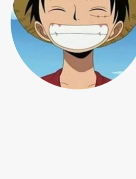


收录于话题
#flink 7 #checkpoint 2



浪尖聊大数据
主要分享大数据框架，如spark, flink, kafka, hbase原理源码，同时会分享数据仓...
366篇原创内容

Official Account

前面，已经有一篇文章讲解了spark的checkpoint:

必会:关于SparkStreaming checkpoint那些事儿

同时，浪尖也在知识星球里发了源码解析的文章。spark streaming的Checkpoint仅仅是针对driver的故障恢复做了数据和元数据的Checkpoint。而本文要讲的flink的checkpoint机制要复杂了很多，它采用的是轻量级的分布式快照，实现了每个操作符的快照，及循环流的在循环的数据的快照。详细的算法后面浪尖会给出文章。

1. 简介

Apache Flink提供容错机制，以持续恢复数据流应用程序的状态。该机制确保即使存在故障，程序的每条消息只会作用于状态一次（exactly-once），当然也可以降级为至少一次（at-least-once）。

容错机制持续地制作分布式流数据流的快照。对于状态较小的流应用程序，这些快照非常轻量级，可以频繁产生快照，而不会对性能产生太大影响。流应用程序的状态存储的位置是可以配置的（例如存储在master节点或HDFS）。

如果程序失败（由于机器，网络或软件故障），Flink任务挂掉，然后利用最近一次成功的checkpoint恢复算子的状态。输入流将重置为状态快照消息的位置。

注意：默认情况下，禁用checkpoint。

注意：要使容错机制完整，数据源(如消息队列或者broker)要支持数据回滚到历史消息的位置。Apache Kafka具有这种能力，Flink与Kafka的连接器利用了该功能。

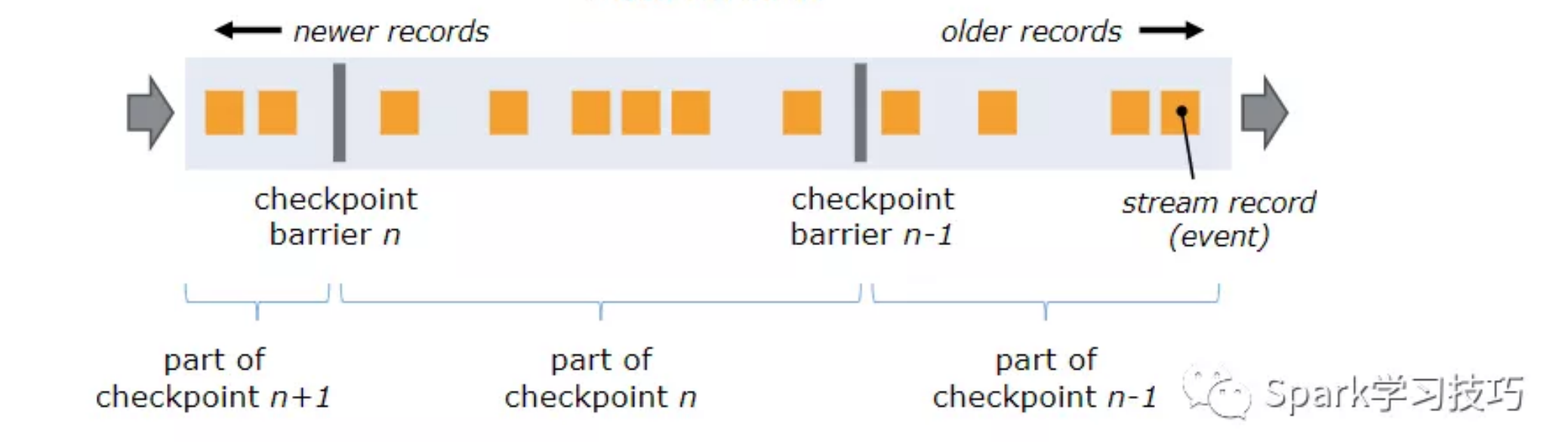
注意：由于Flink的checkpoint是通过分布式快照实现的，因此快照和checkpoint的概念可以互换使用。

