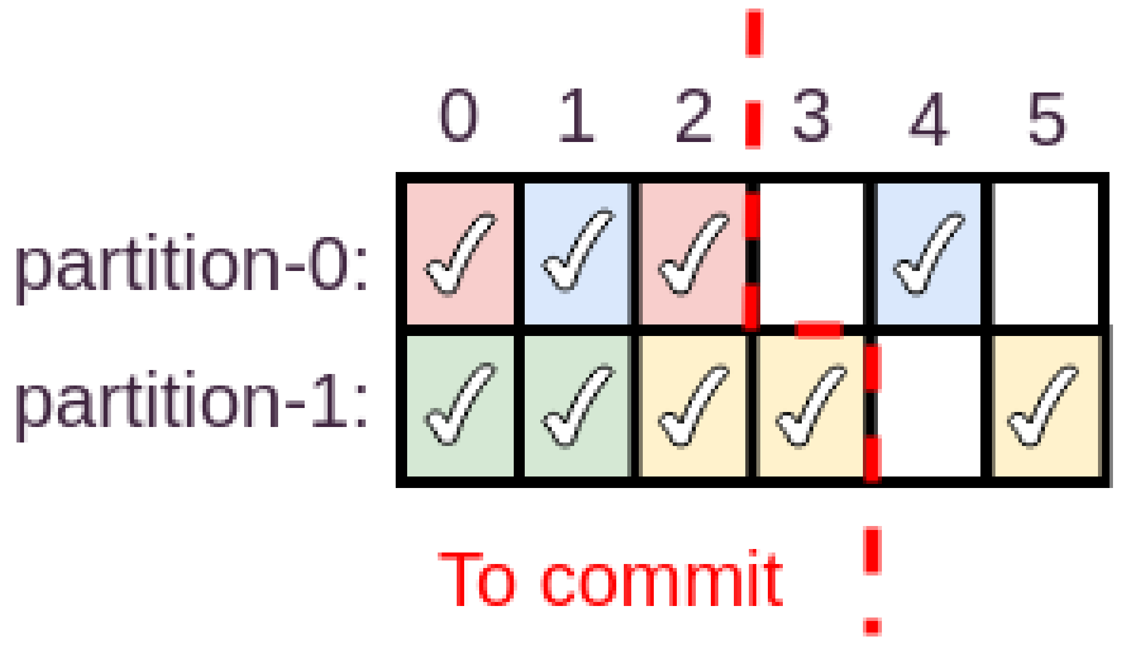
KAFKA WORKERS: MAIN FEATURES

- higher level of distribution
- possibility to pause and resume processing for given partition
- asynchronous processing
  - tighter control of offsets commits
  - backpressure
  - processing timeouts

```
public interface RecordStatusObserver {

    void onSuccess();

    void onFailure(Exception exception);

}
```

KAFKA WORKERS: MAIN FEATURES

- higher level of distribution
- possibility to pause and resume processing for given partition
- asynchronous processing
  - tighter control of offsets commits
  - backpressure
  - processing timeouts



KAFKA WORKERS: MAIN FEATURES

- higher level of distribution
- possibility to pause and resume processing for given partition
- asynchronous processing
  - tighter control of offsets commits
  - backpressure
  - processing timeouts
- at-least-once semantics

KAFKA WORKERS: MAIN FEATURES

- higher level of distribution
- possibility to pause and resume processing for given partition
- asynchronous processing
  - tighter control of offsets commits
  - backpressure
  - processing timeouts
- at-least-once semantics
- handling failures

