Event Streaming and Message Queues: which one should you use?

Domenico Francesco Bruscino & Paolo Patierno

Linux Day Napoli 2022

Who are we?

Senior Principal Software Engineer
Apache Kafka and Strimzi
@ppatierno

Senior Software Engineer
Apache ActiveMQ Artemis
and ArtemisCloud
@bruscinodf

Let’s start from the beginning : Event vs Message

A thing that happens

A fact making the history

A record to retain for longer time

Replayable to rebuild status

A task on a to-do list

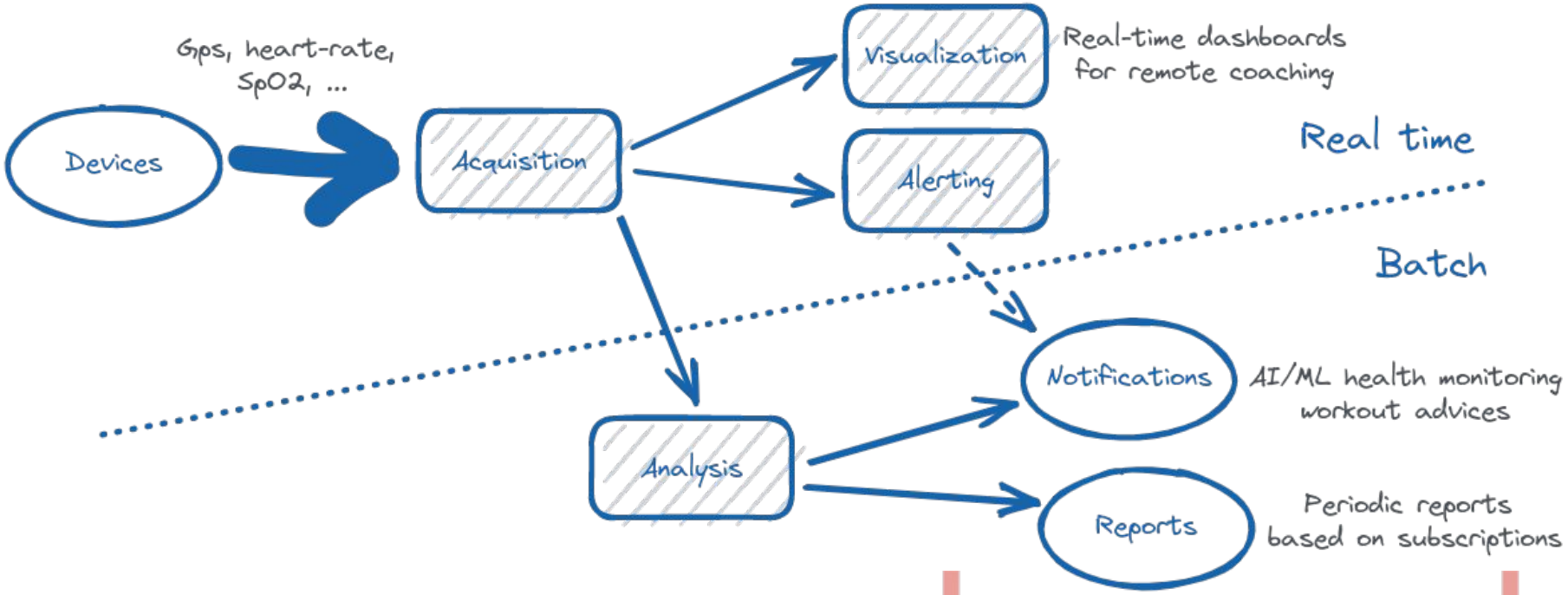
A business process to run

A command to be executed

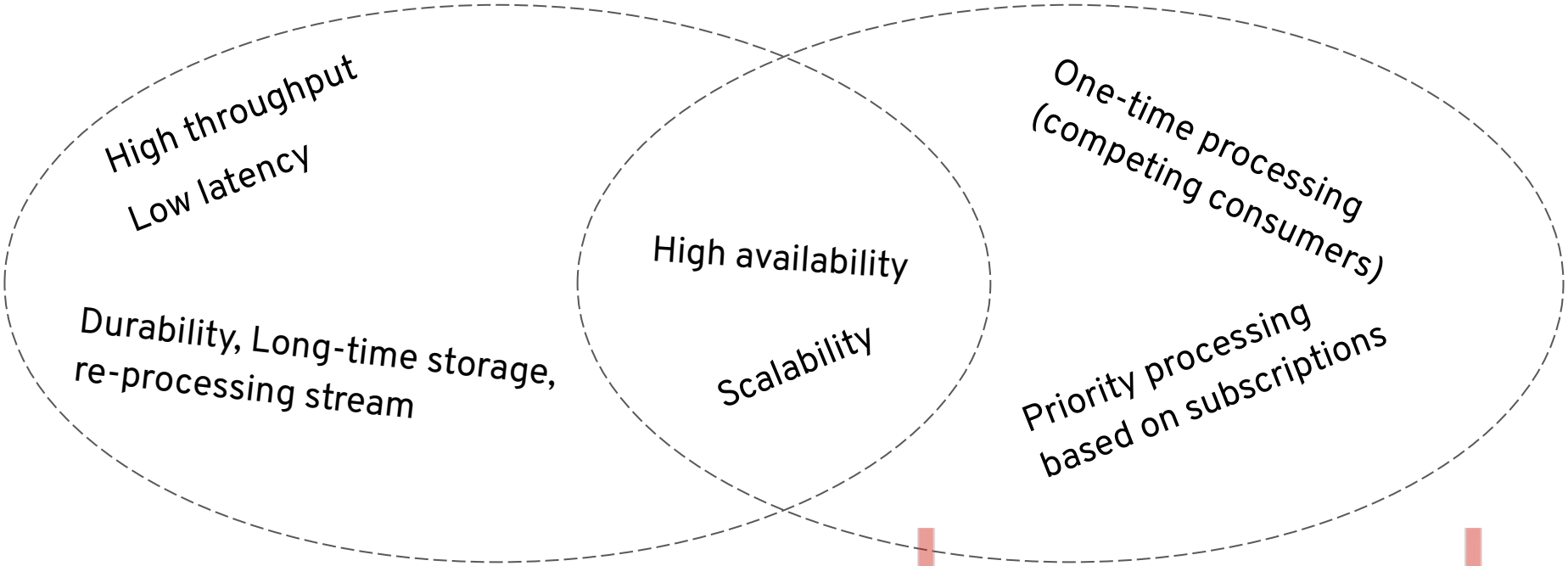
To be deleted when it’s done

*Fun fact … Events **ARE** (anyway) Messages :-)*

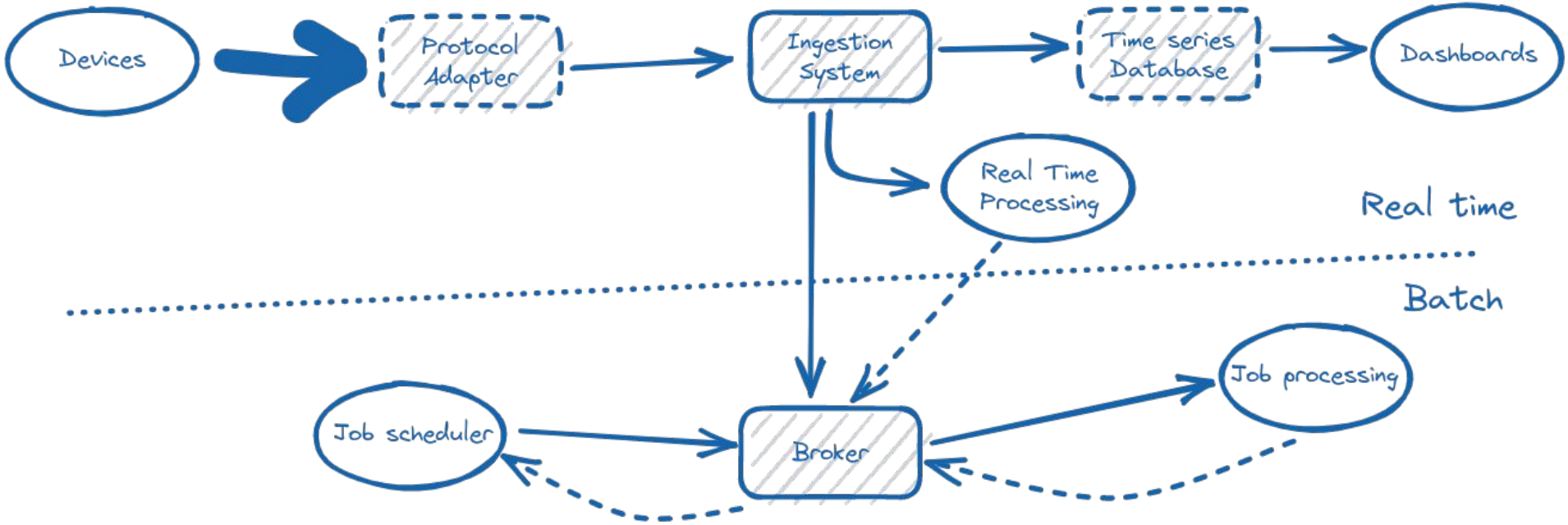
