

Apache Flink结合Apache Kafka实现端到端的一致性语义

周凯波 · 阿里巴巴 / 技术专家

Apache Kafka x Apache Flink·北京— 2019年05月12日

CONTENT

目录 >>

01 / 流计算中的一致性语义

02 / 流计算系统如何支持一致性语义

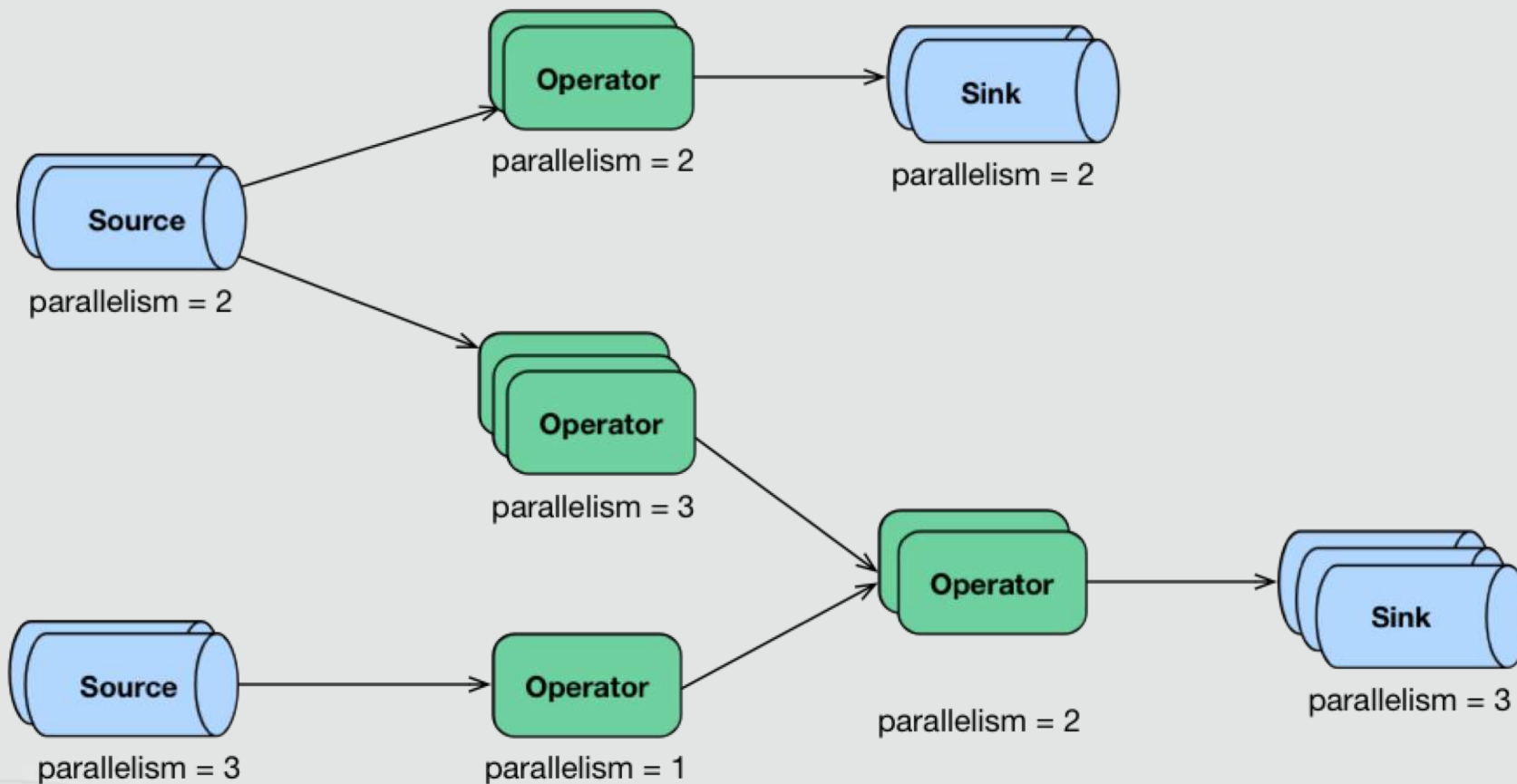
03 / Flink + Kafka如何实现
End-To-End Exactly-Once

