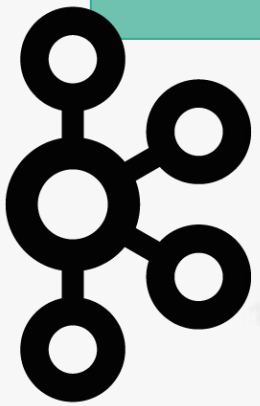
The Stream Processing Landscape: Mindset and Technologies

戴资力 · Ververica / Software Engineer / Apache Flink PMC

Apache Flink Meetup 深圳 - 2019年04月13日



Pravega

LogDevice

? 流式处理重点在于即时性

? 流式处理只用于数据分析

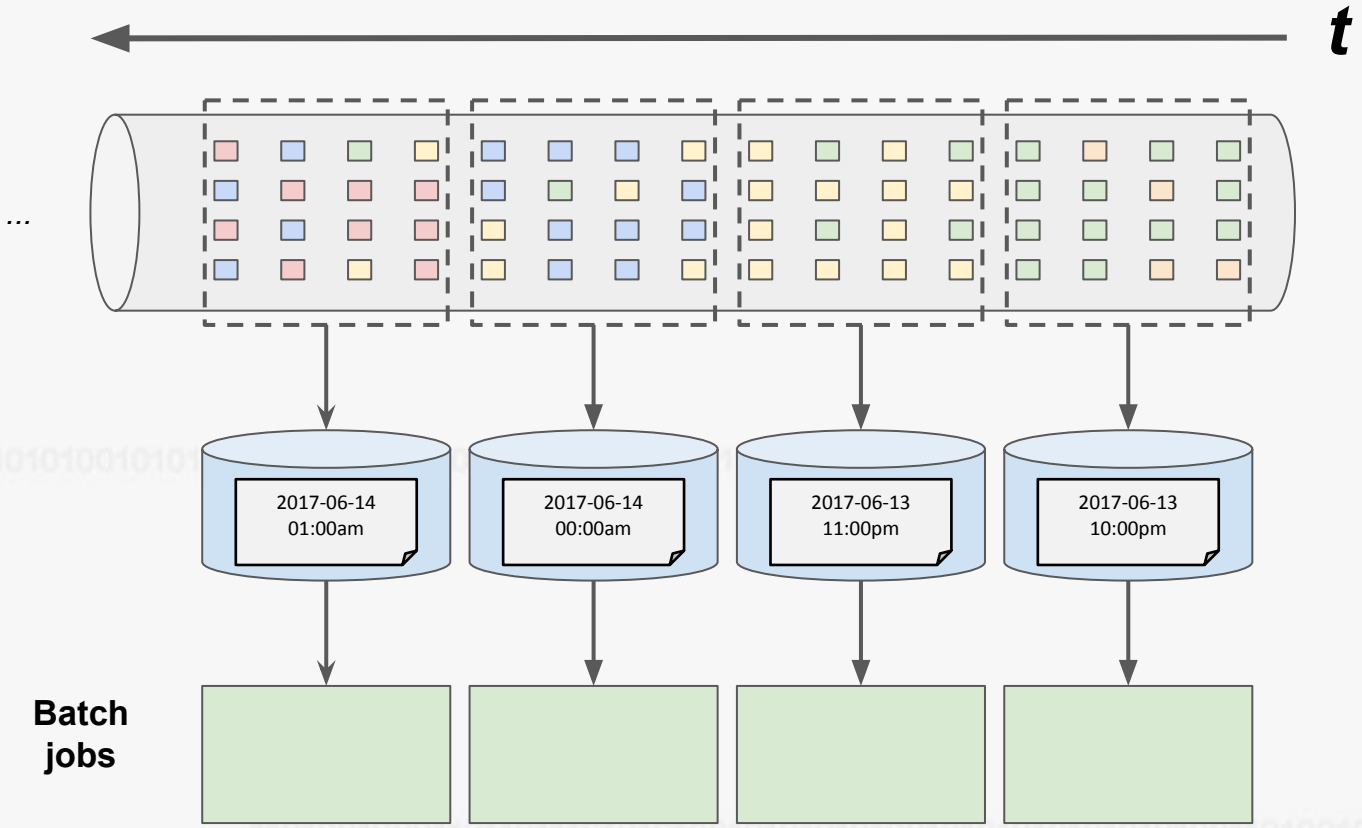
? 流式处理已经发展完毕

？流式处理重点在于即时性

？ 流式处理只用于数据分析

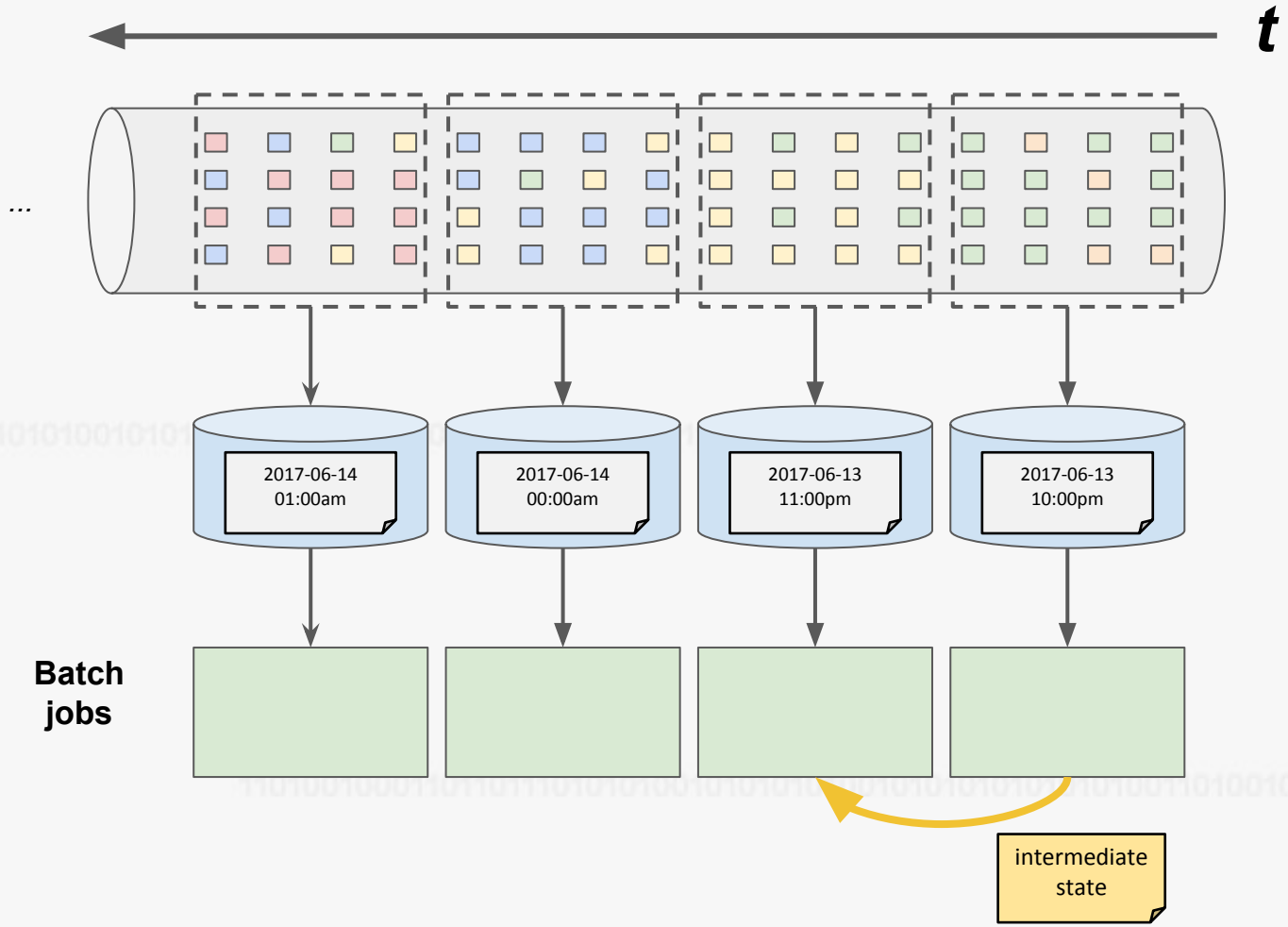
？流式处理已经发展完毕

传统批次处理方法



- 持续收取数据
- 以时间作为划分数个批次档案的依据
- 周期性执行批次运算

传统批次处理方法

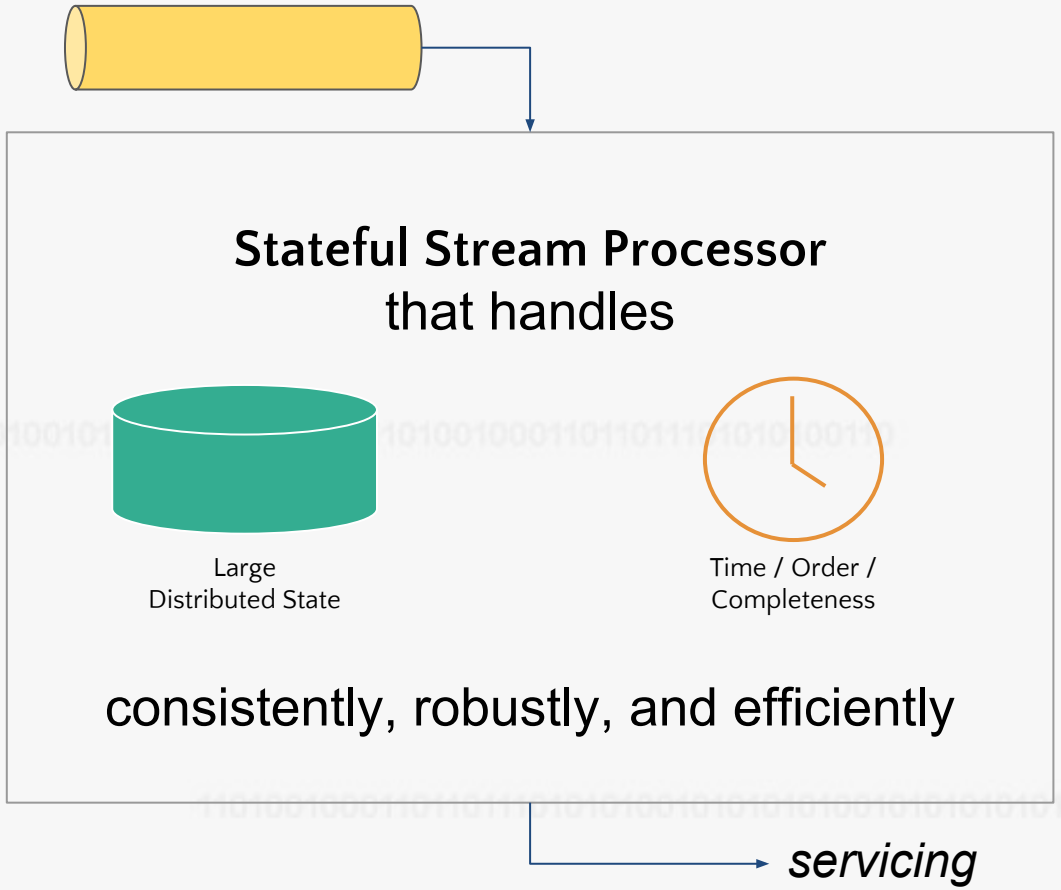


- 假设计算每小时出现特定事件转换的次数
  (# of A → B per hour)
- 如果事件转换跨越了所定义的时间划分?
  → 将中介运算结果(intermediate result) 带到下一个批次运算
- 如果接收到事件的顺序颠倒?

理想方法

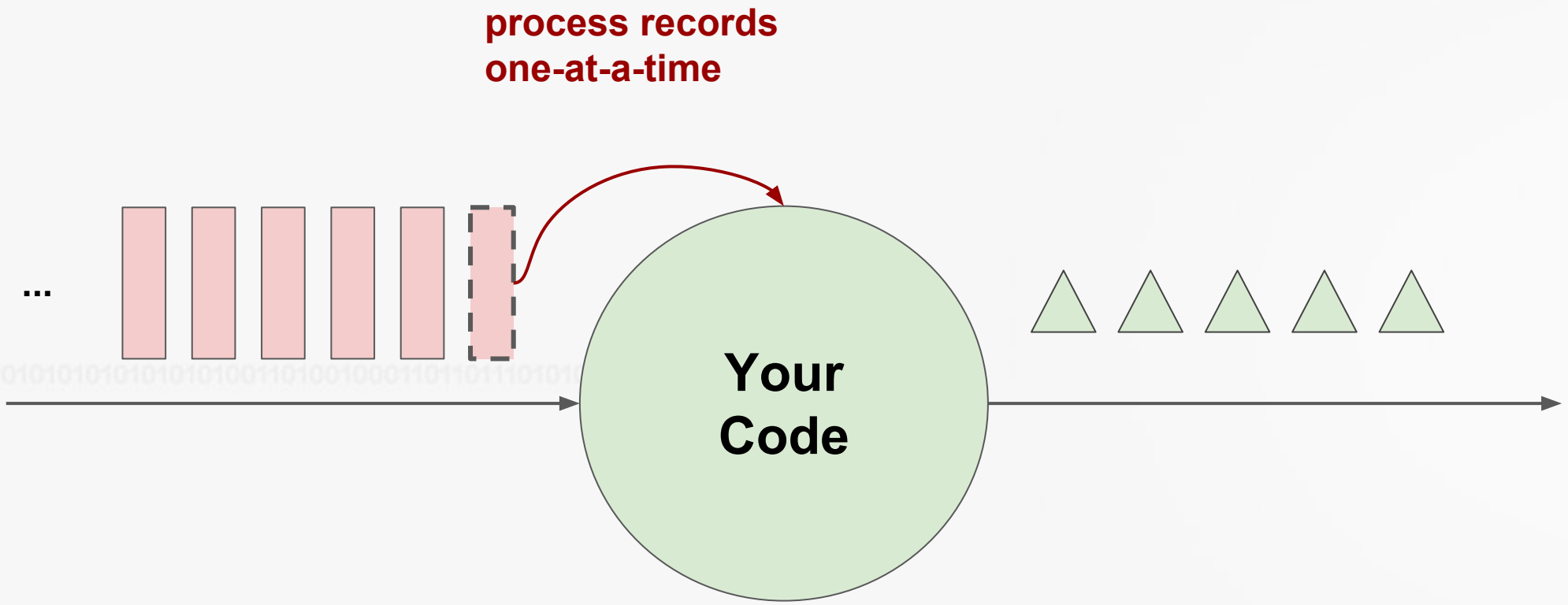

累积状态
累积的状态代表着过去历史中接收过的所有事件
counters, in-progress windows
ML model 的设定参数
累积的状态会影响输出
当所需的数据都完全接收到后，输出运算结果
对于「数据完整性」有机制可以操控
long running computation

理想方法



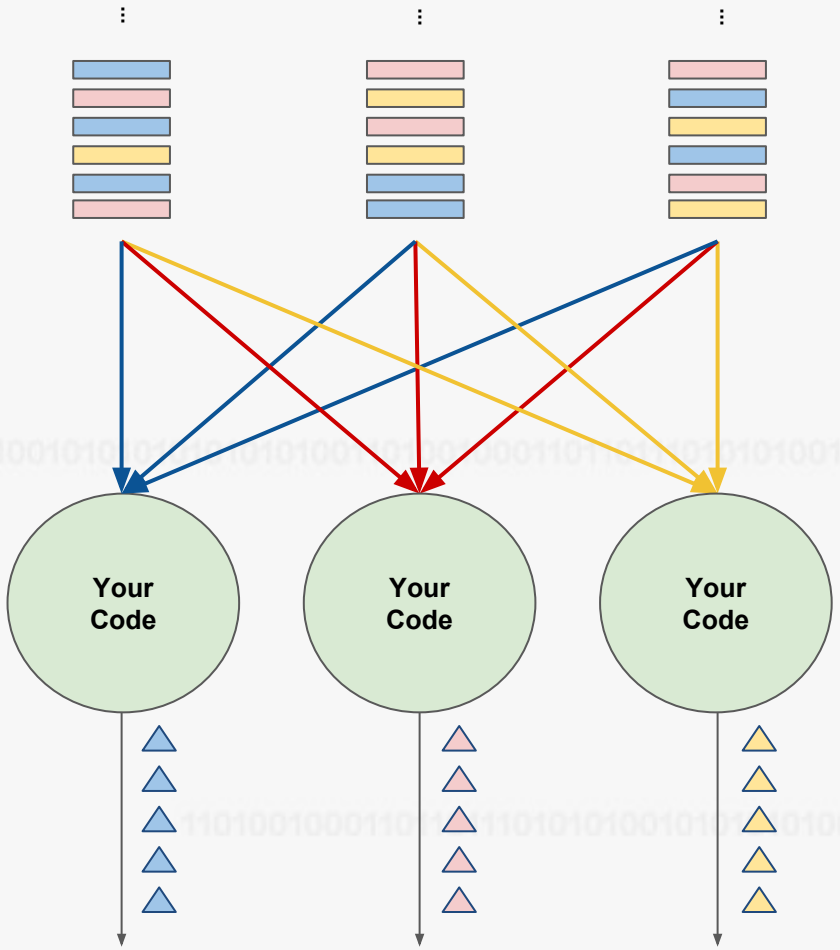
- Stateful stream processing as a **new paradigm** to **continuously process continuous data**
- Produce **accurate** results
- Results available in real-time is only a natural consequence of the model