Welcome to Strada



Real-time vs Batch: challenges



Deep into the workflow

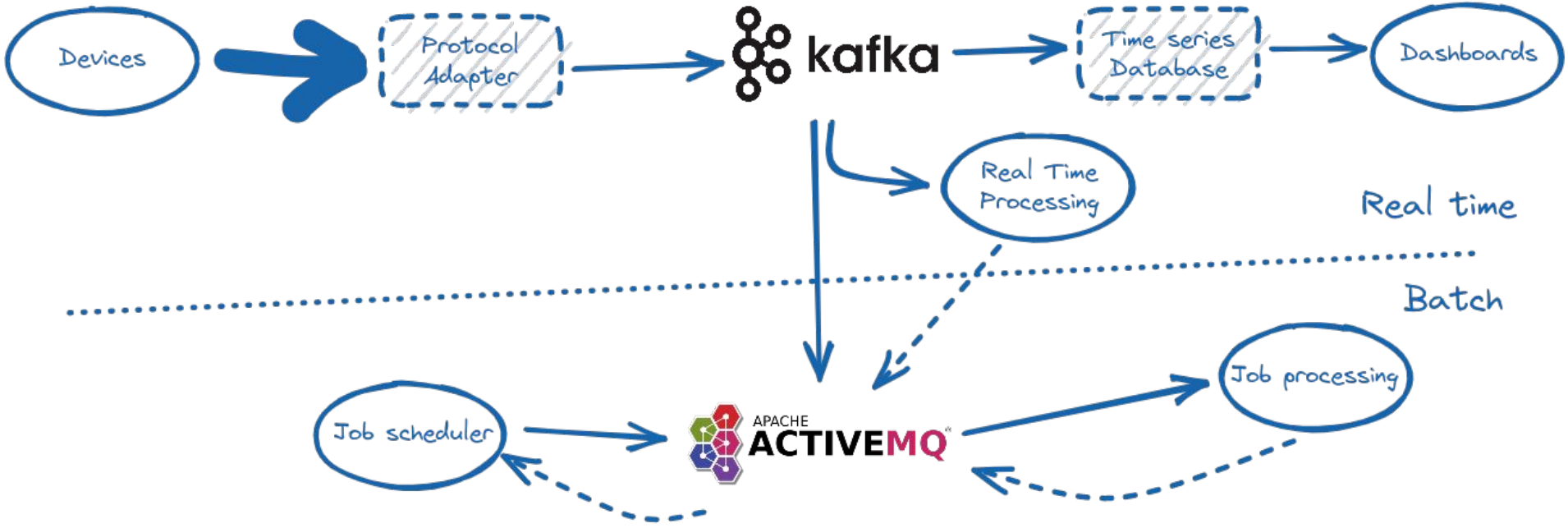


How do we solve this?

Is using one technology enough?

Should we use the right tool for different goals?

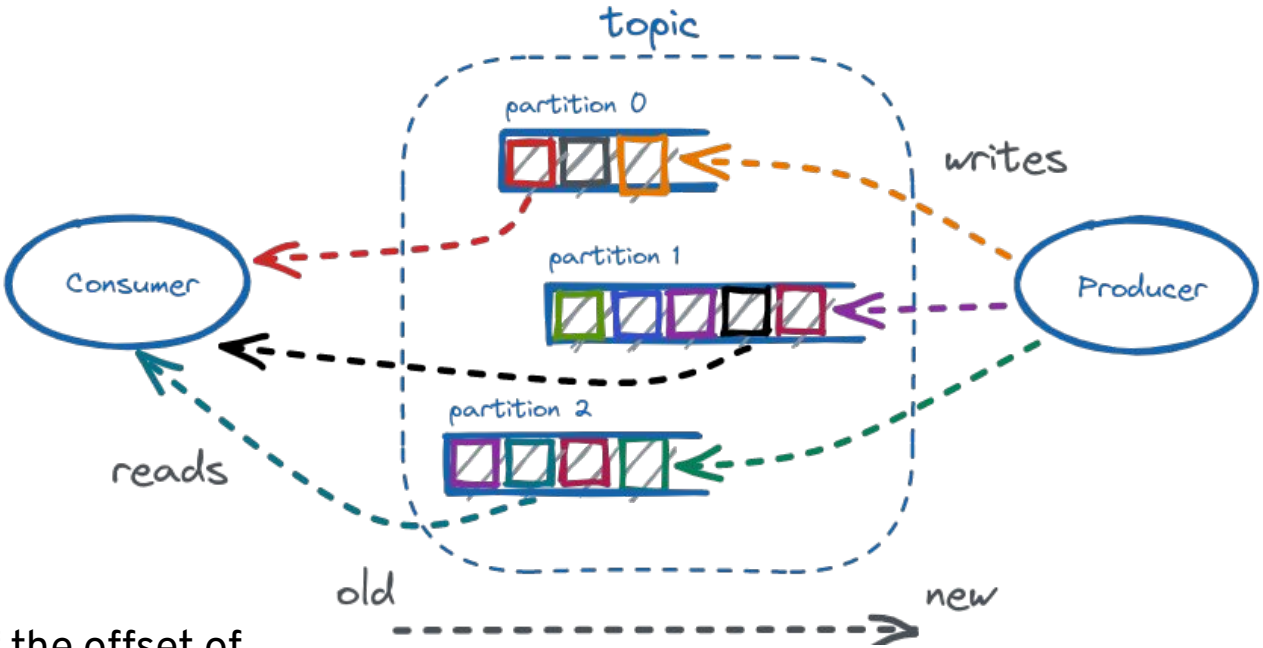
Welcome to … Apache Kafka & Apache ActiveMQ Artemis



Apache Kafka

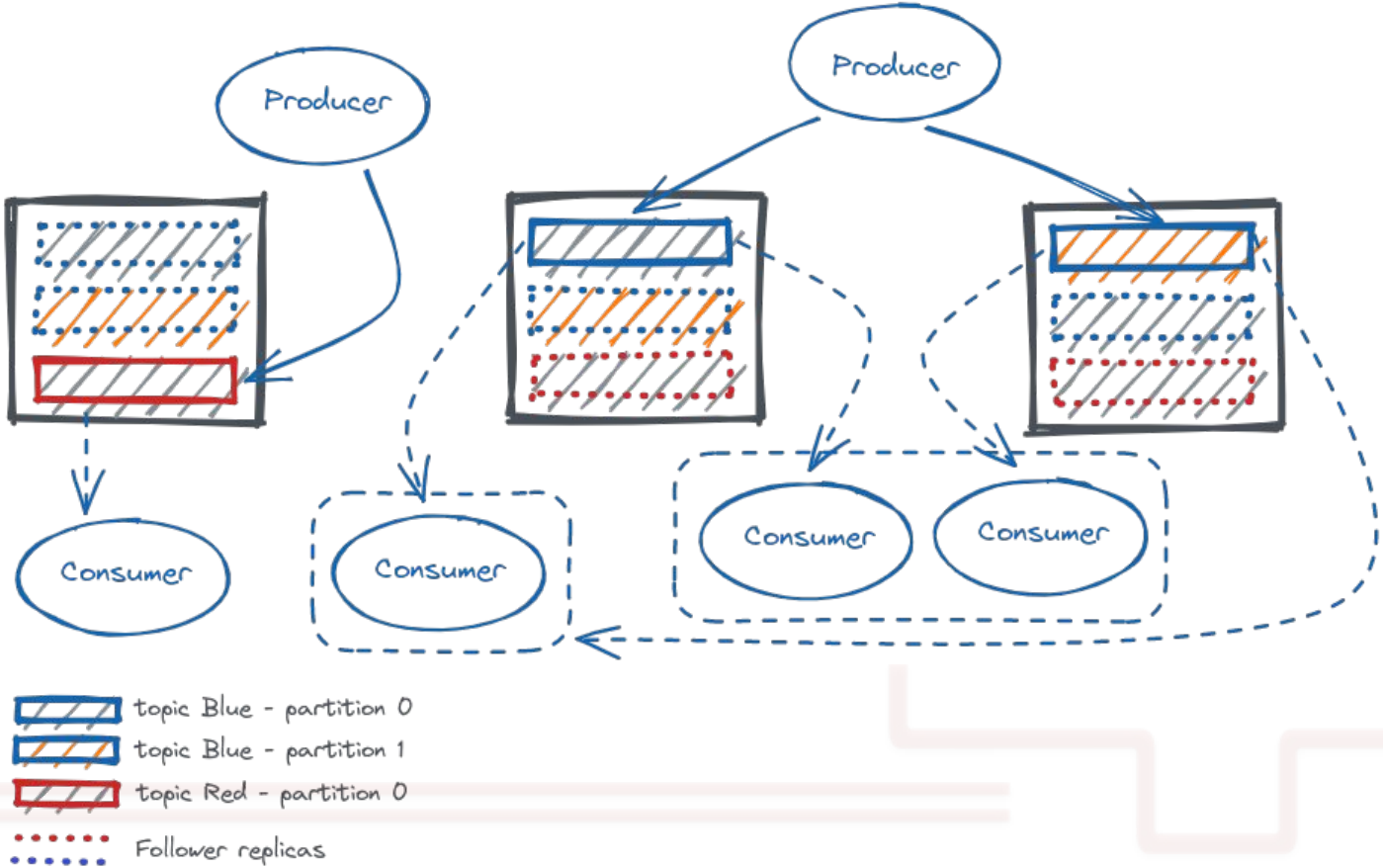
- Messaging system, data/event streaming platform … a commit log
- Developed at Linkedin back in 2010, open sourced in 2011
- Designed to be fast, scalable, durable and available
- Distributed by nature
- Data partitioning (sharding)
- High throughput / low latency
- Ability to handle huge number of consumers
- Dumb broker, smart client