04 / Q&A

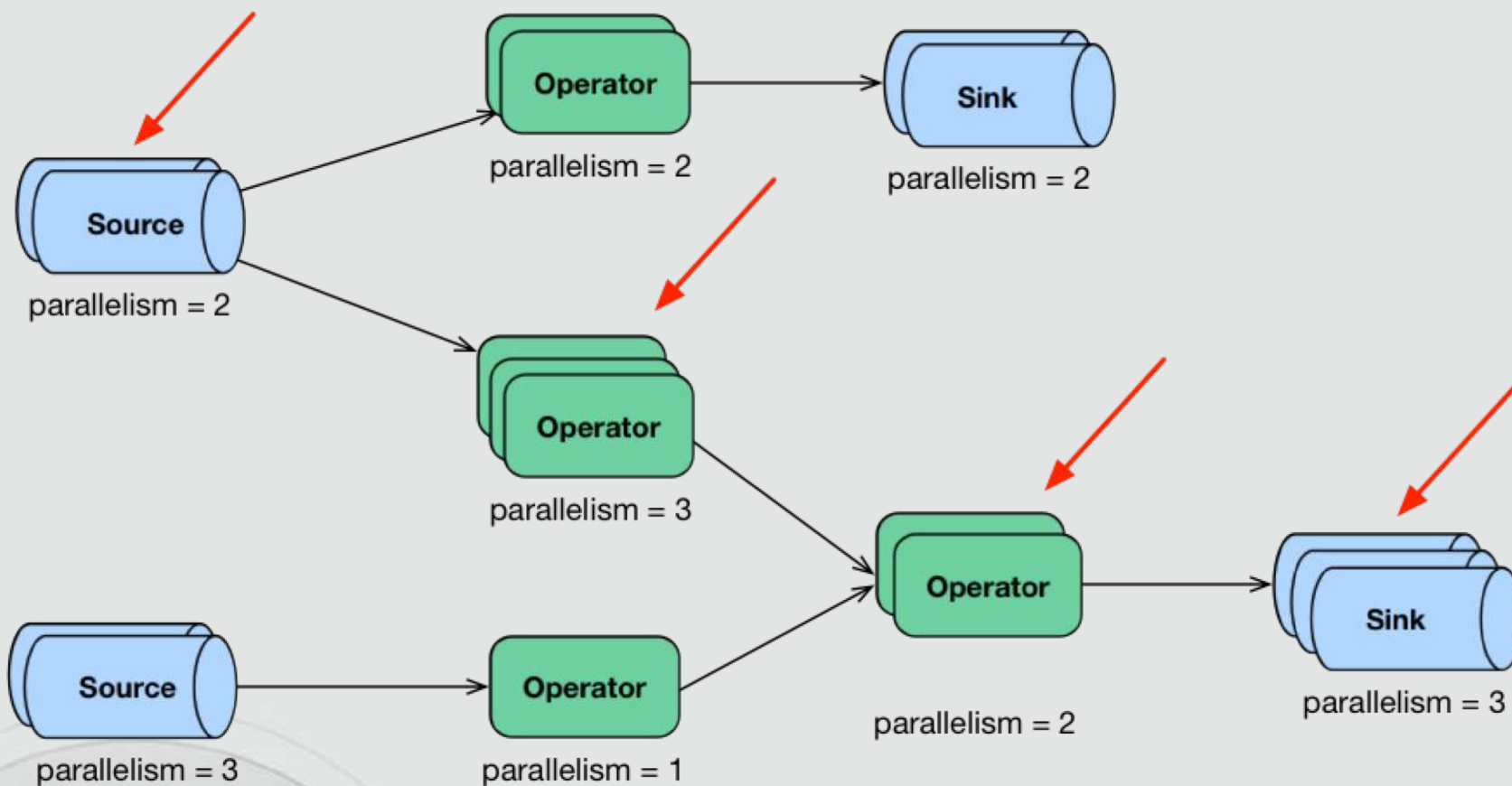
01

流计算中的一致性语义

典型流计算场景



典型流计算场景



流计算中的一致性语义

At Most Once

