Introducing Exactly Once Semantics in Apache® Kafka™

Jason Gustafson, Apurva Mehta, Guozhang Wang, and Sriram Subramaniam

Matthias J. Sax | Software Engineer

matthias@confluent.io

@MatthiasJSax

Outline

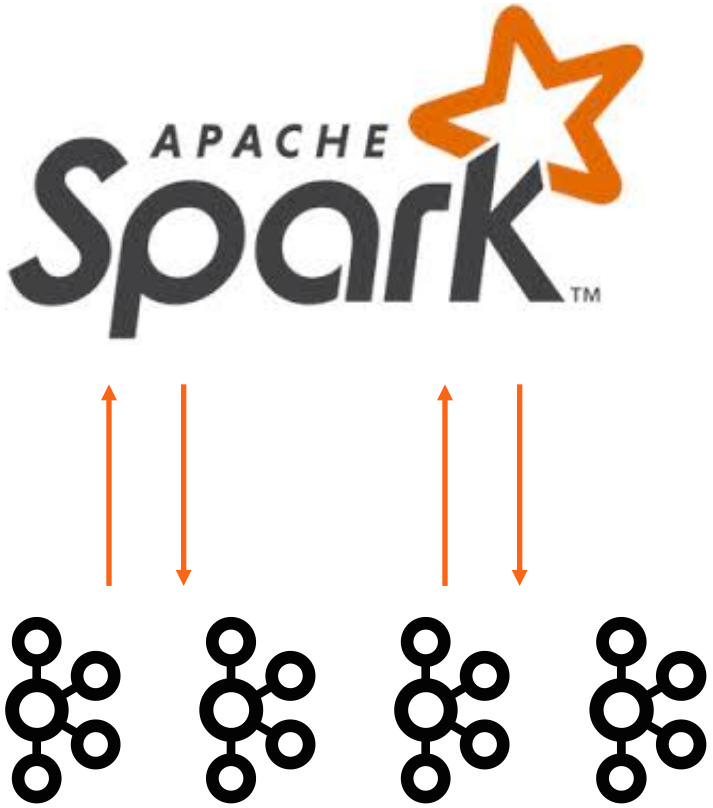
- Kafka's existing delivery semantics.
- What's new?
- How do you use it?
- Summary.

What Kafka offers today

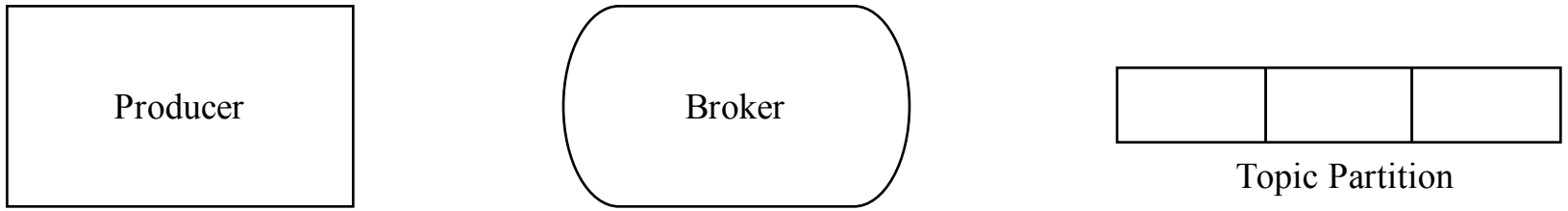
- At-least-once, in-order delivery per partition.
- Producer retries can introduce duplicates.

