大数据实时流计算风云榜

Leo-陈旭

- Big Data is like **teenage sex**: Everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it too

- **2009年**　　毕业于复旦大学数学系
- **2009-2012** 阿里巴巴数据仓库基础架构
- **2012-2014** 大众点评数据平台基础架构
- **2014-至今** 负责平安壹钱包大数据基础架构，基于Hadoop/Spark构建了统一的大数据平台，目前专注于准实时/实时计算平台的构建

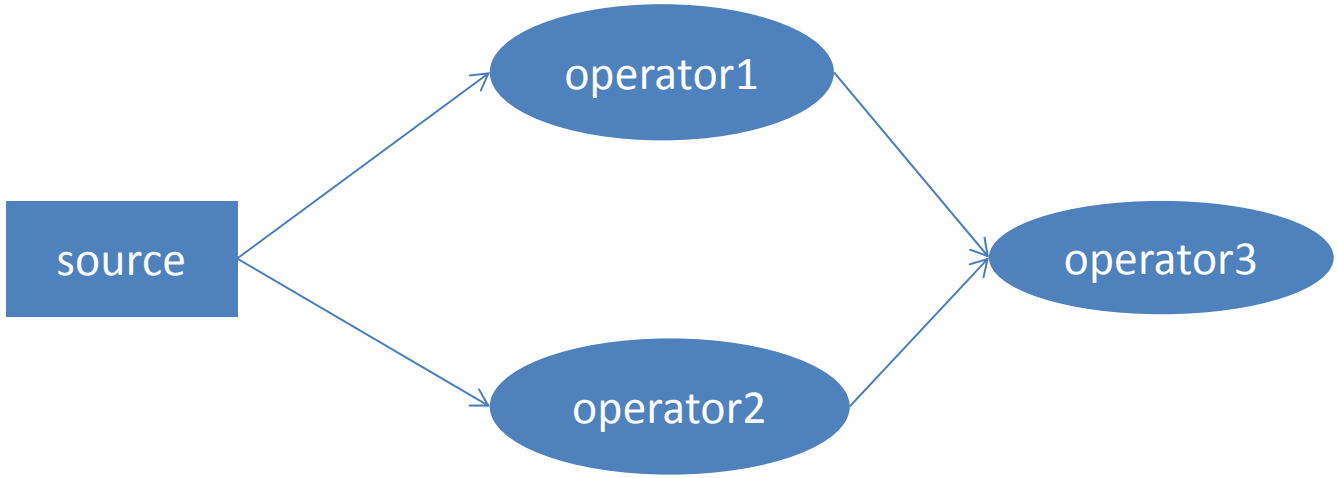
- 业务渴望更快的数据，希望降低延迟（Low Latency)
- 海量、无穷数据集（unbounded data)越来越常见，需要专门处理

- 当我在说流式计算(streaming)的时候我是在说什么

- 批量（batch) –>有边界/有穷数据（bounded data)
- 流式（streaming)->无穷数据（unbounded data)

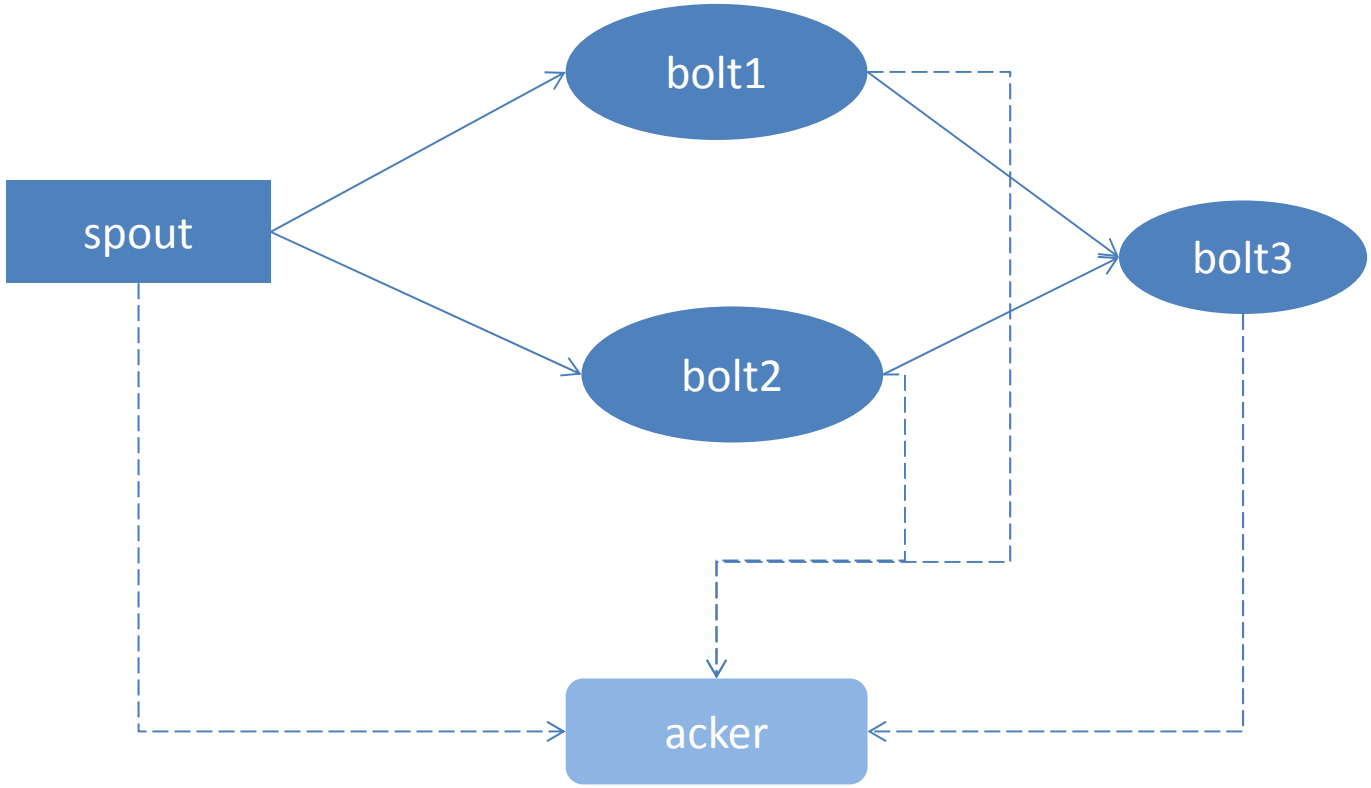
- 处理的难点/挑战在哪里
- 常见框架的核心原理
- 各个框架之间的对比
- 业界的发展

