# Kfaka Stream使用总结



## 一、简介

### 1. 什么是kafka stream?

Kafka Stream是Apache Kafka从0.10版本引入的一个新特性。它是一个用于处理和分析存储在Kafka中的数据，并将得到的数据写回Kafka的**客户端程序库**。

### 2.kafka stream的特点

1. 依赖少.除kafka外，无其他依赖。
2. 提供了一个简单而轻量的jar包，可以很方便的嵌入到java程序中,方便打包与部署。
3. 基于Kafka的分区机制和Rebalance机制，实现水平扩展和在线动态调整并行度。
4. 提供记录级的处理能力，从而实现毫秒级的低延迟。
5. 支持通过状态存储stateStore实现状态操作以及支持基于事件时间的窗口操作。
6. 提供高级别(DSL)、低级别(processor)两套操作API。

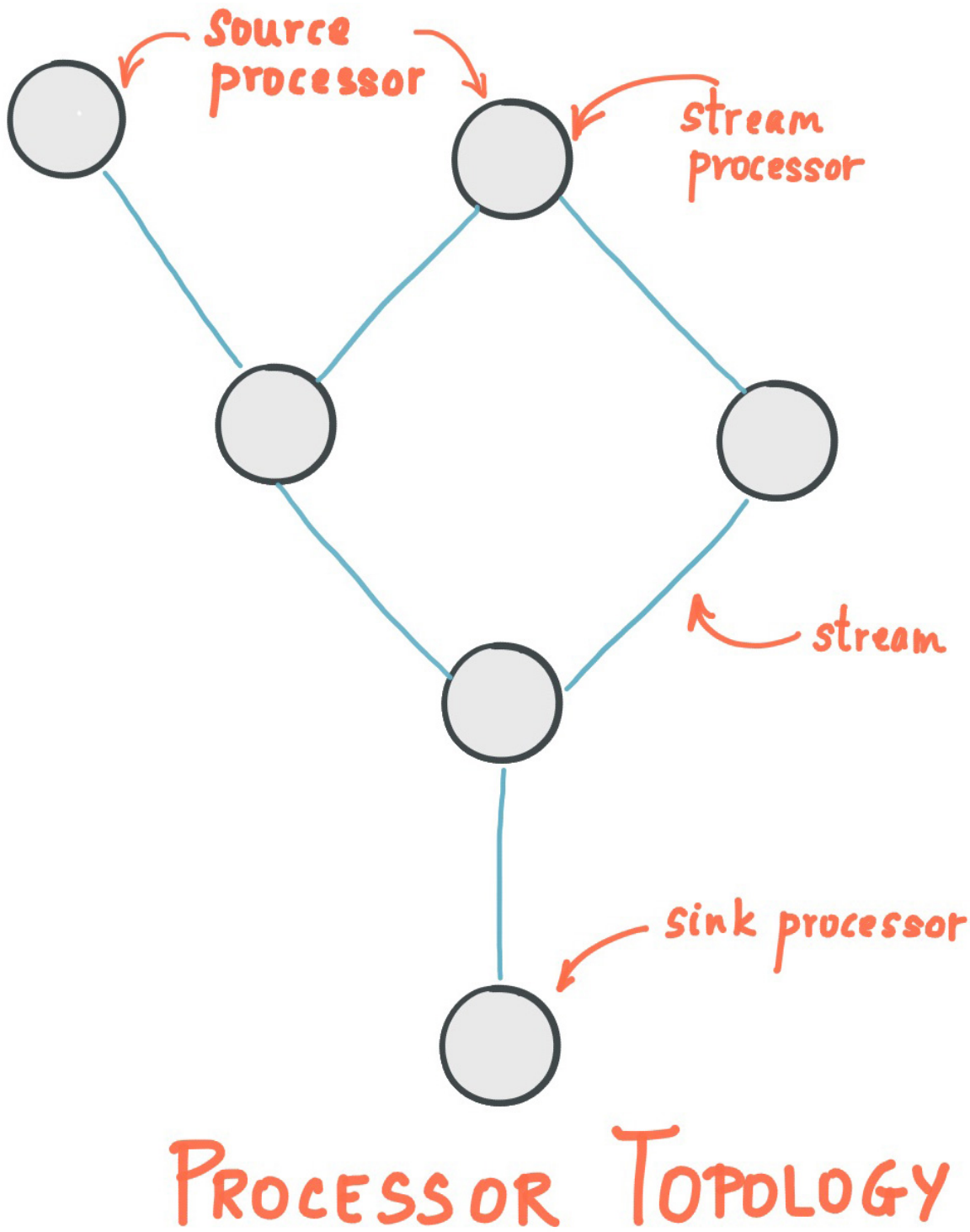
### 3.核心概念

#### 3.1.拓扑(Topology)

拓扑为kafka stream处理的逻辑图谱,由来源,一个个逻辑处理器节点,流的流向以及流的输出构成.kafka stream提供两种方式来构建拓扑:

①kafka stream DSL(高级别)提供了常用的数据转化操作,如:filter,map,count等.