流式处理



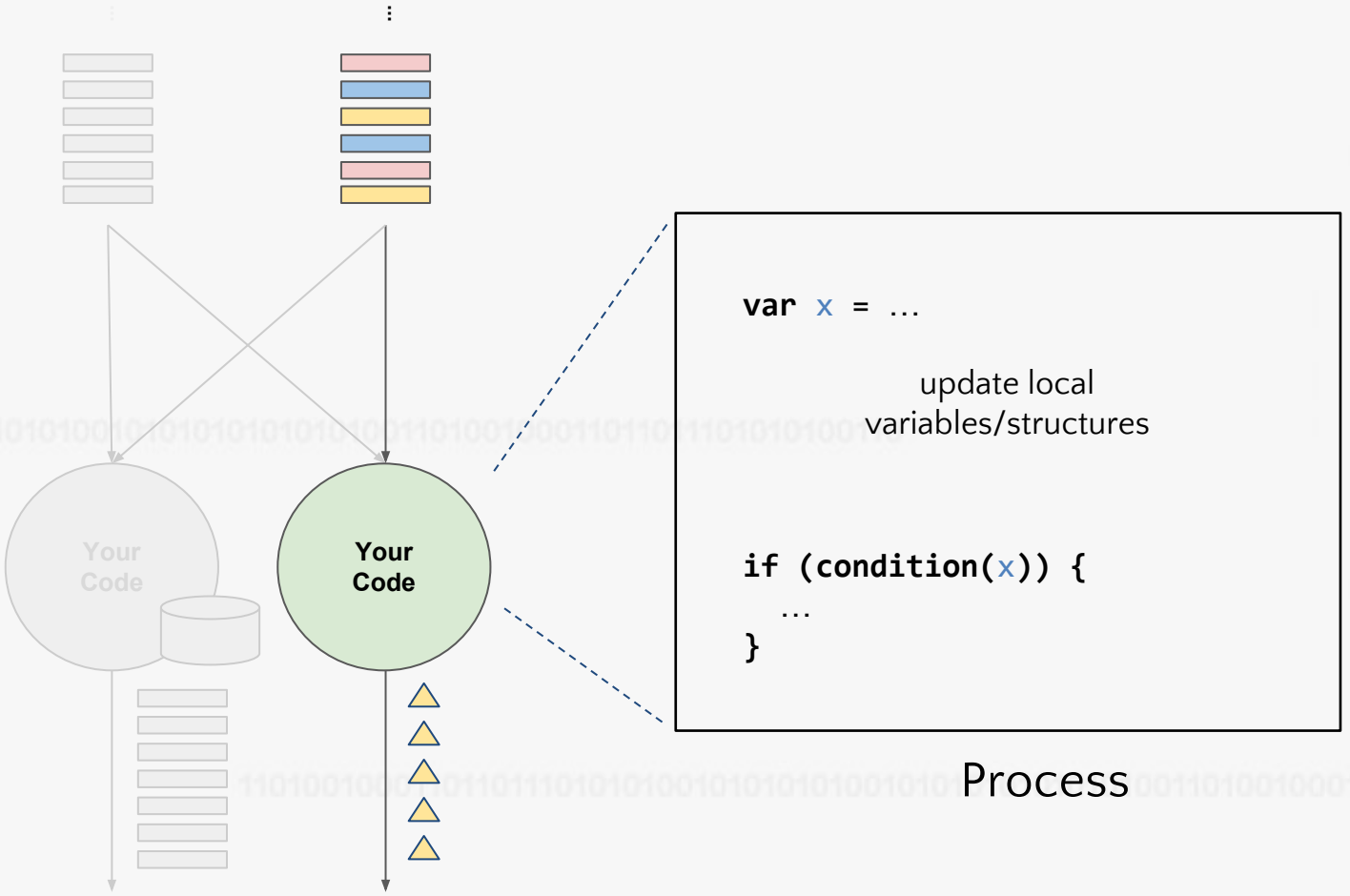
Long running computation, on an endless stream of input

分散式流式处理



- partitions input streams by some key in the data
- distributes computation across multiple instances
- Each instance is responsible for some key range

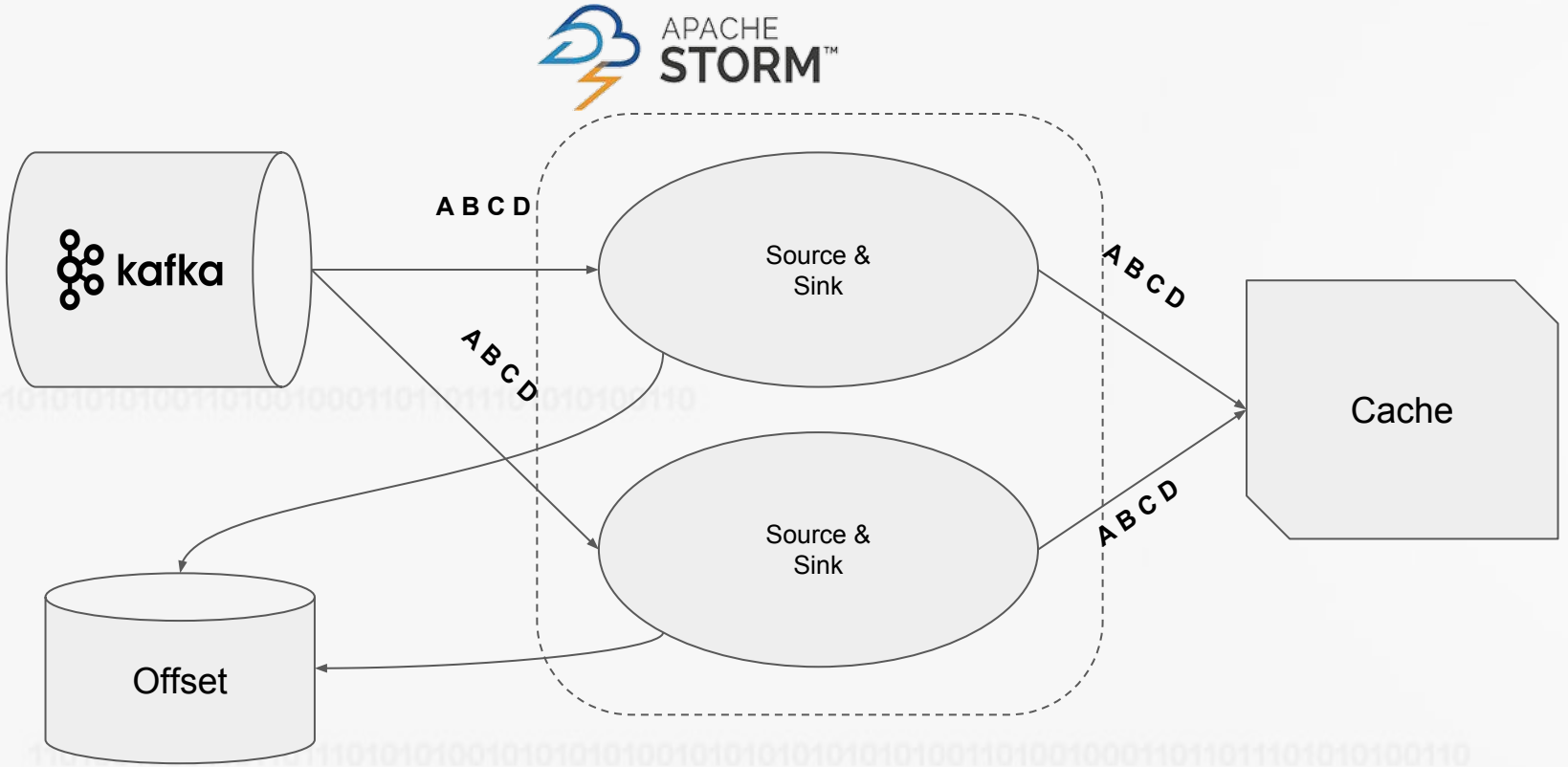
有状态分散式流式处理


Your Code
Your Code
var x = …
update local variables/structures
if (condition(x)) {
…
}
Process

容错性保证

- At-least-once processing
  - May bump into records causing duplicate updates on state

容错性保证

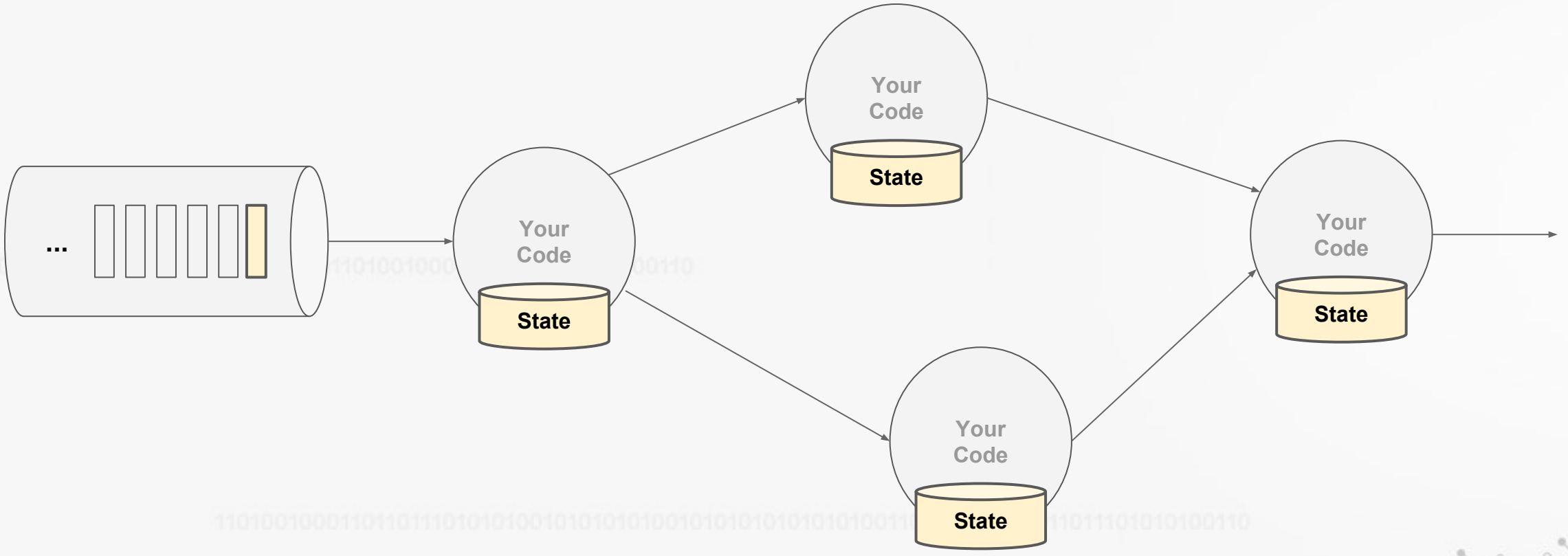

APACHE
STORM™
kafka
A B C D
A B C D
Source &
Sink
Source &
Sink
A B C D
A B C D
Cache
Offset

容错性保证

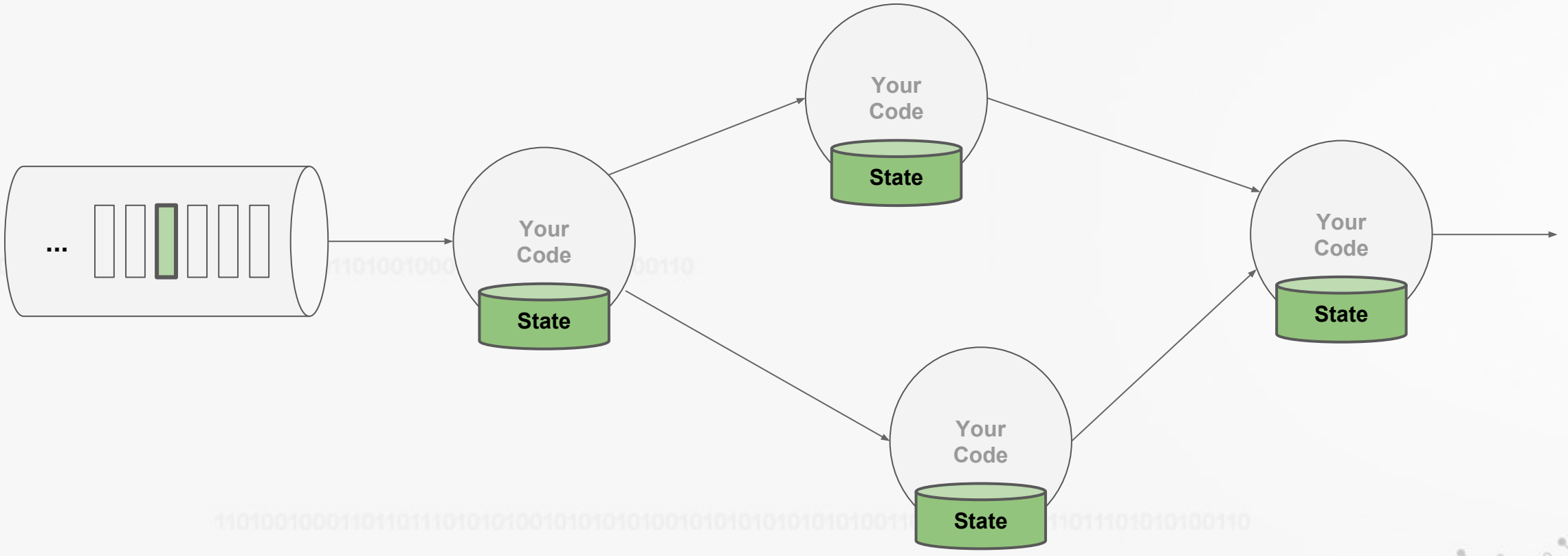
- At-least-once processing
  - May bump into records causing duplicate updates on state

- Exactly-once processing
  - Guarantee that all records affect state exactly-once

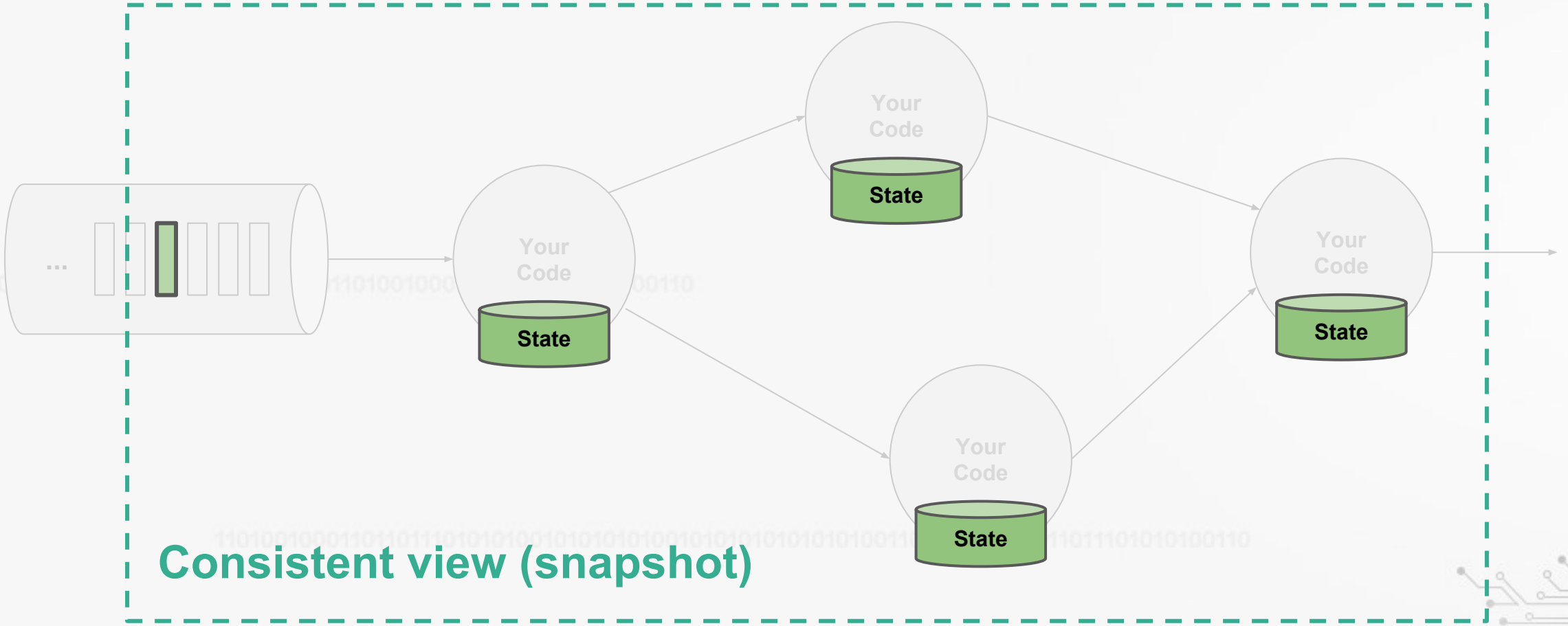
容错性保证


...
Your Code
State
Your Code
State
Your Code
State
Your Code
State

容错性保证


...
Your Code
State
Your Code
State
Your Code
State
Your Code
State

容错性保证

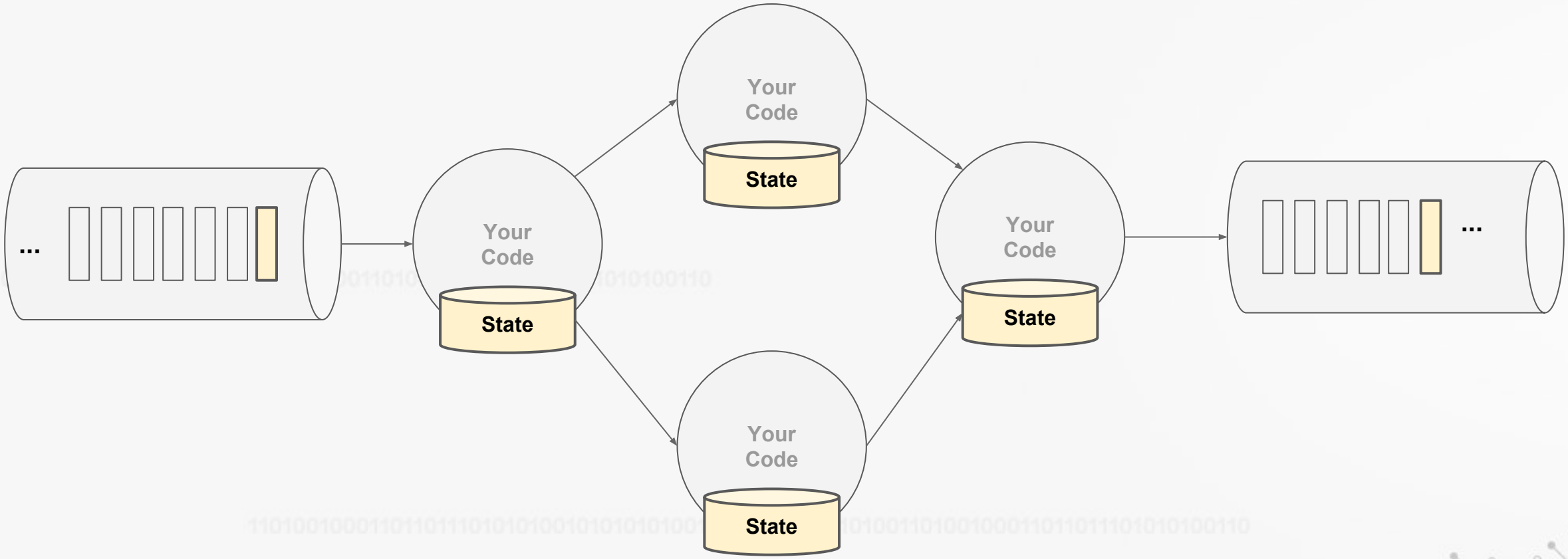

...
Your Code
State
Your Code
State
Your Code
State
Your Code
State
Consistent view (snapshot)