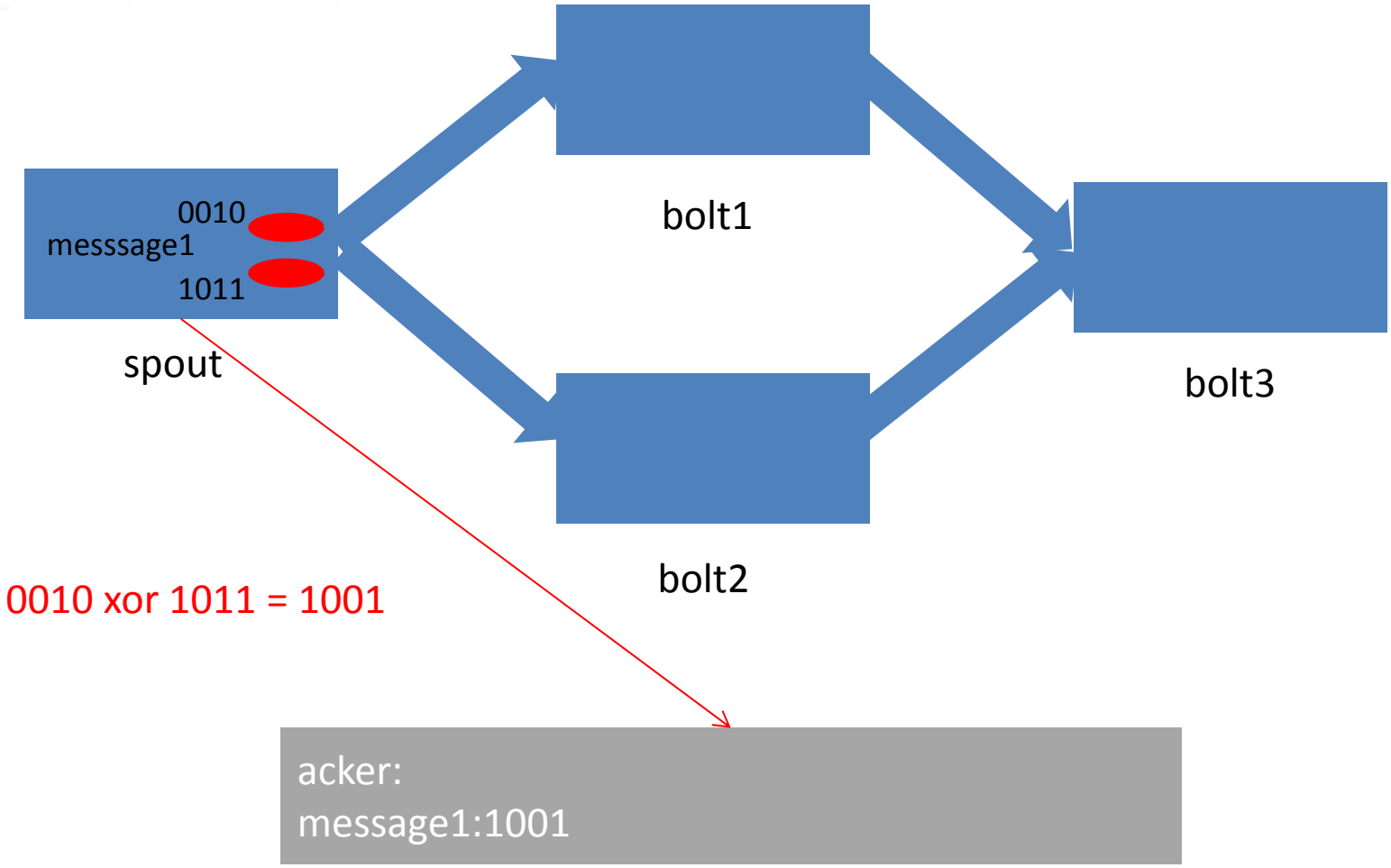
挑战一：如何保证数据被完全处理?