②processor(低级别) 允许开发者自己定义处理逻辑,以及基于状态仓库(stateStore)做计算.




**3.2.时间**

kafka stream中时间的概念:

- 事件时间: 当一个数据记录发生的时间点，也就是数据被创建的时间。
- 处理时间: 数据记录被流处理的时间,也即是数据被kafka stream消费的时间.
- 摄取时间: 数据记录被kafka broker存储在topic分区的时间.

kafka stream在0.10后允许实现`org.apache.kafka.streams.processor.TimestampExtractor`接口,基于该接口,可根据业务需求自定义执行不同的时间.

当kafka stream处理完数据写回到kafka中时,kafka stream将分配时间戳给新的消息.分配规则有上下文决定:

- 当通过处理一些输入记录（例如，在process（）函数调用中触发的context.forward（））生成新的输出记录时，输出记录时间戳直接从输入记录时间戳继承。
- 当通过周期性函数（如punctuate()）生成新的输出记录时。输出记录时间戳被定义为流任务的当前内部时间（通过context.timestamp()获取）。
- 对于聚合操作，生成的聚合更新的记录时间戳将被最新到达的输入记录触发更新。

### 3.3.状态

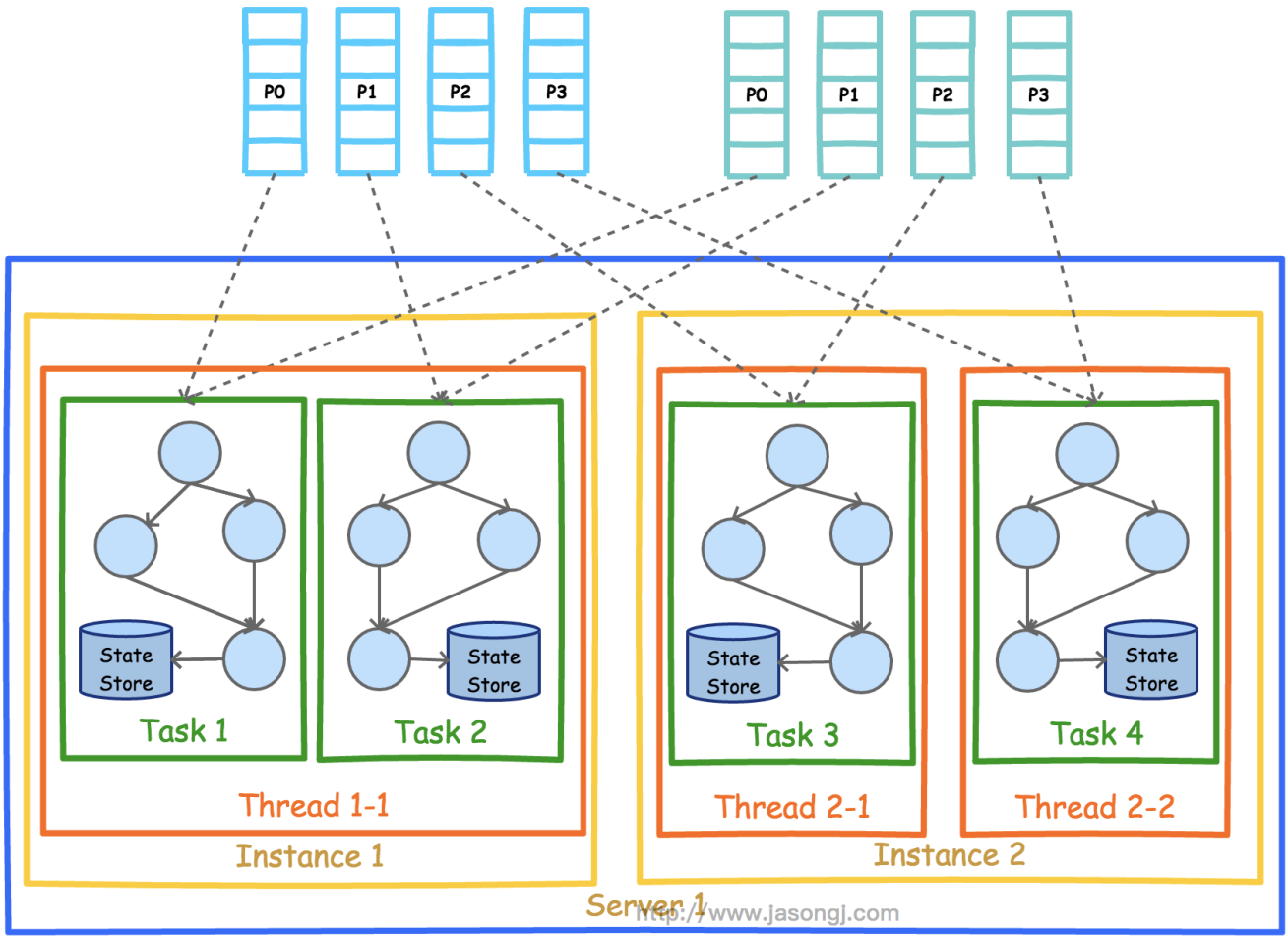
一些流处理程序不需要状态,也就是每条消息处理独立于其他的消息处理,如:过滤字符,文本打标签等.而另一些流处理程序是需要状态的,如:网站PV量的计算等,kafka stream提供了状态存储功能,流处理程序可以用来存储和查询数据.kafka stream默认将数据存储在本地RocksDB数据库中.