容错性保证

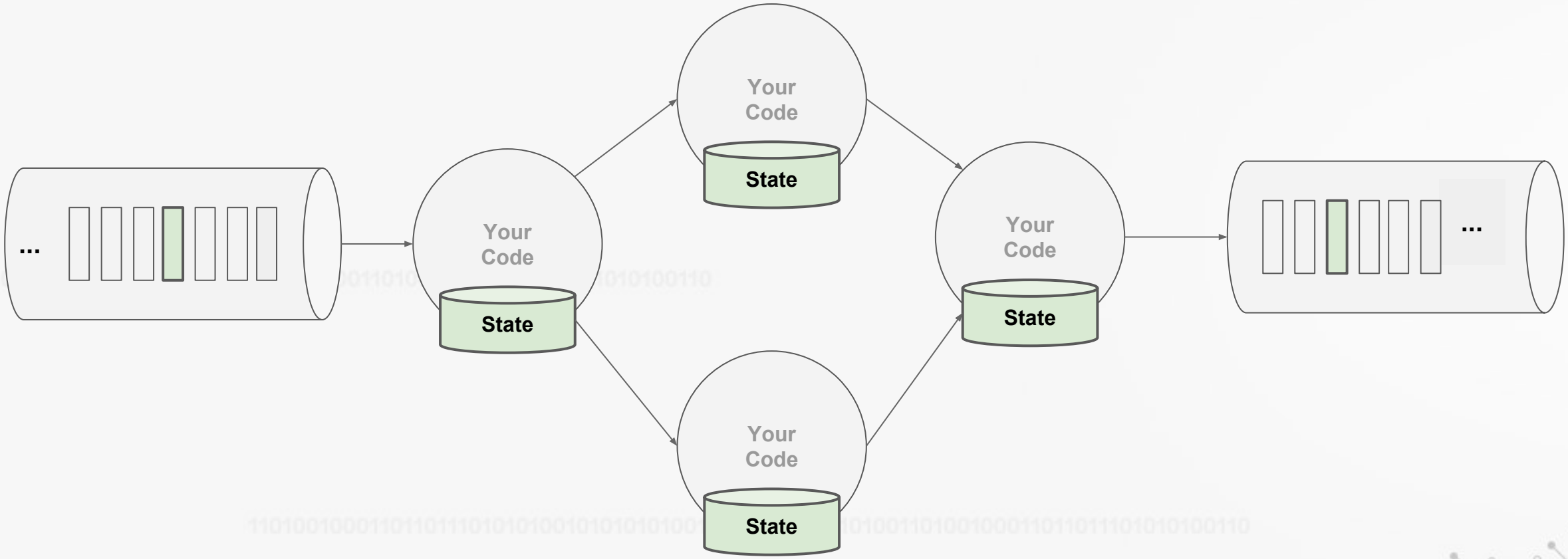
- At-least-once processing
  - May bump into records causing duplicate updates on state
- Exactly-once processing
  - Guarantee that all records affect state exactly-once
- End-to-end exactly-once delivery
  - Guarantee that all records affect internal AND external state exactly-once

Apache Flink

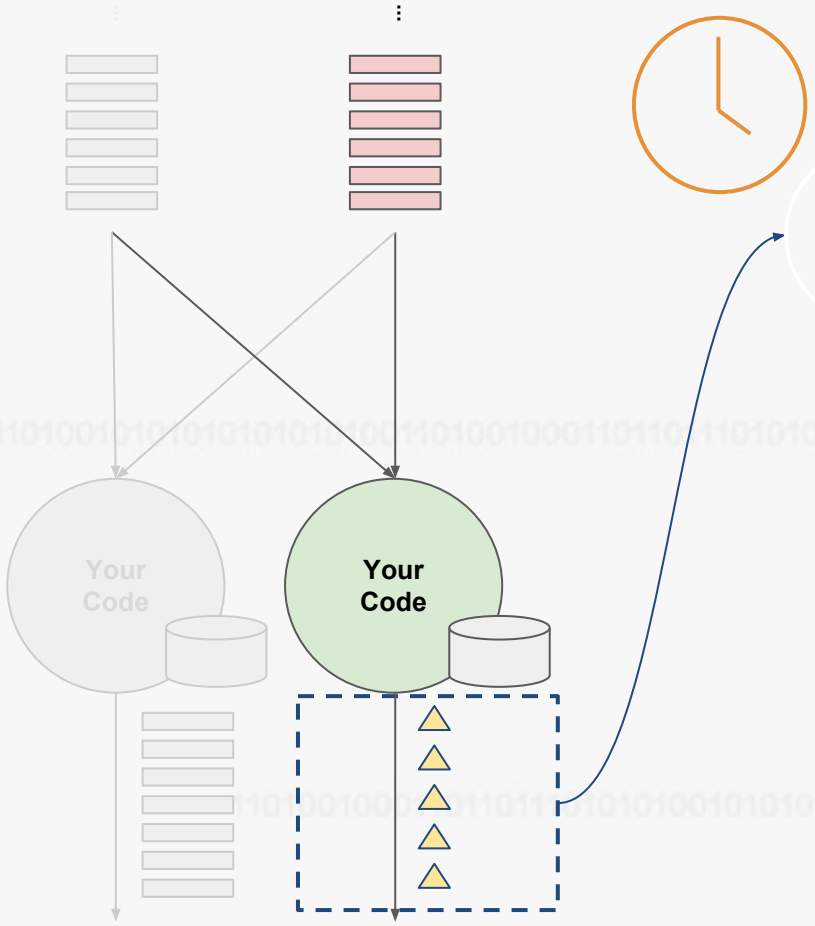
容错性保证



容错性保证


...
Your Code
State
Your Code
State
Your Code
State
Your Code
State
...

Event-Time 处理



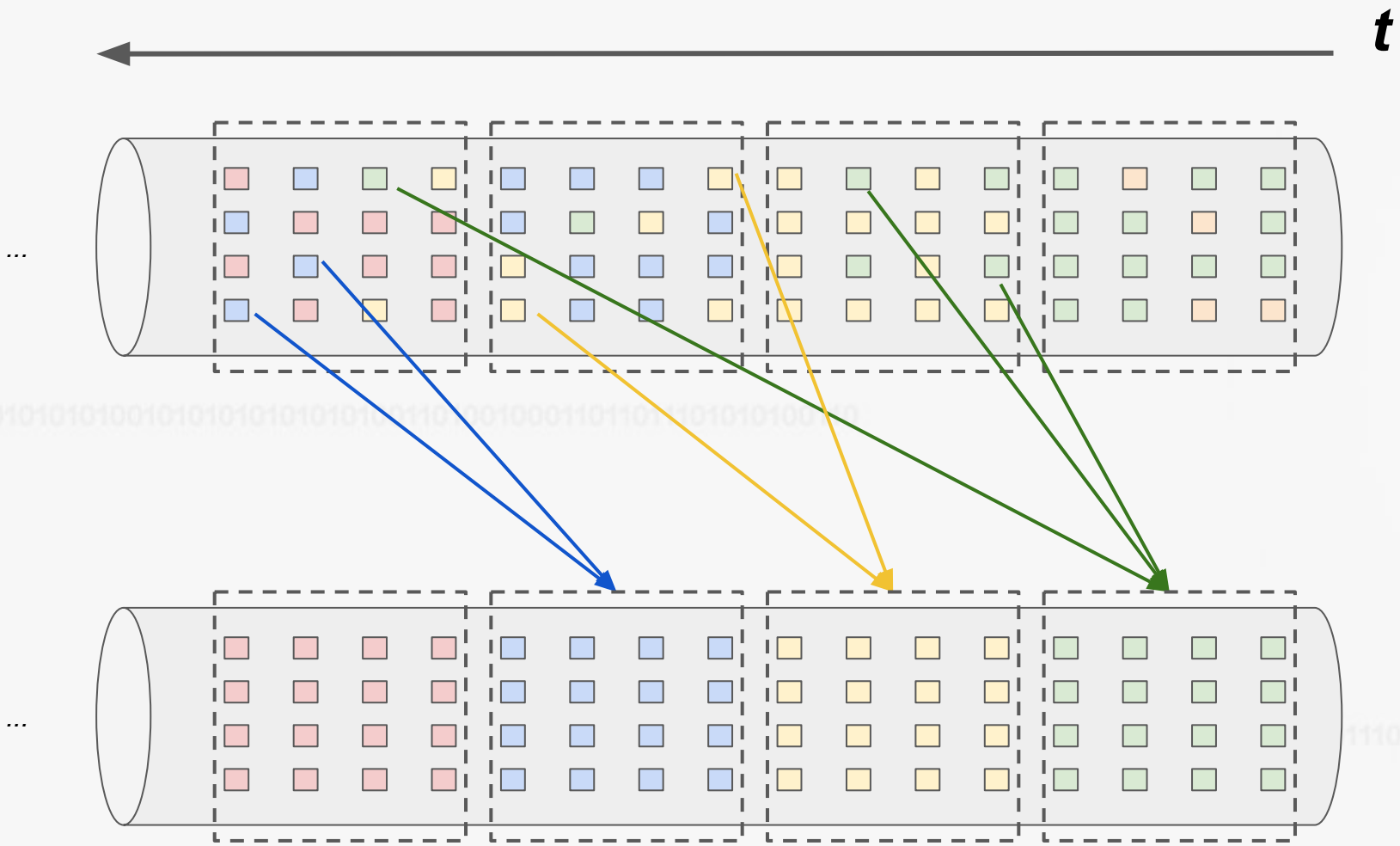
何时才能输出运算结果?

- 是否已经完整收到运算所需的所有数据？
- 这个问题通常都跟时间有关 e.g. 是否已经收到 3 - 4 pm 之间的所有数据?
- **要回答这个问题必须让处理引擎有 event-time 的认知**