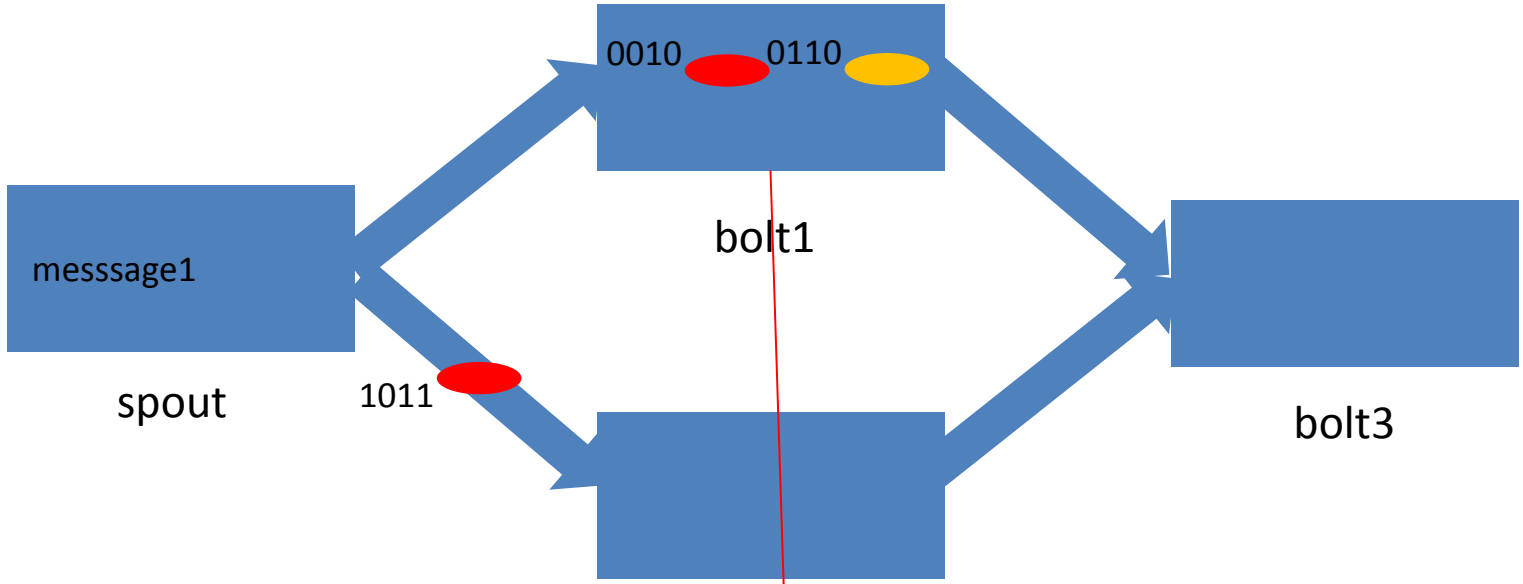


容错机制/可靠性
Sematic: at-least-once

- Acker per record: Storm 记录级容错



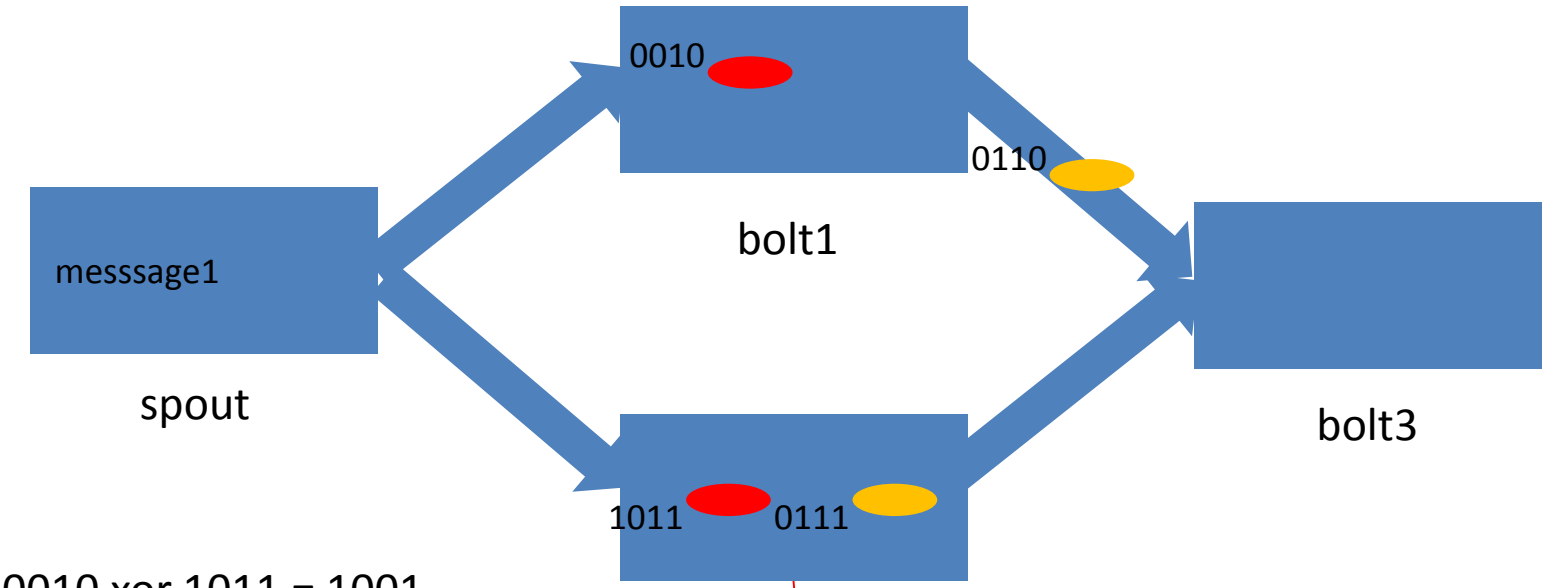

0010
messsage1
1011
spout
bolt1
bolt2
bolt3
0010 xor 1011 = 1001
acker:
message1:1001



