Spark Streaming和Kafka整合是如何保证数据零丢失

Kafka 2016-03-02 21:03:12 10639 16评论 下载为PDF

当我们正确地部署好Spark Streaming，我们就可以使用Spark Streaming提供的零数据丢失机制。为了体验这个关键的特性，你需要满足以下几个先决条件：

1、输入的数据来自可靠的数据源和可靠的接收器；
2、应用程序的metadata被application的driver持久化了(checkpointed)；
3、启用了WAL特性(Write ahead log)。

下面我将简单地介绍这些先决条件。

文章目录

1 可靠的数据源和可靠的接收器
2 元数据持久化(Metadata checkpointing)
3 可能存在数据丢失的场景
4 WAL（Write ahead log）

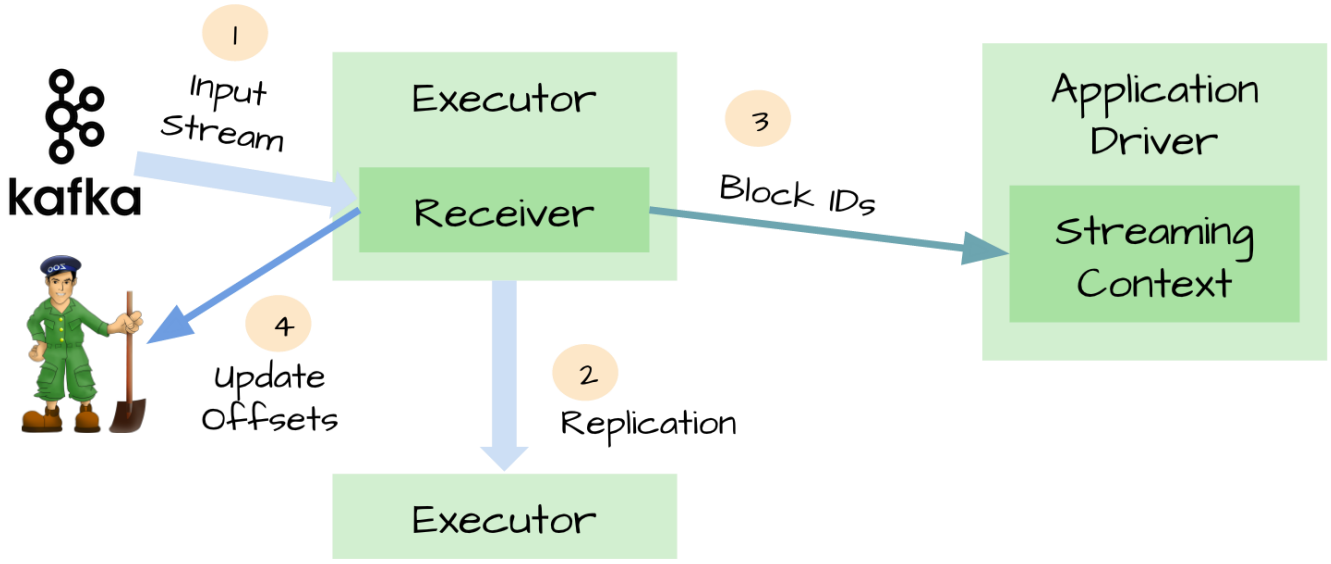
5 At-least-once语义

6 WAL的缺点

7 Kafka direct API

可靠的数据源和可靠的接收器

对于一些输入数据源（比如Kafka），Spark Streaming可以对已经接收的数据进行确认。输入的数据首先被接收器（receivers ）所接收，然后存储到Spark中（默认情况下，数据保存到2个执行器中以便进行容错）。数据一旦存储到Spark中，接收器可以对它进行确认（比如，如果消费Kafka里面的数据时可以更新Zookeeper里面的偏移量）。这种机制保证了在接收器突然挂掉的情况下也不会丢失数据：因为数据虽然被接收，但是没有被持久化的情况下是不会发送确认消息的。所以在接收器恢复的时候，数据可以被原端重新发送。