Apache Kafka : the basics



- Consumer keeps track of the offset of the last read message

Apache Kafka : brokers & replicas

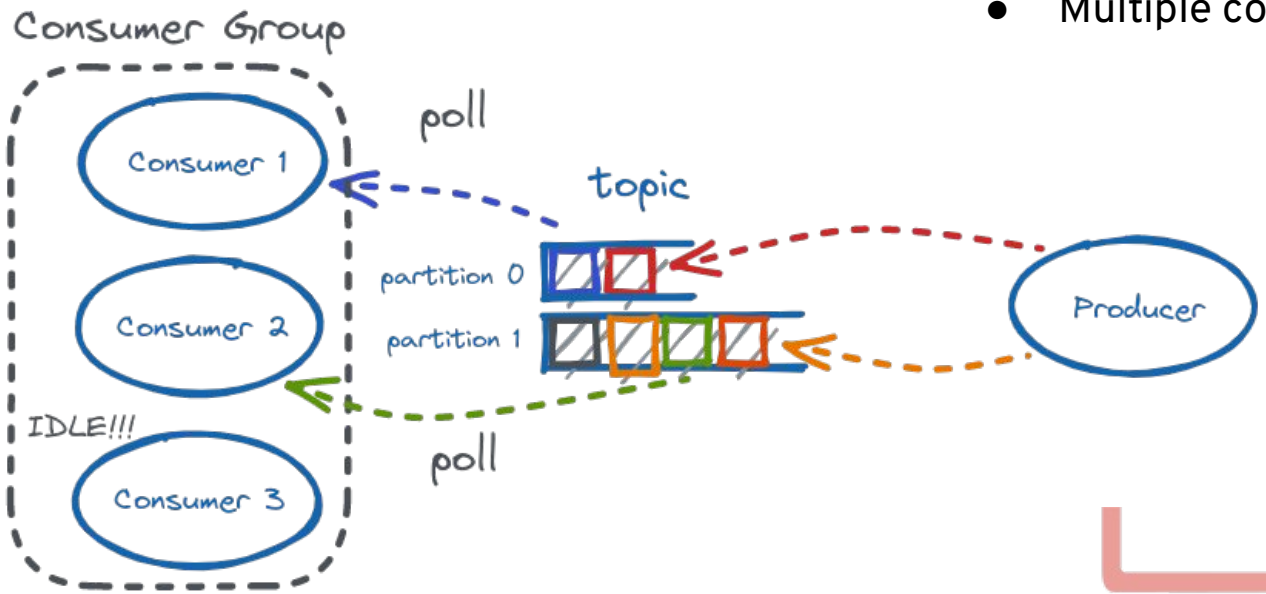


Apache Kafka ... good at

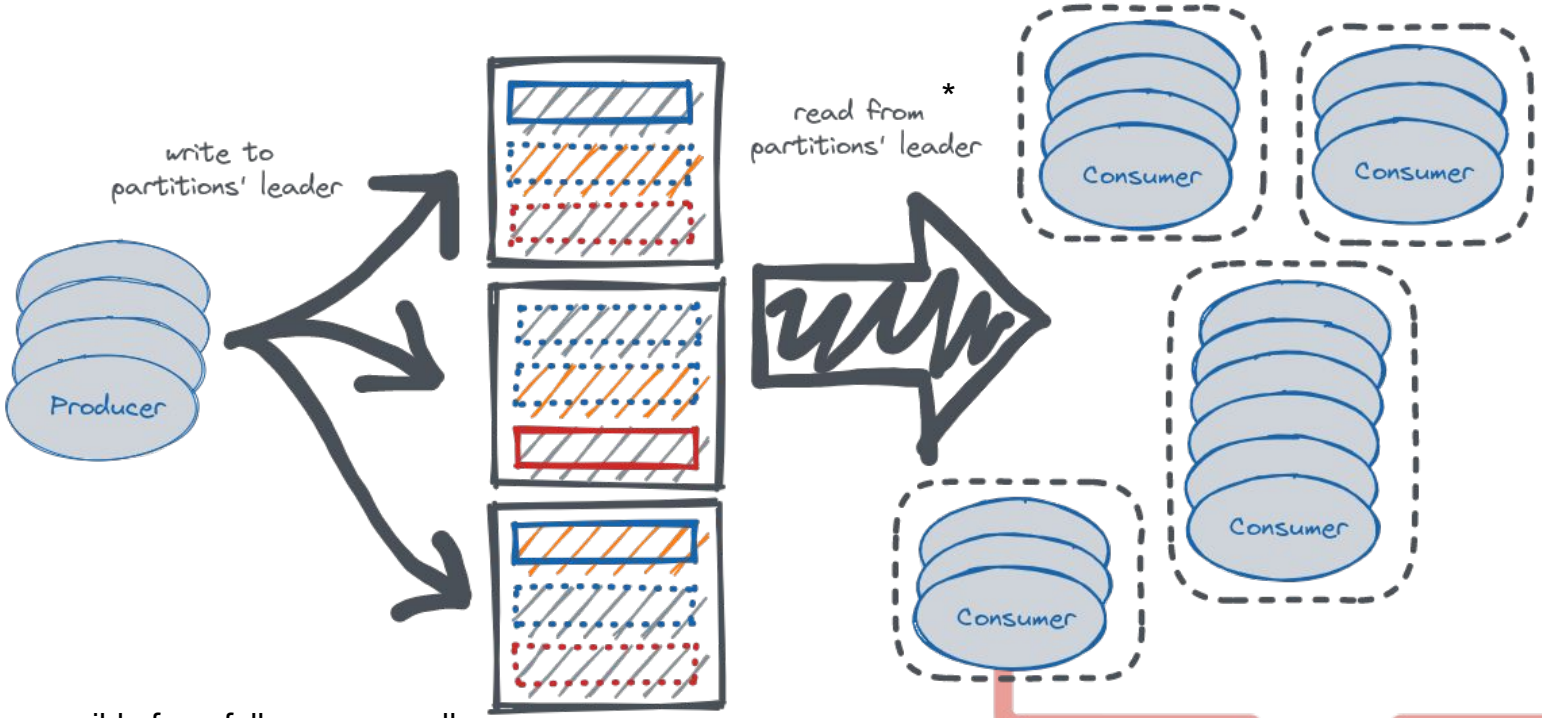
- Long-time durable storage
  - Messages stored and available for re-processing
- High throughput, Low latency
  - No complex process on the broker, just get and store the message
  - Consumers keep track of offset, Producers select destination and do batching
- Scalability
  - Topic partitions allow to scale consumers
- High Availability
  - Topic partitions are replicated across brokers

Apache Kafka … good at : scalability

- Partitioning is the key for scalability
- Partitions spread across consumers
- Multiple consumer group allows pub/sub