0010 xor 1011 = 1001

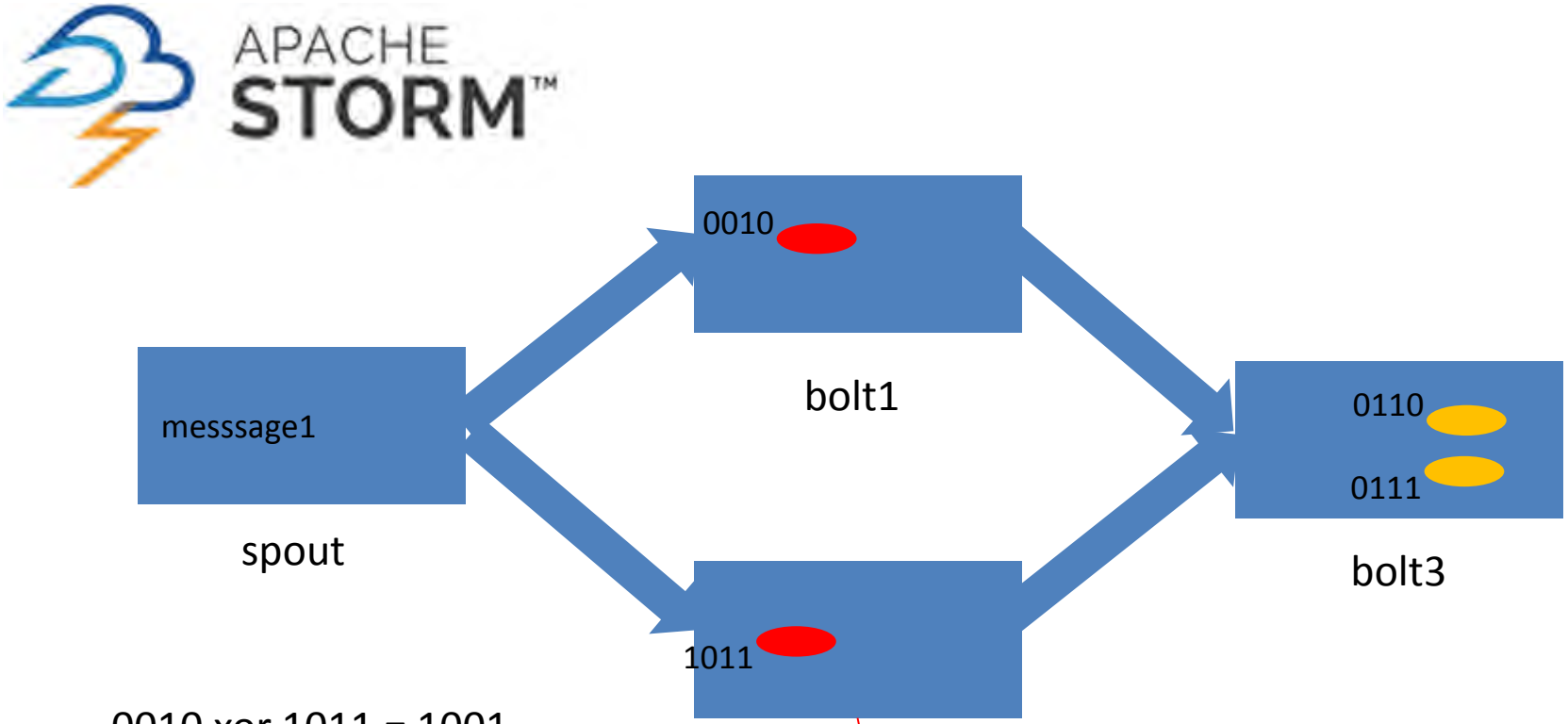
0010 xor 0110 = 0100

acker:
message1:1001 xor 0100 =1101



0010 xor 1011 = 1001
0010 xor 0110 = 0100
1011 xor 0111 = 1100

acker:
message1:1001 xor 0100 xor 1100 = 0001

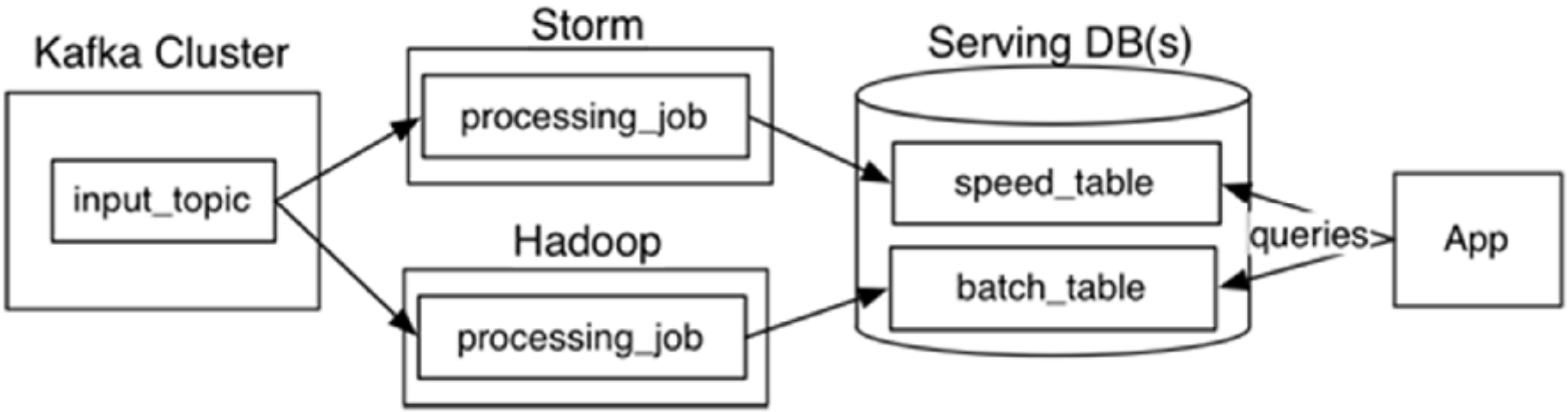


0010 xor 1011 = 1001
0010 xor 0110 = 0100
1011 xor 0111 = 1100
0110 xor 0111 = 0001

acker:
message1:1001 xor 0100 xor 1100 xor 0001= 0000

- 可回溯的数据源（replay)

- Lambda架构(storm author)
  - **Low-latency**, approximate, and/or speculative results
  - High-latency, correct results

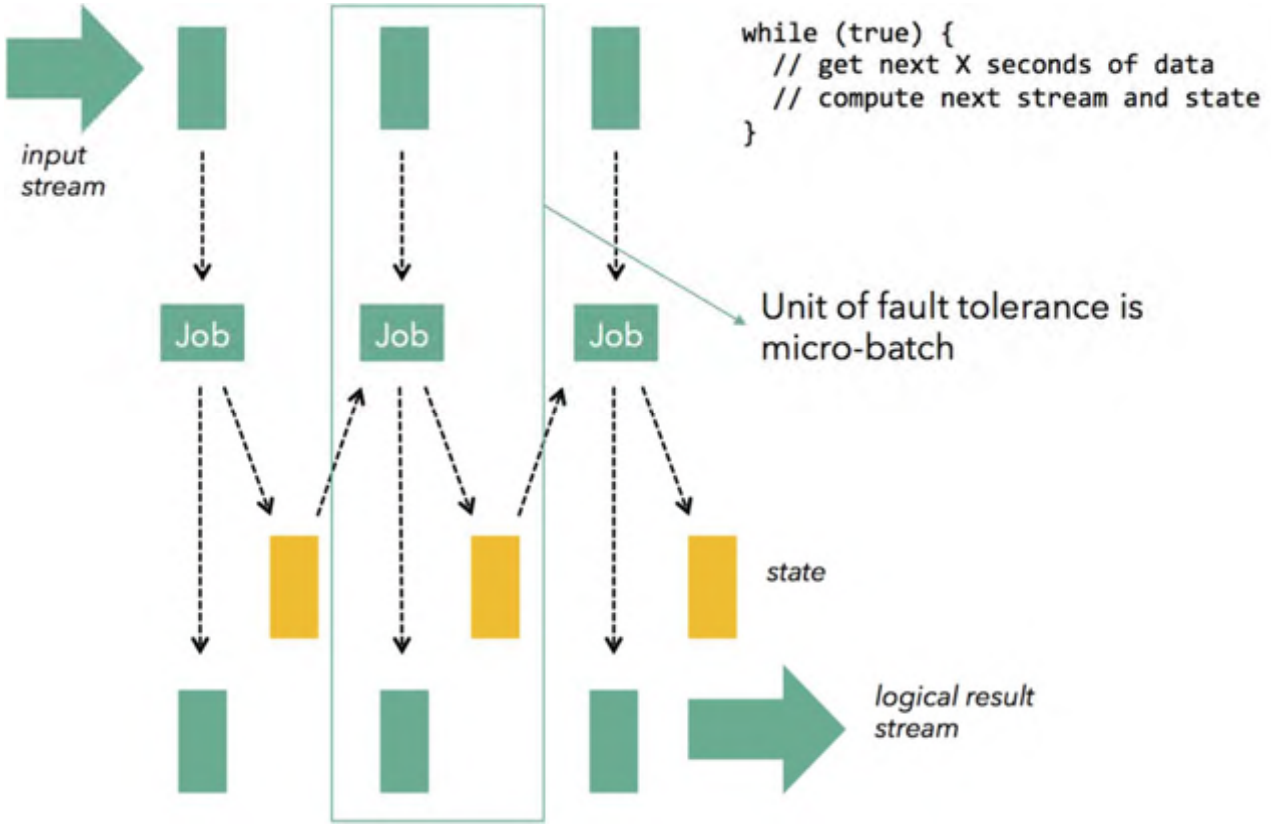


挑战二:如何保证消息只被处理一次

从原理上来讲，需要在发送tuple的时候带上txid，在需要事务处理的时候，根据该txid是否以前已经处理成功来决定是否进行处理，当然需要把txid和处理结果一起做保存。并且需要保障顺序性，在当前请求txid提交前，所有比自己低txid请求都已经提交