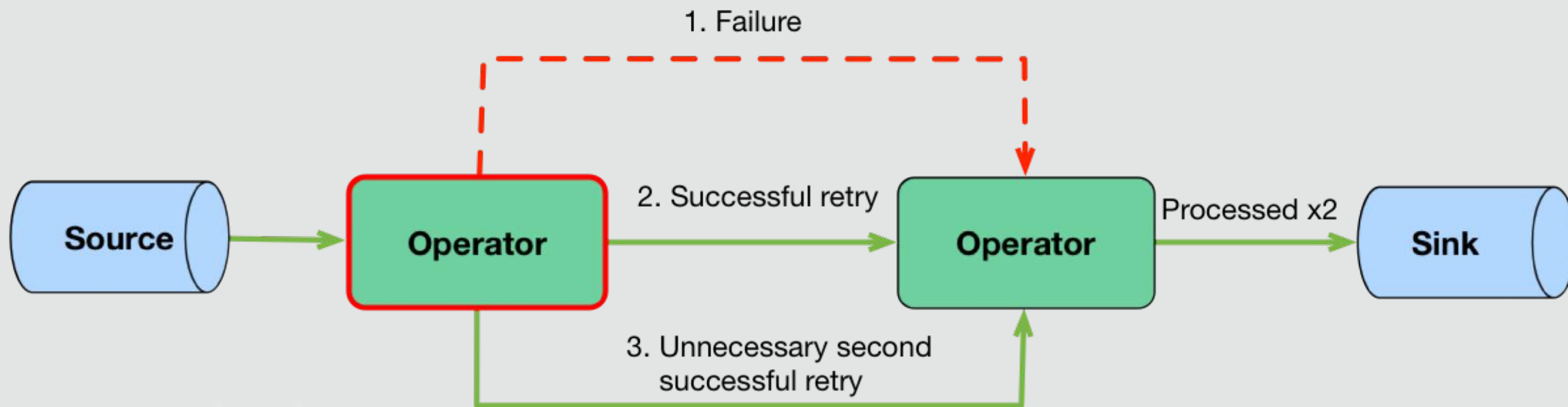
最多计算一次



流计算中的一致性语义

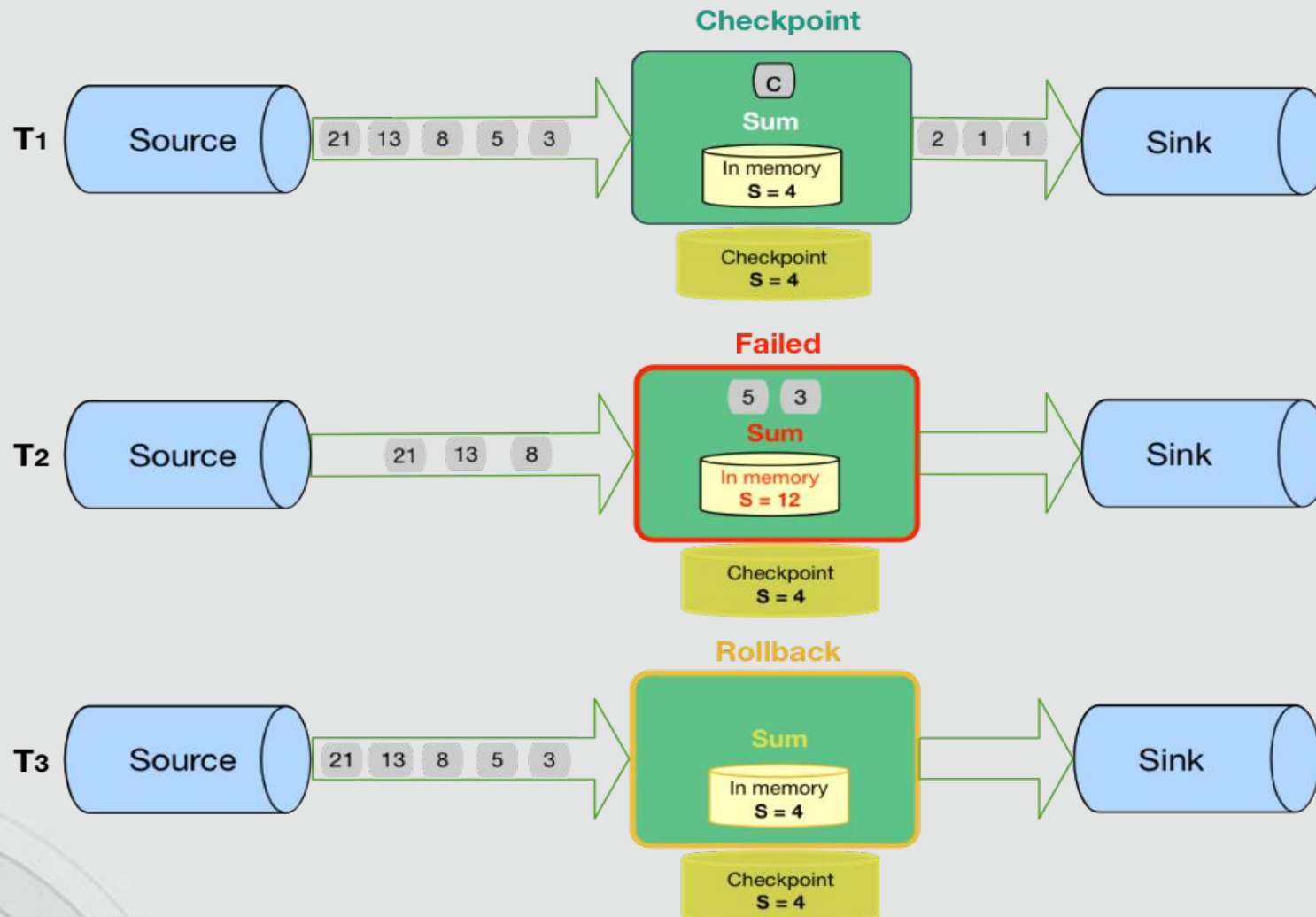
At Least Once

至少计算一次



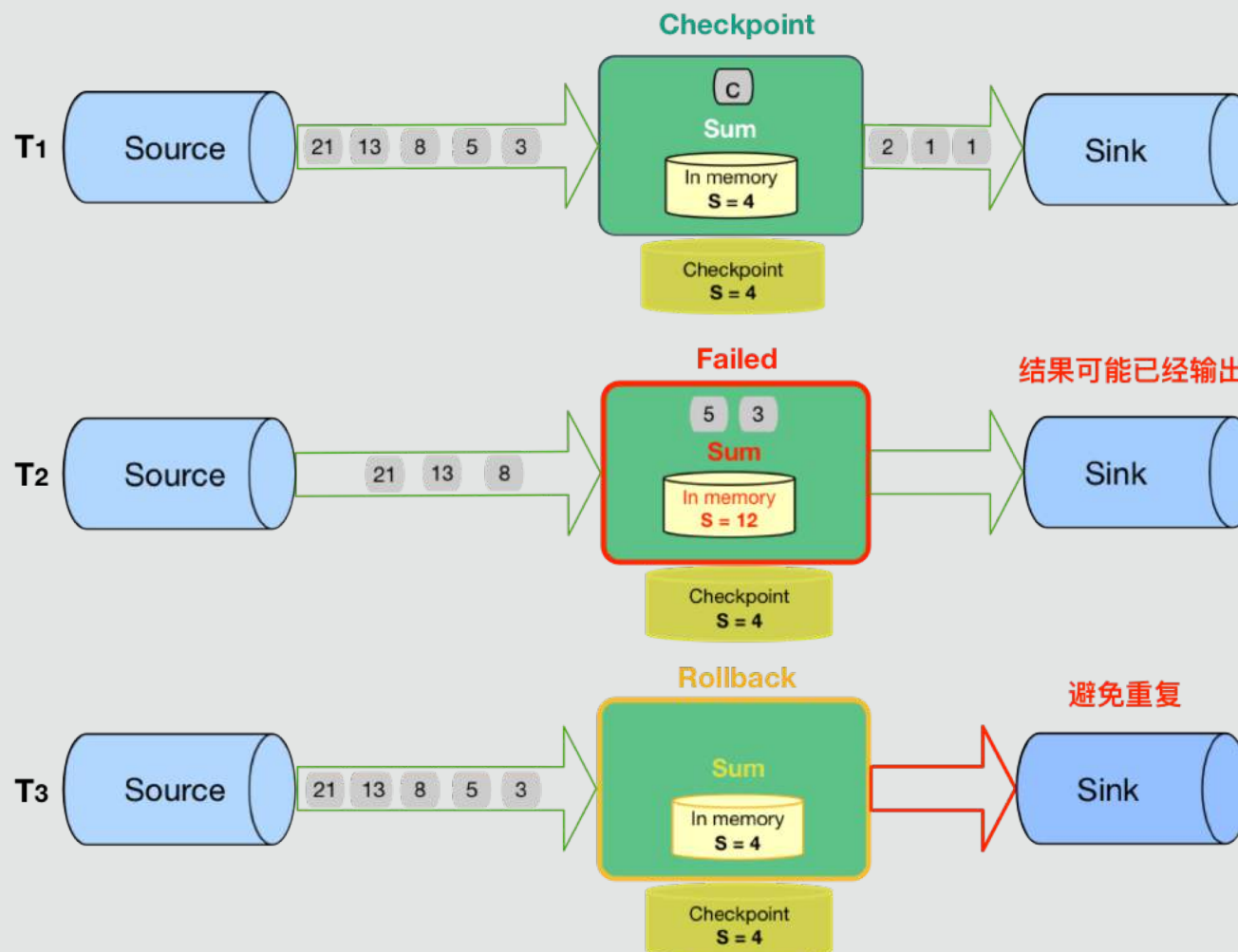