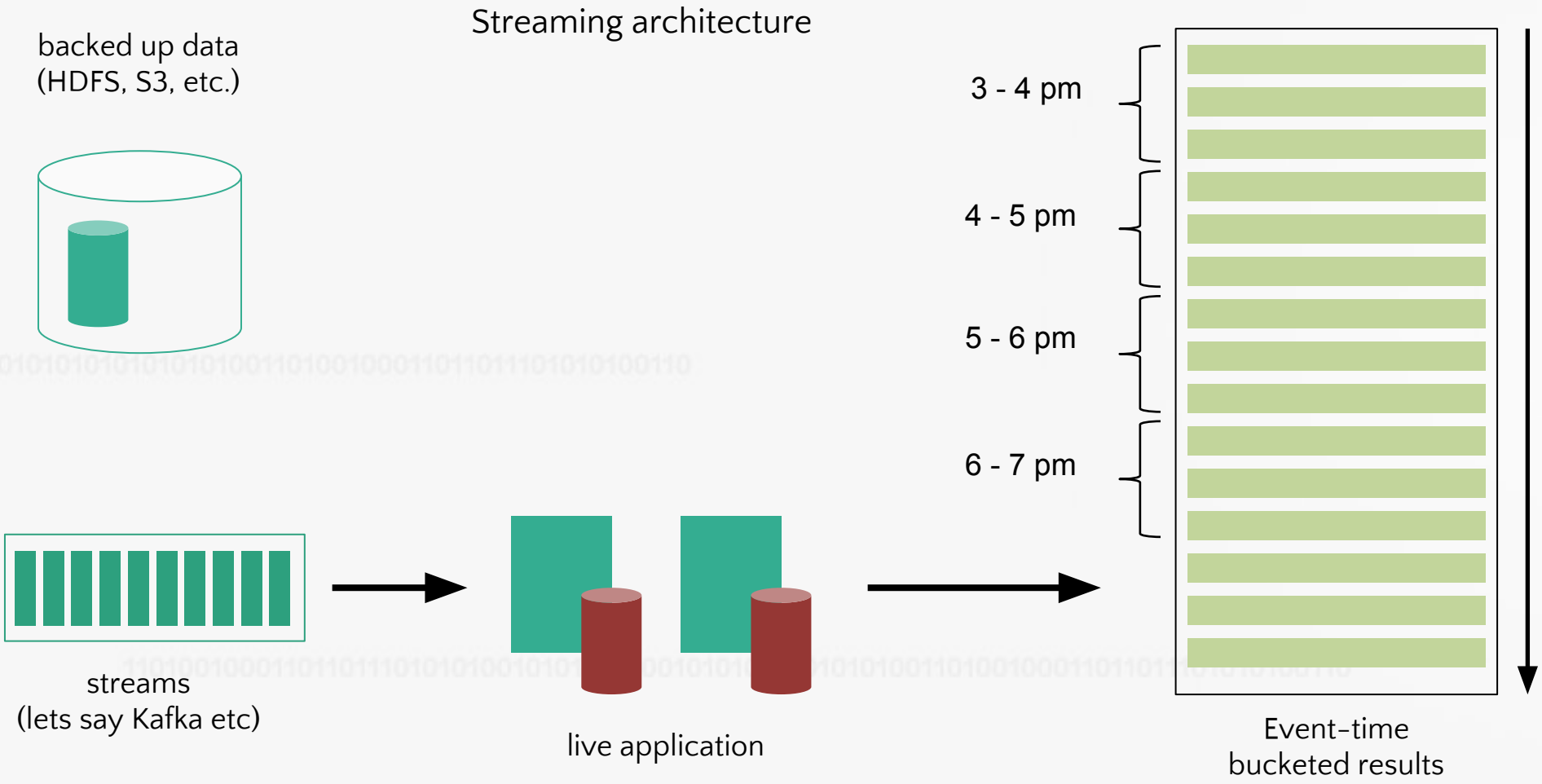
Event-Time 处理

processing time

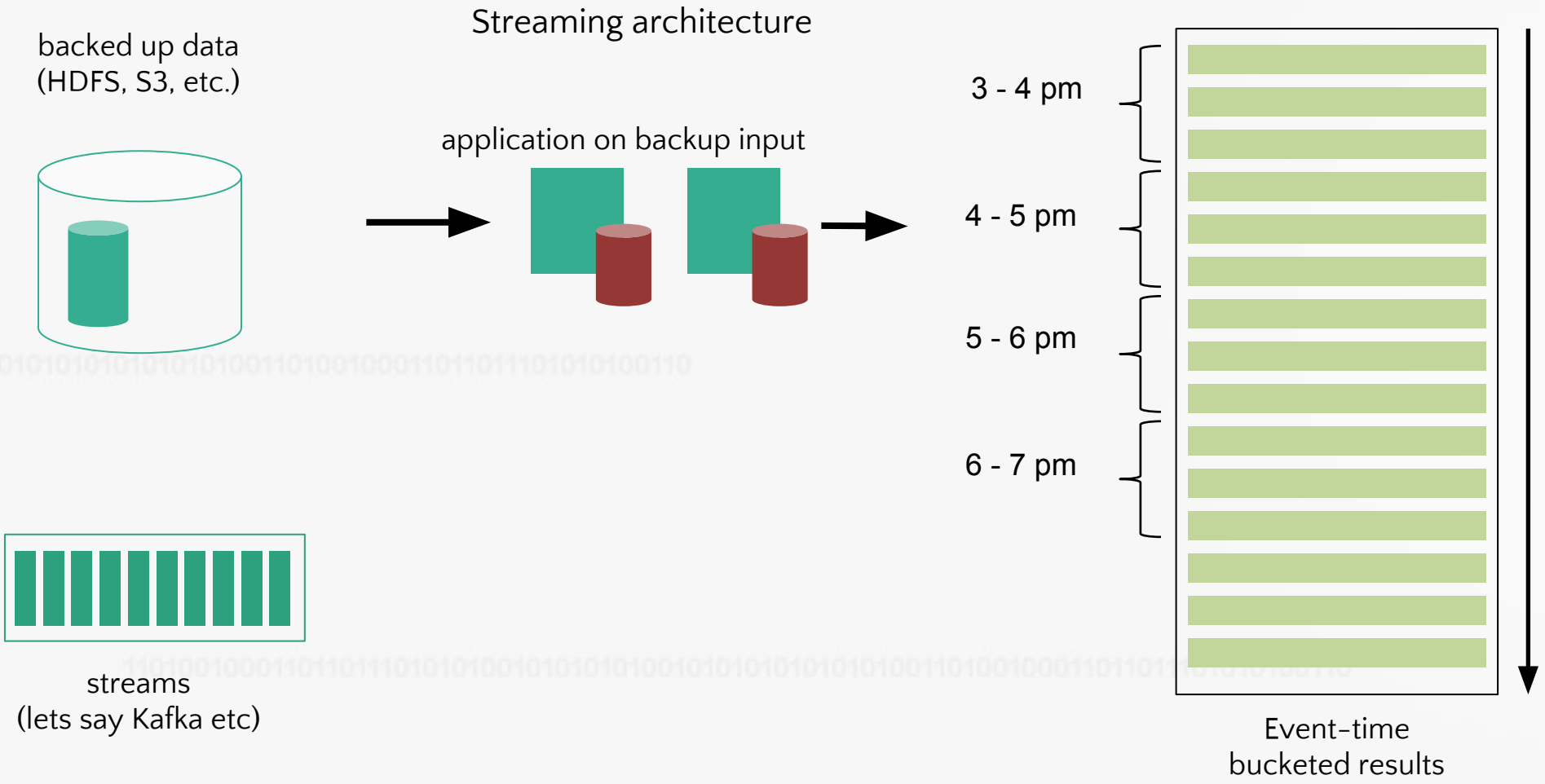
Event-Time 处理



Deterministic replaying



Deterministic replaying



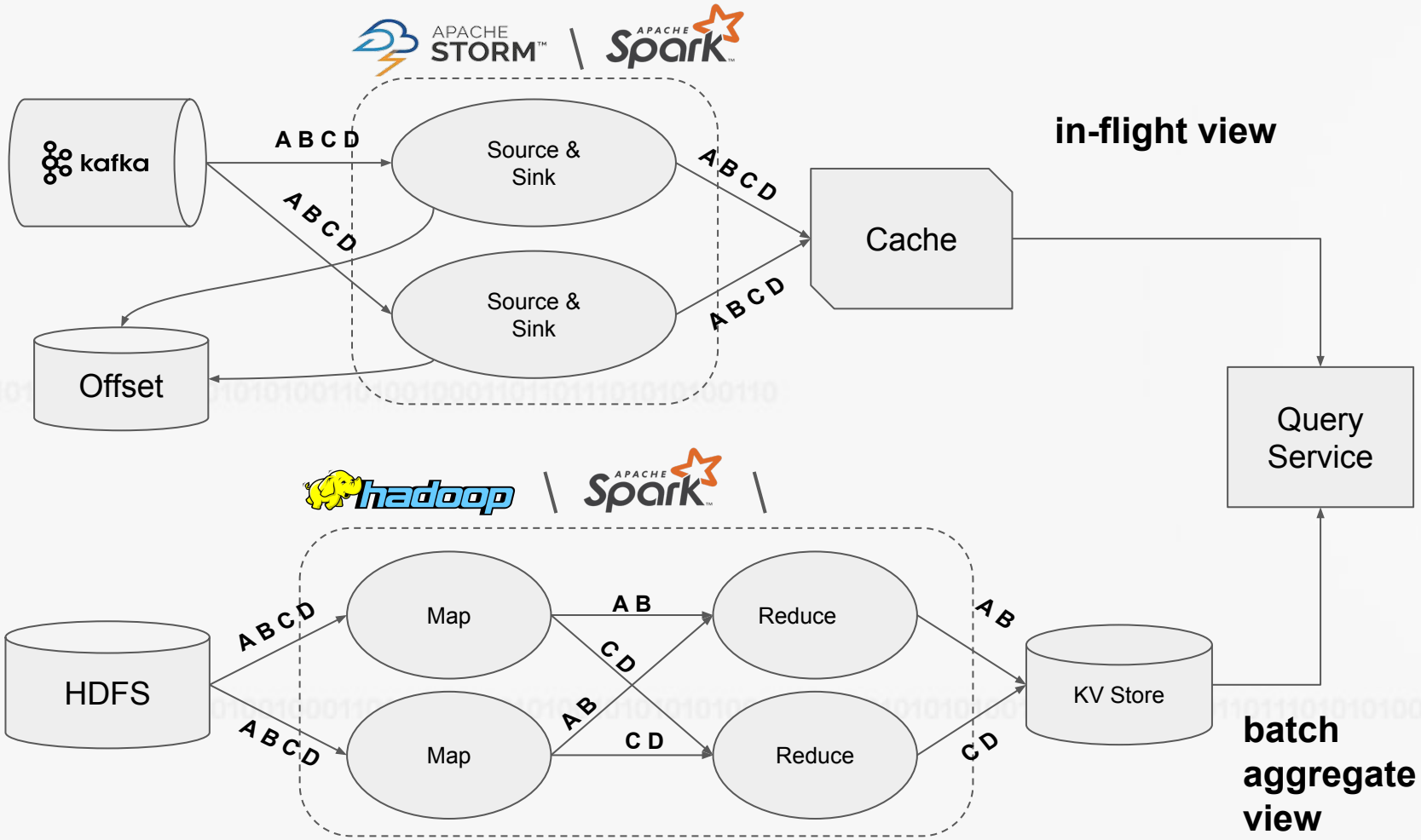
？流式处理重点在于即时性

✖ 流式处理重点在于即时性

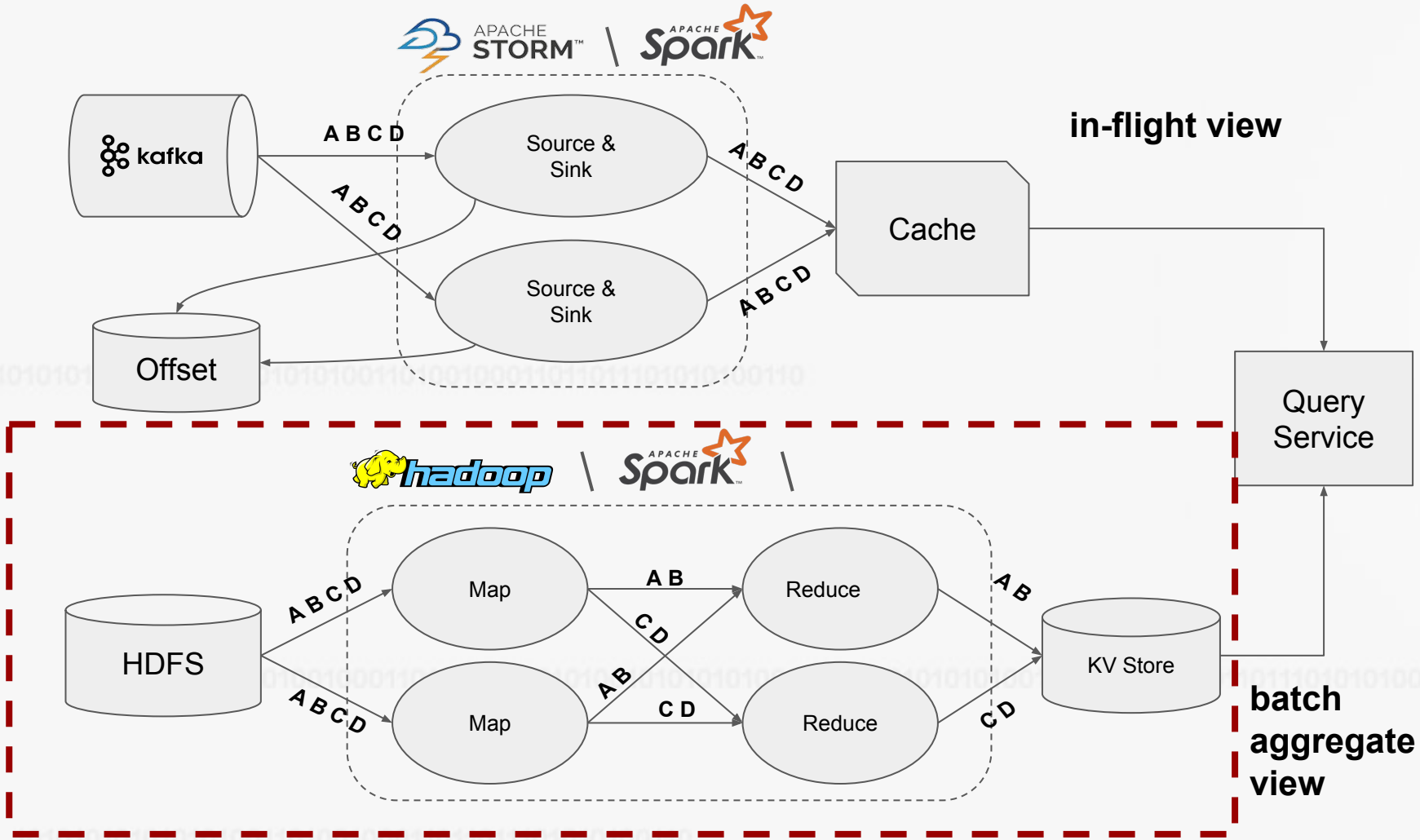
✓ 用于处理 Continuous Data 最自然的运算框架

✓ 可以完全取代批次处理

Lambda 架构怎么看？



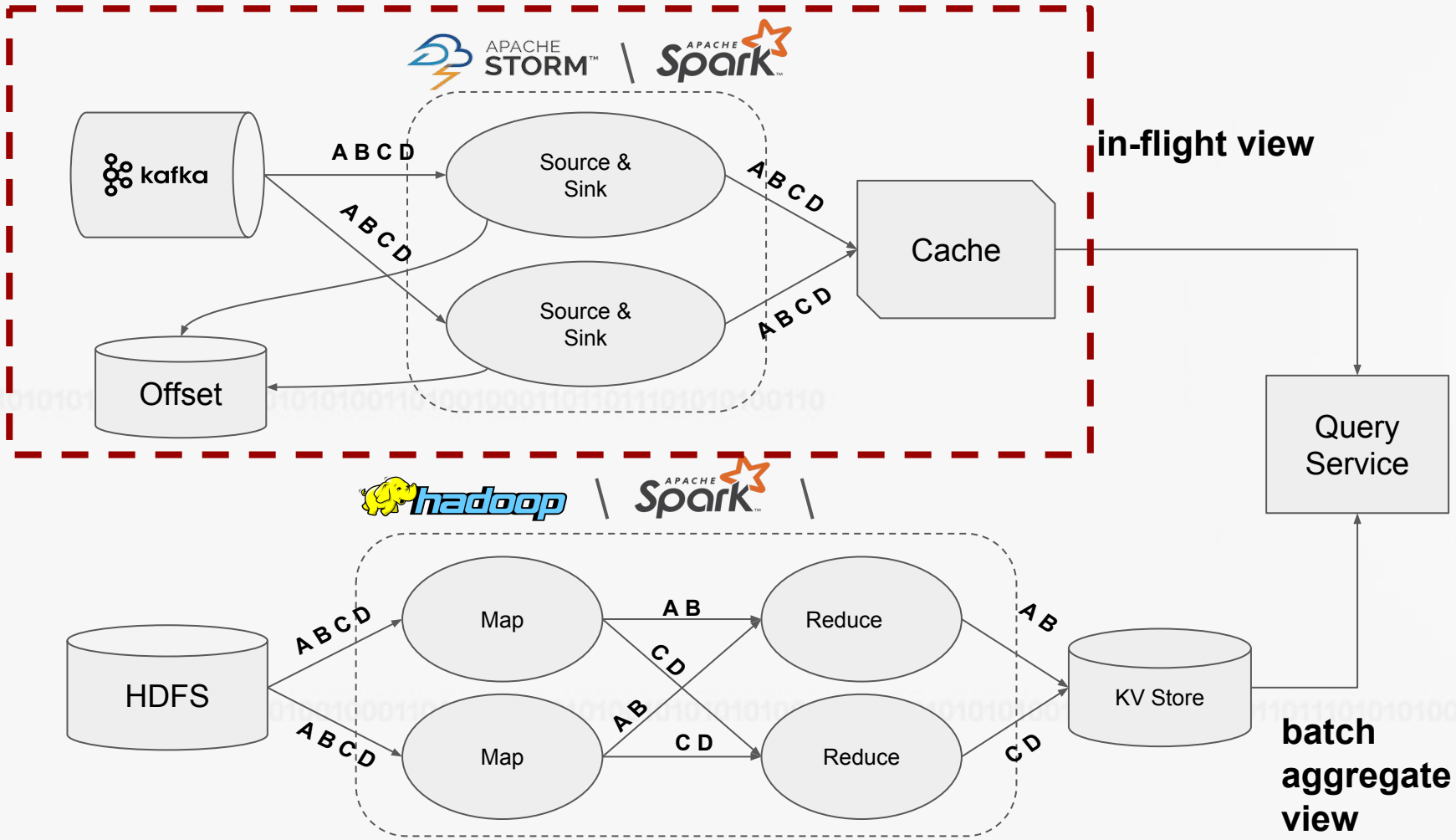
Lambda 架构怎么看？



QPS	Bulk Load
Fault Tolerant?	Yes
Result Correct?	No

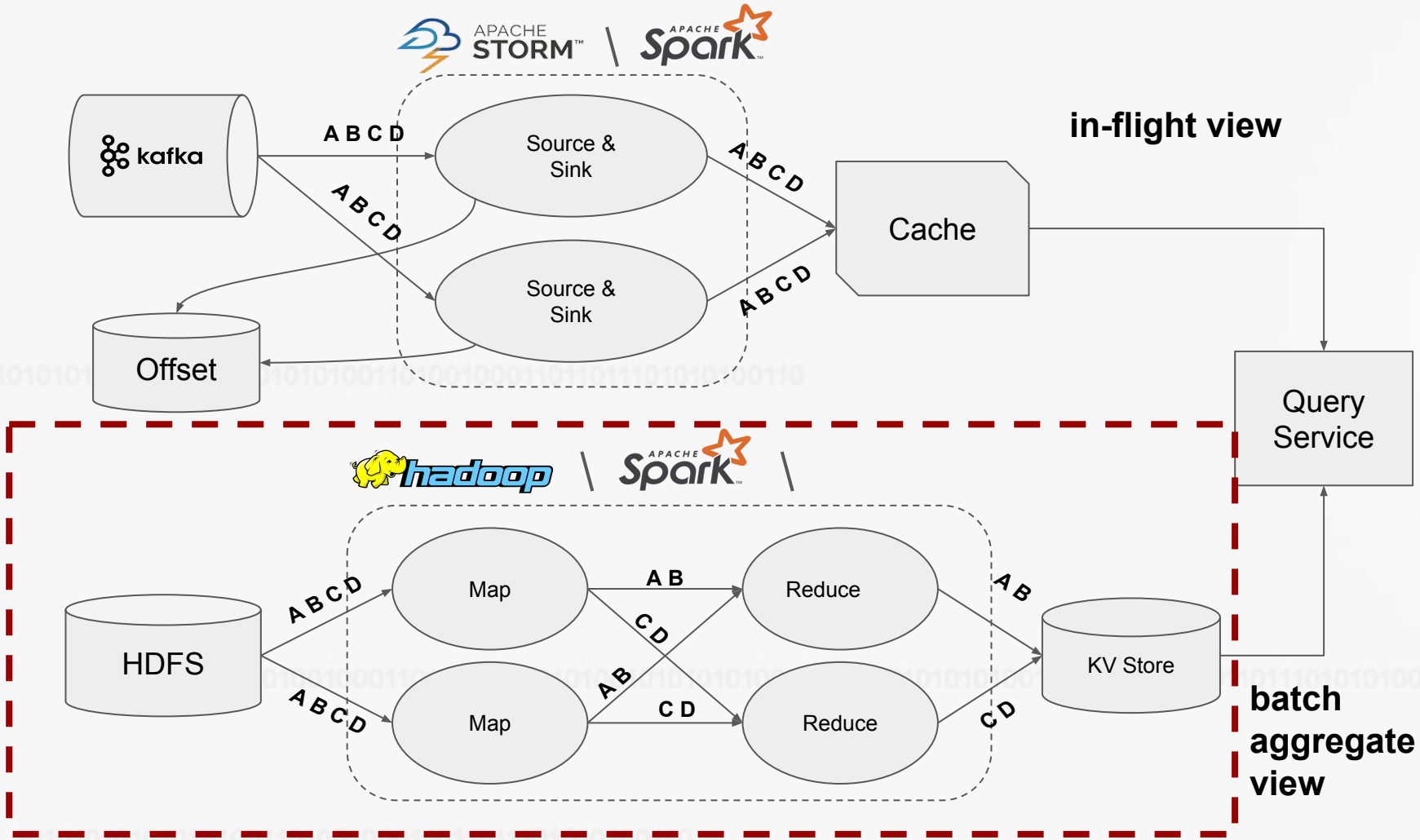
Apache Flink

Lambda 架构怎么看？

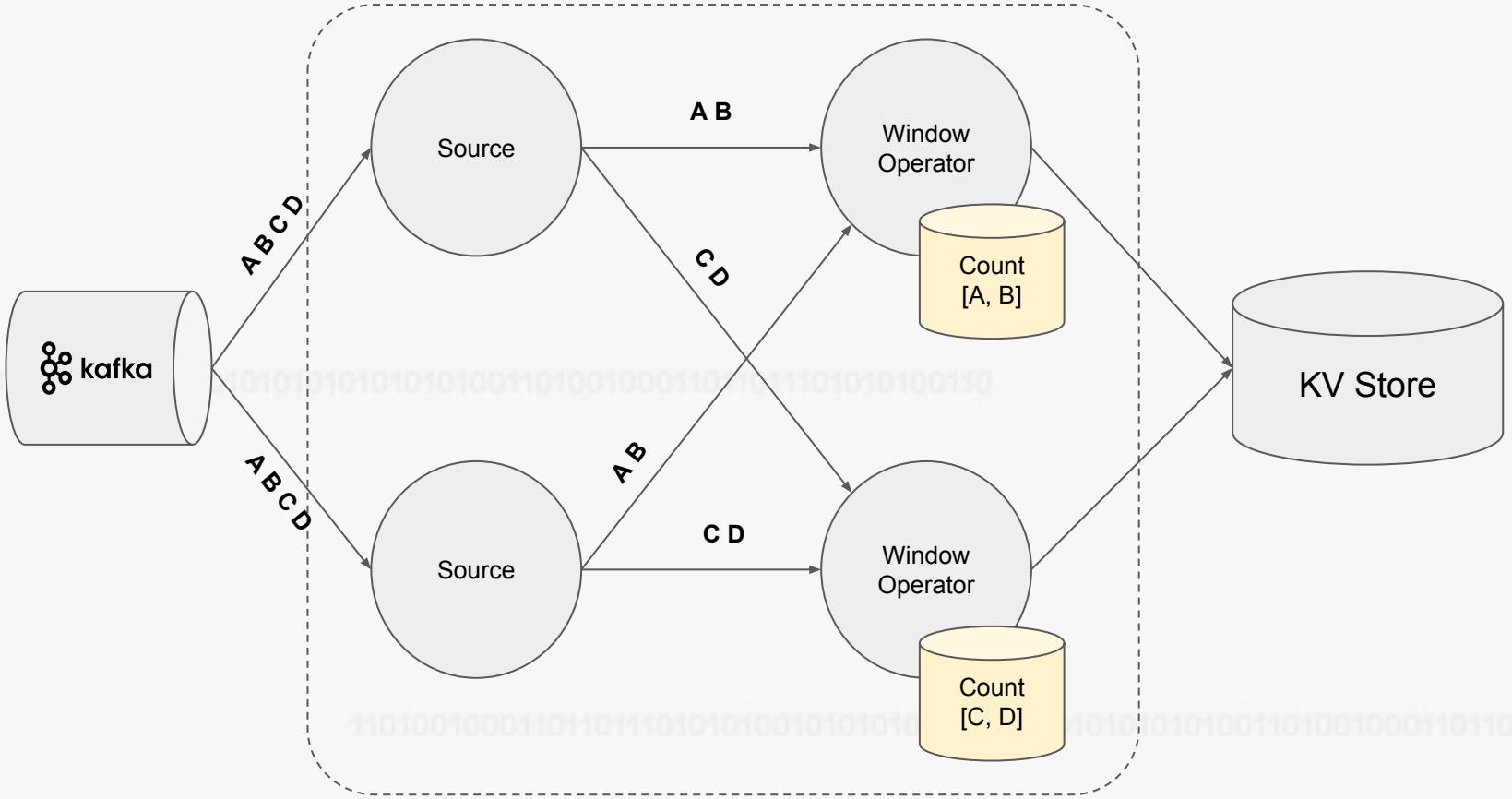


QPS	2 * x / sec
Fault Tolerant?	No
Result Correct?	No

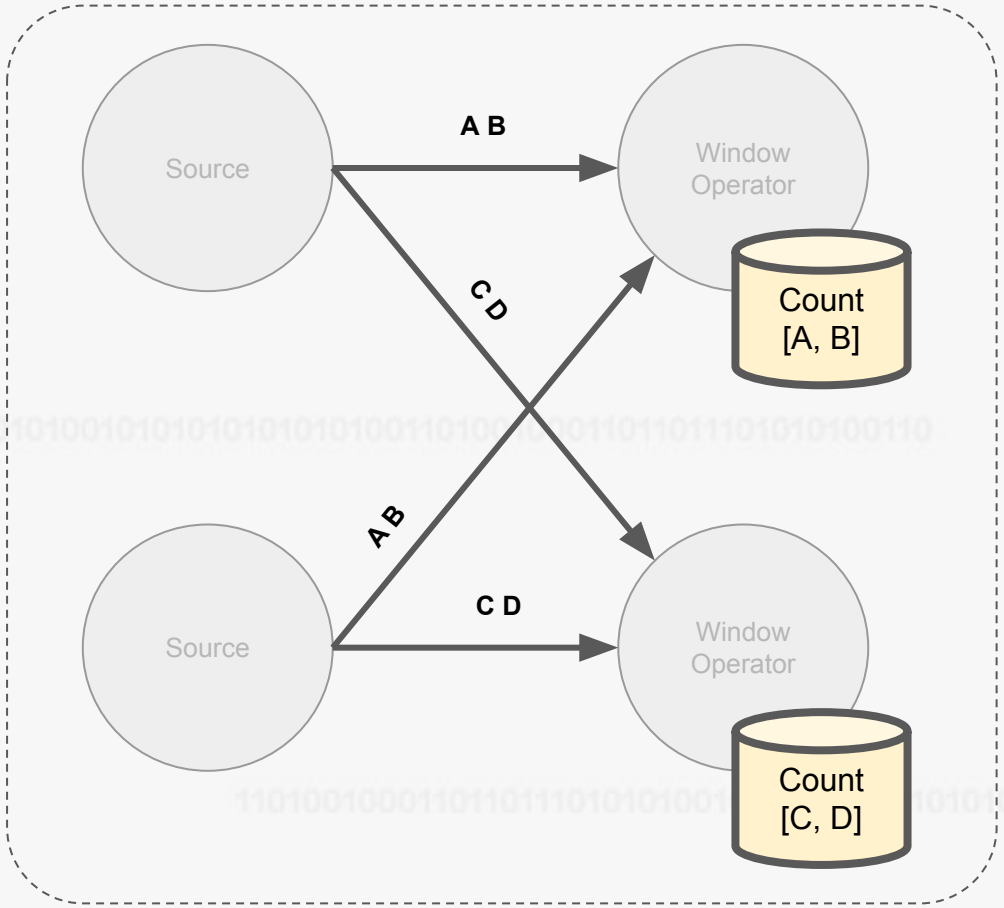
Lambda 架构怎么看？



批流统一数据应用架构

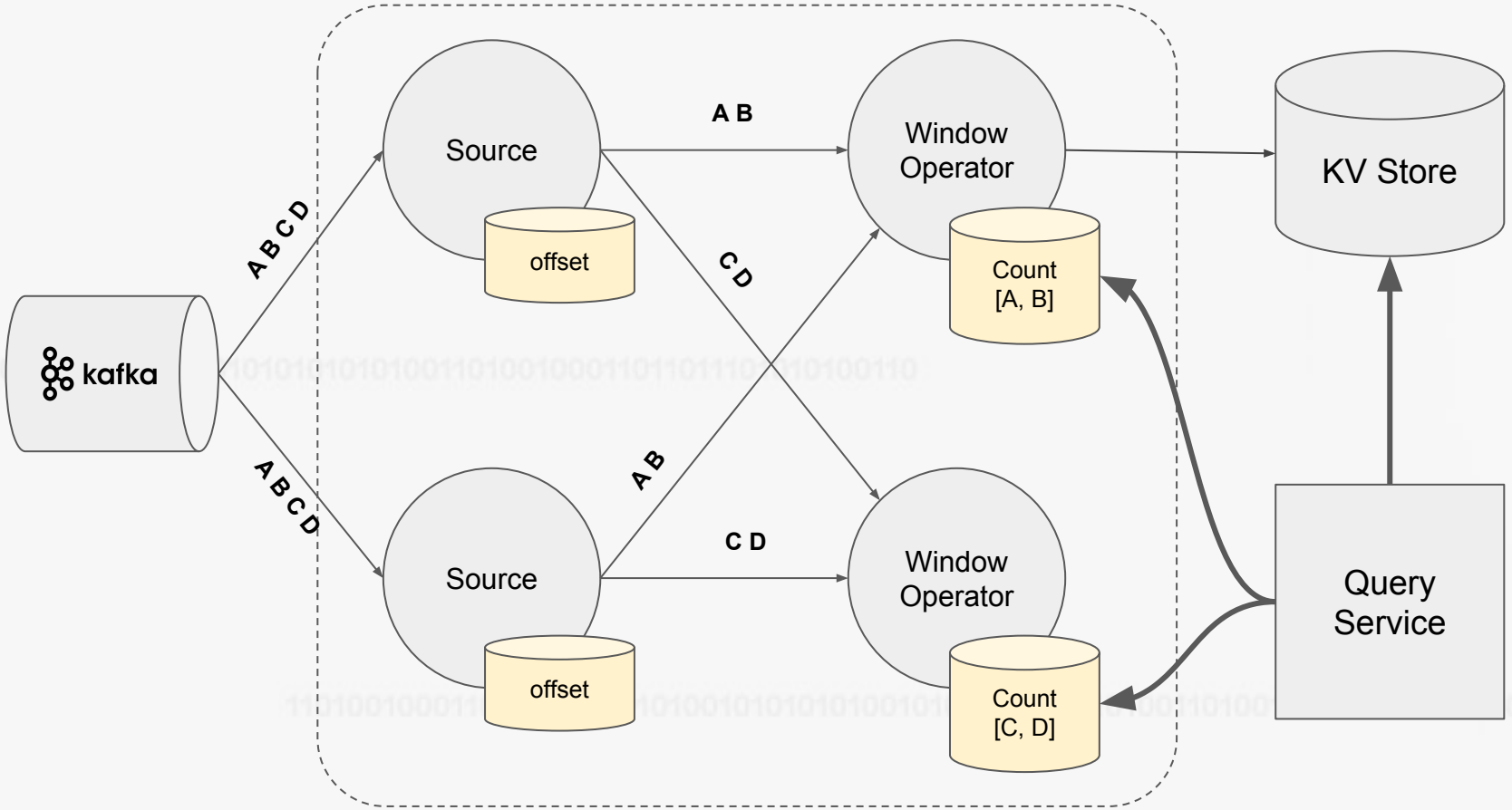

kafka
A B C D
A B C D
Source
Source
A B
C D
A B
C D
Window Operator