2. Checkpointing

Flink的容错机制的核心部分是制作分布式数据流和操作算子状态的一致性快照，这些快照充当一致性checkpoint，系统可以在发生故障时回滚。Flink用于制作这些快照的机制在“分布式数据流的轻量级异步快照”中进行了描述。它受到分布式快照的标准Chandy-Lamport算法的启发，专门针对Flink的执行模型而定制。

2.1 Barriers

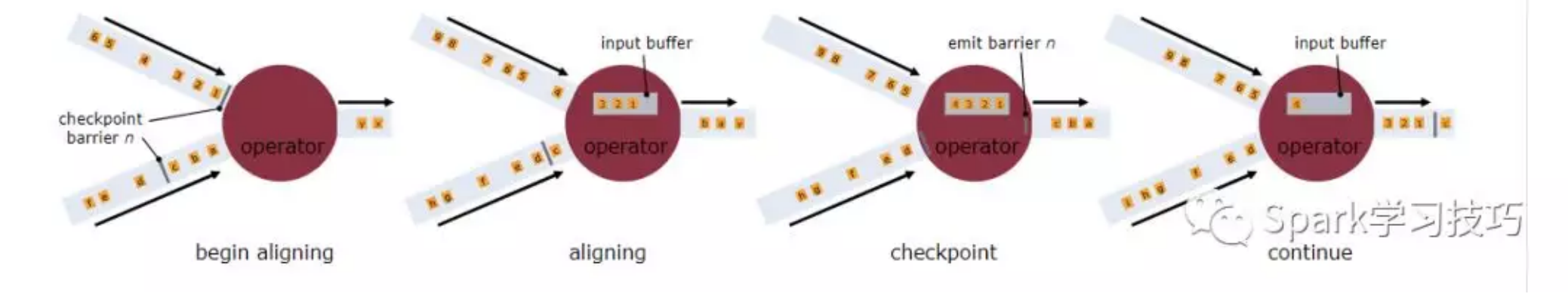
Flink分布式快照的核心概念之一是barriers。这些barriers被注入数据流并与消息一起作为数据流的一部分向下流动。barriers永远不会超过前面的消息，数据流严格有序。barriers将数据流中的消息分为进入当前快照的消息和进入下一个快照的消息。每个barriers都带有快照的ID，并且barriers之前的消息都进入了该快照。barriers不会中断消息流的流动，非常轻量级。来自不同快照的多个barriers可以同时流中出现，这意味着可以同时发生各种快照。



barriers在数据流source处被注入并行数据流中，快照n的barriers被插入的位置（我们称之为Sn）是快照所包含的数据在数据源中最大位置。例如，在Apache Kafka中，此位置将是分区中最后一条消息的偏移量。将该位置Sn报告给checkpoint协调器（Flink的JobManager）。

然后barriers向下游流动。当一个中间操作算子从其所有输入流中收到快照n的barriers时，它将为快照n发出barriers进入其所有输出流中。一旦sink操作算子（流式DAG的末端）从其所有输入流接收到barriers n，它就向checkpoint协调器确认快照n完成。在所有sink确认快照后，意味快照着已完成。

一旦完成快照n，job将永远不再向数据源请求Sn之前的消息，因为此时这些消息（及其后续消息）都已经通过整个数据流拓扑，也即是已经被处理结束啦。



接收多个输入流的运算符需要基于快照barriers对齐输入流。上图说明了这一点：

- 一旦操作算子从一个输入流接收到快照barriers n，它就不能处理来自该流的任何消息，直到它接收到其他输入算子barriers n为止。否则，它会搞混属于快照n的消息和属于快照n + 1的消息。
- barriers n所属的流暂时会被搁置。从这些流接收的消息不会被处理，而是放入输入缓冲区。
- 一旦从最后一个流接收到barriers n，操作算子就会发出所有挂起的向后传送的消息，然后自己发出快照n的barriers。
- 之后，它恢复处理来自所有输入流的消息，在处理来自流的消息之前优先处理来自输入缓冲区的信息。