Stream Processing with Apache Spark

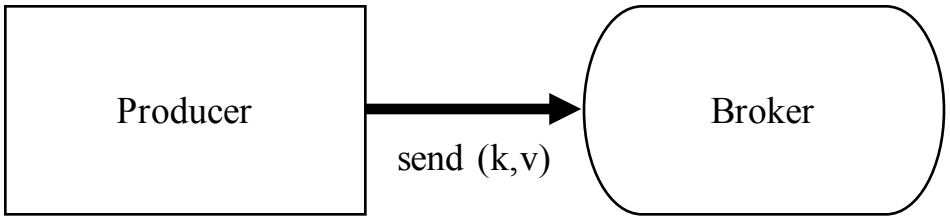
Read-process-write pattern:



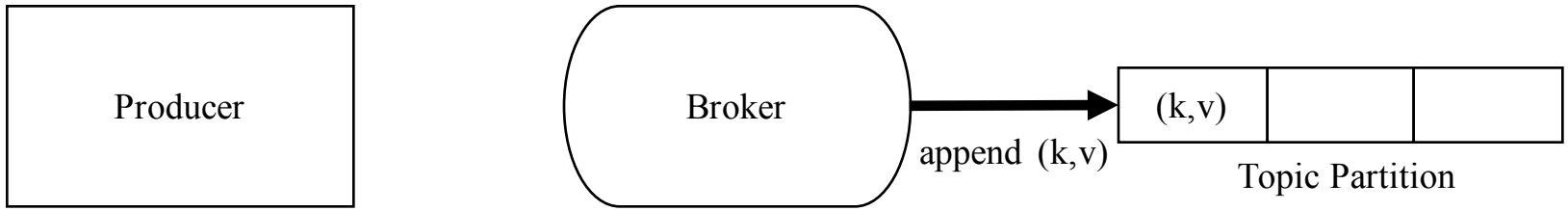
Example: duplicate write



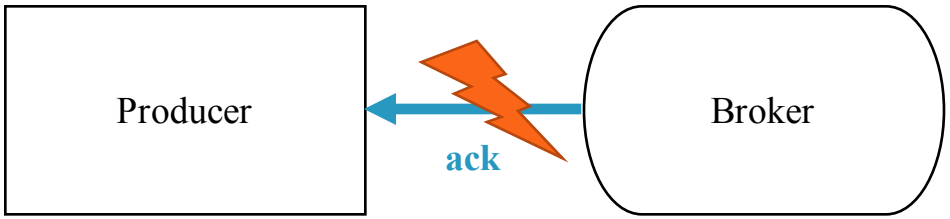
Example: duplicate write



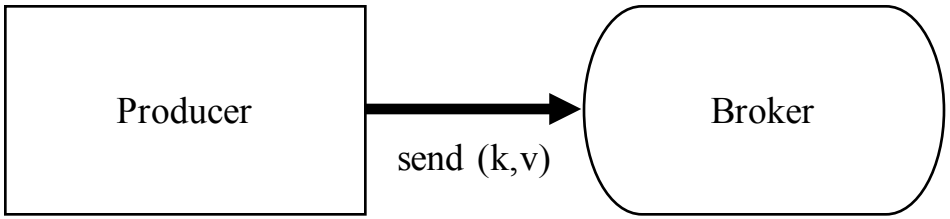
Example: duplicate write



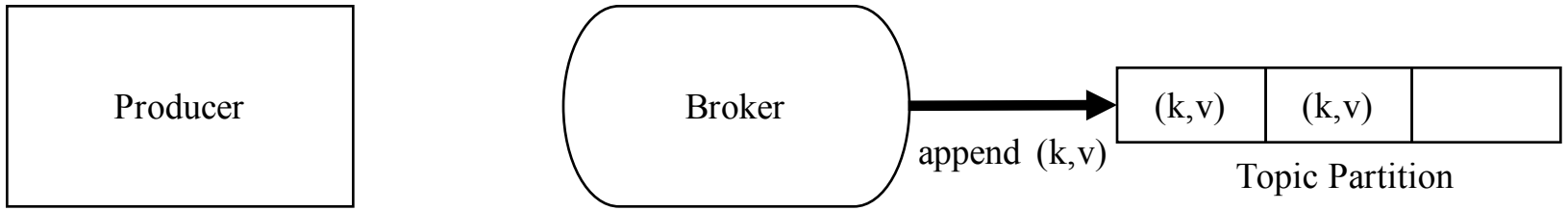
Example: duplicate write



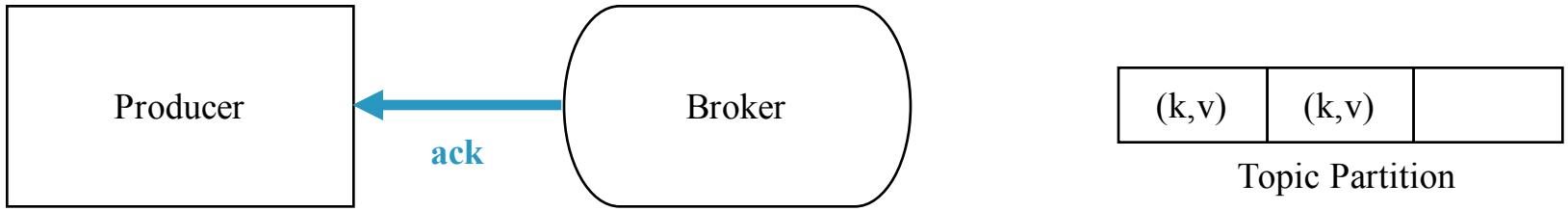
Example: duplicate write



Example: duplicate write



Example: duplicate write



