Flink

1. Flink 的容错机制（checkpoint）

Flink可靠性的基石-checkpoint机制详细解析

2. Flink checkpoint与 Spark Flink 有什么区别或优势吗

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 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，实现了每个算子的快照，及流动中的数据的快照。

3… Flink 中的 Time 有哪几种

在flink中被划分为事件时间，提取时间，处理时间三种。

如果以EventTime为基准来定义时间窗口那将形成EventTimeWindow,要求消息本身就应该携带EventTime。

如果以IngesingtTime为基准来定义时间窗口那将形成IngestingTimeWindow,以source的systemTime为准。

如果以ProcessingTime基准来定义时间窗口那将形成ProcessingTimeWindow，以operator的systemTime为准。

4. 对于迟到数据是怎么处理的

Flink中 WaterMark 和 Window 机制解决了流式数据的乱序问题，对于因为延迟而顺序有误的数据，可以根据eventTime进行业务处理，对于延迟的数据Flink也有自己的解决办法，主要的办法是给定一个允许延迟的时间，在该时间范围内仍可以接受处理延迟数据

设置允许延迟的时间是通过allowedLateness(lateness: Time)设置

保存延迟数据则是通过sideOutputLateData(outputTag: OutputTag[T])保存

获取延迟数据是通过DataStream.getSideOutput(tag: OutputTag[X])获取

文章推荐：

Flink 中极其重要的 Time 与 Window 详细解析

5. Flink 的运行必须依赖 Hadoop组件吗

Flink可以完全独立于Hadoop，在不依赖Hadoop组件下运行。但是做为大数据的基础设施，Hadoop体系是任何大数据框架都绕不过去的。Flink可以集成众多Hadooop 组件，例如Yarn、Hbase、HDFS等等。例如，Flink可以和Yarn集成做资源调度，也可以读写HDFS，或者利用HDFS做检查点。

6. Flink集群有哪些角色？各自有什么作用

有以下三个角色：

JobManager处理器：

也称之为Master，用于协调分布式执行，它们用来调度task，协调检查点，协调失败时恢复等。Flink运行时至少存在一个master处理器，如果配置高可用模式则会存在多个master处理器，它们其中有一个是leader，而其他的都是standby。

TaskManager处理器：

也称之为Worker，用于执行一个dataflow的task(或者特殊的subtask)、数据缓冲和data stream的交换，Flink运行时至少会存在一个worker处理器。

Clint客户端：

Client是Flink程序提交的客户端，当用户提交一个Flink程序时，会首先创建一个Client，该Client首先会对用户提交的Flink程序进行预处理，并提交到Flink集群中处理，所以Client需要从用户提交的Flink程序配置中获取JobManager的地址，并建立到JobManager的连接，将Flink Job提交给JobManager

7. Flink 资源管理中 Task Slot 的概念

在Flink中每个TaskManager是一个JVM的进程, 可以在不同的线程中执行一个或多个子任务。

为了控制一个worker能接收多少个task。worker通过task slot（任务槽）来进行控制（一个worker至少有一个task slot）。

8. Flink的重启策略了解吗

Flink支持不同的重启策略，这些重启策略控制着job失败后如何重启：

固定延迟重启策略

固定延迟重启策略会尝试一个给定的次数来重启Job，如果超过了最大的重启次数，Job最终将失败。在连续的两次重启尝试之间，重启策略会等待一个固定的时间。

失败率重启策略

失败率重启策略在Job失败后会重启，但是超过失败率后，Job会最终被认定失败。在两个连续的重启尝试之间，重启策略会等待一个固定的时间。

无重启策略

Job直接失败，不会尝试进行重启。

9. Flink是如何保证Exactly-once语义的

Flink通过实现两阶段提交和状态保存来实现端到端的一致性语义。分为以下几个步骤：

开始事务（beginTransaction）创建一个临时文件夹，来写把数据写入到这个文件夹里面

预提交（preCommit）将内存中缓存的数据写入文件并关闭

正式提交（commit）将之前写完的临时文件放入目标目录下。这代表着最终的数据会有一些延迟

丢弃（abort）丢弃临时文件

若失败发生在预提交成功后，正式提交前。可以根据状态来提交预提交的数据，也可删除预提交的数据。

10. 如果下级存储不支持事务，Flink 怎么保证 exactly-once

端到端的exactly-once对sink要求比较高，具体实现主要有幂等写入和事务性写入两种方式。

幂等写入的场景依赖于业务逻辑，更常见的是用事务性写入。而事务性写入又有预写日志（WAL）和两阶段提交（2PC）两种方式。

如果外部系统不支持事务，那么可以用预写日志的方式，把结果数据先当成状态保存，然后在收到 checkpoint 完成的通知时，一次性写入 sink 系统。

11. Flink是如何处理反压的

Flink 内部是基于 producer-consumer 模型来进行消息传递的，Flink的反压设计也是基于这个模型。Flink 使用了高效有界的分布式阻塞队列，就像 Java 通用的阻塞队列（BlockingQueue）一样。下游消费者消费变慢，上游就会受到阻塞。

12. Flink中的状态存储

Flink在做计算的过程中经常需要存储中间状态，来避免数据丢失和状态恢复。选择的状态存储策略不同，会影响状态持久化如何和 checkpoint 交互。Flink提供了三种状态存储方式：MemoryStateBackend、FsStateBackend、RocksDBStateBackend。

13. Flink是如何支持批流一体的

这道题问的比较开阔，如果知道Flink底层原理，可以详细说说，如果不是很了解，就直接简单一句话：Flink的开发者认为批处理是流处理的一种特殊情况。批处理是有限的流处理。Flink 使用一个引擎支持了DataSet API 和 DataStream API。

14. Flink的内存管理是如何做的

Flink 并不是将大量对象存在堆上，而是将对象都序列化到一个预分配的内存块上。此外，Flink大量的使用了堆外内存。如果需要处理的数据超出了内存限制，则会将部分数据存储到硬盘上。Flink 为了直接操作二进制数据实现了自己的序列化框架。

15. Flink CEP 编程中当状态没有到达的时候会将数据保存在哪里

在流式处理中，CEP 当然是要支持 EventTime 的，那么相对应的也要支持数据的迟到现象，也就是watermark的处理逻辑。CEP对未匹配成功的事件序列的处理，和迟到数据是类似的。在 Flink CEP的处理逻辑中，状态没有满足的和迟到的数据，都会存储在一个Map数据结构中，也就是说，如果我们限定判断事件序列的时长为5分钟，那么内存中就会存储5分钟的数据，这在我看来，也是对内存的极大损伤之一。