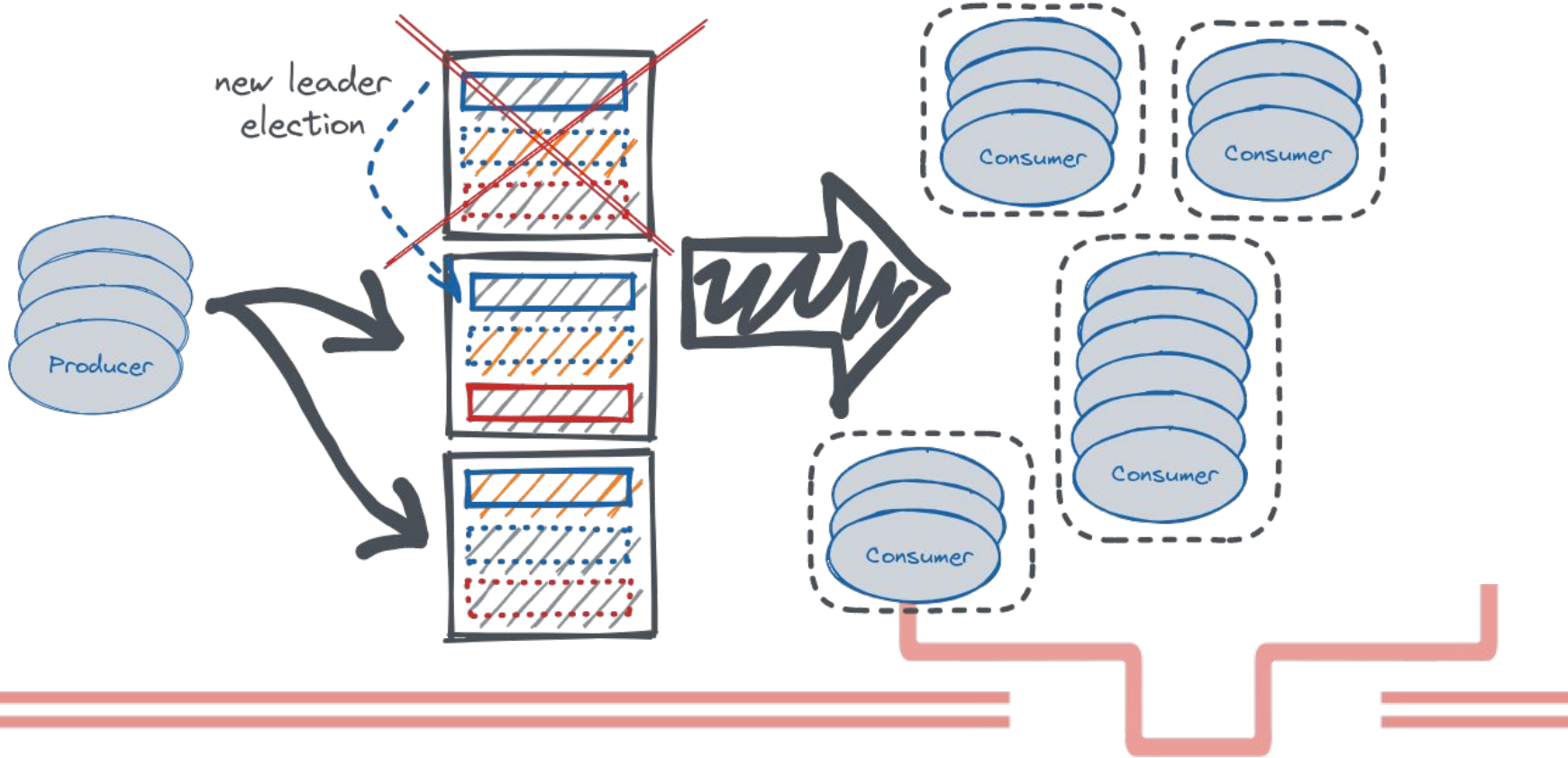


Apache Kafka … good at : high availability



* actually possible from followers as well

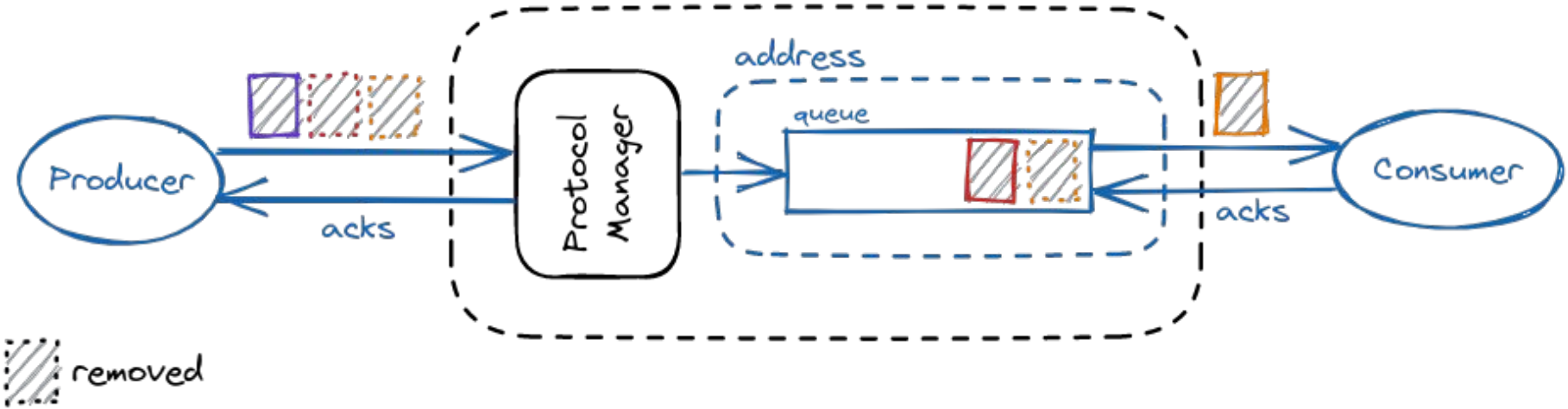
Apache Kafka … good at : high availability



Apache ActiveMQ Artemis

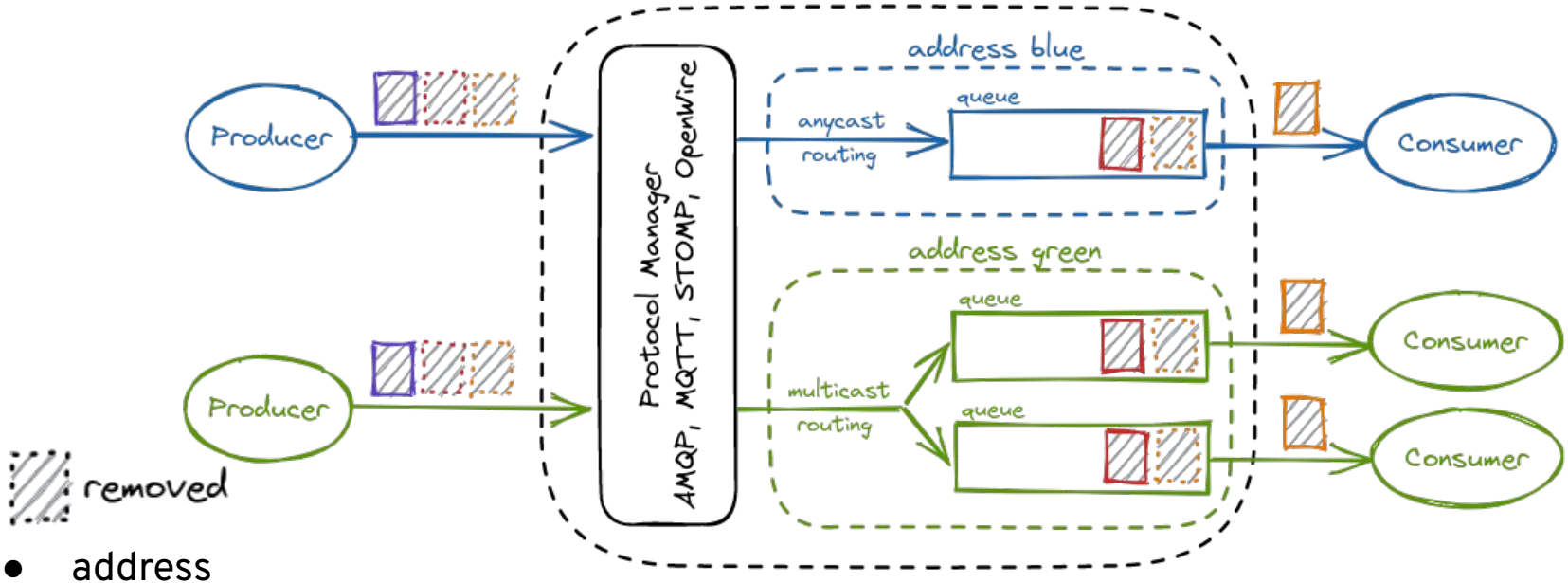
- Standards based (JMS 1.x/2.x/3.x)
- Multi protocol (AMQP, MQTT, STOMP, OpenWire)
- Powerful and flexible address model
- Cluster ready and highly available
- Trusted and used in many highly critical use cases
- Dumb client, smart broker

Apache ActiveMQ Artemis : the basics



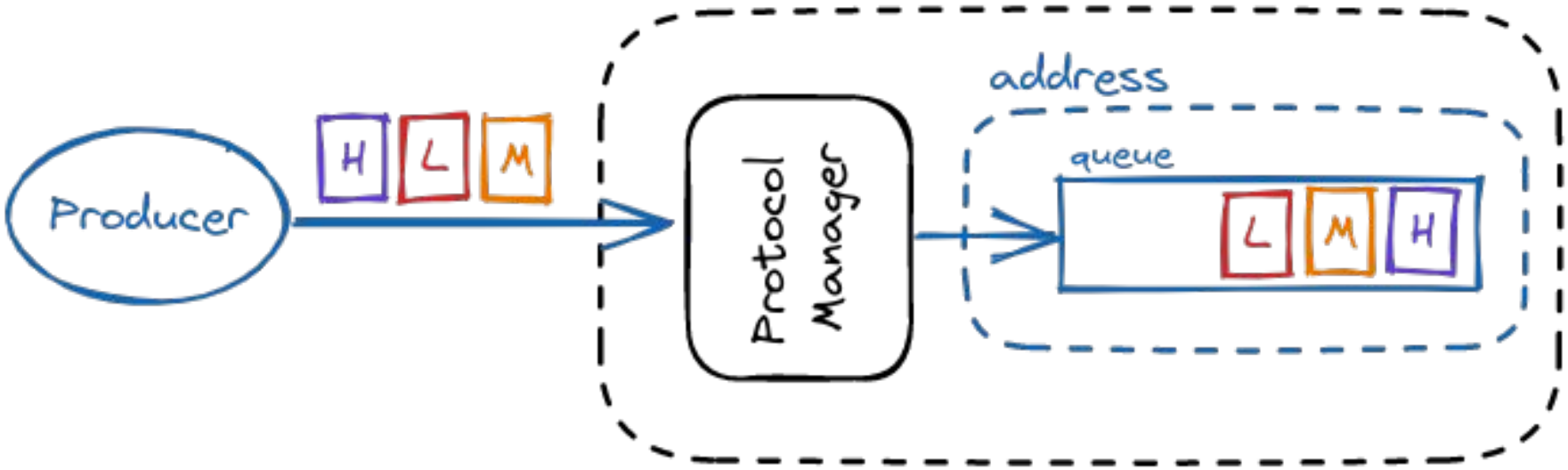
- Broker sends acknowledgments to confirm messages received by producers
- Broker removes messages from queues when it receives acknowledgments from consumers