2.2 state

当运算符包含任何形式的状态时，此状态也必须是快照的一部分。操作算子状态有不同的形式：

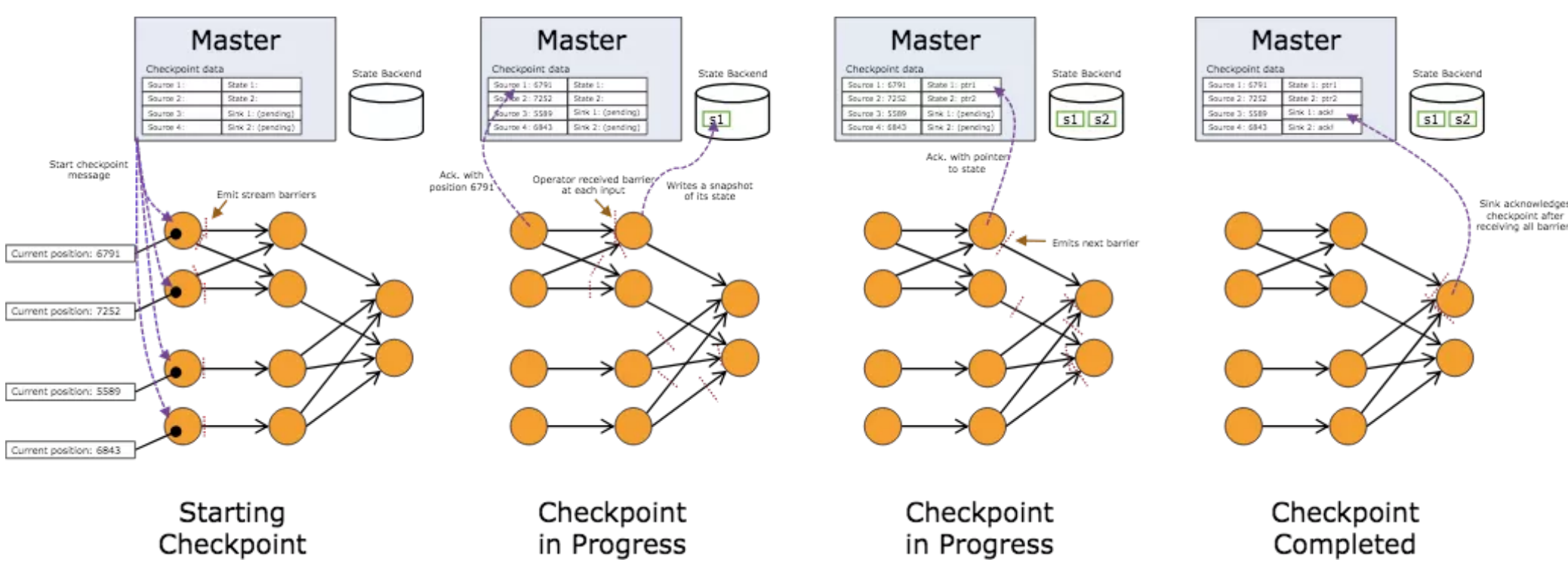
用户定义的状态：这是由转换函数（如map（）或filter（））直接创建和修改的状态。

系统状态：此状态是指作为运算符计算一部分的数据缓冲区。此状态的典型示例是窗口缓冲区，系统在其中收集（和聚合）窗口里的消息，直到窗口被计算和抛弃。

操作算子在他们从输入流接收到所有快照barriers时，以及在向其输出流发出barriers之前，会对其状态进行写快照。此时，在barrier之前的数据对状态的更新已经完成，barrier之后的数据不会更新状态。由于快照的状态可能很大，因此它存储在可配置的状态后端中。默认情况下，是存储到JobManager的内存，但对于生产使用，应配置分布式可靠存储（例如HDFS）。在存储状态之后，操作算子确认checkpoint完成，将快照barriers发送到输出流中，然后继续。

生成的快照现在包含：

- 对于每个并行流数据源，创建快照时流中的偏移/位置
- 对于每个运算符，存储在快照中的状态指针



2.3 Exactly Once vs. At Least Once

对齐步骤可能增加流式程序的等待时间。通常，这种额外的延迟大约为几毫秒，但也会见到一些延迟显著增加的情况。对于要求所有消息始终具有超低延迟（几毫秒）的应用程序，Flink可以在checkpoint期间跳过流对齐。一旦操作算子看到每个输入流的checkpoint barriers，就会写checkpoint快照。