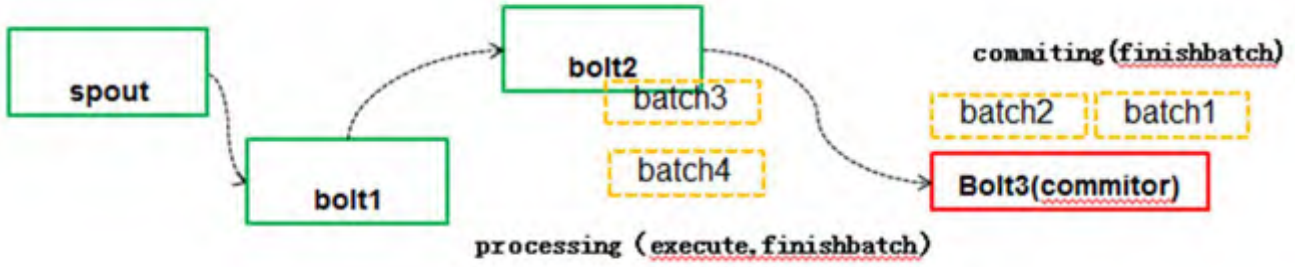
本质：状态事务性、数据一致性

- Micro-batch:Spark Streaming



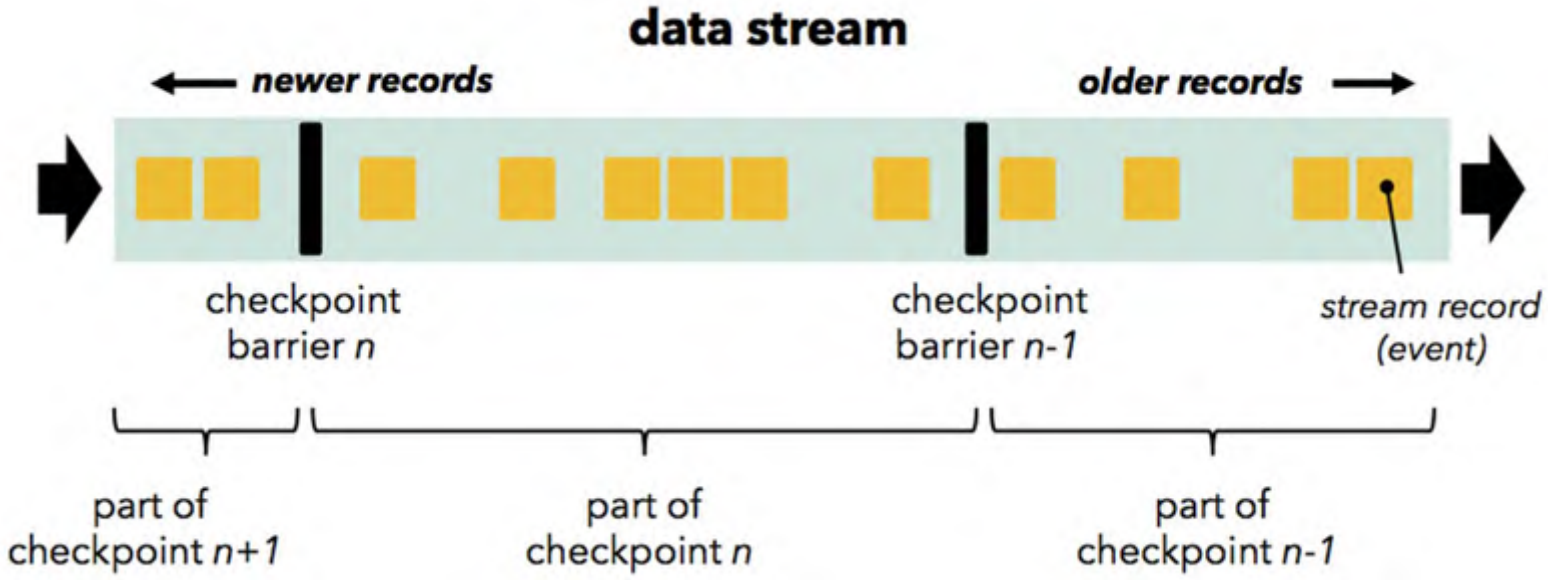
TRIDENT
STORM

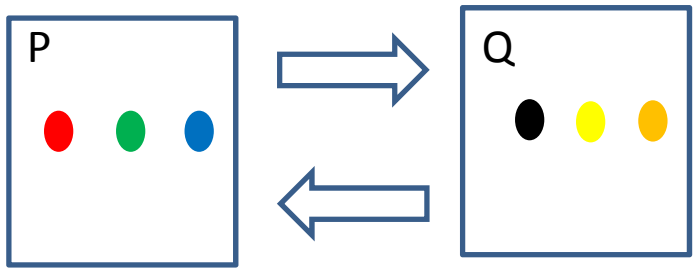
- Transactional Storm (Trident)



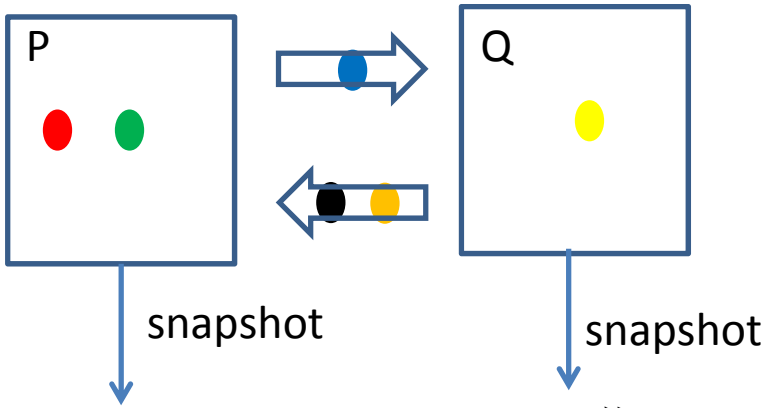
- Exactly-once,
- high throughput
- High latency