Window Operator
Count [A, B]
Count [C, D]
KV Store

批流统一数据应用架构



- **The state in the stream procesor:**
  - Essentially a **key-partitioned**, sharded **KV store**
  - Represents the in-flight aggregations
  - Exactly-once, fault tolerant
  - **Why not just query that?**

批流统一数据应用架构



- Directly query state in real-time
- The **CQRS** (Command Query Request Segmentation) model

✖ 流式处理重点在于即时性

✓ 用于处理 Continuous Data 最自然的运算框架

✓ 可以完全取代批次处理

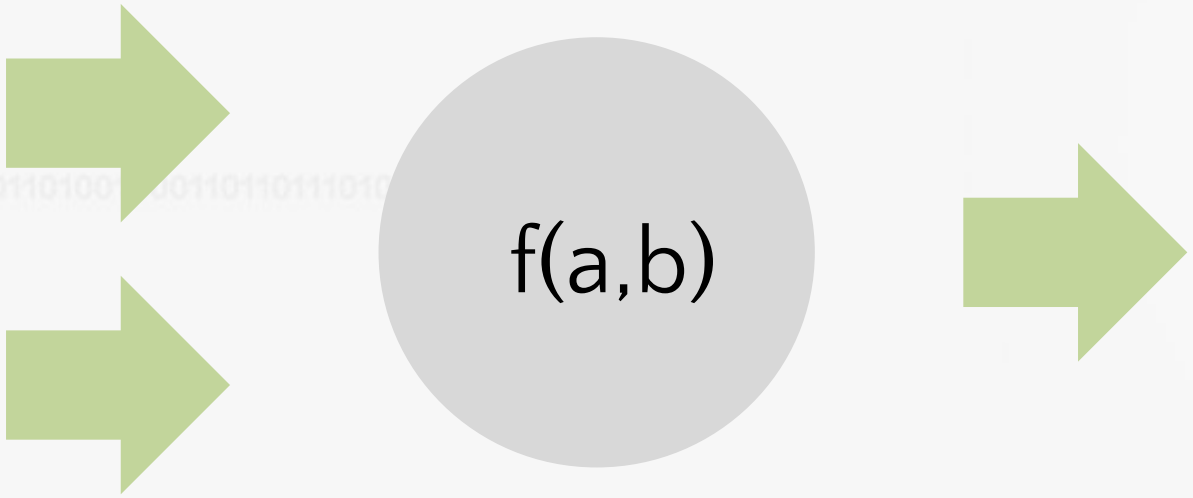
✓ 不再有 Lambda 架构 - 趋向于批流统一架构

× 流式处理重点在于即时性

？ 流式处理只用于数据分析

？流式处理已经发展完毕

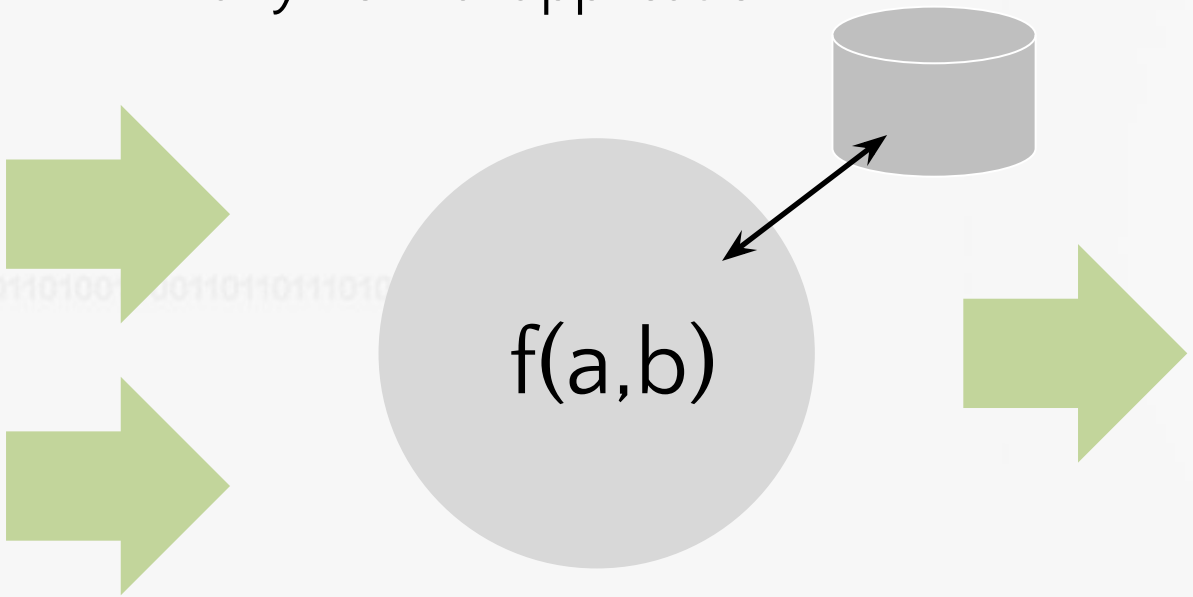
现代流式处理引擎



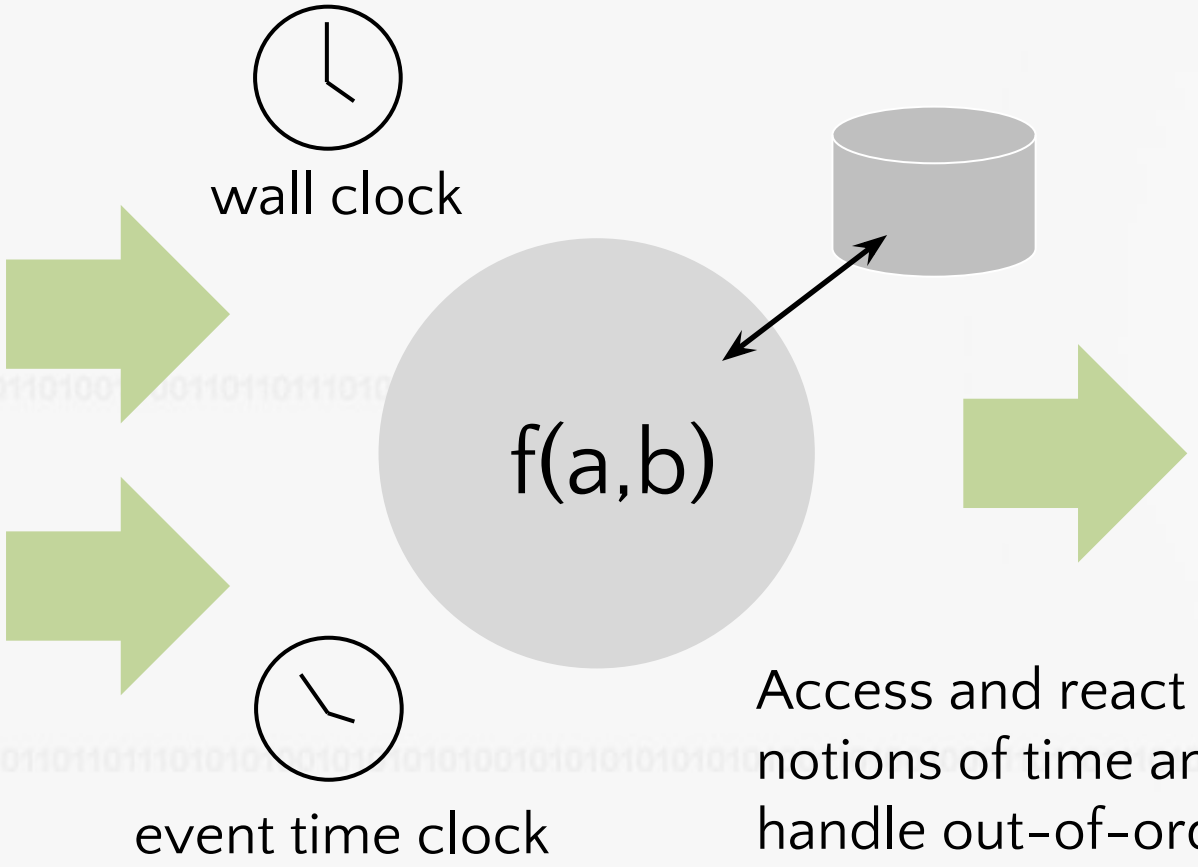
Event-driven function
executed distributedly

现代流式处理引擎

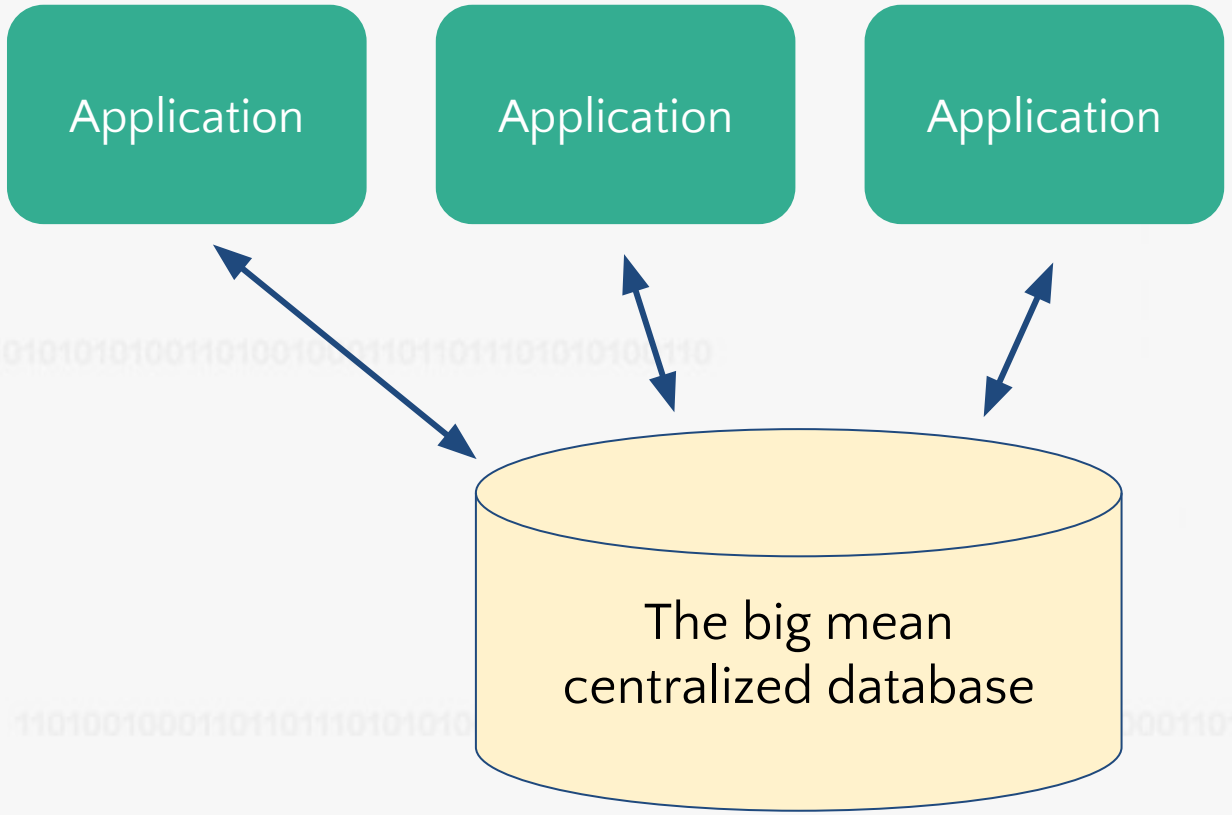
Maintain fault tolerant local state similar to any normal application