流计算中的一致性语义

Exactly Once
精确计算一次

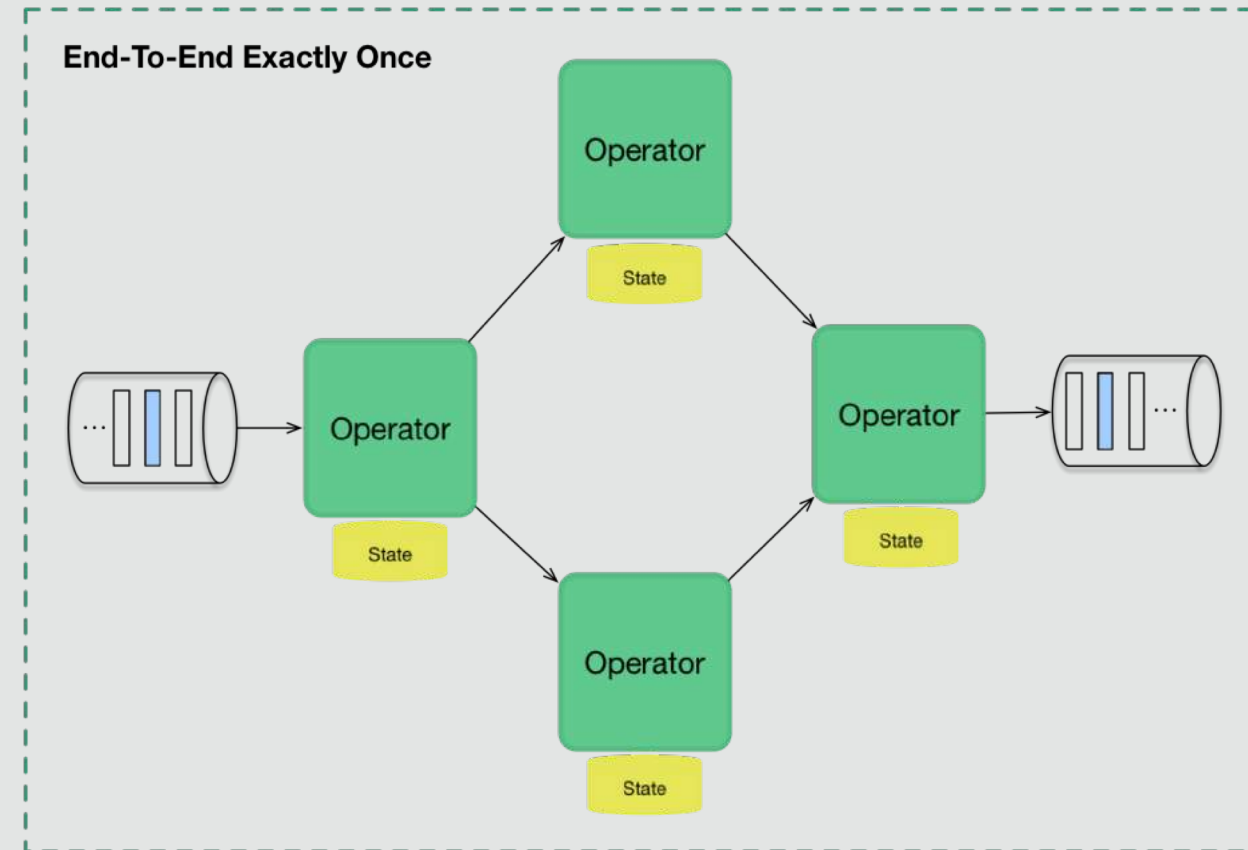
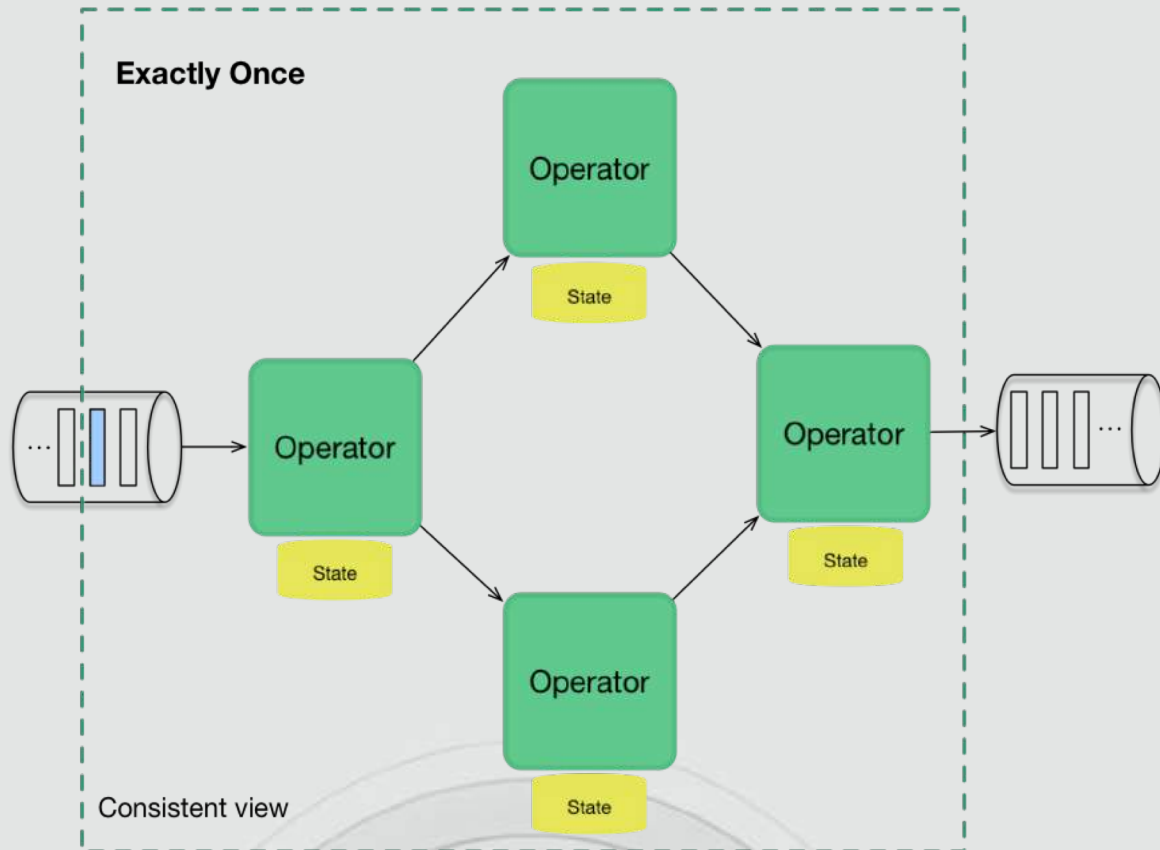