Apache ActiveMQ Artemis address model



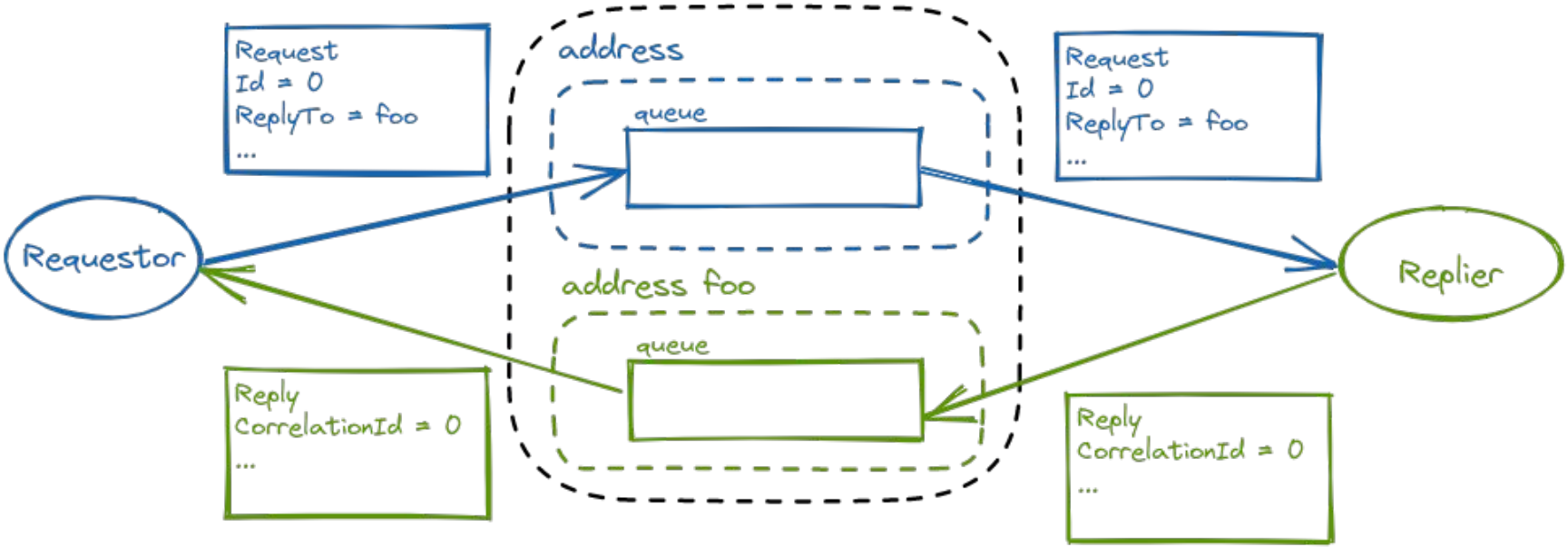
- address
- queue
- routing type

Apache ActiveMQ Artemis ... good at : message priority



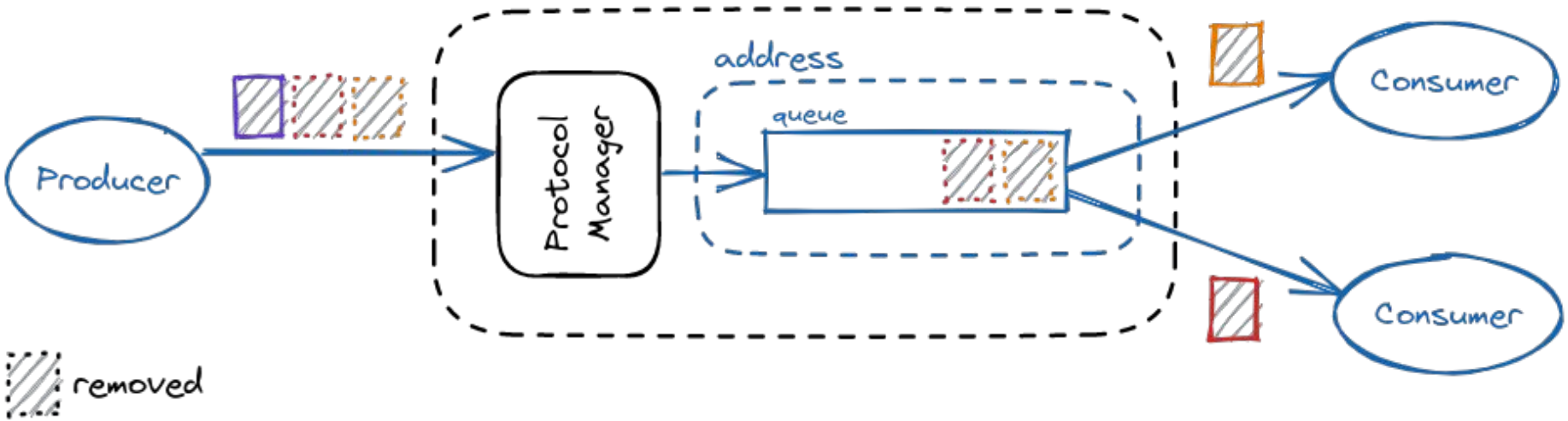
Messages are enqueued in order of priority

Apache ActiveMQ Artemis ... good at : request/reply



Broker automatically create a temporary address and queue for the reply

Apache ActiveMQ Artemis ... good at : competing consumers



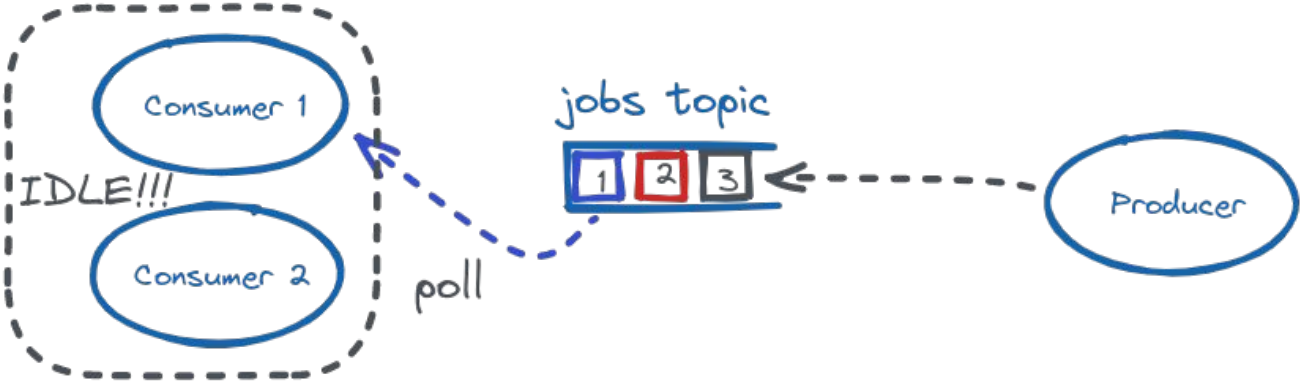
- Message order is preserved
- Delivered messages are removed
- No required change to attach a new consumer

Why these choices?

What’s wrong with using Kafka for job processing?

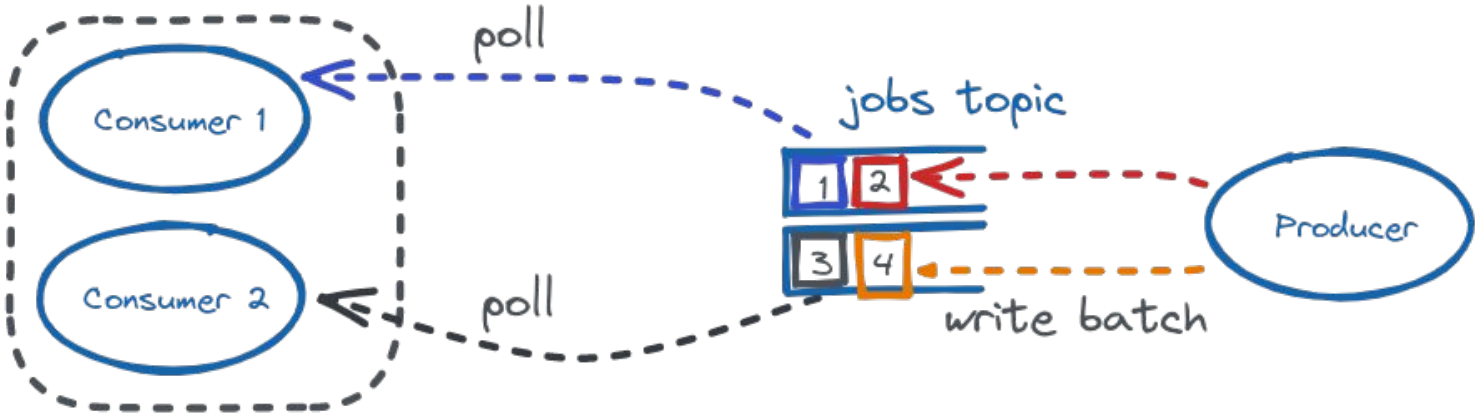
Or with using Artemis ActiveMQ for event streaming?

