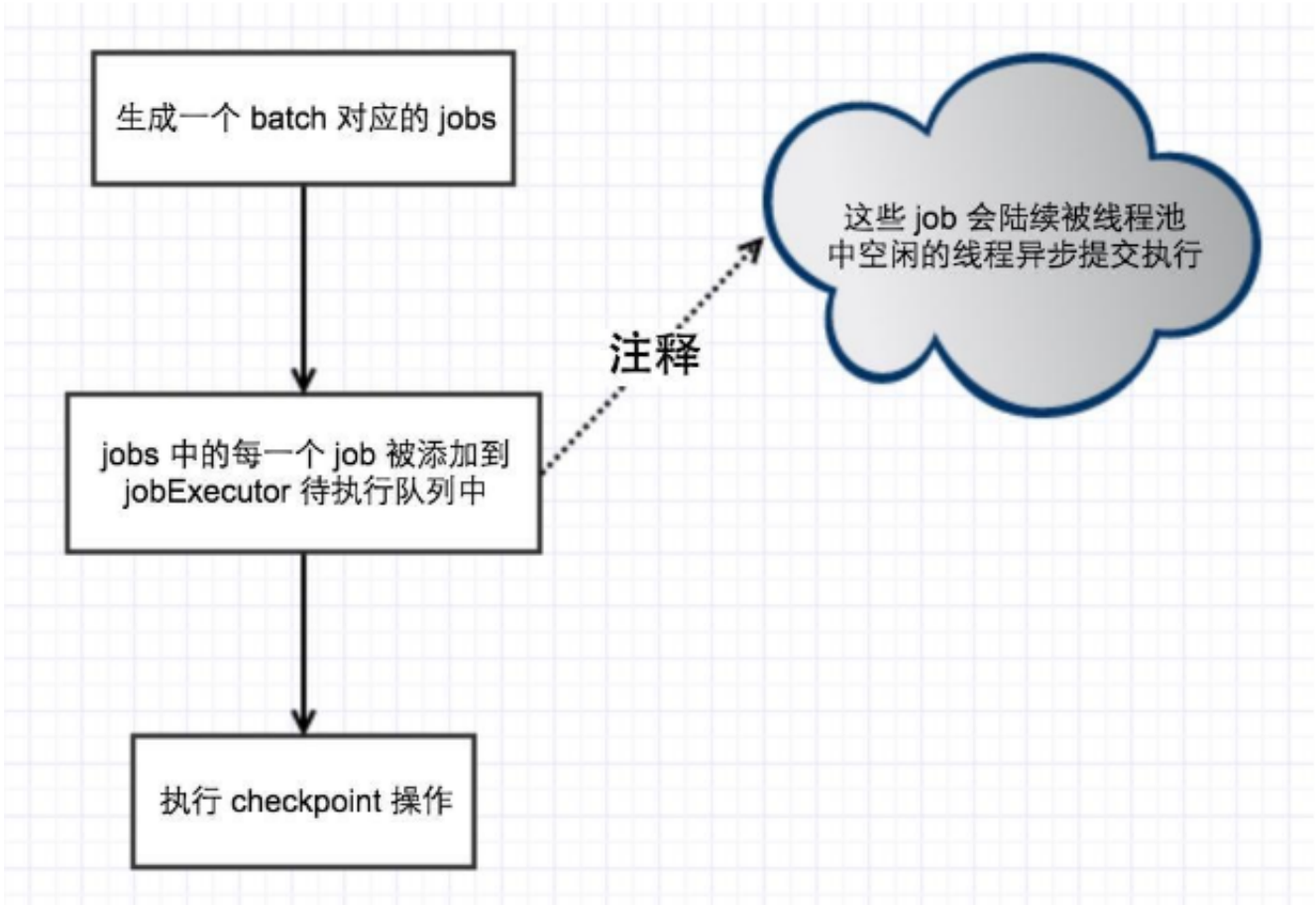
为什么Spark Streaming + Kafka很难保证exactly once？

Streaming job 的调度与执行

我们先来看看如下 job 调度执行流程图：



为什么很难保证 exactly once

上面这张流程图最主要想说明的就是，job 的提交执行是异步的，与 checkpoint 操作并不是原子操作。这样的机制会引起数据重复消费问题：

为了简化问题容易理解，我们假设一个 batch 只生成一个 job，并且 spark.streaming.concurrentJobs 值为1，该值代表 jobExecutor 线程池中线程的个数，也即可以同时执行的 job 的个数。

假设，batch duration 为2s，一个 batch 的总共处理时间为1s，此时，一个 batch

开始了，第一步生成了一个 job，假设花了0.1s，然后把该 job 丢到了 jobExecutor 线程池中等待调度执行，由于 checkpoint 操作和 job 在线程池中执行是异步的，在0.2s 的时候，checkpoint 操作完成并且此时开始了 job 的执行。

注意，这个时候 checkpoint 完成了并且该 job 在 checkpoint 中的状态是未完成的，随后在第1s 的时候 job 完成了，那么在这个 batch 结束的时候 job 已经完成了但该 job 在 checkpoint 中的状态是未完成的。

在下一个 batch 运行到 checkpoint 之前就挂了（比如在拉取数据的时候挂了、OOM 挂了等等异常情况），driver 随后从 checkpoint 中恢复，那么上述的 job 依然是未执行的，根据使用的 api 不同，对于这个 job 会再次拉取数据或从 wal 中恢复数据重新执行该 job，那么这种情况下该 job 的数据就就会被重复处理。比如这时记次的操作，那么次数就会比真实的多。

如果一个 batch 有多个 job 并且spark.streaming.concurrentJobs大于1，那么这种情况就会更加严重，因为这种情况下就会有多个 job 已经完成但在 checkpoint 中还是未完成状态，在 driver 重启后这些 job 对应的数据会被重复消费处理。

另一种会导致数据重复消费的情况主要是由于 Spark 处理的数据单位是 partition 引起的。比如在处理某 partition 的数据到一半的时候，由于数据内容或格式会引起抛异常，此时 task 失败，Spark 会调度另一个同样的 task 执行，那么此时引起 task 失败的那条数据之前的该 partition 数据就会被重复处理，虽然这个 task 被再次调度依然会失败。若是失败还好，如果某些特殊的情况，新的 task 执行成功了，那么我们就很难发现数据被重复消费处理了。

如何保证 exactly once

至于如何才能保证 exactly once，其实要根据具体情况而定。总体来说，可以考虑以下几点：

1、业务是否不能容忍即使是极少量的数据差错，如果是那么考虑 exactly once。如果可以容忍，那就没必要非实现 exactly once 不可
2、即使重复处理极小部分数据会不会对最终结果产生影响。若不会，那重复处理就重复吧，比如排重统计
3、若一定要保证 exactly once，应该考虑将对 partition 处理和 checkpoint或自己实现类似 checkpoint 功能的操作做成原子的操作；并且对 partition 整批数据进行类似事物的处理