Why improve?

- Stream processing is becoming a bigger part of the data landscape.
- Apache Kafka is the foundation for such stream processing.
- Strengthening Kafka's semantics expands the universe of streaming applications.

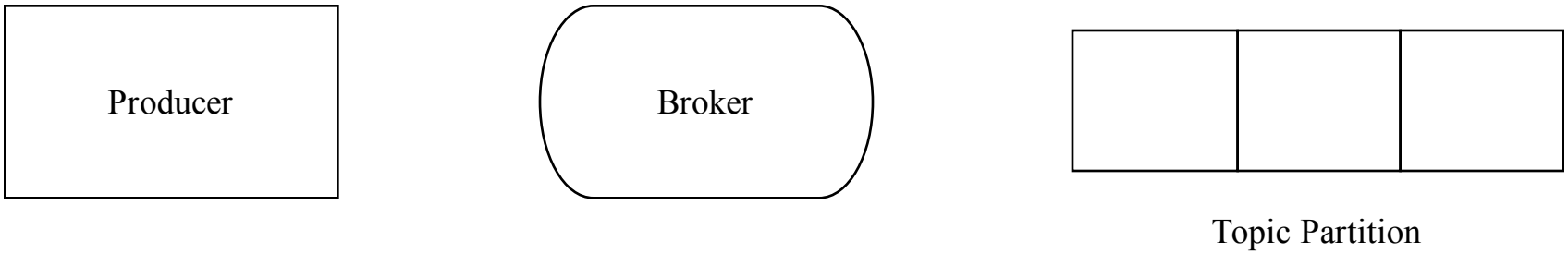
What's new?

What's new

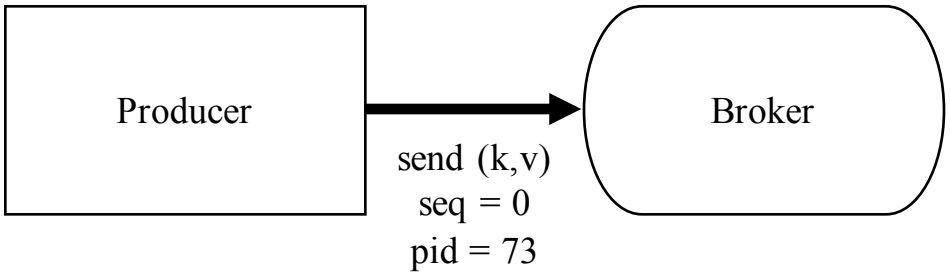
- Idempotent producer: exactly-once writes.
- Transactional producer: Atomic writes across multiple partitions.
- Exactly-once stream processing: read-process-write.