Apache Kafka … not good at : job ordering



- Order is per partition not per topic
- Only one consumer per partition to get each job and execute it
- To keep jobs ordering, we cannot scale
- Jobs get stuck if consumer is slow, or bad message (head of line blocking)

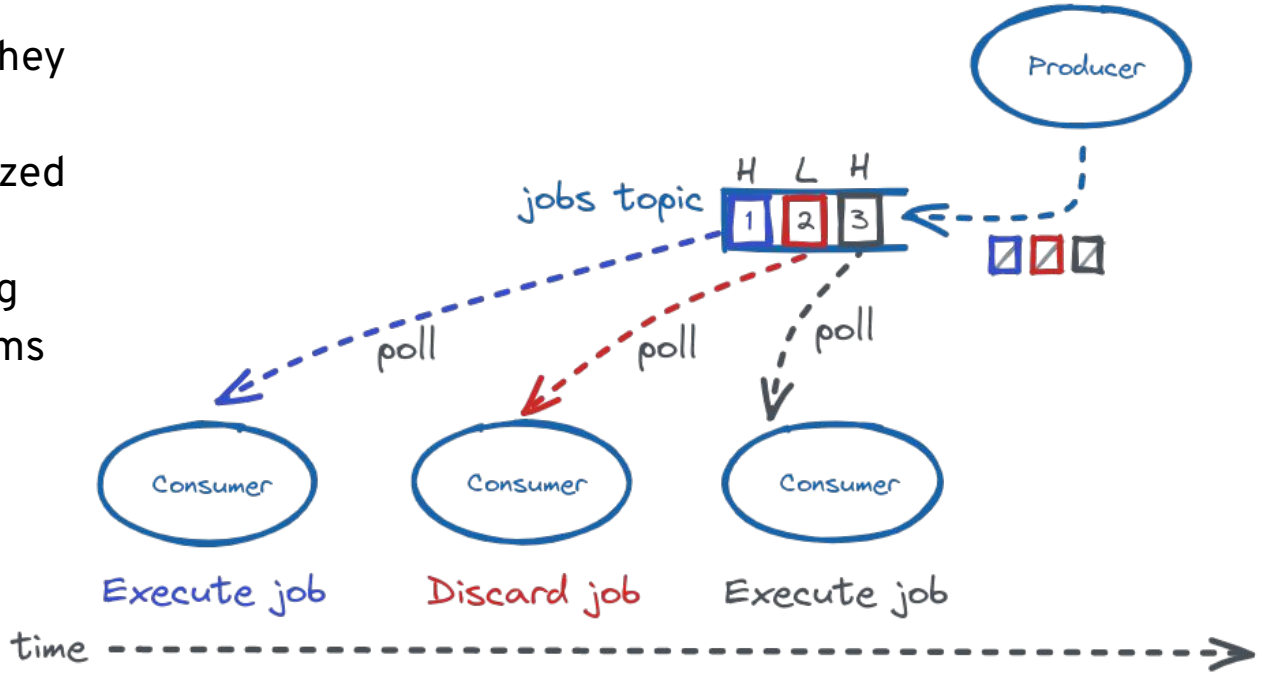
Apache Kafka … not good at : job ordering



- Trying to scale consumers but …
- … Producer writes in “batch”
- … More partitions can screw up the jobs order

Apache Kafka … not good at : message priority

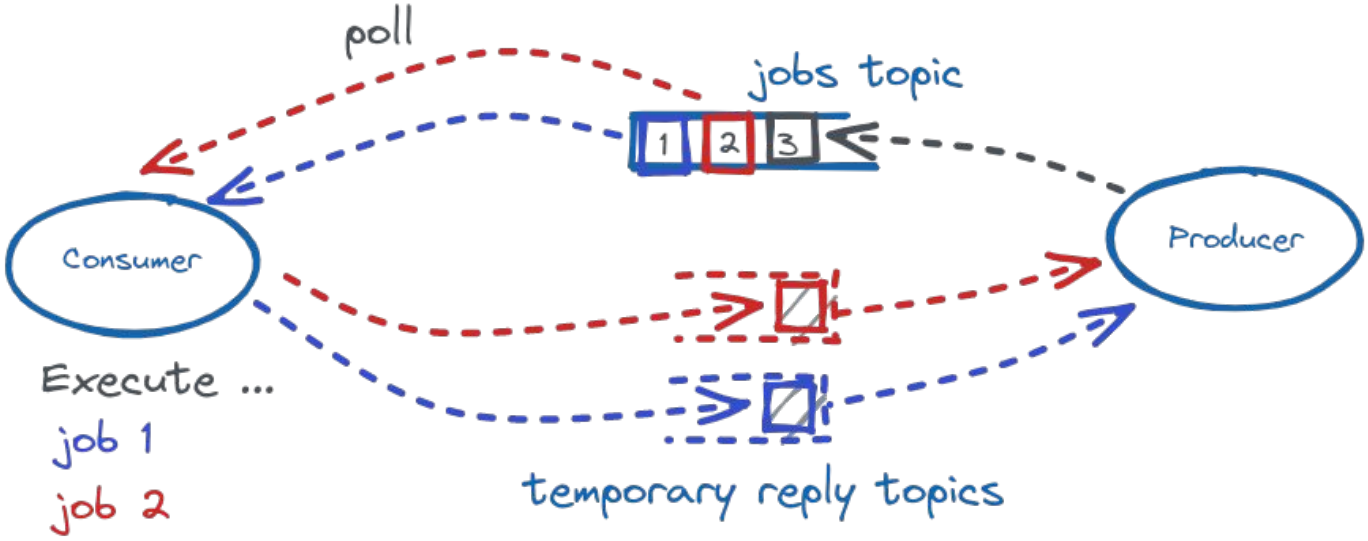
- Messages are appended as they arrive
- Broker cannot return prioritized messages
- Need for an application doing the filtering (i.e. Kafka Streams API based)



Apache Kafka … not good at : request/reply

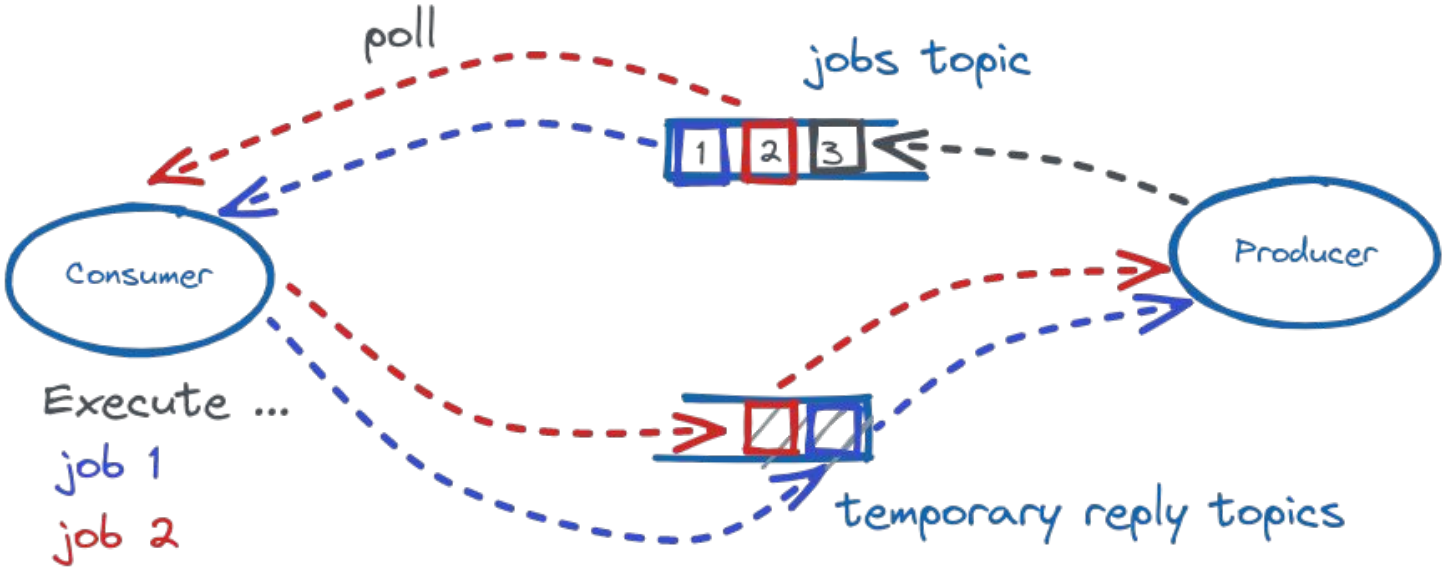
- Request/Reply pattern
  - Thought for an event-driven architecture which doesn’t cope well with a request/reply
  - Correlate response with request is not native
    - No correlation-id
    - No reply-to for creating a “temporary” topic for reply
- Onerous reply topic(s) handling

Apache Kafka … not good at : request/reply



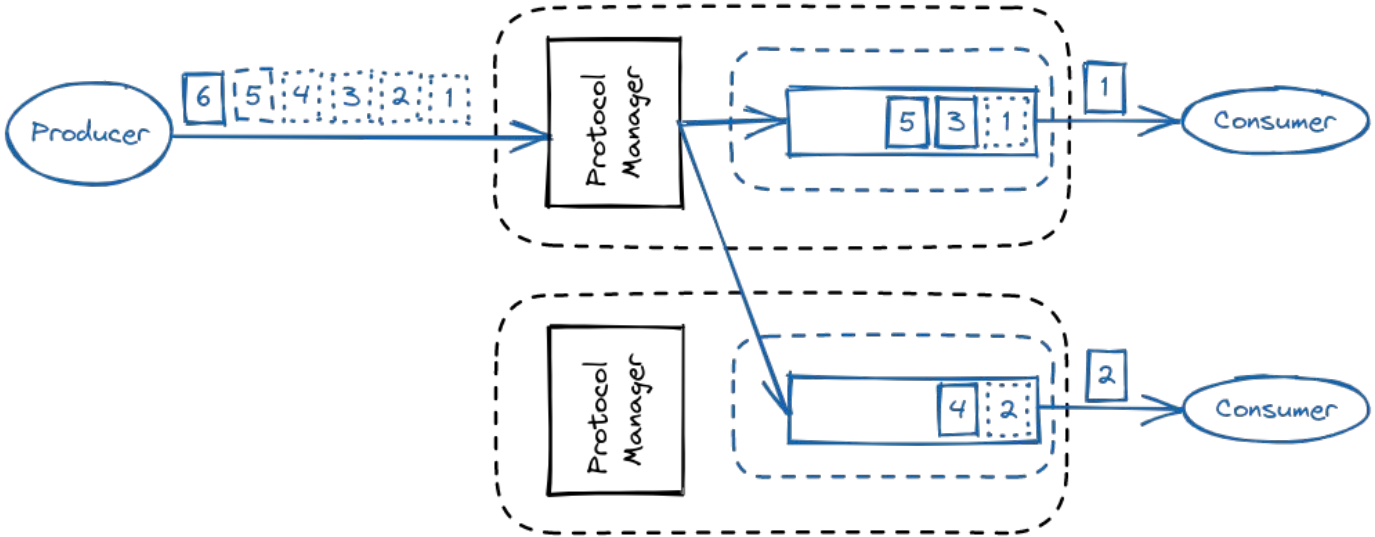
- “Temporary” topic creation is onerous
- Disk allocation, replication set up, …