当跳过对齐时，即使在checkpoint n的某些checkpoint barriers到达之后，操作算子仍继续处理所有输入。这样，操作算子还可以在创建checkpoint n的状态快照之前，继续处理属于checkpoint n + 1的数据。在还原时，这些消息将作为重复消息出现，因为它们都包含在checkpoint n的状态快照中，并将作为checkpoint n之后数据的一部分进行重复处理。

注意：对齐仅适用于具有多个输入（join）的运算符以及具有多个输出的运算符（在流重新分区/shuffle之后）。正因为如此，对于只有map（），flatMap（），filter（）等操作，实际上即使在至少一次模式下也能提供一次保证。

2.4 异步状态快照

注意，上述机制意味着操作算子在将状态的快照存储在状态后端时，停止处理输入消息。每次写快照时，这种同步状态快照操作都会引入延迟。

可以让操作算子在存储状态快照时继续处理，高效地让状态快照存储在后台异步发生。为此，操作算子必须能够生成一个状态对象，该状态对象应以某种方式存储，以便对操作算子状态的进一步修改不会影响该状态对象。例如，RocksDB中使用的写时复制(copy-on-write)数据结构具有这种能力。

在接收到输入的checkpoint的barriers后，操作算子启动其状态的异步快照复制。它立即释放其barriers到输出，并继续进行常规流处理。后台复制过程完成后，它会向checkpoint协调器（JobManager）确认checkpoint完成。checkpoint仅在所有sink都已收到barriers并且所有状态操作算子已确认其完成备份（可能在barriers到达sink之后）之后才算完成。

2.5 Recovery

在这种机制下的恢复是很直接的：当失败时，Flink选择最新完成的checkpoint k。然后，系统重新部署整个分布式数据流，并为每个操作算子重置作为checkpoint k的一部分的快照的状态。数据源设置为从位置Sk开始读取。例如在Apache Kafka中，这意味着告诉消费者从偏移量Sk开始读取。

如果状态以递增方式写快照，则操作算子从最新完整快照的状态开始，然后对该状态应用一系列增量快照更新。

2.6 操作算子快照的实现

在创建操作算子快照时，有两部分：同步部分和异步部分。

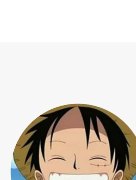
操作算子和状态后端将其快照提供为Java FutureTask。该任务包含同步部分已完成且异步部分处于挂起状态的状态。然后，异步部分由该checkpoint的后台线程执行。

完全同步的checkpoint返回已经完成的FutureTask的运算符。如果需要执行异步操作，则在FutureTask的run（）方法中执行。

任务是可取消的，可以释放流和其他资源消耗的句柄。

推荐阅读：

- 干货:Flink+Kafka 0.11端到端精确一次处理语义实现
- Spark Streaming VS Flink



未命名公众号
本公众号主要分享Spark使用及源码，spark 机器学习，图计算，同时会涉及到，如...
11篇原创内容

Official Account

Read more

People who liked this content also liked

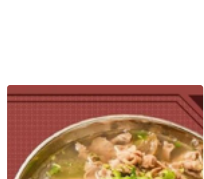
用户画像-标签体系

浪尖聊大数据



撷尽四川之味，乐山比成成都更好吃

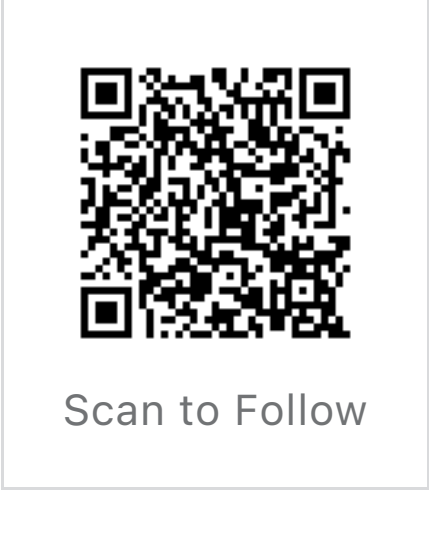
吃喝鸡冻队



铁汉柔情！边疆军人的浪漫告白，你慕了吗？

喀喇昆仑卫士





Scan to Follow