Idempotent Producer

Example: Idempotent Producer

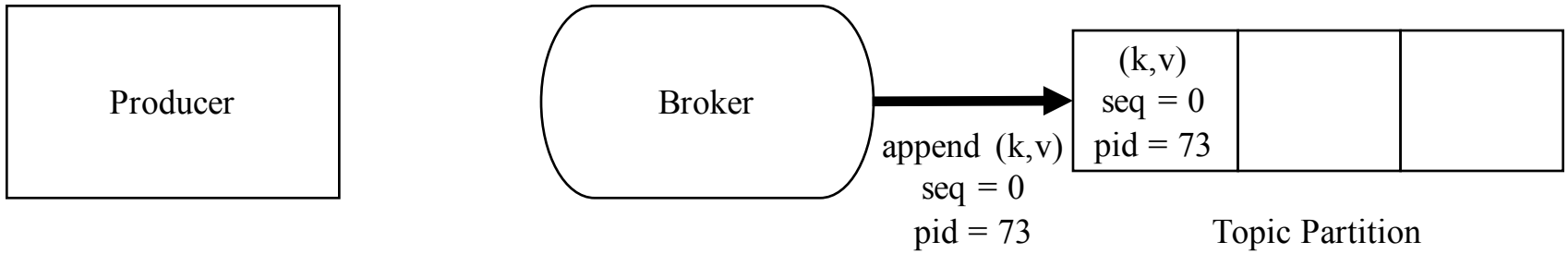


Example: Idempotent Producer

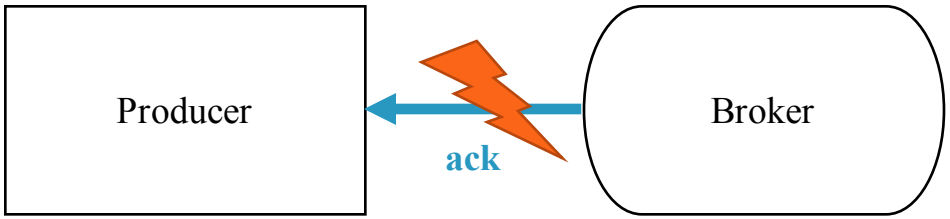


Topic Partition

Example: Idempotent Producer

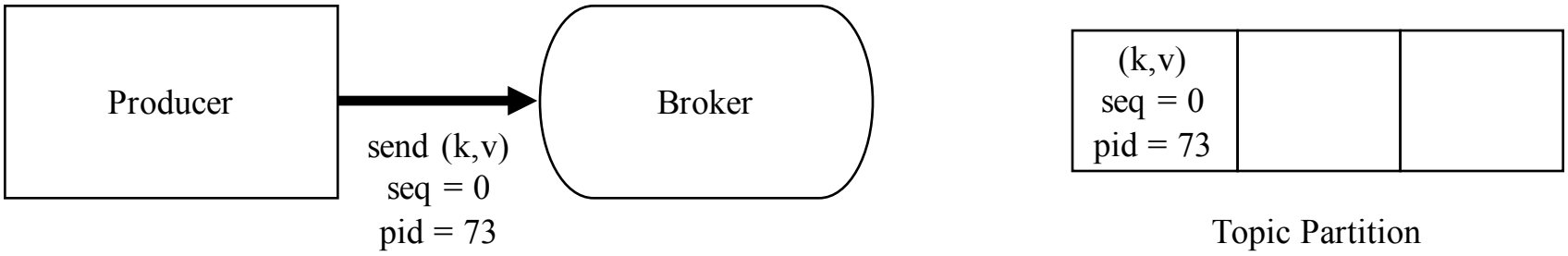


Example: Idempotent Producer