- 分布式快照(Distributed Snapshots)：Flink
- Chandy and Lamport 算法 --1985

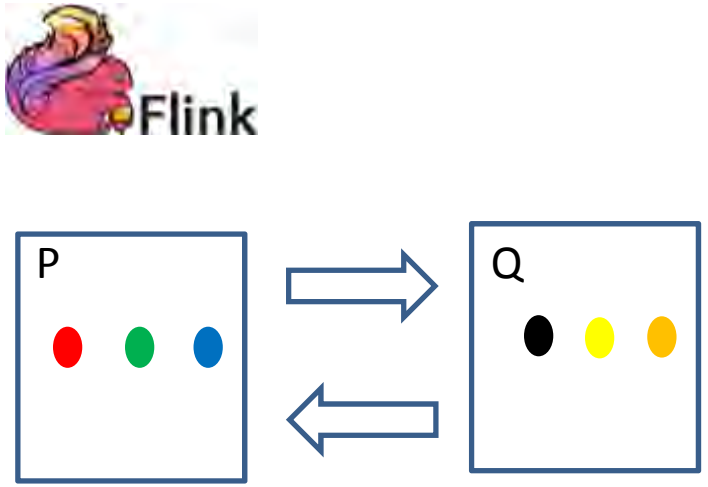




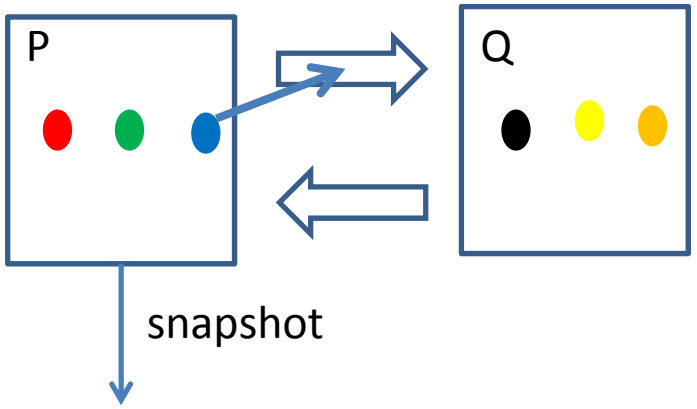
Initial State:
（P:红、绿、蓝）
（Q:黑、黄、橙)



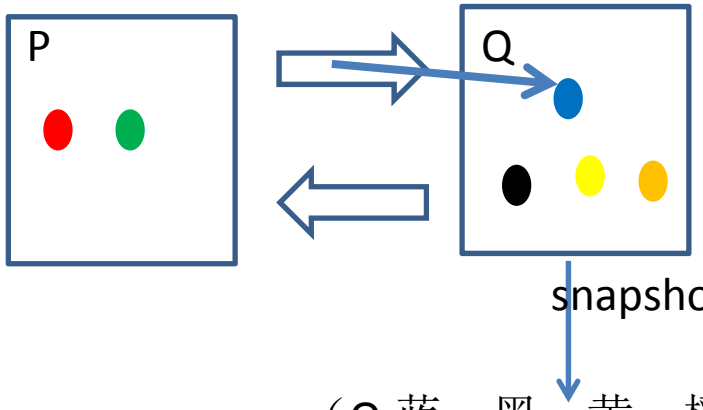
（P:红、绿）
（Q:黄)



Initial State:
（P:红、绿、蓝）
（Q:黑、黄、橙)