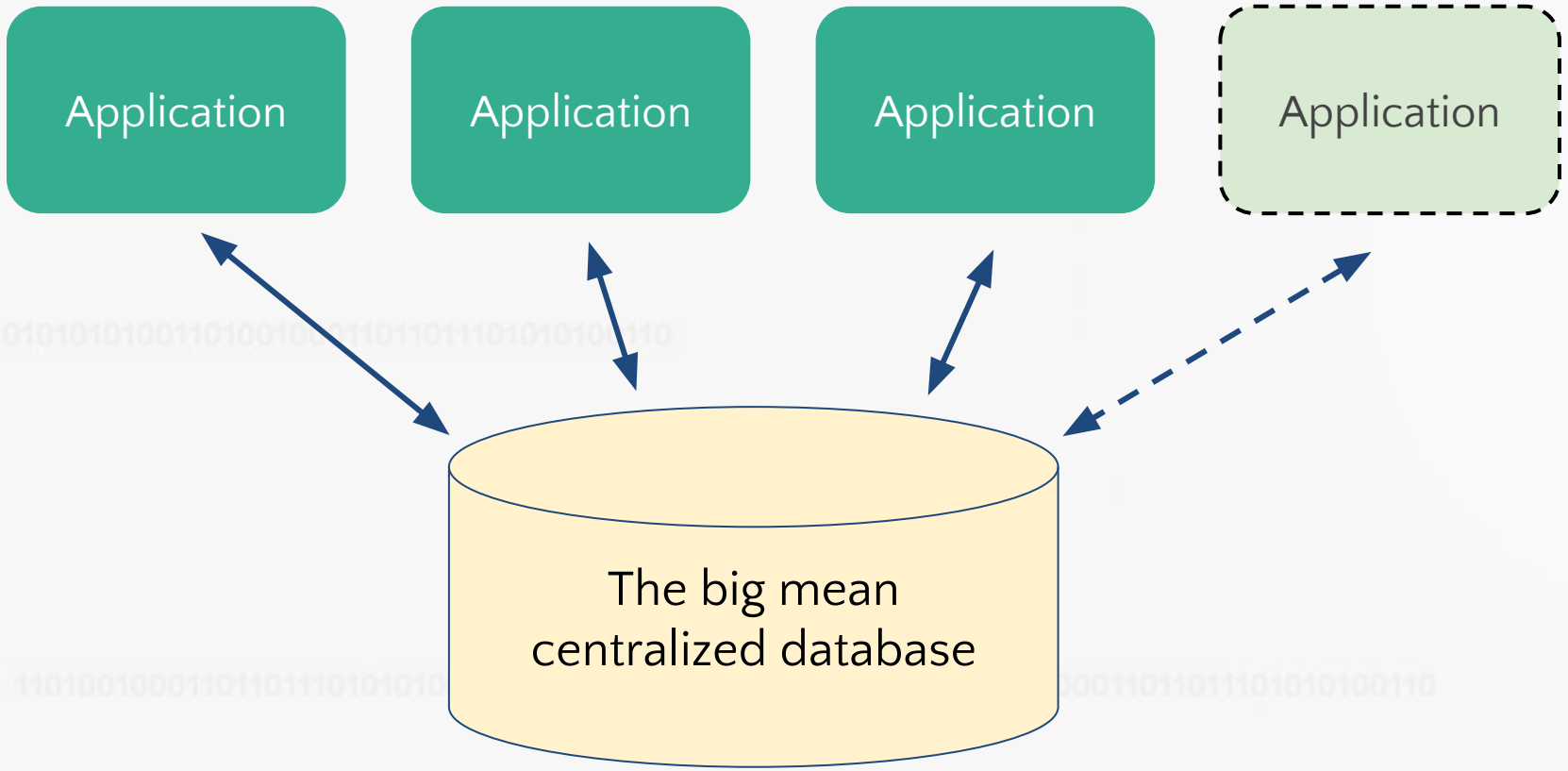
现代流式处理引擎


wall clock
f(a,b)
event time clock
Access and react to
notions of time and progress,
handle out-of-order events

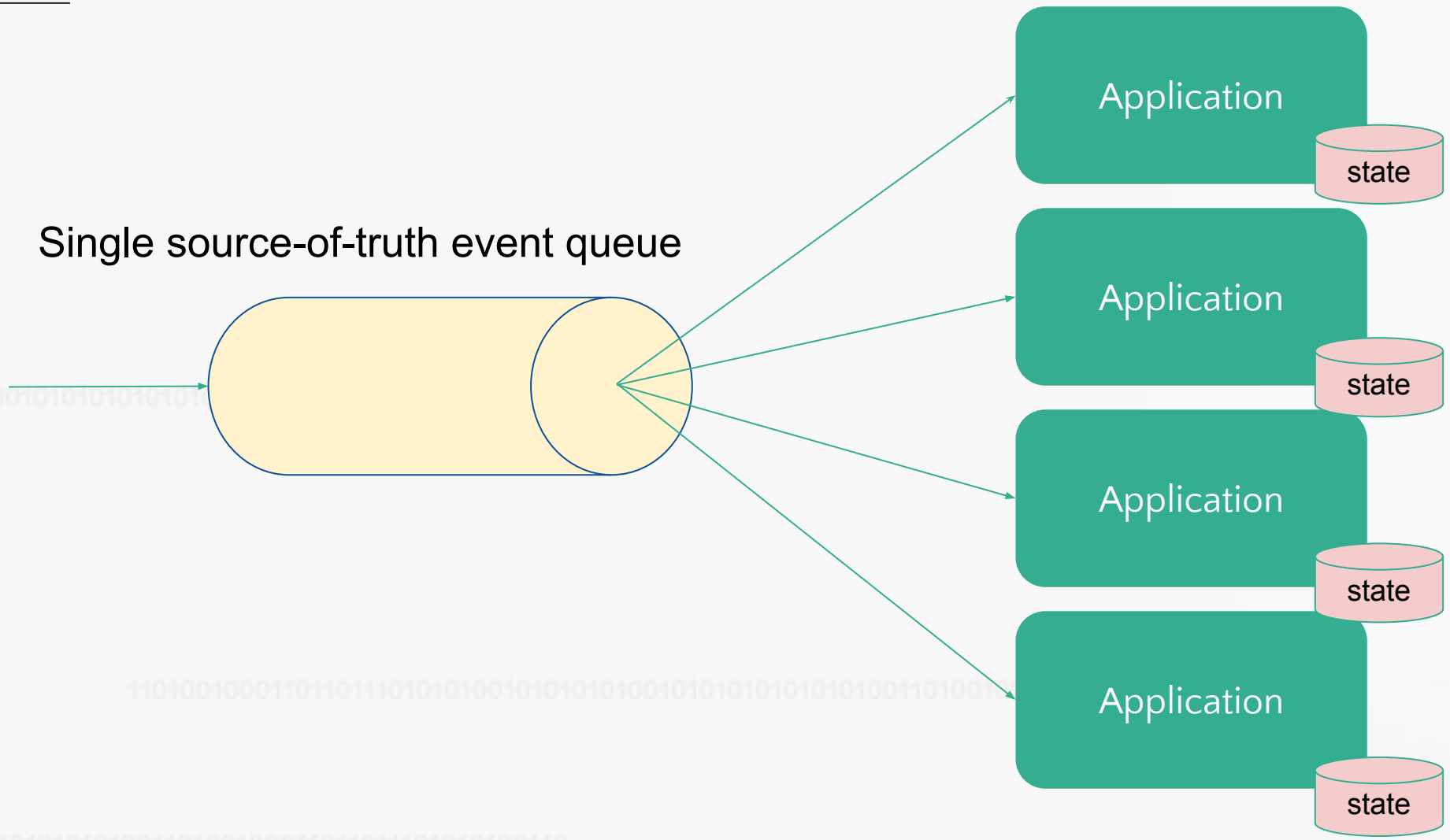
问题:传统应用架构


Application
Application
Application
The big mean
centralized database

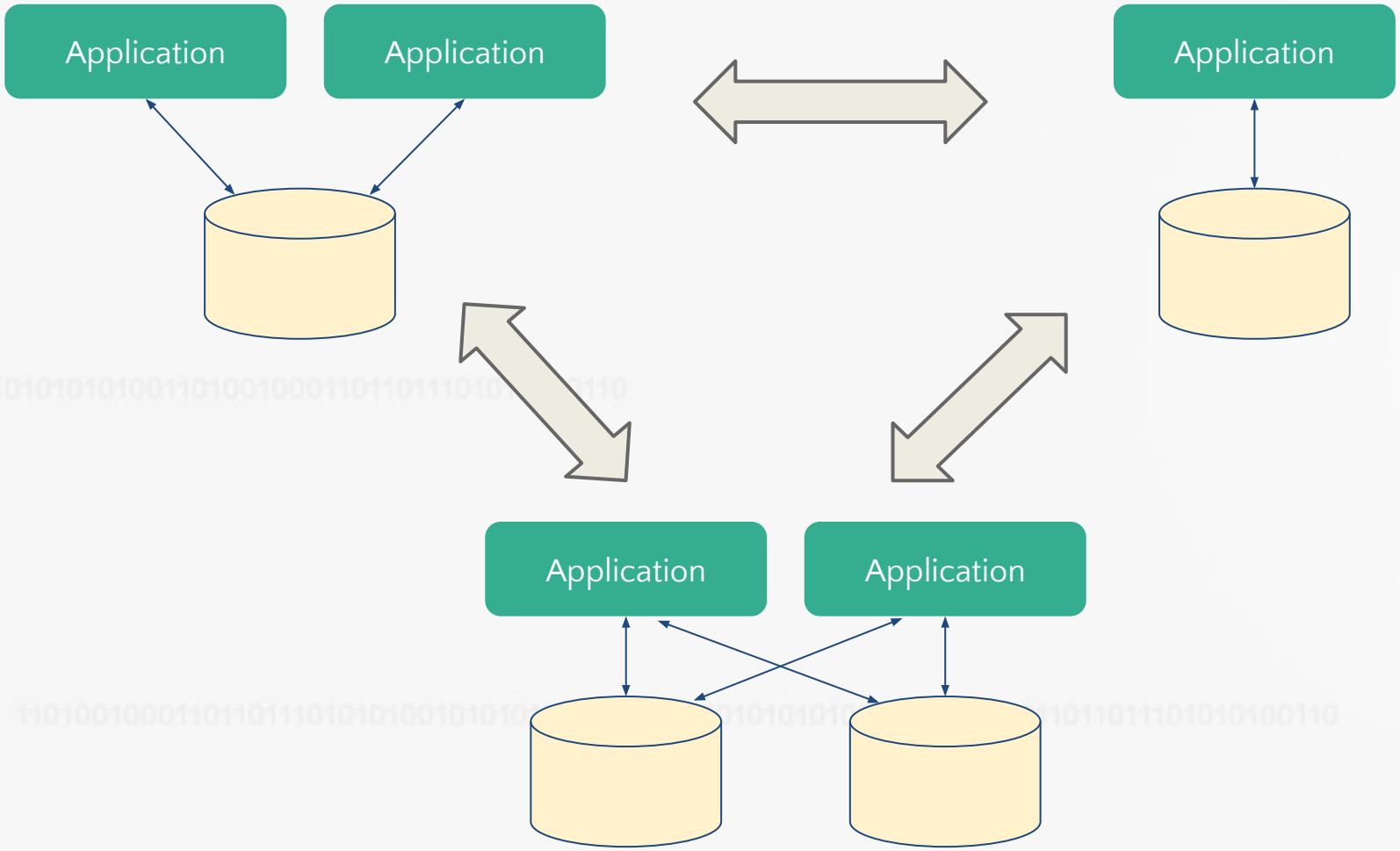
问题:传统应用架构


Application
Application
Application
Application
The big mean
centralized database

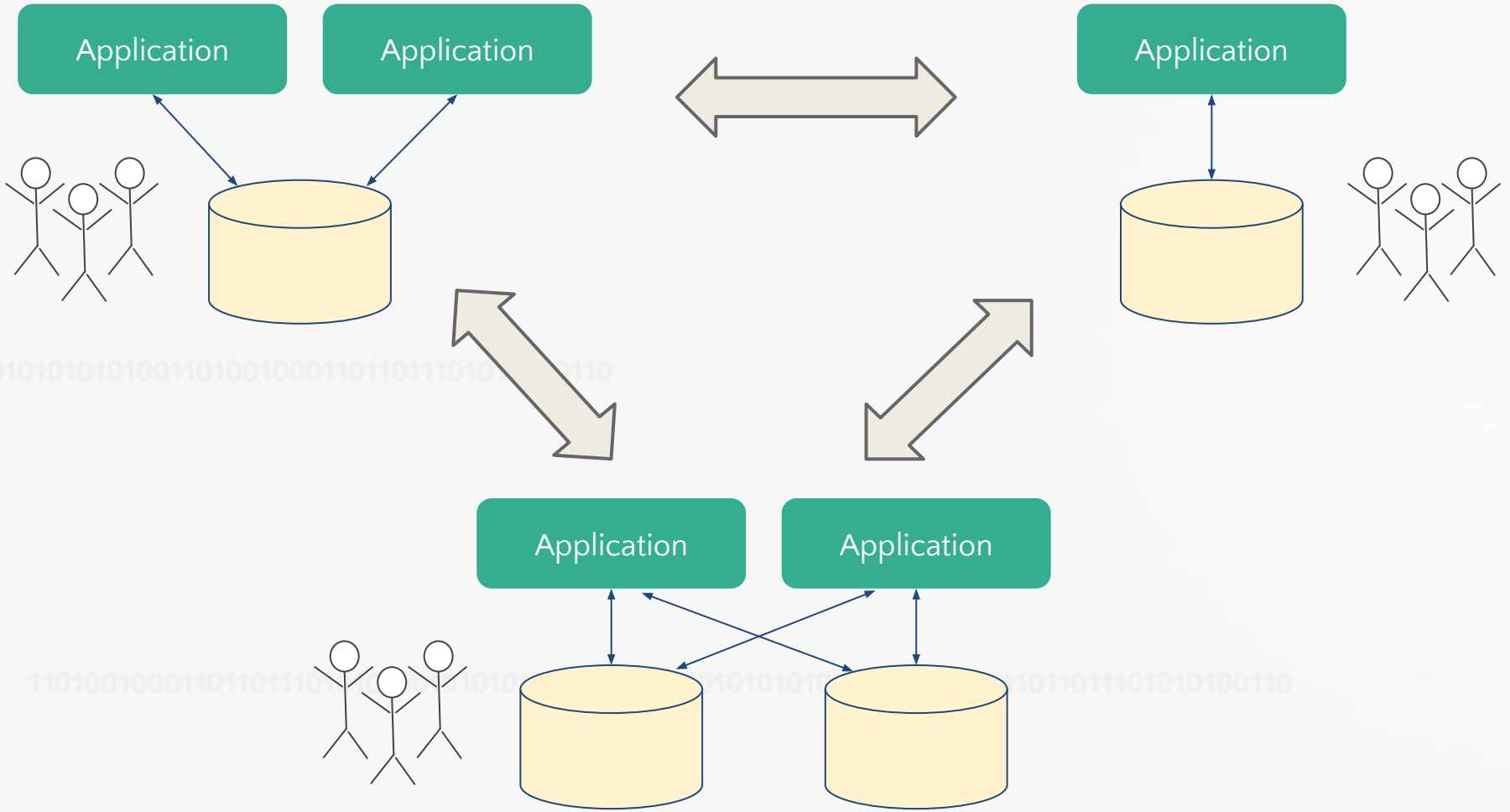
如何解决？


Single source-of-truth event queue
Application
state
Application
state
Application
state
Application
state