流计算中的一致性语义

End-To-End Exactly Once
端到端的一致性



Exactly Once vs End-To-End Exactly Once



Exactly Once vs End-To-End Exactly Once

Exactly Once

- 保证所有记录仅影响内部状态一次

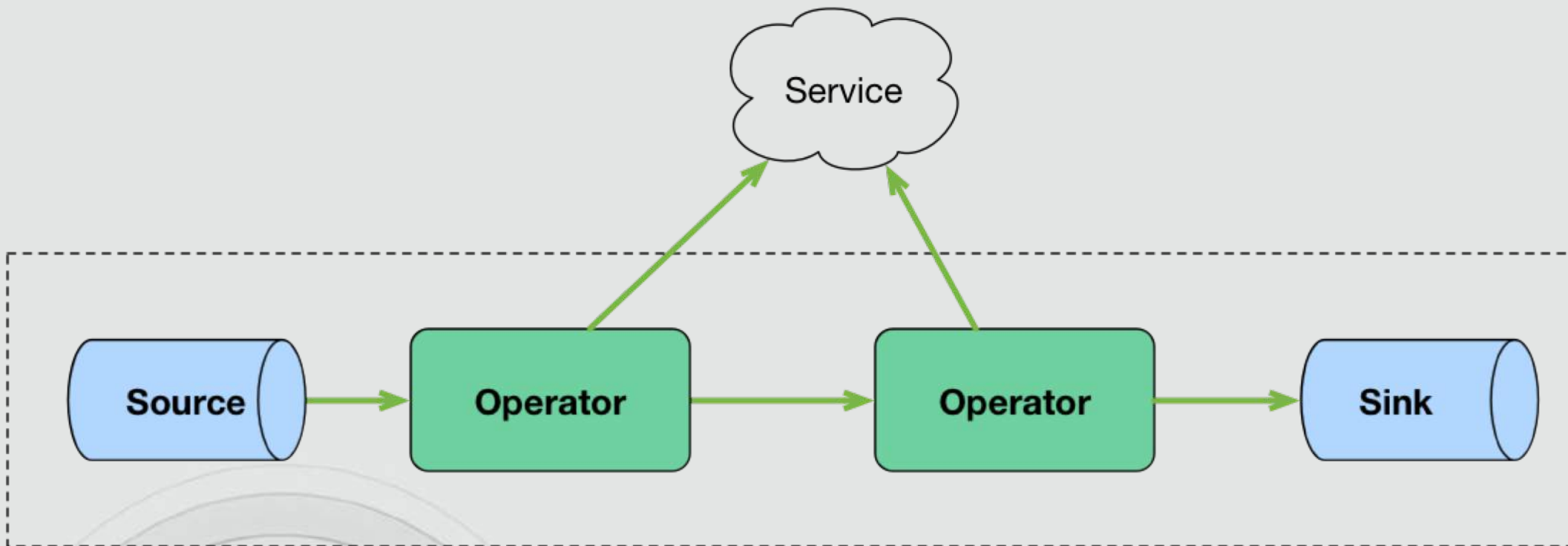
End-To-End Exactly Once

- 保证所有记录仅影响内部和外部状态一次

Side-effect

UDF 等访问的外部服务

- 例如：计数器，GUID生成



01 总结

一致性由弱到强

At most once < At least once < Exactly once < End-To-End Exactly once

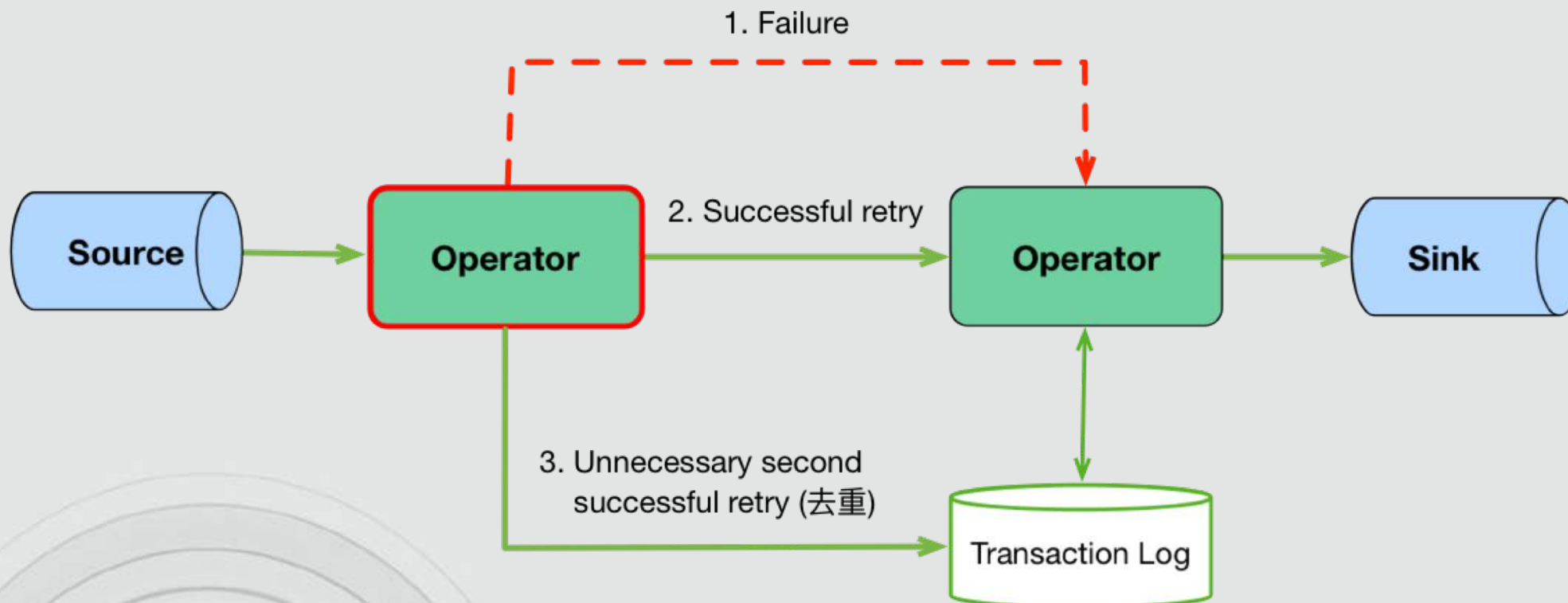
02

流计算系统如何支持一致性语义

At least once + 去重

每个算子维护一个事务日志，跟踪已处理的事件
重放失败事件，在事件进入下个算子之前，移除重复事件

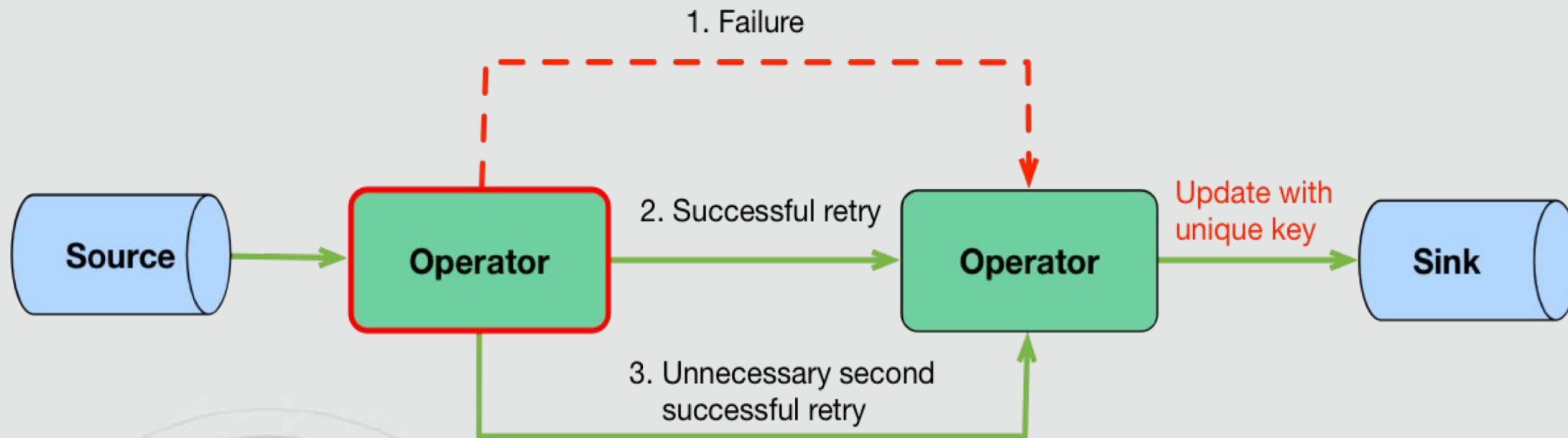
- 如 Google MillWheel



At least once + 幂等

依赖 Sink 端存储的去重性和数据特征

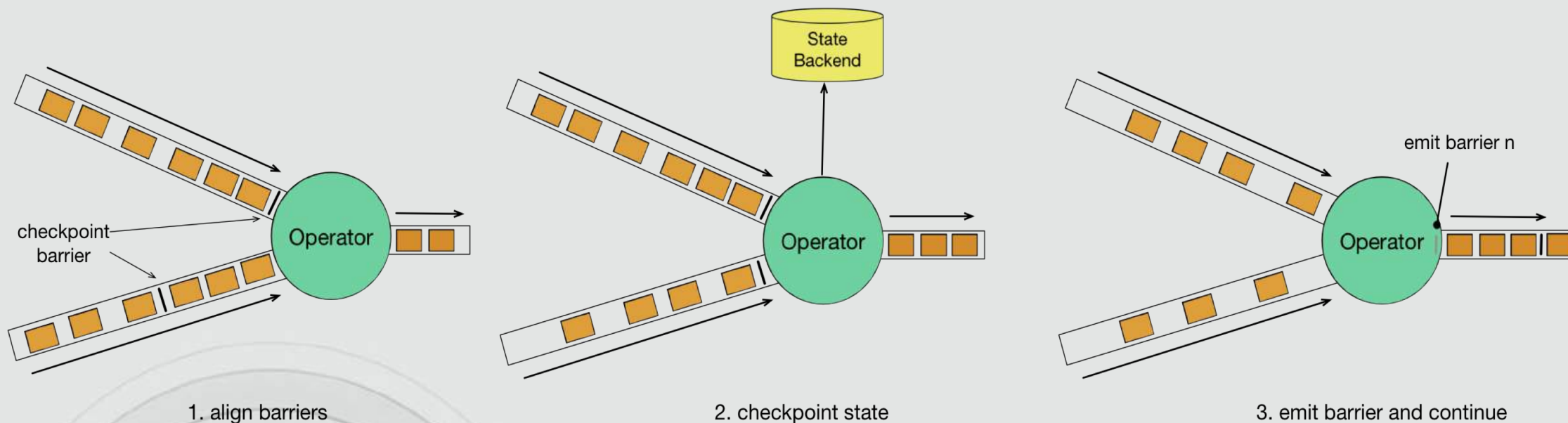
- 如输出到数据库，通过 replace into + unique key



分布式快照

Chandy-Lamport 分布式快照算法

- 引入barrier，把 input stream 分为 preshot records 和 postshot records
- Operator 收到所有上游 barrier 的时候做一个snapshot，继续往下处理
- 当所有 Sink Operator 都完成了Snapshot，这一轮 Snapshot 就完成了



02 总结

Exactly Once实现方式	优点	缺点
At least once + 去重	<ul style="list-style-type: none">故障对性能的影响是局部的故障的影响不一定会随着拓扑的大小而增加	<ul style="list-style-type: none">可能需要大量的存储和基础设施来支持每个算子的每个事件的性能开销
At least once + 幂等	<ul style="list-style-type: none">实现简单，开销较低	<ul style="list-style-type: none">依赖存储特性和数据特征
分布式快照	<ul style="list-style-type: none">较小的性能和资源开销	<ul style="list-style-type: none">barrier 同步任何算子发生故障，都需要发生全局暂停和状态回滚（Region Failover: FLINK-4256）拓扑越大，对性能的潜在影响越大