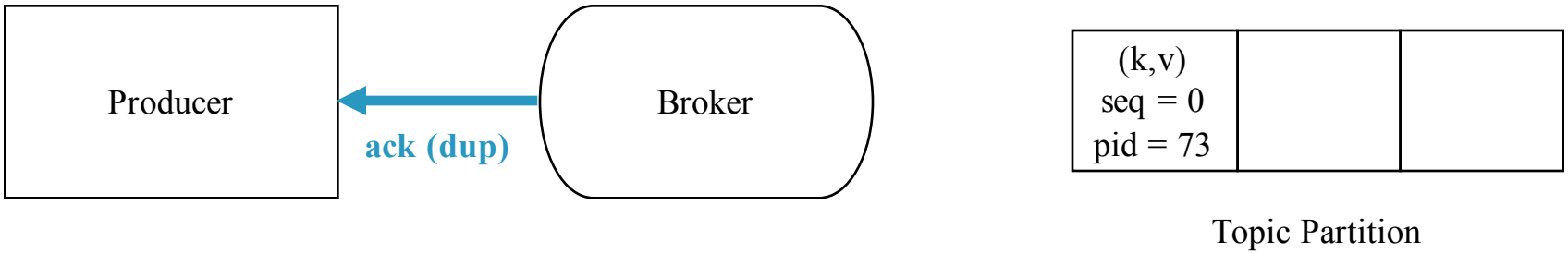


Topic Partition

Example: Idempotent Producer



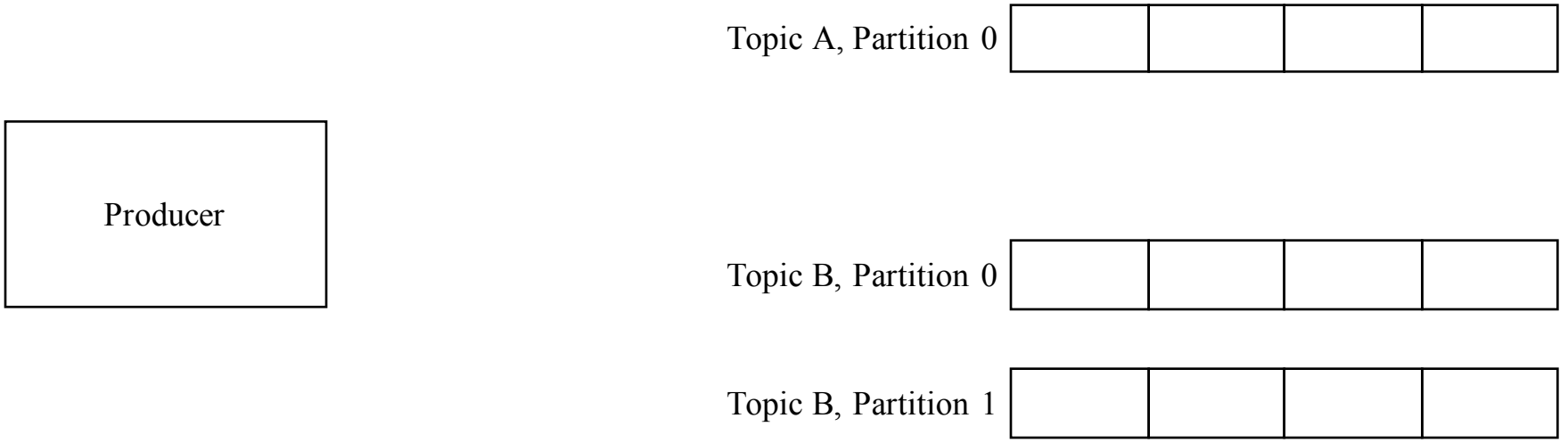
Example: Idempotent Producer



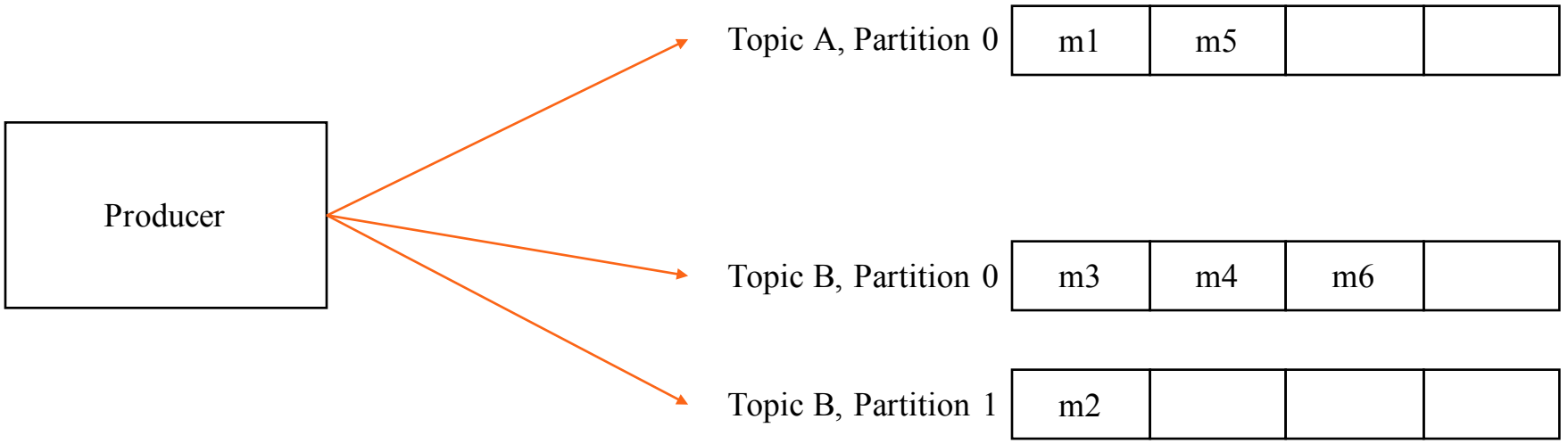
Atomic Multi-Partition Writes

(aka "transactions")