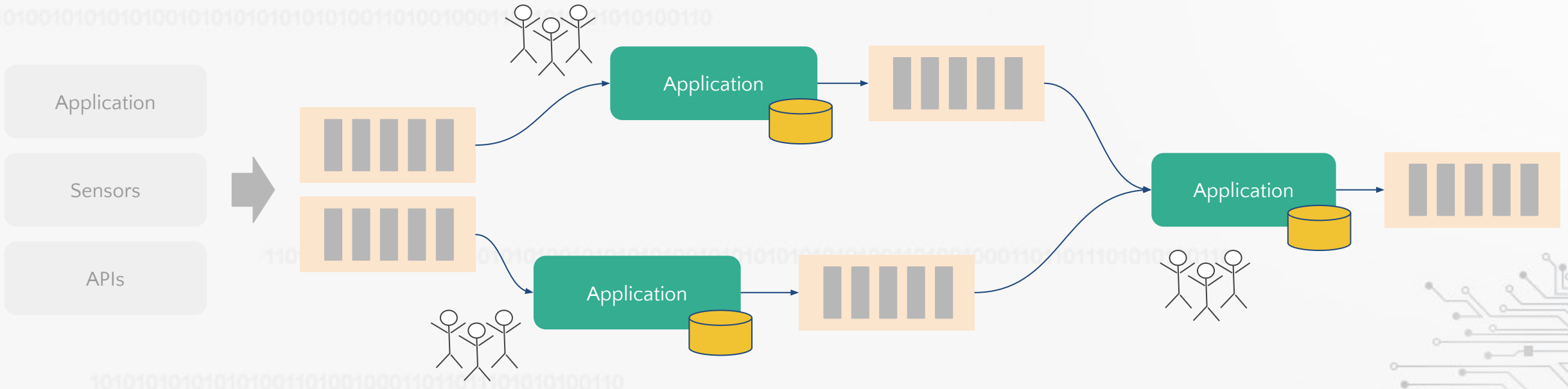
微服务 …


Application
Application
Application
Application
Application

微服务 …


Application
Application
Application
Application
Application

… 与有状态流式处理！


Application
Sensors
APIs
Application
Application
Application

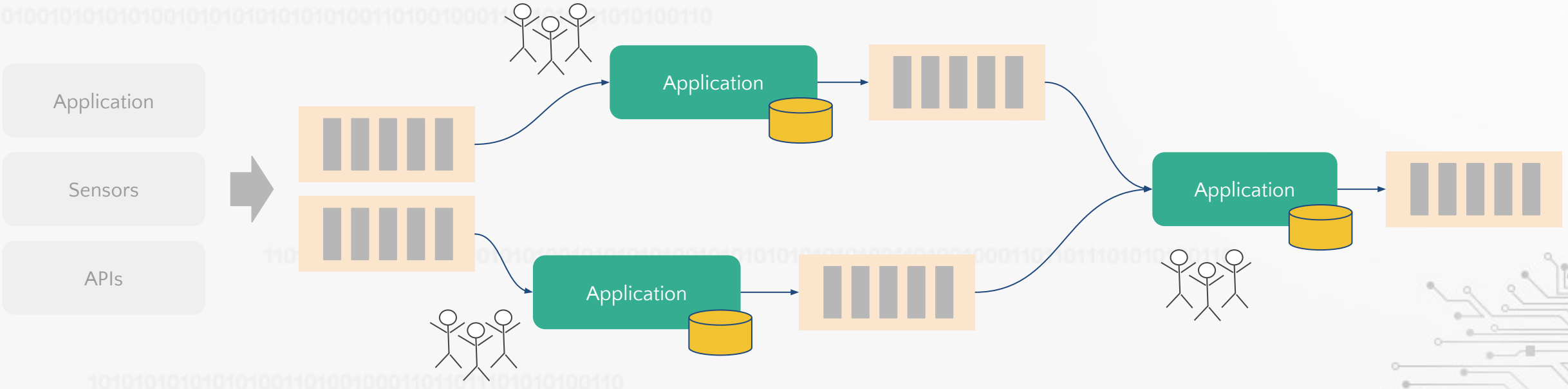
Apache Flink

… 与有状态流式处理！

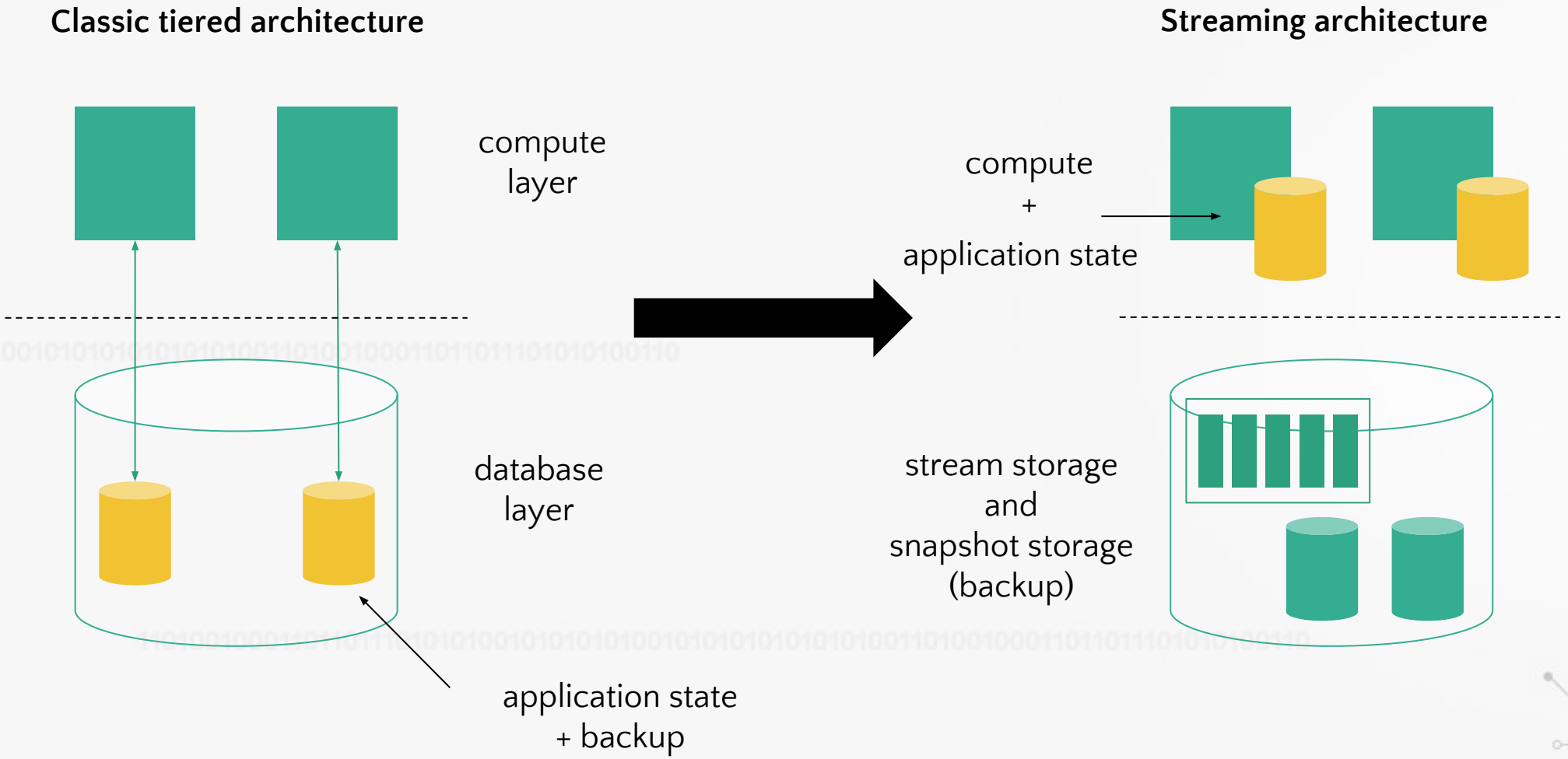
Very simple: **state is just part of the application**

Microservices on steroids!

encourages to build even more specialized apps in a lightweight manner

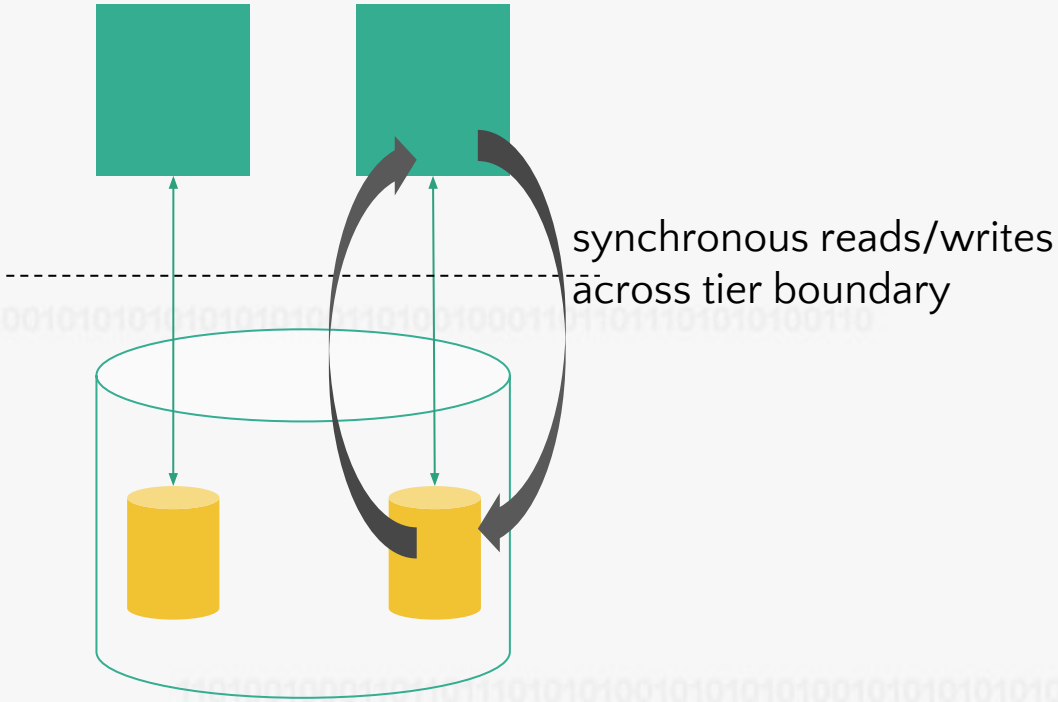


运算、状态、与储存

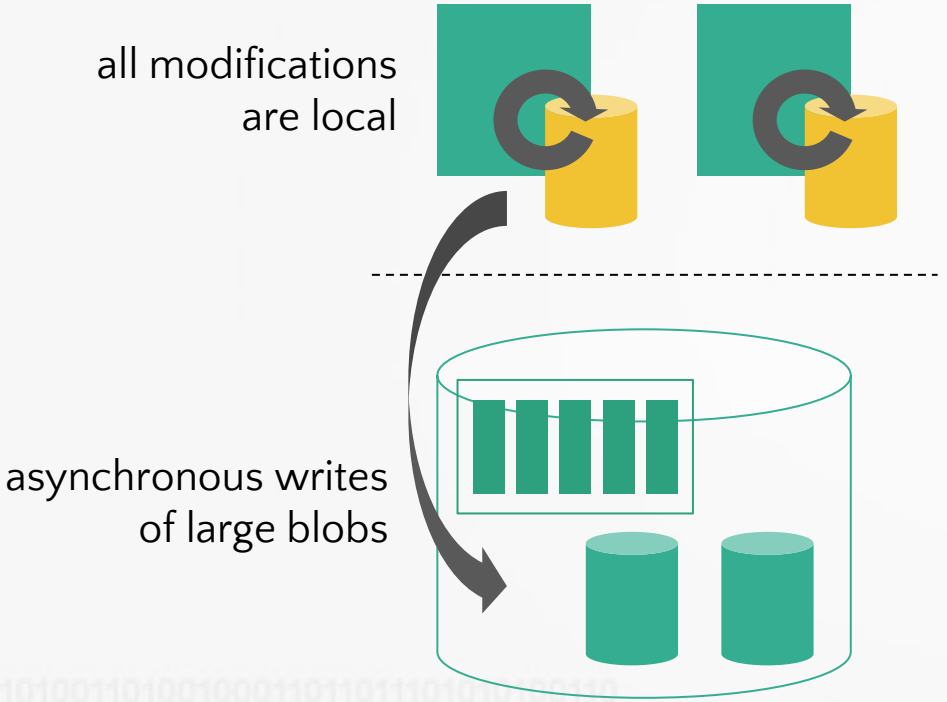

Classic tiered architecture
compute layer
database layer
application state + backup
Streaming architecture
compute + application state
stream storage and snapshot storage (backup)

效能

Classic tiered architecture

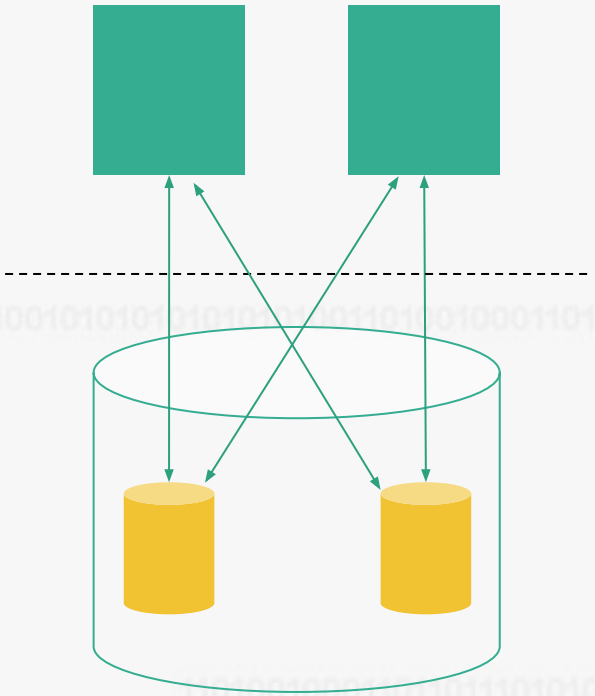


Streaming architecture



一致性

Classic tiered architecture



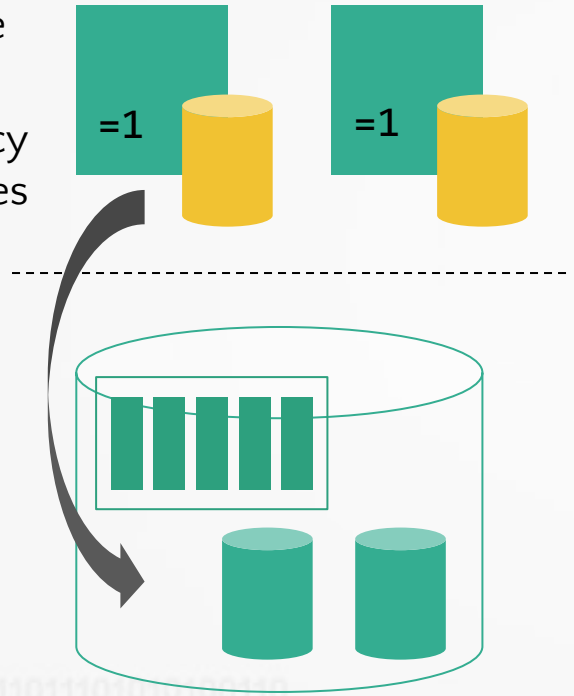
distributed transactions

at scale typically
at-most / at-least once

Streaming architecture

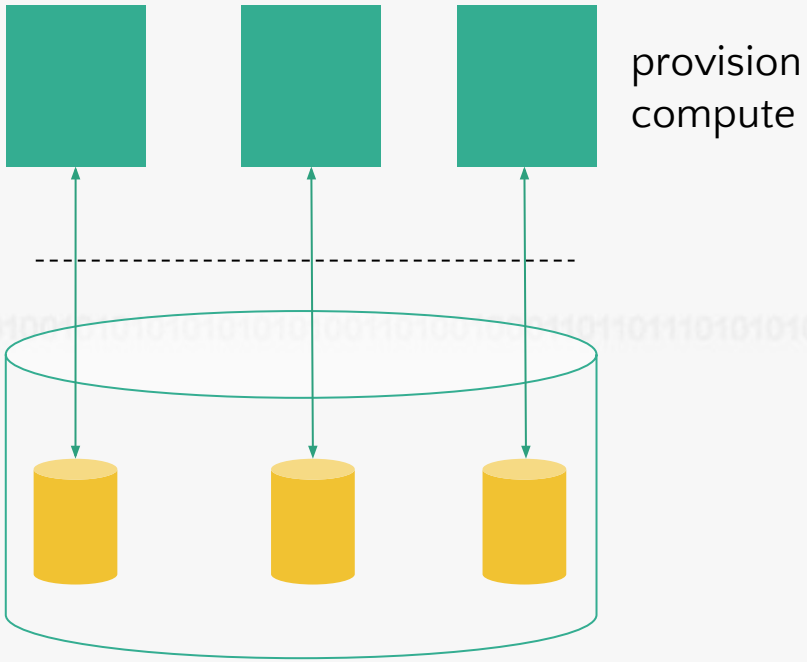
exactly once
per state

snapshot consistency
across states

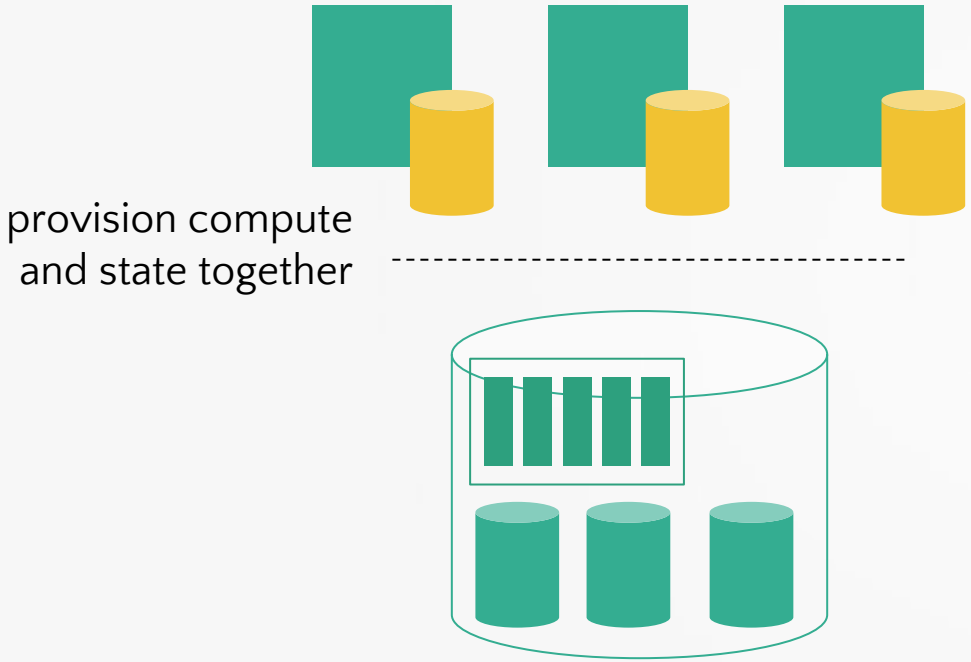


服务延展性

Classic tiered architecture

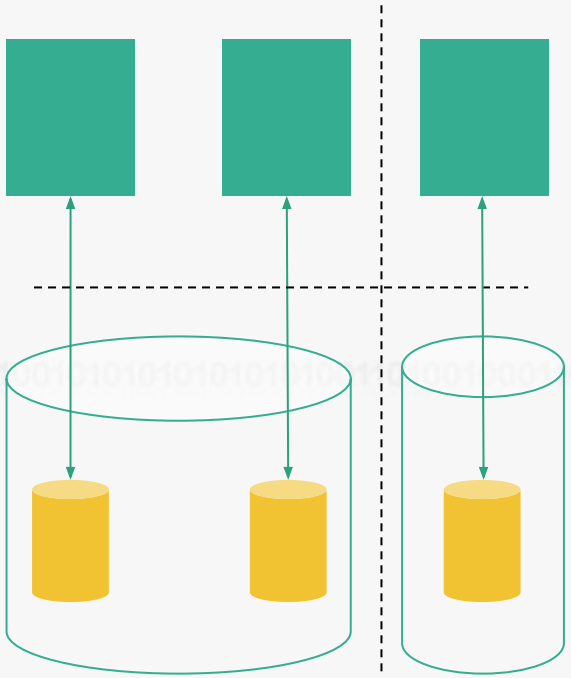


Streaming architecture



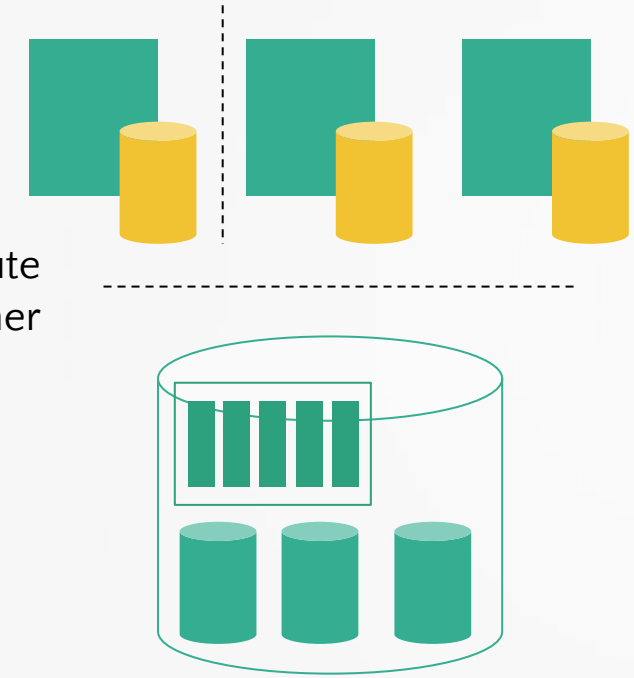
服务扩充性

Classic tiered architecture



provision a new database
(or add capacity to an existing one)

Streaming architecture



provision compute
and state together

simply occupies some
additional backup space

？流式处理只用于数据分析

✕ 流式处理只用于数据分析

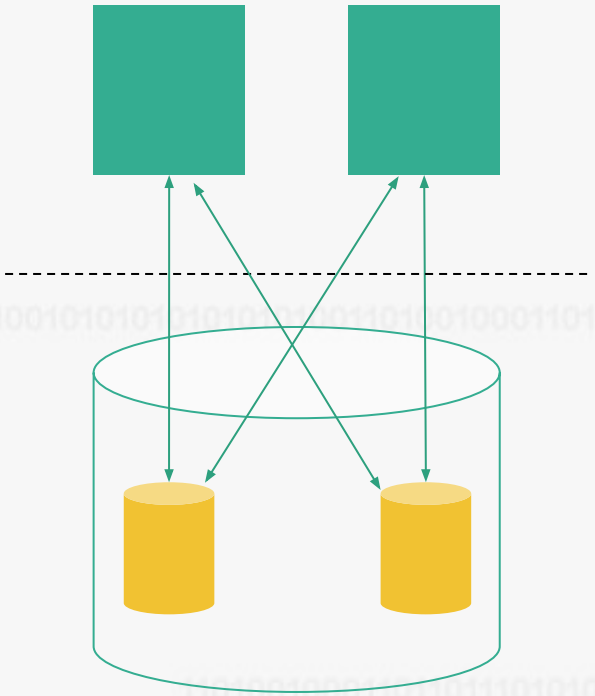
✓ 有状态流式处理（Stateful Stream Processing）为 event-driven 应用的重点技术