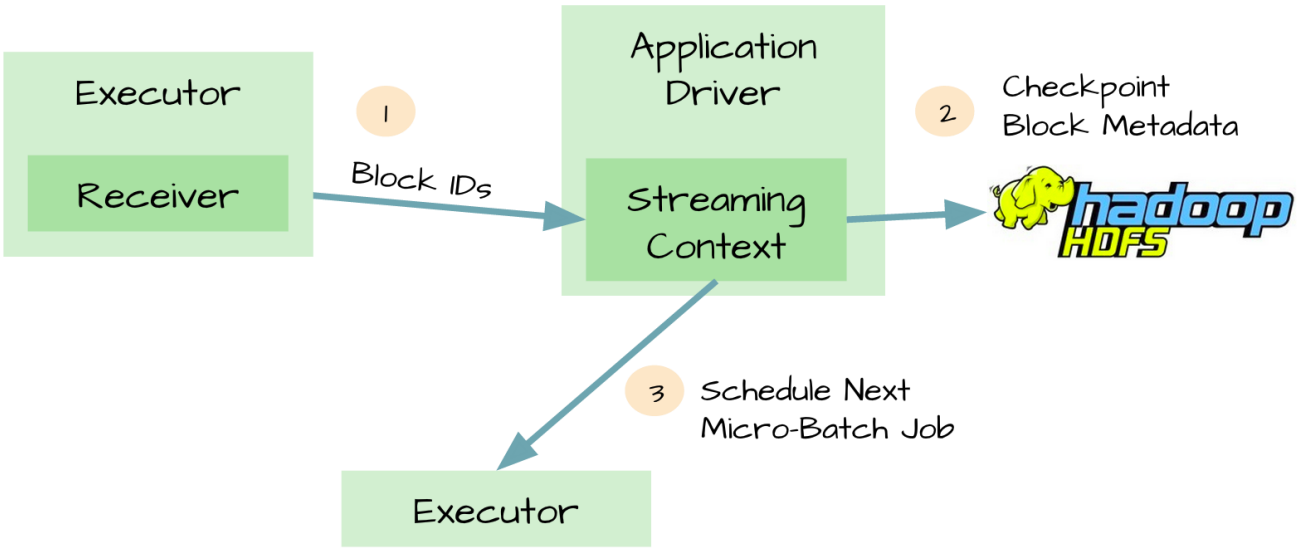
元数据持久化(Metadata checkpointing)

可靠的数据源和接收器可以让我们从接收器挂掉的情况下恢复（或者是接收器运行的Exectuor和服务器挂掉都可以）。但是更棘手的问题是，如果Driver挂掉如何恢复？对此开发者们引入了很多技术来让Driver从失败中恢复。其中一个就是对应用程序的元数据进行Checkpint。利用这个特性，Driver可以将应用程序的重要元数据持久化到可靠的存储中，比如HDFS、S3；然后Driver可以利用这些持久化的数据进行恢复。元数据包括：

1、配置；

2、代码；

3、那些在队列中还没有处理的batch（仅仅保存元数据，而不是这些batch中的数据）



由于有了元数据的Checkpint，所以Driver可以利用他们重构应用程序，而且可以计算出Driver挂掉的时候应用程序执行到什么位置。

可能存在数据丢失的场景

令人惊讶的是，即使是可靠的数据源、可靠的接收器和对元数据进行Checkpint，仍然不足以阻止潜在的数据丢失。我们可以想象出以下的糟糕场景：

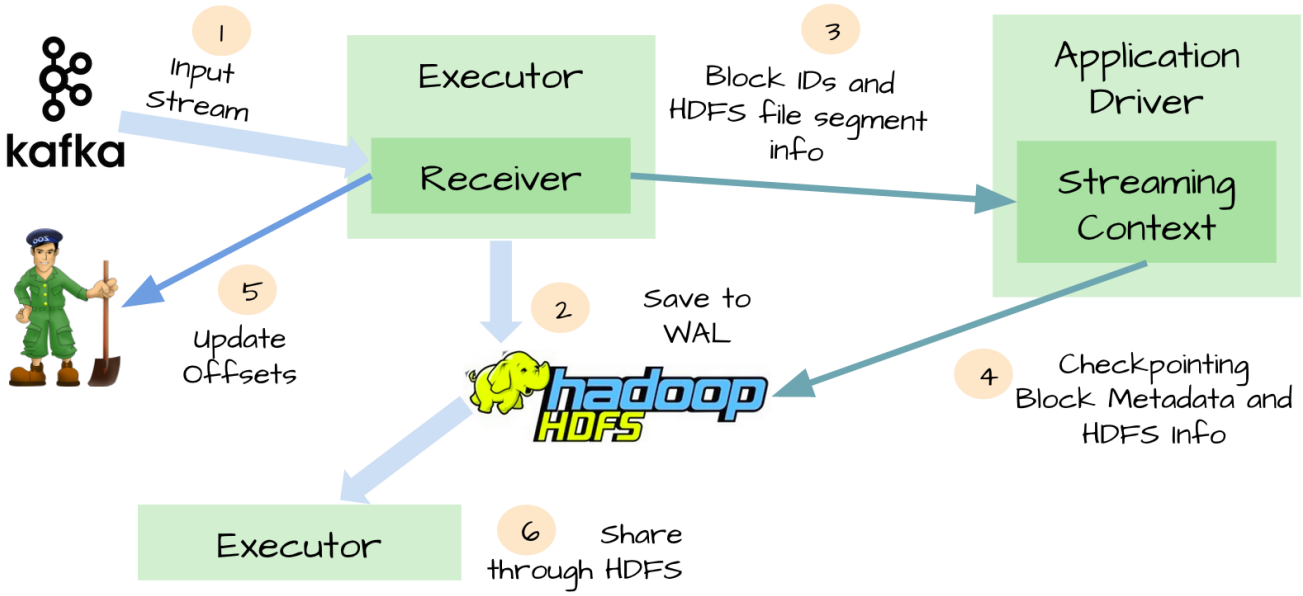
1、两个Exectuor已经从接收器中接收到输入数据，并将它缓存到Exectuor的内存中；
2、接收器通知输入源数据已经接收；
3、Exectuor根据应用程序的代码开始处理已经缓存的数据；
4、这时候Driver突然挂掉了；
5、从设计的角度看，一旦Driver挂掉之后，它维护的Exectuor也将全部被kill；
6、既然所有的Exectuor被kill了，所以缓存到它们内存中的数据也将被丢失。结果，这些已经通知数据源但是还没有处理的缓存数据就丢失了；
7、缓存的时候不可能恢复，因为它们是缓存在Exectuor的内存中，所以数据被丢失了。