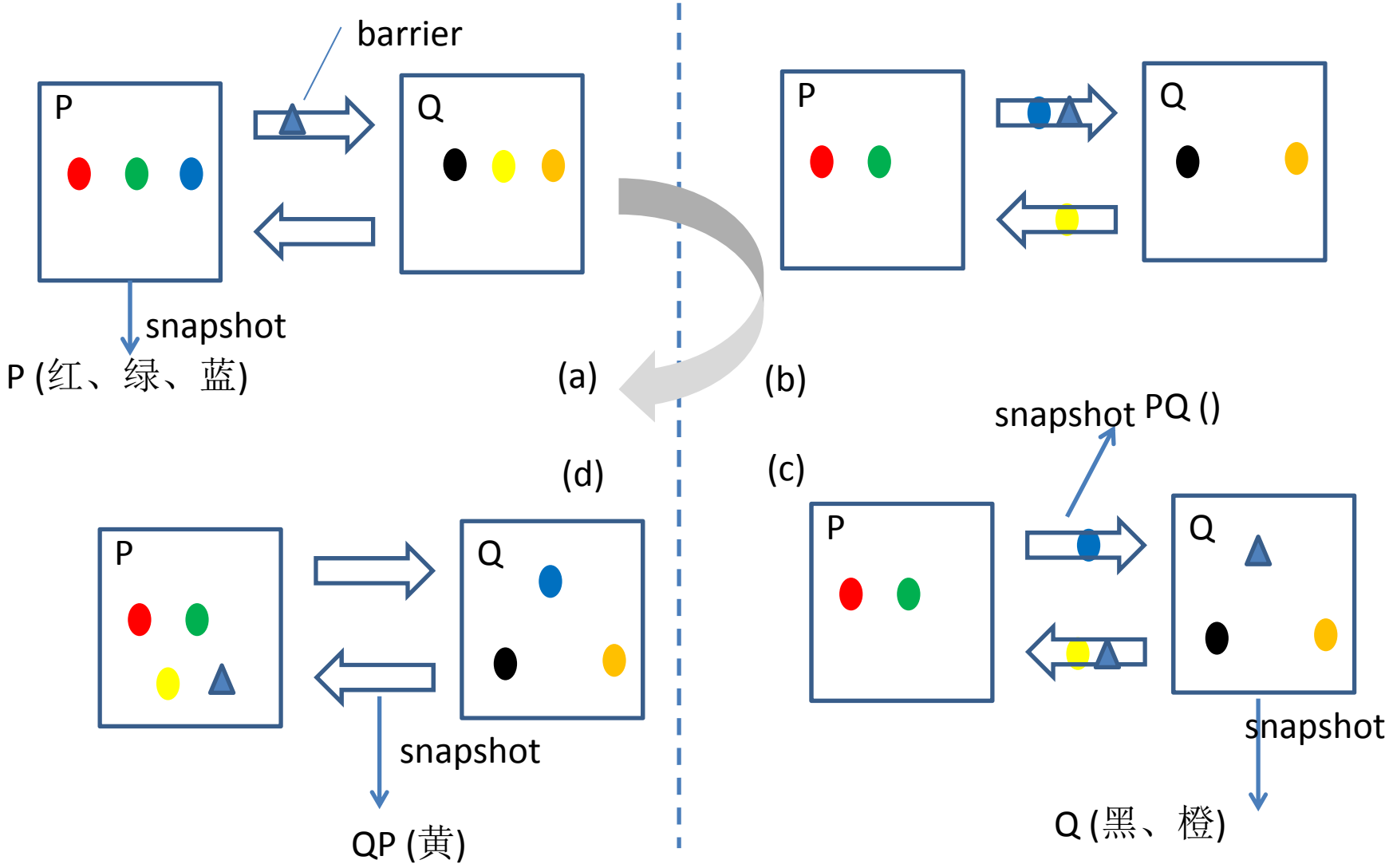


（P:红、绿、蓝）



（Q:蓝、黑、黄、橙）

- Chandy and Lamport 算法描述

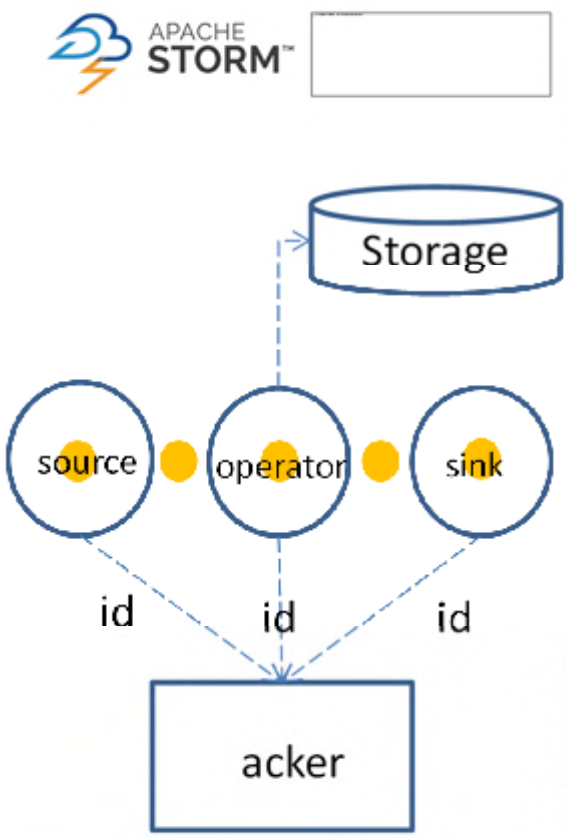


- 最终snapshot：
  - P (红、绿、蓝)
  - Channel PQ ()
  - Q （黑、橙)
  - Channel QP (黄)

常见框架对比

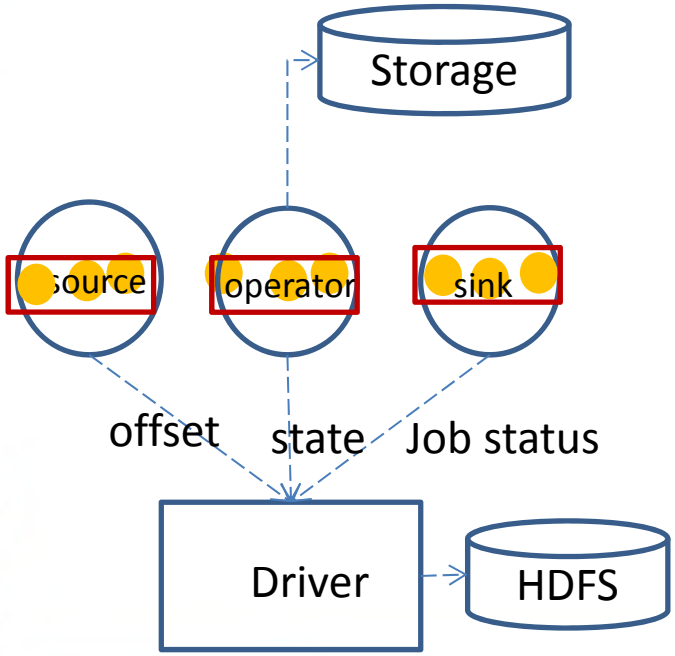
Continuous Streaming

Acker per Record



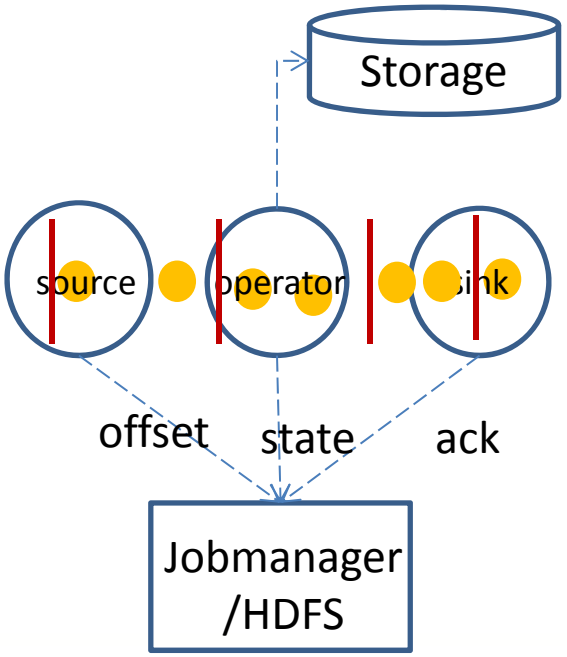
Micro-Batch

Checkpoint per batch



Continuous Streaming

Checkpoint per batch



Continuous Streaming

Acker per Record

Low Latency

High Overhead

Low Throughput

Micro-Batch

Checkpoint per batch

High Latency

Continuous Streaming

Checkpoint per batch

Low Latency

Low Overhead

High Throughput

Continuous Streaming

Acker per Record

Micro-Batch

Checkpoint per batch

Continuous Streaming

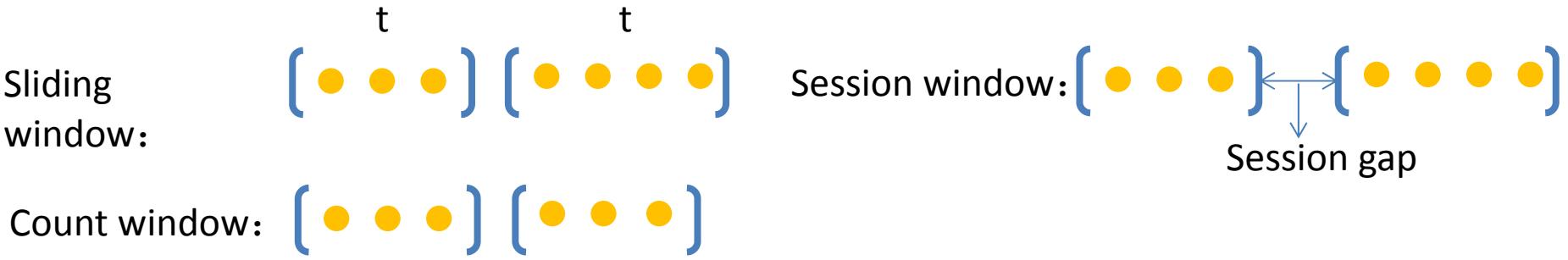
Checkpoint per batch

At least once

- Ackers know about if a record is processed successfully or not.if It failed ,replay it
- This is no state consitency guarantee