✓ 与微服务架构的核心理念吻合

- ✕ 流式处理重点在于即时性
- ✕ 流式处理只用于数据分析
- ？ 流式处理已经发展完毕

一致性

Classic tiered architecture



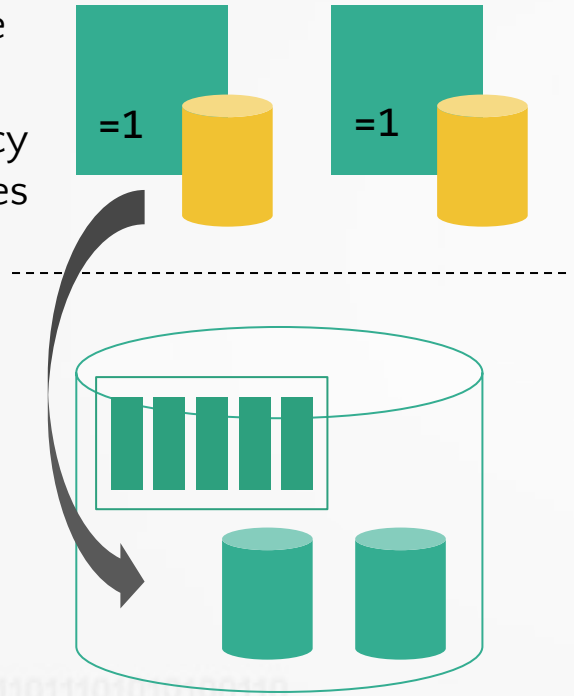
distributed transactions

at scale typically
at-most / at-least once

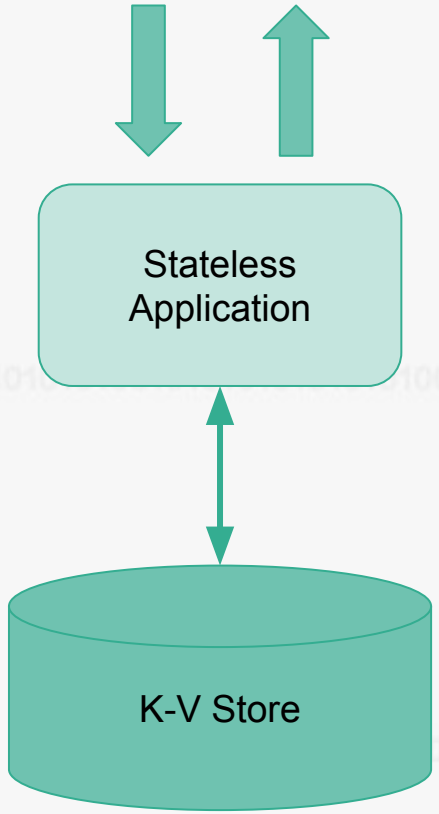
Streaming architecture

exactly once
per state

snapshot consistency
across states

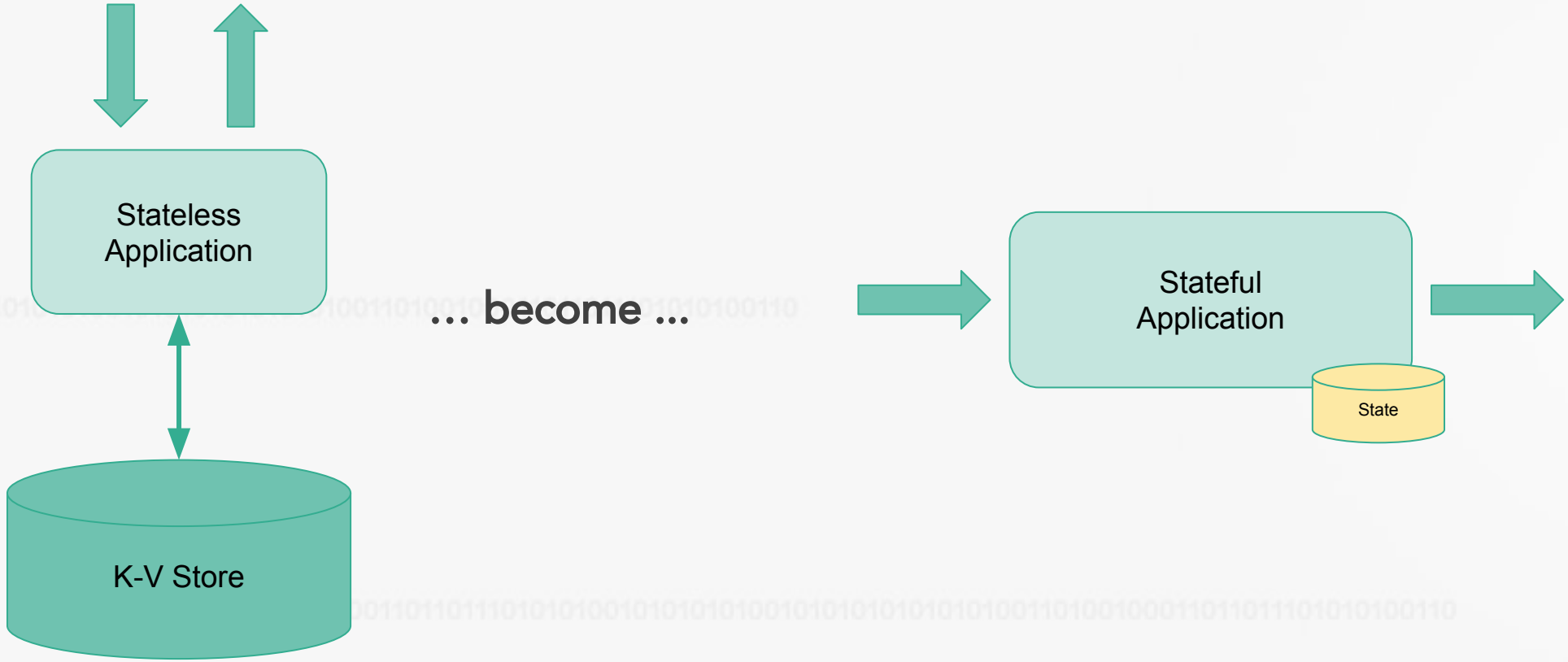


数据应用架构演变

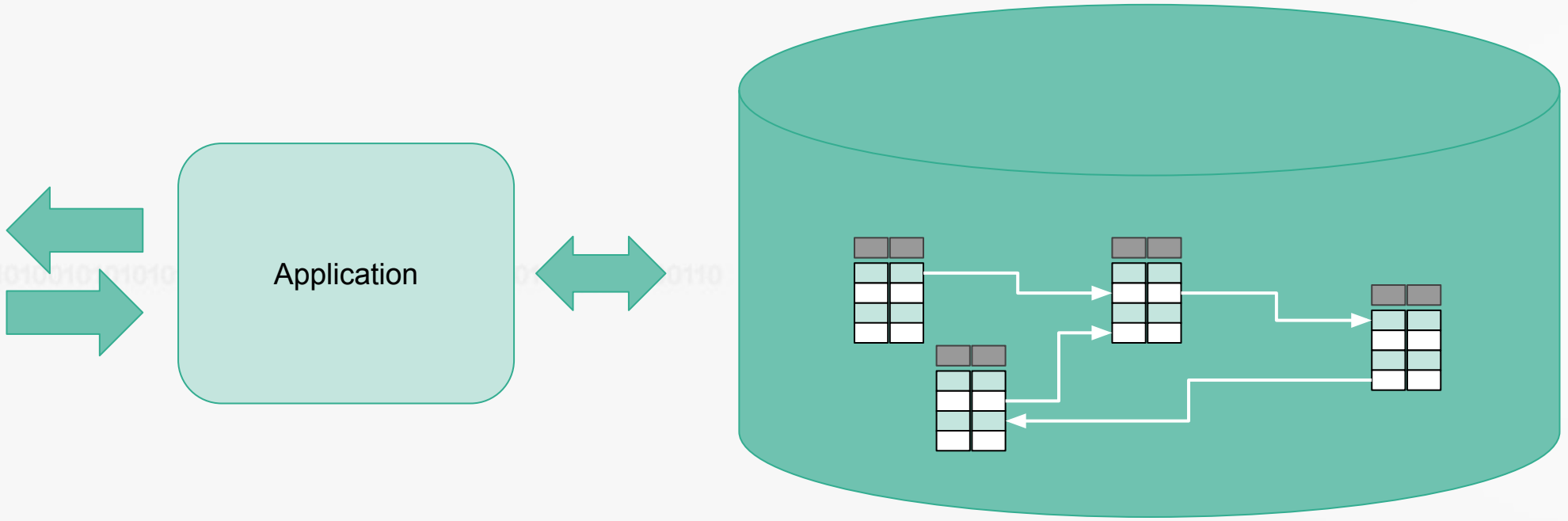


... become ...

数据应用架构演变


Stateless
Application
K-V Store
... become ...
Stateful
Application
State

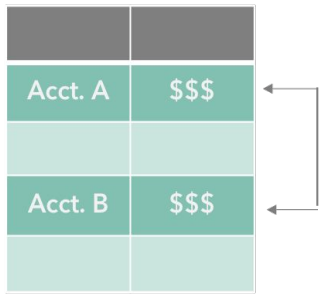
仍有些应用尚无法利用流式处理表示

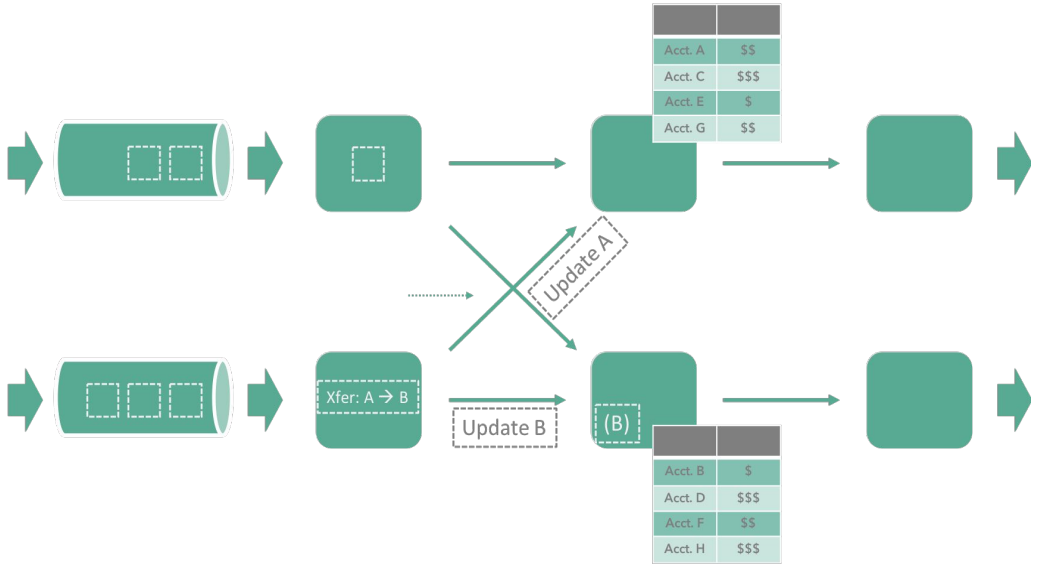


Apache Flink

目前有状态流式处理引擎的限制

LIMITATION	EXAMPLE
Up until now, stream processors could only update a single key with strong correctness guarantees (exactly once)	Wiring money from one account (key) to another with transactional guarantees is not feasible





Apache Flink

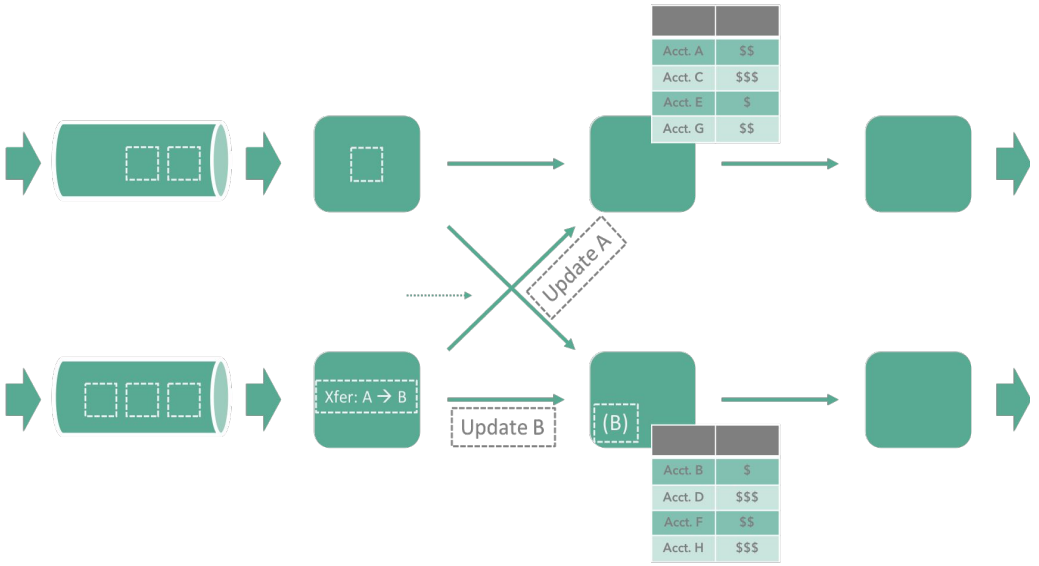
目前有状态流式处理引擎的限制

ACID

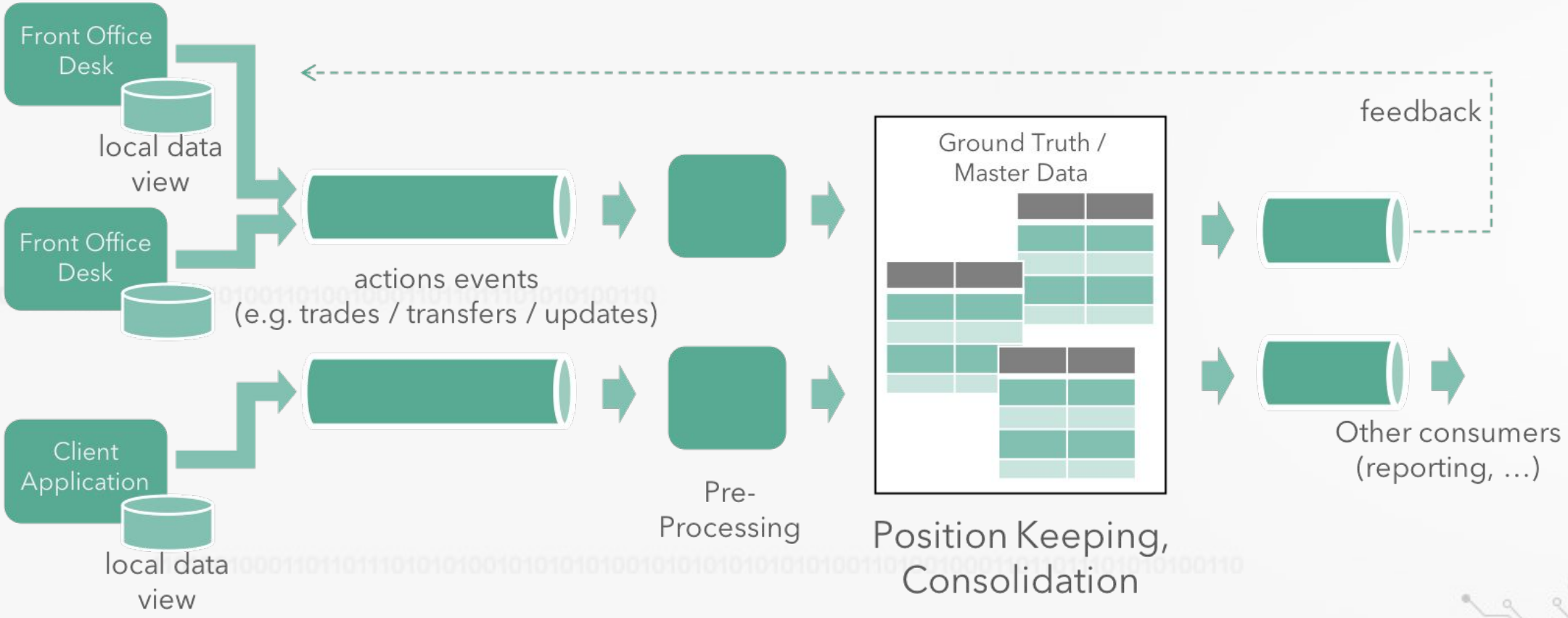
- **Atomicity:** the transfer affects either both accounts or none
- **Consistency:** the transfer must only happen if the account have sufficient funds
- **Isolation:** no other operation can interfere and cause an incorrect result
- **Durability:** the result of the transfer is durable

EXAMPLE

Wiring money from one account (key) to another with transactional guarantees is not feasible



例如:银行金融业常见应用



过去 …

At least once guarantees

Approximate real time analytics

- The first stream processors (Storm) offered at least once guarantees: no data loss but possible duplications to support real-time approximate analytics (Lambda)
- Think eventual consistency in distributed databases

… 现在 …

Accurate single-key applications

- Exactly-once guarantees for real-time streaming applications operating on a single key at a time
- Think k/v stores with single-key consistency

Approximate real time analytics

- The first stream processors (Storm) offered at least once guarantees: no data loss but possible duplications to support real-time approximate analytics (Lambda)
- Think eventual consistency in distributed databases

… 与未来

Accurate general applications

- ACID guarantees supporting applications that read and modify several keys
- Think relational database systems with ACID guarantees

Accurate single-key applications

- Exactly-once guarantees for real-time streaming applications operating on a single key at a time
- Think k/v stores with single-key consistency

Approximate real time analytics

- The first stream processors (Storm) offered at least once guarantees: no data loss but possible duplications to support real-time approximate analytics (Lambda)
- Think eventual consistency in distributed databases

? 流式处理重点在于即时性

? 流式处理只用于数据分析

? 流式处理已经发展完毕

✓ 流式处理重点在于是对运算
Continuous Data 最自然的运算框架

? 流式处理只用于数据分析

? 流式处理已经发展完毕

✓ 流式处理重点在于是对运算
Continuous Data 最自然的运算框架

✓ 流式处理正渐渐改变现代应用的架构方式

? 流式处理已经发展完毕

✓ 流式处理重点在于是对运算
Continuous Data 最自然的运算框架

✓ 流式处理正渐渐改变现代应用的架构方式

✓ 流式处理仍很有演进空间，
「Stream Processor as a Database」
即是其中的重点方向之一

THANKS

Flink China社区大群

扫一扫群二维码，立刻加入该群。