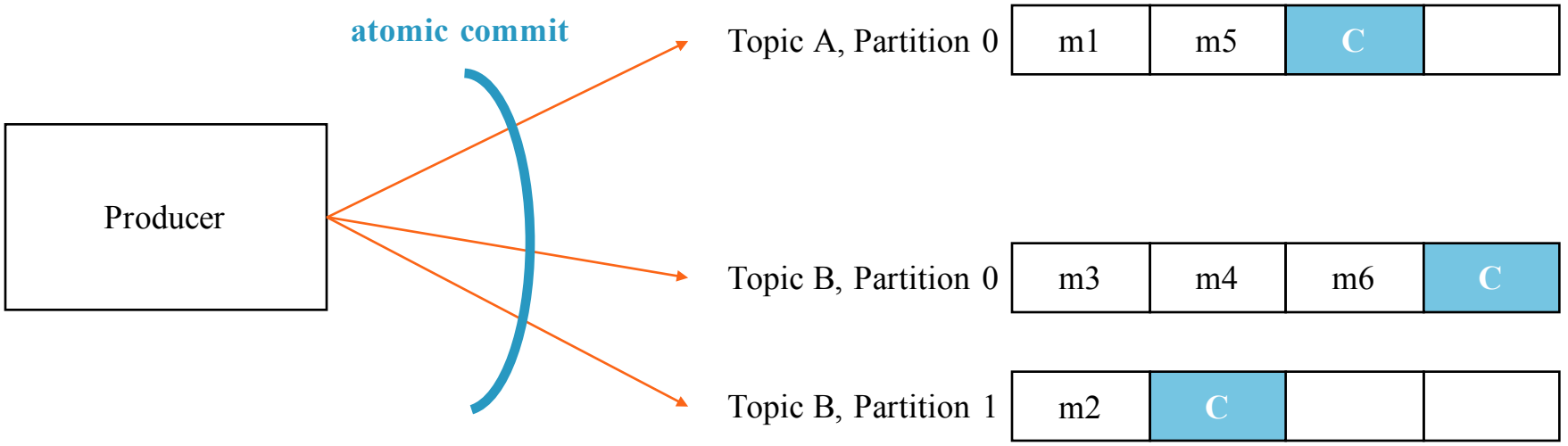
Atomic Multi-Partition Writes



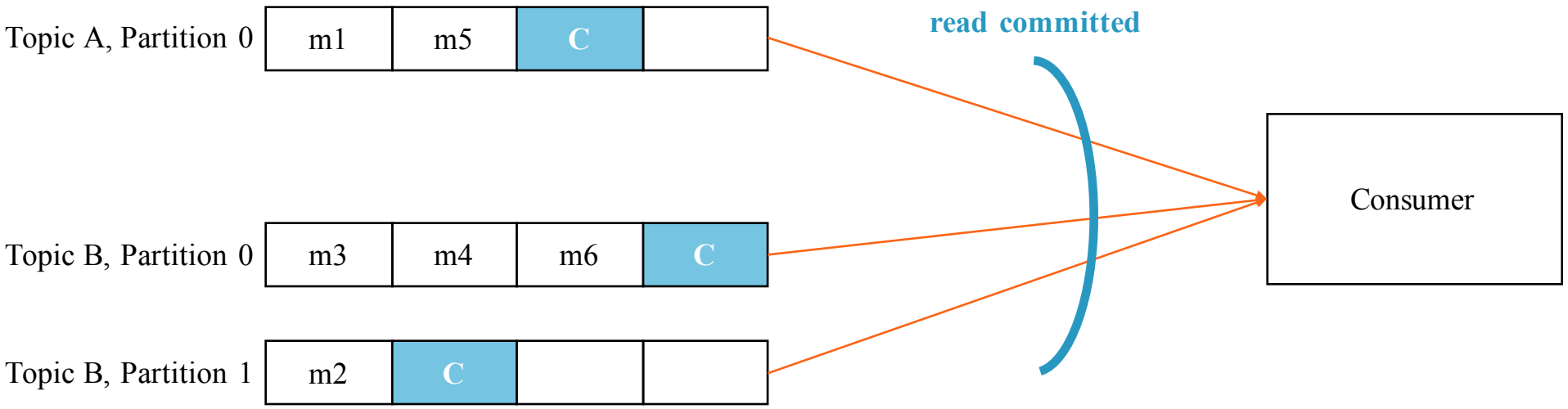
Atomic Multi-Partition Writes



Atomic Multi-Partition Writes



Atomic Multi-Partition Writes



Transactional API

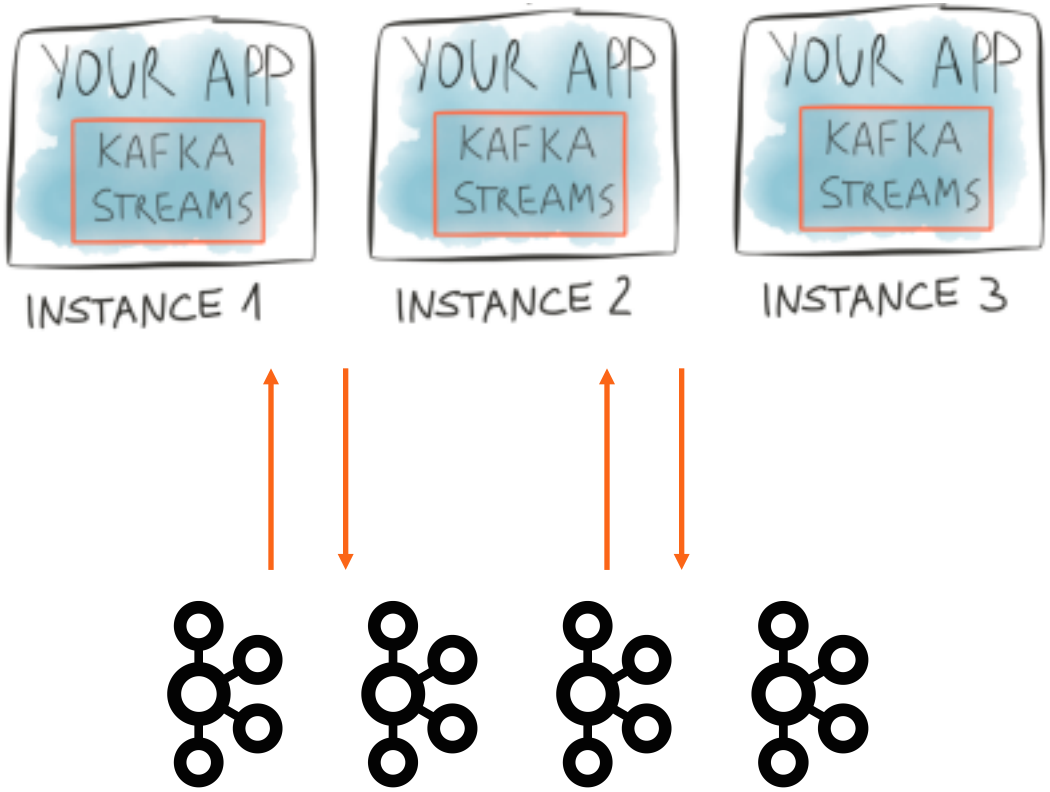
```
producer.initTransactions();
try {
     producer.beginTransaction();
     producer.send(record0);
     producer.send(record1);
     producer.sendOffsetsToTxn(…);
     producer.commitTransaction();
} catch (ProducerFencedException e) {
     producer.close();
} catch (KafkaException e) {
     producer.abortTransaction();
}
```