这对于很多关键型的应用程序来说非常的糟糕，不是吗？

WAL（Write ahead log）

为了解决上面提到的糟糕场景，Spark Streaming 1.2开始引入了WAL机制。

启用了WAL机制，所以已经接收的数据被接收器写入到容错存储中，比如HDFS或者S3。由于采用了WAl机制，Driver可以从失败的点重新读取数据，即使Exectuor中内存的数据已经丢失了。在这个简单的方法下，Spark Streaming提供了一种即使是Driver挂掉也可以避免数据丢失的机制。

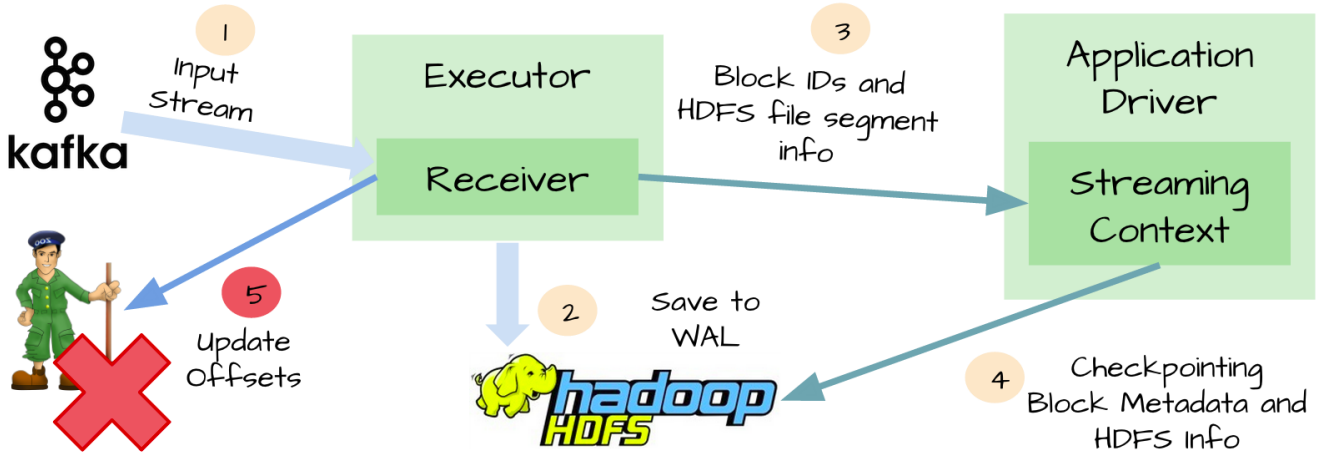


At-least-once语义

虽然WAL可以确保数据不丢失，它并不能对所有的数据源保证exactly-once语义。想象一下可能发生在Spark Streaming整合Kafka的糟糕场景。

1、接收器接收到输入数据，并把它存储到WAL中；

2、接收器在更新Zookeeper中Kafka的偏移量之前突然挂掉了；



3、Spark Streaming假设输入数据已成功收到（因为它已经写入到WAL中），然而Kafka认为数据被没有被消费，因为相应的偏移量并没有在Zookeeper中更新；