### 3.4.分区(Partition)和任务(Task)

Kafka分区数据的消息层用于存储和传输。Kafka Streams分区数据用于处理。基于kafka topic分区的并行性模型,kafka stream使用了分区和任务的概念.

Kafka Streams根据输入流分区创建固定数量的Task，其中每个Task分配一个输入流的topic. 分区对Task的分配不会改变，因此每个Task是应用程序并行性的固定单位。然后，Task可以基于分配的分区实现自己的处理器拓扑.如果某个Stream的输入Topic有多个(比如2个Topic，1个Partition数为4，另一个Partition数为3)，则总的Task数等于Partition数最多的那个Topic的Partition数。这是因为Kafka Stream使用了Consumer的Rebalance机制，每个Partition对应一个Task。

并行模式下的分布:



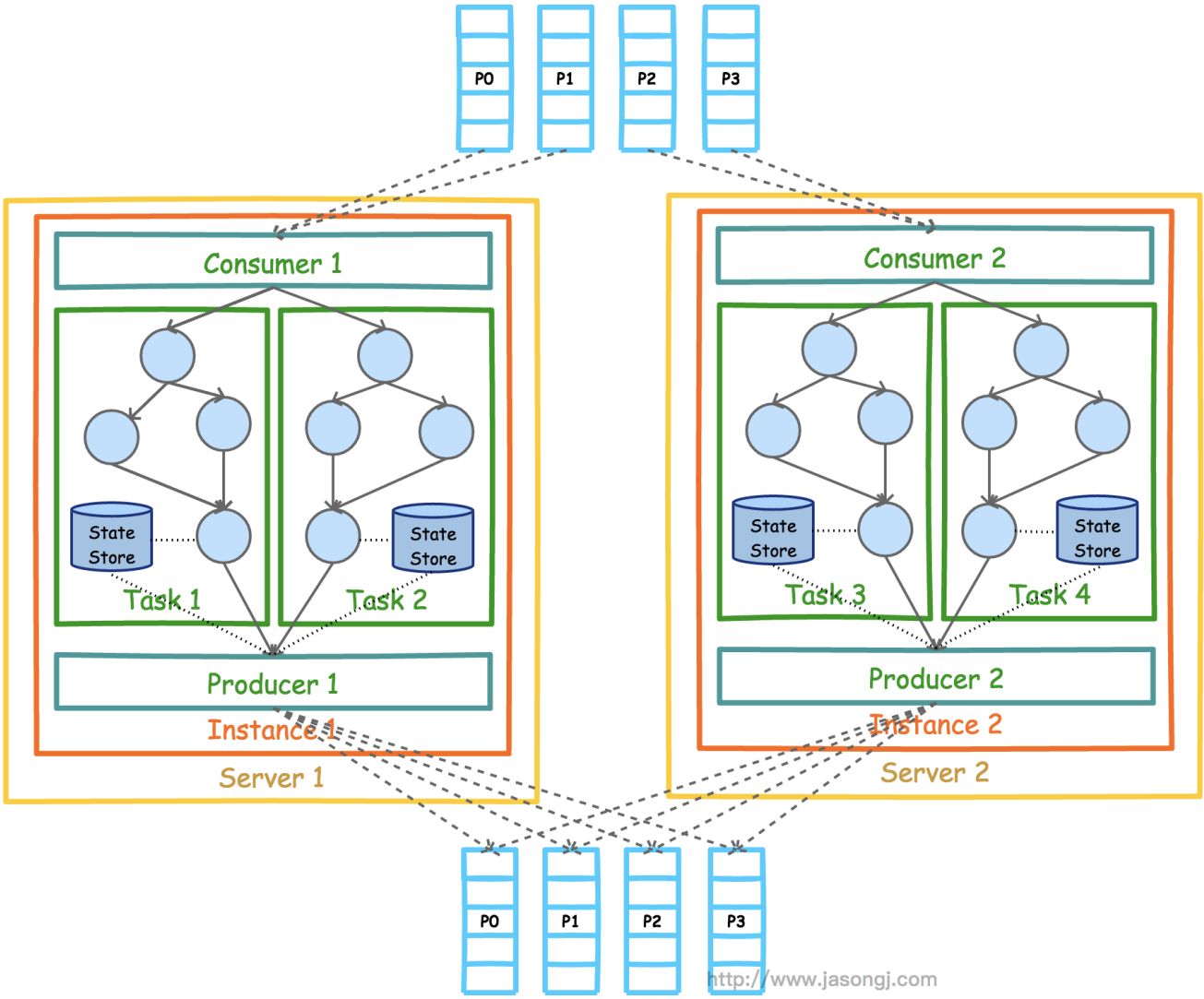
### 3.5 数据抽象

1. **KStream**: data as record stream,KStream为一个insert队列,新数据不断增加进来.
2. **KTable**: data as change log stream,KTable为一个update队列,新数据和已有数据有相同的key,则用新数据覆盖原有原来的数据.

**流表二元性**:

- 流作为表: 一个流可以认为是一个表的变更日志，其中在流中的每个的数据记录捕获表的状态变化。因此，流其实是一个伪装的表，并且可以通过从开始到结束重放变更日志来很容地重构表。
- 表作为流: 表可以认为是在流中的每个key的最新value的一个时间点的快照(流的数据记录是k-v键值对)。因此，表也可以认为是伪装的流，它可以通过对表中每个k-v进行迭代而容易的转换成流。

# 二、架构




如图所示，kafka stream支持接收多个topic中传来的数据，并且通过kafka的rebalance机制，各个程序之间支持水平扩展。kafka stream从kafka中获取数据，并且内置了consumer和producer。通过内置的consumer接收到kafka中的数据，处理后再通过内置的producer将数据返回到kafka中。