Apache Kafka … not good at : request/reply



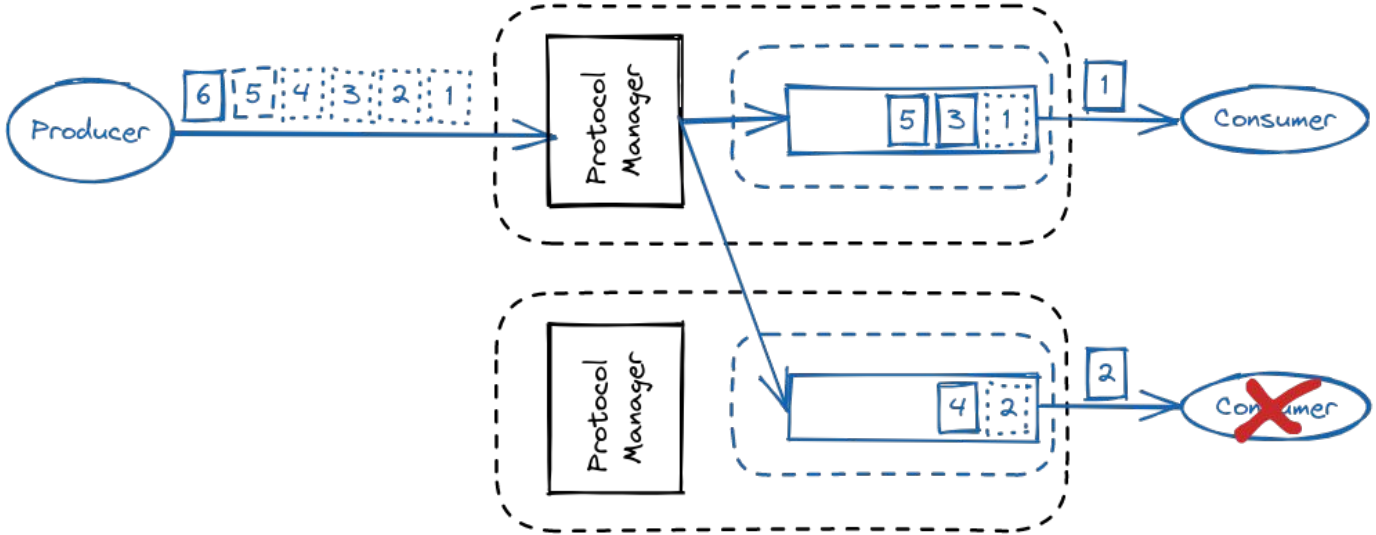
- Error prone, re-reading same acks
- Higher level logic needed

Apache ActiveMQ Artemis ... not good at : horizontal scaling



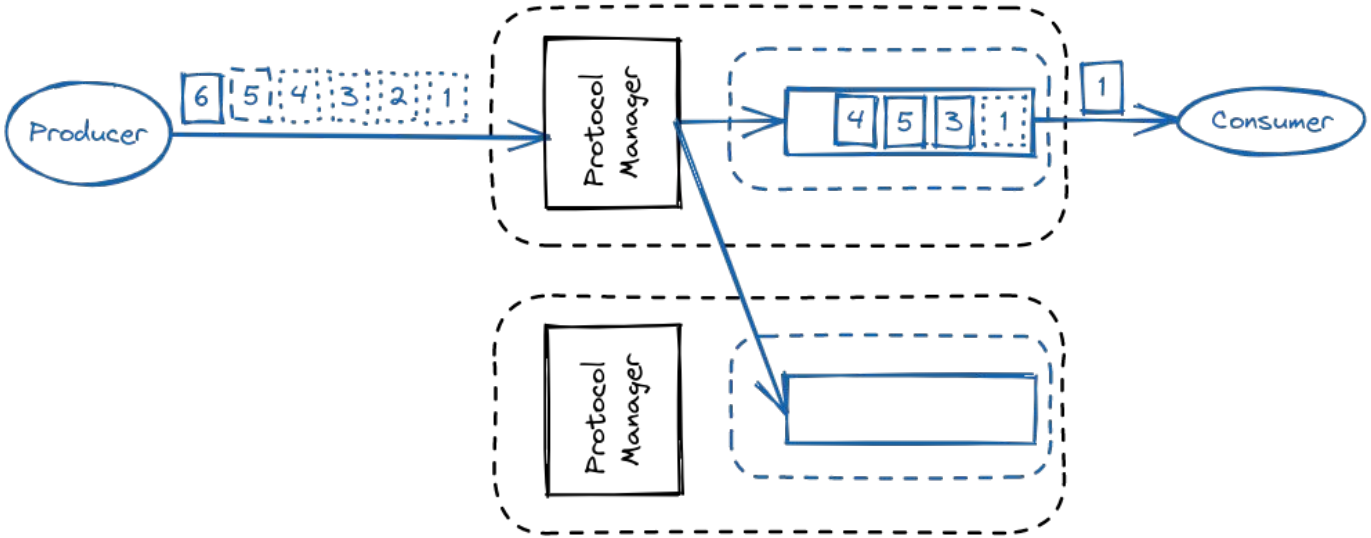
Broker 1 needs to process messages before routing