03

Flink + Kafka如何实现End-To-End Exactly Once

Flink对Exactly Once的支持

Flink 1.4 版本之前

- 支持 Exactly-Once 语义，仅限于应用程序内部

Flink 1.4 之后

- 通过两阶段提交 (TwoPhaseCommitSinkFunction) 支持 End-To-End Exactly Once
- kafka 0.11 +

分布式快照

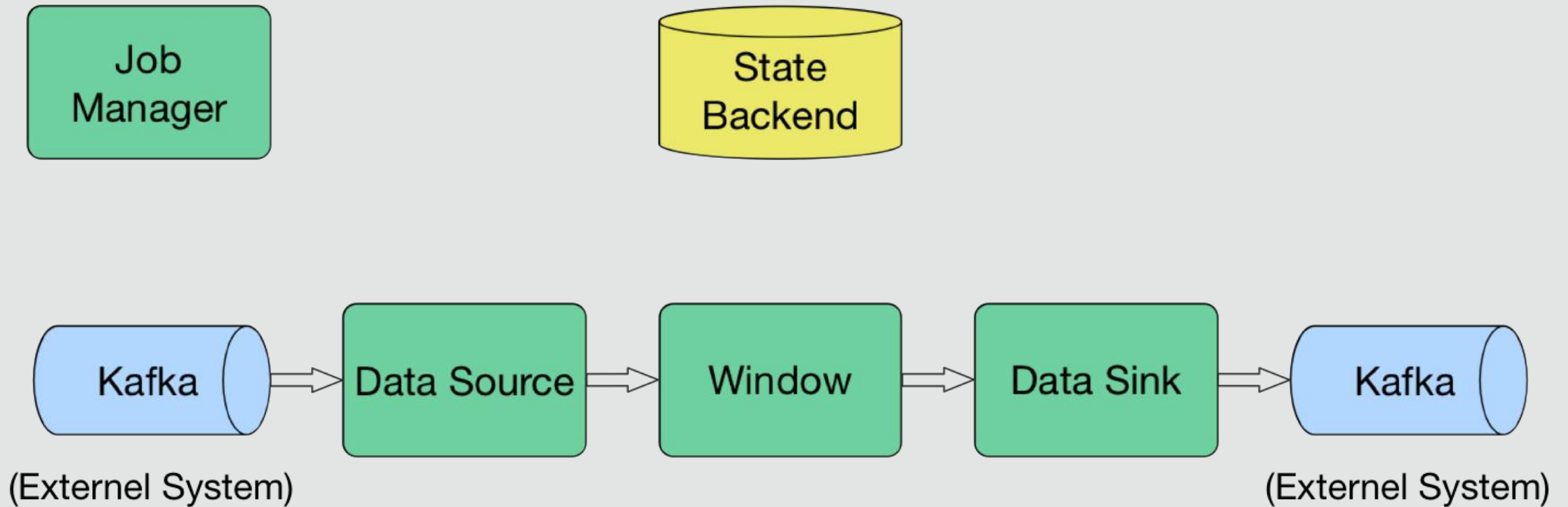
- 输入流的位置
- 当前状态

Flink对Exactly Once的支持

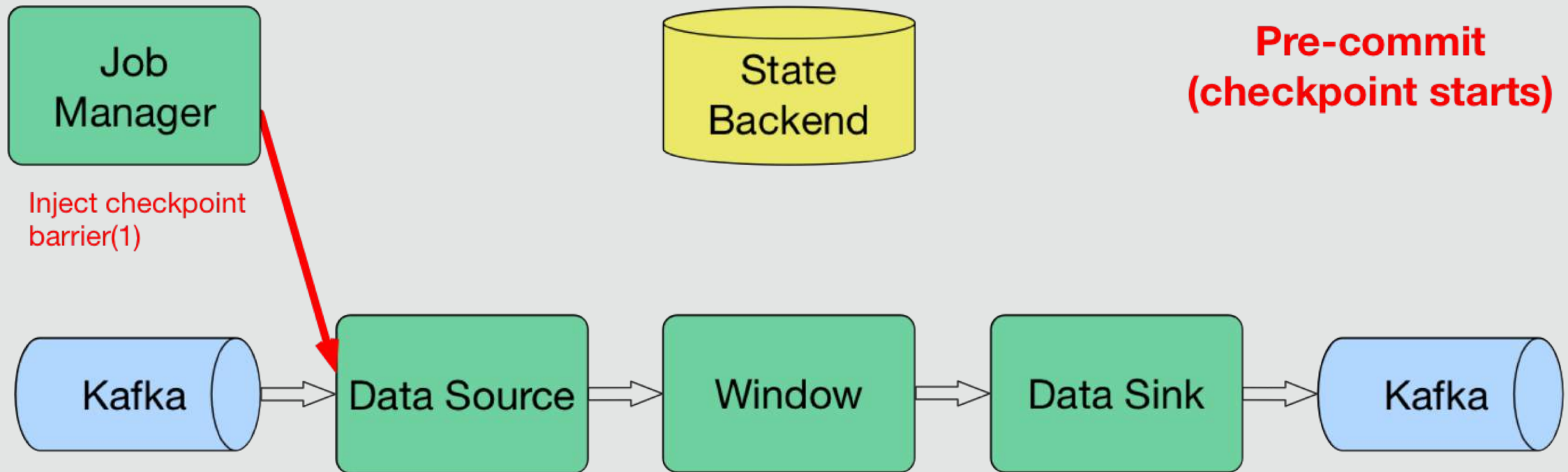
TwoPhaseCommitSinkFunction

- CheckpointedFunction
 - initializeState
 - snapshotState
- CheckpointListener
 - notifyCheckpointComplete