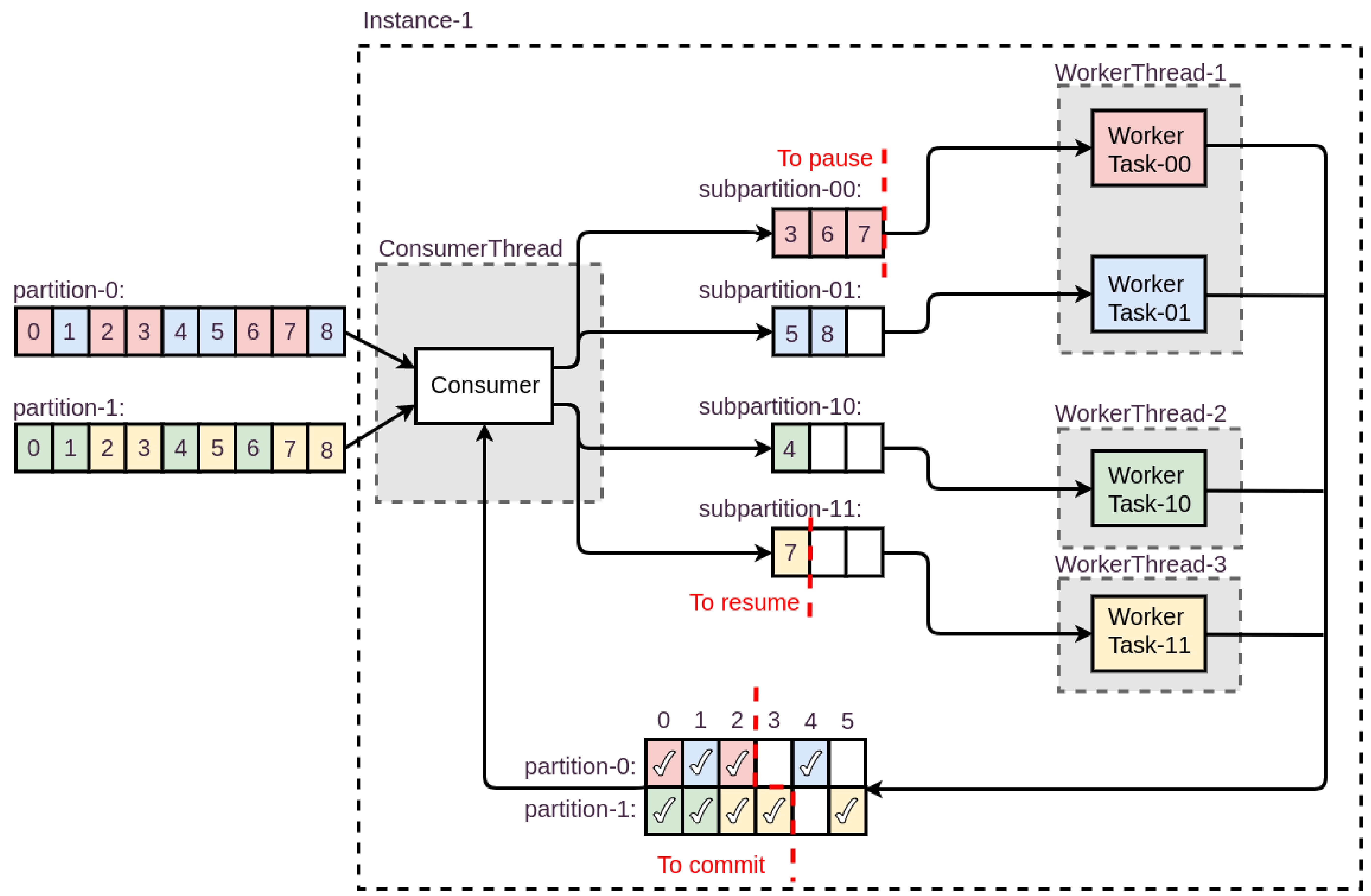
KAFKA WORKERS: MAIN FEATURES

- higher level of distribution
- possibility to pause and resume processing for given partition
- asynchronous processing
    - tighter control of offsets commits
    - backpressure
    - processing timeouts
- at-least-once semantics
- handling failures
- kafka-to-kafka, hdfs, bigquery, elasticsearch connectors

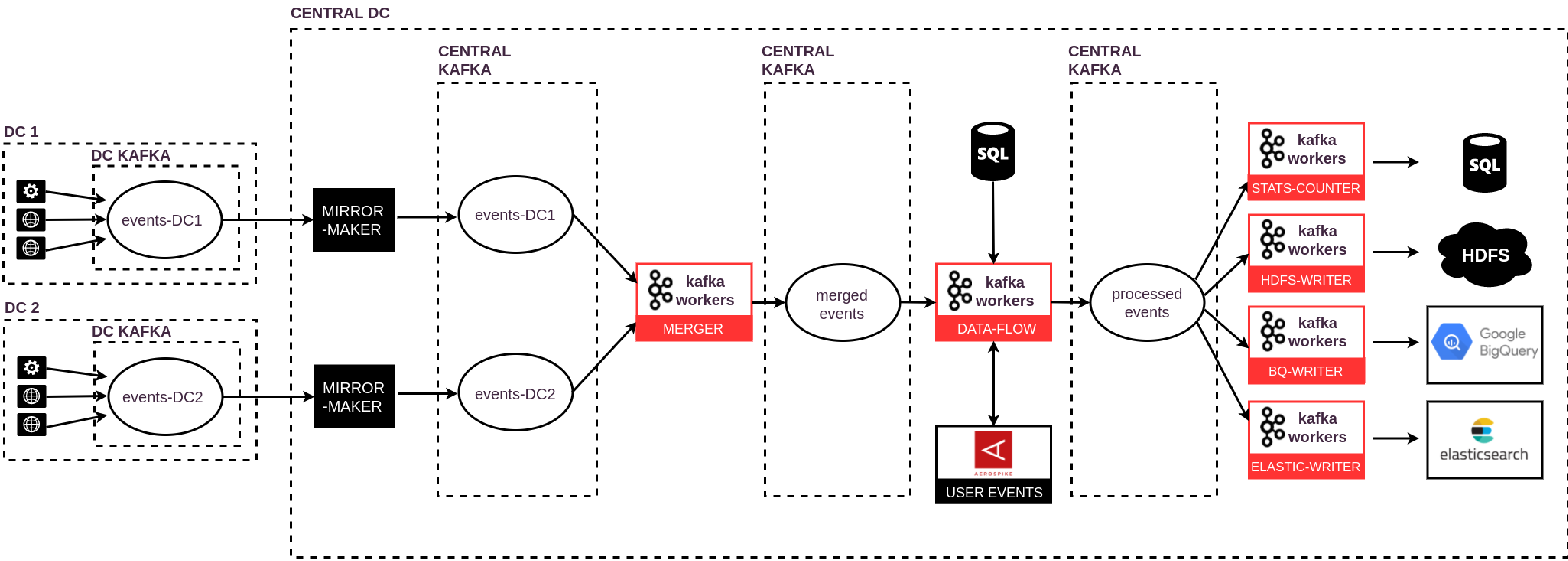
KAFKA WORKERS: MAIN FEATURES

- higher level of distribution
- possibility to pause and resume processing for given partition
- asynchronous processing
  - tighter control of offsets commits
  - backpressure
  - processing timeouts
- at-least-once semantics
- handling failures
- kafka-to-kafka, hdfs, bigquery, elasticsearch connectors
- github.com/RTBHOUSE/kafka-workers

KAFKA WORKERS: PARALLELISM MODEL



THE 5TH ITERATION: KAFKA WORKERS



RTB HOUSE

WE ARE HIRING

techblog.rtbhouse.com/jobs