Exactly once

- State is persisted in durable storage
- Checkpoint is linked with state storage per batch

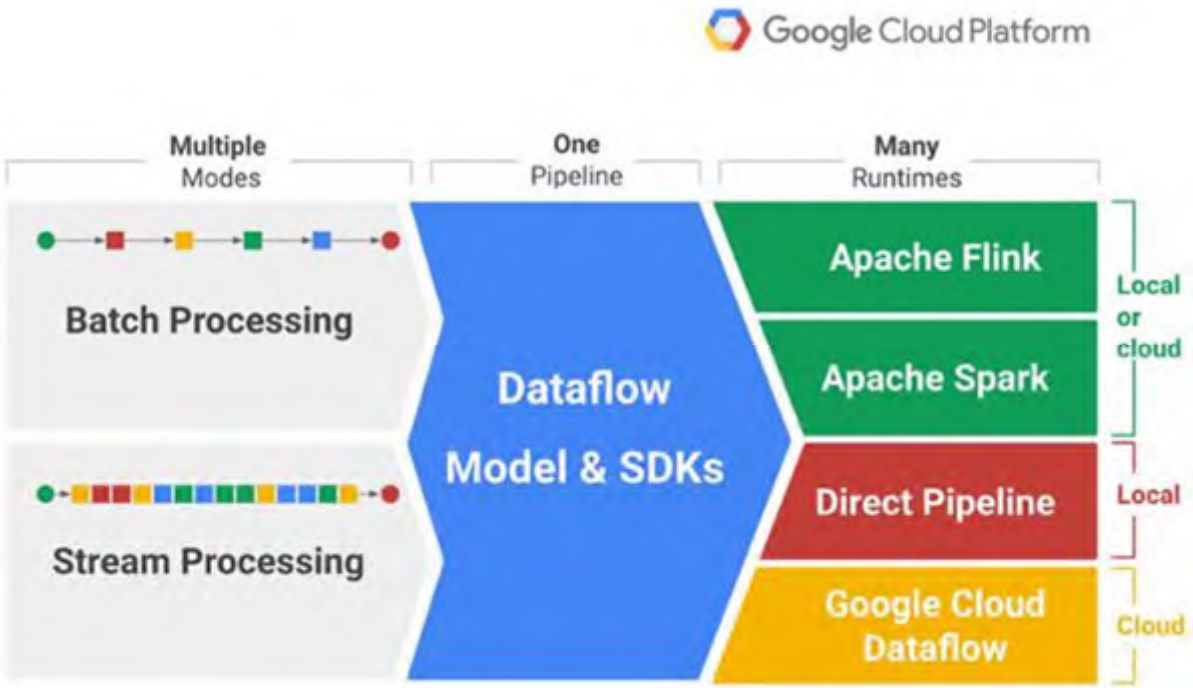


	Sliding window	Count window	Session window
Spark Streaming	√	×	×
APACHE STORM™	√	√	×
Flink	√	√	√

业界发展

- 数据天生就是流式的
- 批量是流式计算的一个特例
- 如果批量处理是第一次数据革命，流式处理则是第二次数据革命

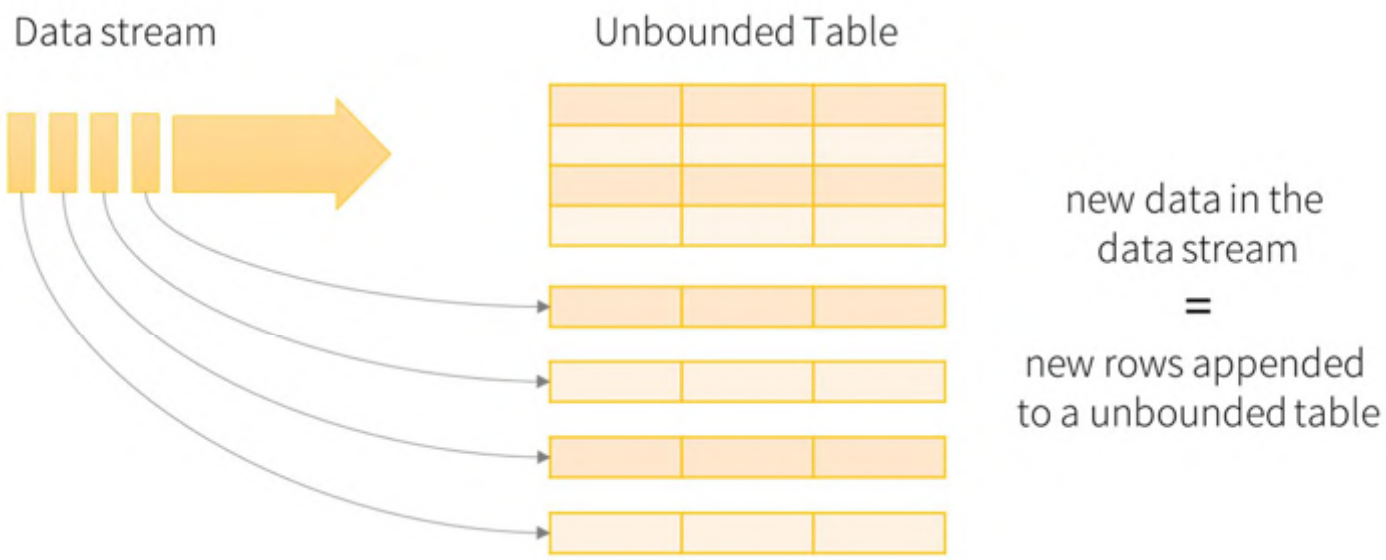
Dataflow SDK



beam

壹钱包

Spark 2.0 Structured Streaming



Data stream as an unbounded table

Thanks!

Q & A