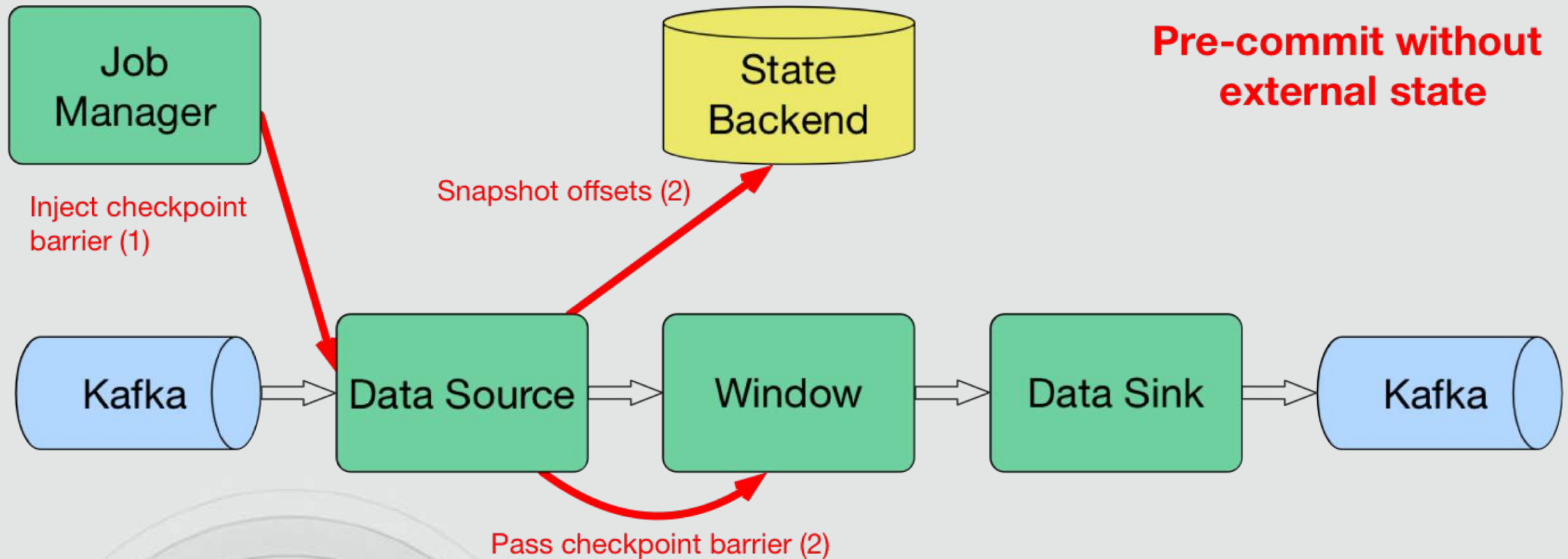
Exactly Once two-phase commit



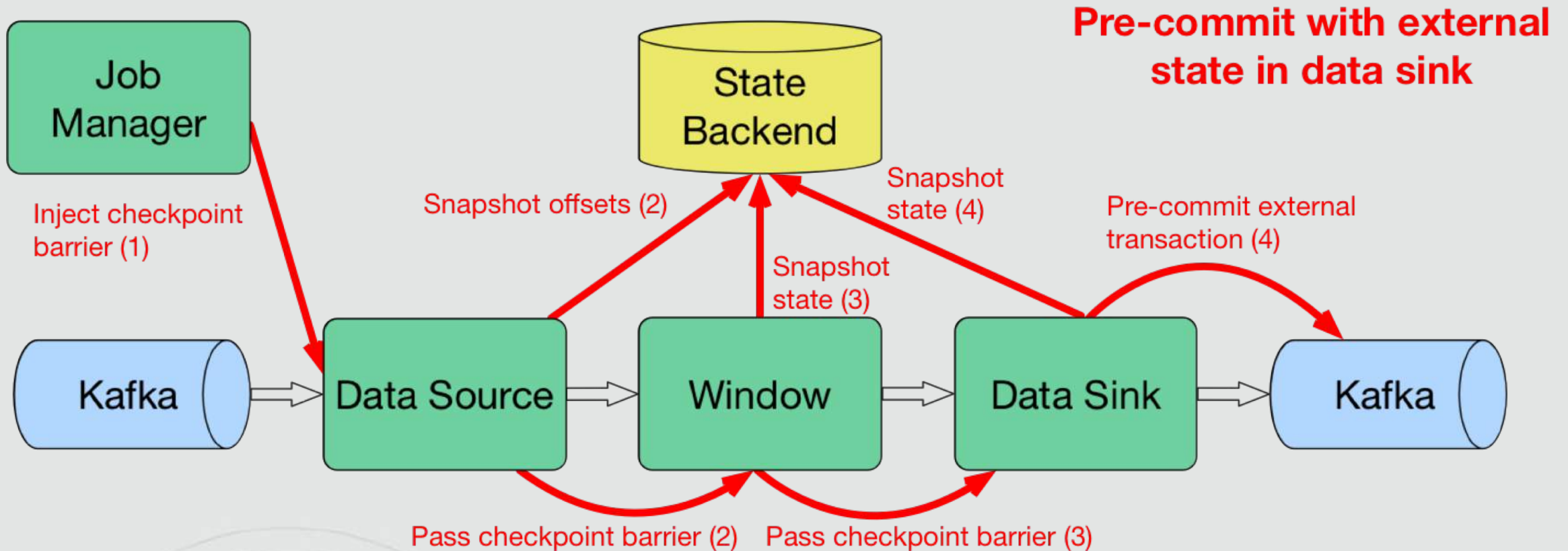
Exactly Once two-phase commit



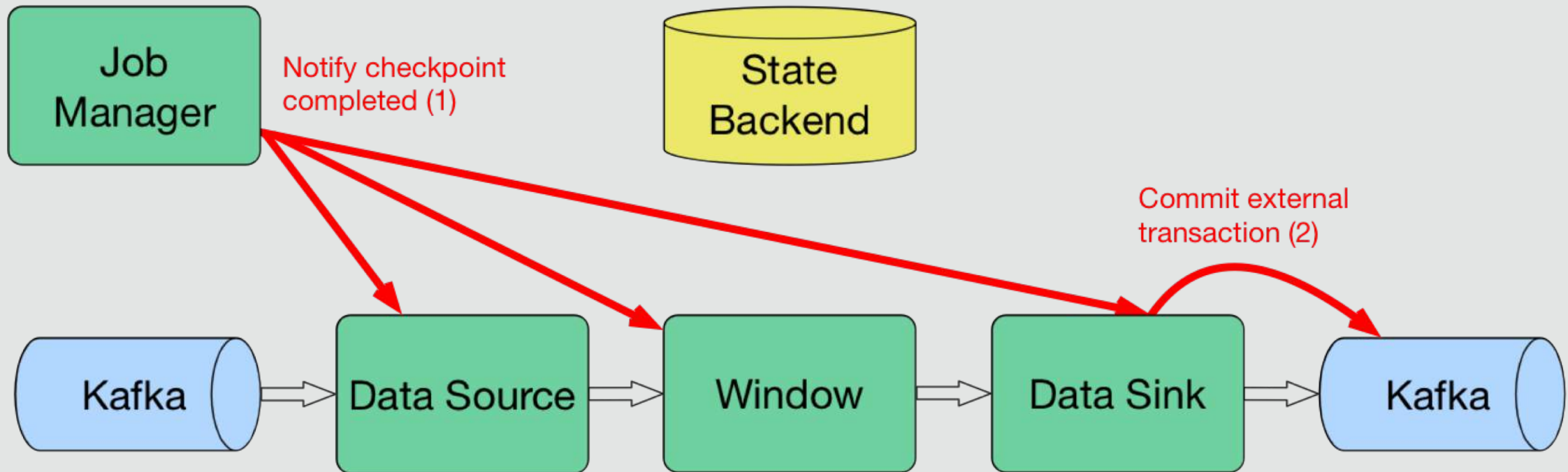
Exactly Once two-phase commit



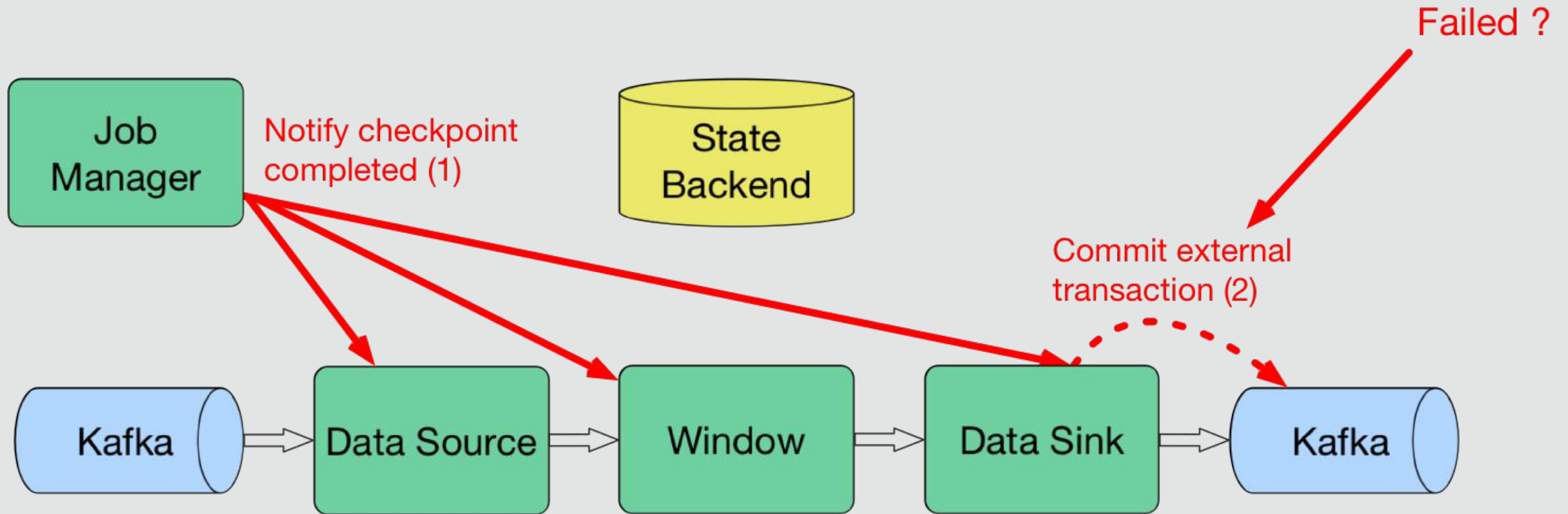
Exactly Once two-phase commit



Exactly Once two-phase commit



Exactly Once two-phase commit



Exactly Once two-phase commit

```
public interface CheckpointListener {  
  
    /**  
     * This method is called as a notification once a distributed checkpoint has been completed.  
     *  
     * Note that any exception during this method will not cause the checkpoint to  
     * fail any more.  
     *  
     * @param checkpointId The ID of the checkpoint that has been completed.  
     * @throws Exception  
     */  
    void notifyCheckpointComplete(long checkpointId) throws Exception;  
}
```

03 总结

- Flink 通过两阶段提交支持 End-To-End Exactly Once
- 需要存储系统支持事务提交


04

Q&A

THANKS

Flink China社区大群



 扫一扫群二维码，立刻加入该群。