# 三、为什么选择Kafka Stream?

**1. 使用成本低**

与Spark和Storm等流式处理框架相比，kafka stream提供的是一个基于kafka的流式处理类库。且kafka stream作为流式处理类库，直接提供具体的类和接口给开发者，整个程序处理逻辑全都由开发者自己控制，方便开发和调试。

**2.轻量易部署**

由于kafka stream是作为类库嵌入程序中，使得kafka stream打包部署非常方便。并且kafka stream利用了kafka的分区机制和consumer的rebalance机制，使得kafka stream程序可以非常方便的进行水平扩展，并且可以在线动态调整并行度。

**3.性能**

Kafka Stream的并行模型中，最小粒度为Task，而每个Task包含一个特定子Topology的所有Processor，使得所有处理逻辑都在同一线程内完成。这一特点跟Storm的Topology完全不一样。Storm的Topology的每一个Task只包含一个Spout或Bolt的实例。因此Storm的一个Topology内的不同Task之间需要通过网络通信传递数据，而Kafka Stream的Task包含了完整的子Topology，所以Task之间不需要传递数据，也就不需要网络通信。这一点降低了系统复杂度，也提高了处理效率。

# 四、缺点

- 暂不支持异步操作.所以在处理逻辑中避免使用高开销的操作,否则整个处理线程将会阻塞.
- 不支持像spark streaming那样使用SQL完成实时的日志数据统计.
- 数据来源单一,只支持kafka作为数据来源.

# 应用示例

# 疑问解答

1. **Task和线程之间的关系**

- kafka stream通过`props.put(StreamsConfig.NUM_STREAM_THREADS_CONFIG,2);`属性可以设置并行的线程数.
- Task的数量由Topic的分区数决定,取监听的topic中最大的分区数作为Task的数量,Task和Thread之间的分配由线程数决定.若有4个Task,但是只有一个Thread,则4个Task位于同一线程中串行.若有4个Task,和4个Thread,则每个Task独享一个线程,并行处理.若Task数大于Thread数,则有kafka stream自行做分配.若Task数小于线程数,则会出现某些线程不能执行Task的情况.

2. **当某一实例处理数据时宕机了,数据是否会丢失**

   分区与任务的分配永远不改变,当应用实例执行任务失败时,则其被分配的任务将自动在其他实例中被创建,并从相同的流分区重新消费.