How to use exactly-once capabilities:

- Streams API *(the easiest way to use exactly-once semantics)*
  - Config parameter `processing.mode = "exactly_once"`
- Idempotent Producer
  - Config parameter `enable.idempotence = true`
- Transactional Producer
  - Config parameter `transactional.id = "my-unique-tid"`
  - ***And Transactional API (hard to use!)***
- Transactional Consumer
  - Config parameter `isolation.level = "read_committed"` (default: "`read_uncommitted`")

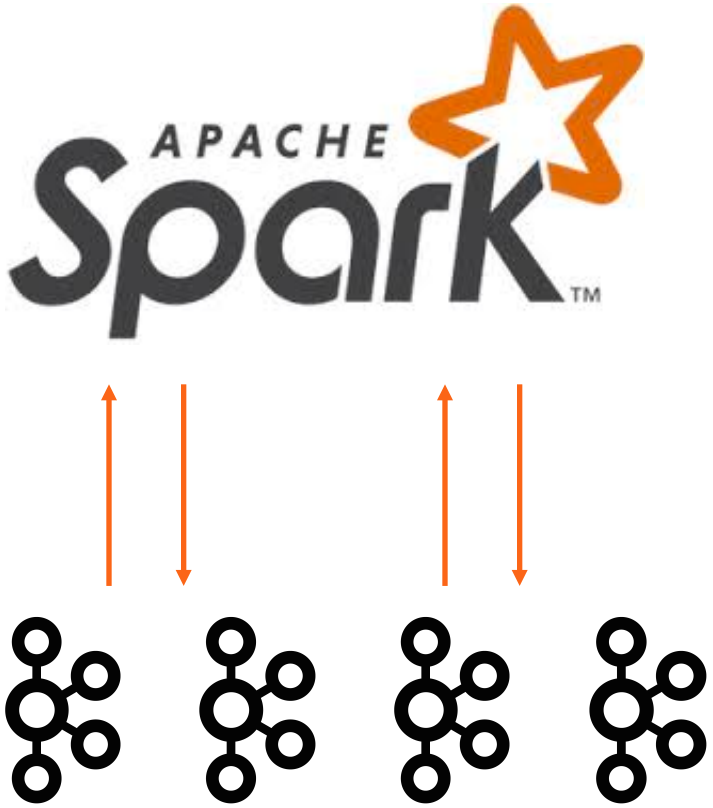
Stream Processing with Kafka's Streams API

Transactional read-process-write-commit pattern:



Stream Processing with Apache Spark

Transactional read-process-write-commit pattern:



When to use this?

Available in Kafka 0.11, June 2017. Try it out!

Putting it together

- We understood Kafka's existing delivery semantics.
- Learned how these have been strengthened.
- Learned how the new semantics work.
- Saw, it's easy to use with higher level APIs like Kafka Streams or Apache Spark.

Thank You

We are hiring!