Apache ActiveMQ Artemis ... not good at : horizontal scaling



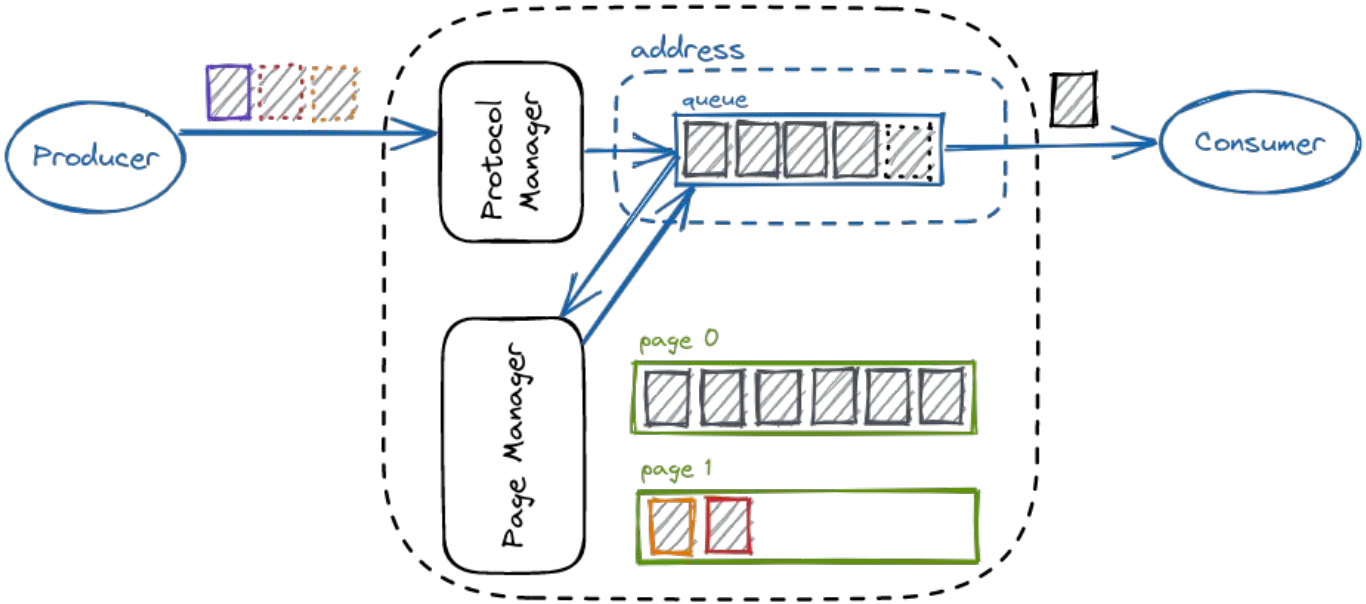
Broker 2 needs to redistribute messages

Apache ActiveMQ Artemis ... not good at : horizontal scaling



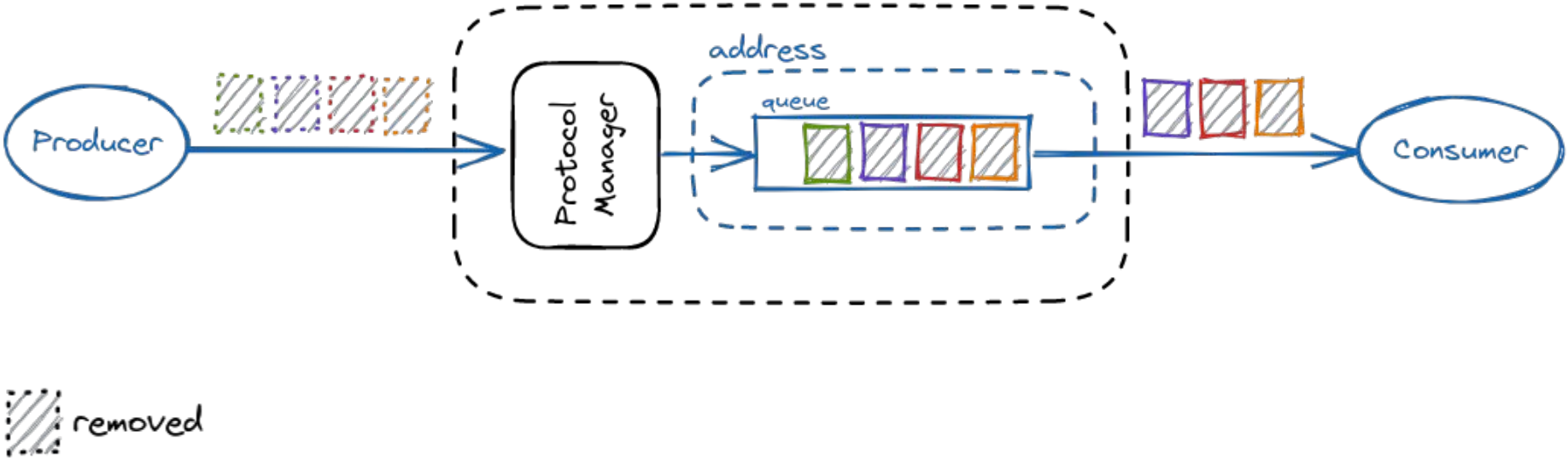
Messages lose order after redistribution

Apache ActiveMQ Artemis ... not good at : long-time storage



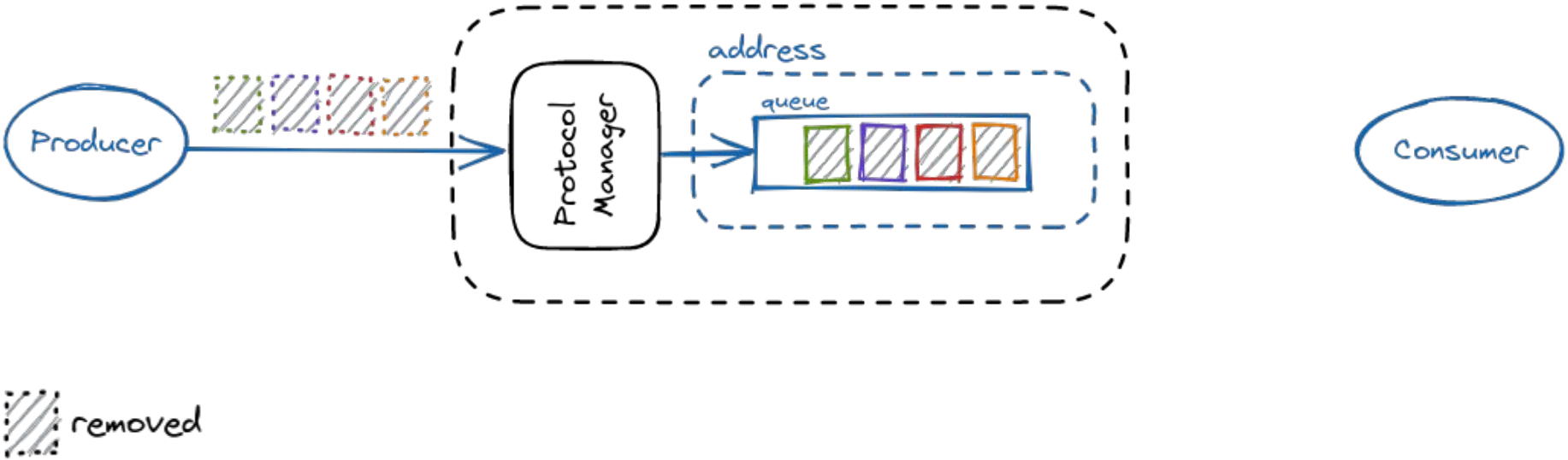
Fast producers and slow consumers cause paging

Apache ActiveMQ Artemis ... not good at : long-time storage



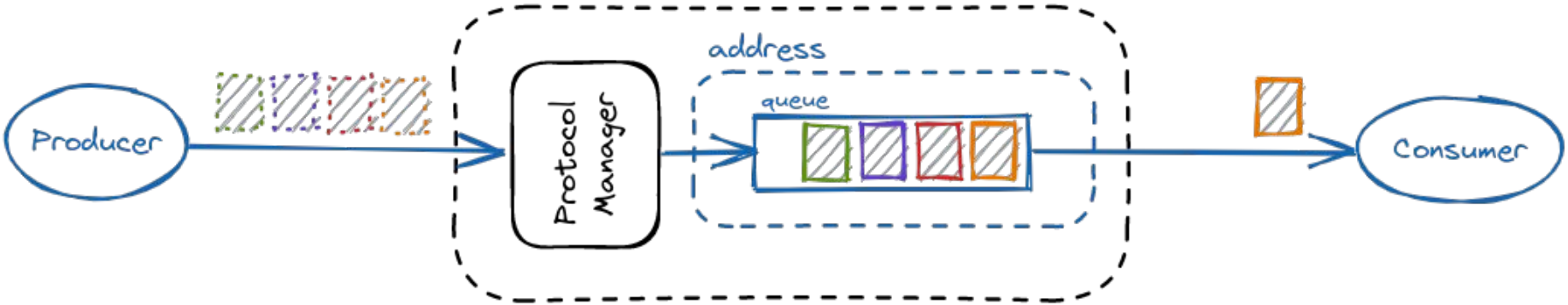
- The queue pointer is managed by the broker
- Non destructive queue messages are not removed

Apache ActiveMQ Artemis ... not good at : long-time storage



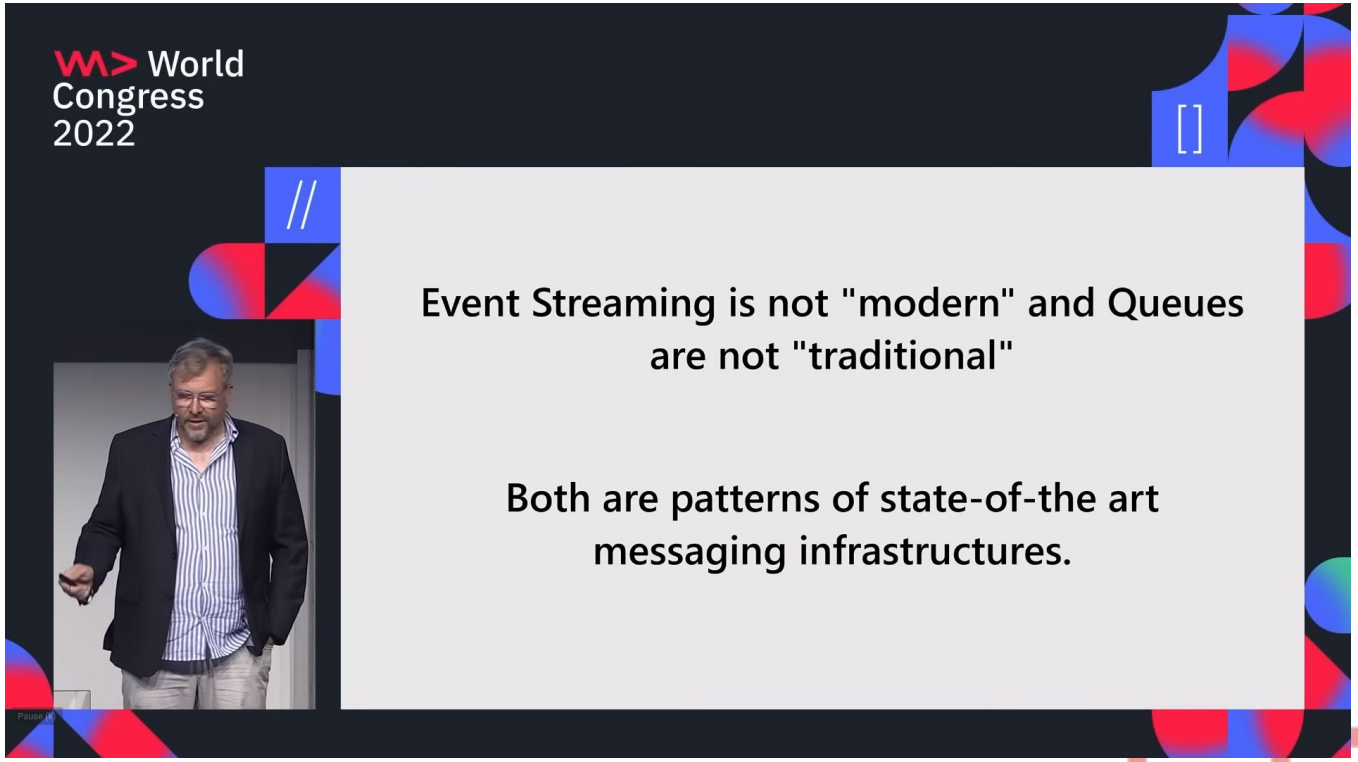
- The queue pointer is lost when the session is closed

Apache ActiveMQ Artemis ... not good at : long-time storage



- The queue pointer starts from the beginning for new sessions

Clemens Vasters (Microsoft) says …



Thanks!