4、过了一会，接收器从失败中恢复；

5、那些被保存到WAL中但未被处理的数据被重新读取；

6、一旦从WAL中读取所有的数据之后，接收器开始从Kafka中消费数据。因为接收器是采用Kafka的High-Level Consumer API实现的，它开始从Zookeeper当前记录的偏移量开始读取数据，但是因为接收器挂掉的时候偏移量并没有更新到Zookeeper中，所有有一些数据被处理了2次。

WAL的缺点

除了上面描述的场景，WAL还有其他两个不可忽略的缺点:

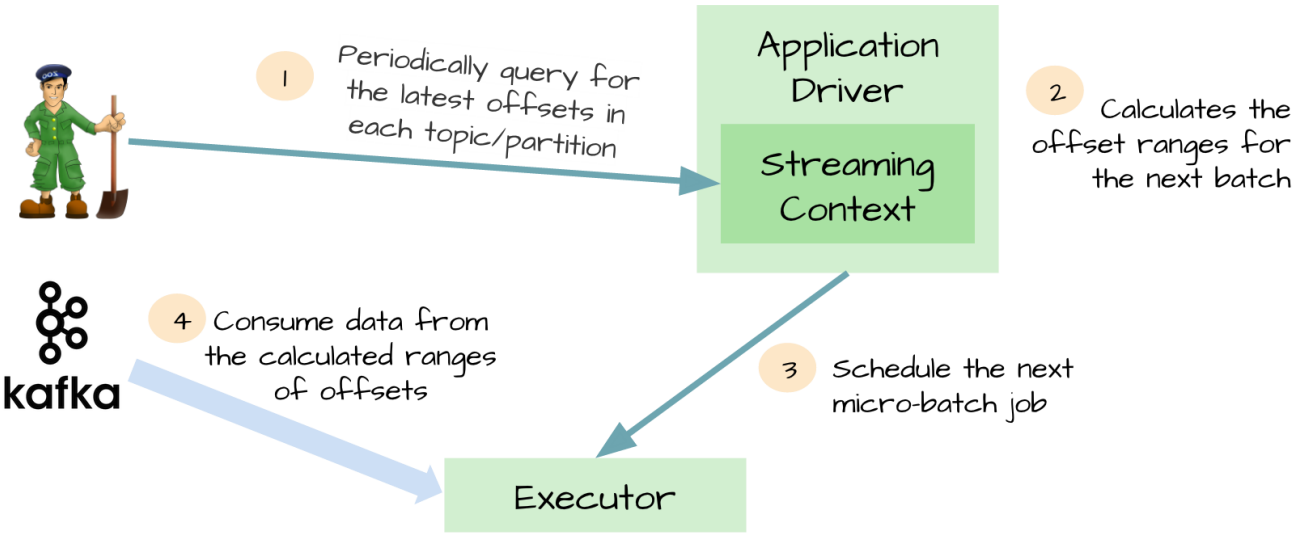
1、WAL减少了接收器的吞吐量，因为接受到的数据必须保存到可靠的分布式文件系统中。

2、对于一些输入源来说，它会重复相同的数据。比如当从Kafka中读取数据，你需要在Kafka的brokers中保存一份数据，而且你还得在Spark Streaming中保存一份。

Kafka direct API

为了解决由WAL引入的性能损失，并且保证 exactly-once 语义，Spark Streaming 1.3中引入了名为Kafka direct API。

这个想法对于这个特性是非常明智的。Spark driver只需要简单地计算下一个batch需要处理Kafka中偏移量的范围，然后命令Spark Exectuor直接从Kafka相应Topic的分区中消费数据。换句话说，这种方法把Kafka当作成一个文件系统，然后像读文件一样来消费Topic中的数据。



在这个简单但强大的设计中:

1、不再需要Kafka接收器，Exectuor直接采用Simple Consumer API从Kafka中消费数据。

2、不再需要WAL机制，我们仍然可以从失败恢复之后从Kafka中重新消费数据；

3、exactly-once语义得以保存，我们不再从WAL中读取重复的数据。

本文翻译至：Recent Evolution of Zero Data Loss Guarantee in Spark Streaming With Kafka：http://getindata.com/blog/post/recent-evolution-of-zero-data-loss-guarantee-in-spark-streaming-with-kafka/

Kafka Spark

« 转发微博有机会获取《Spark大数据分析实战》

自定义Spark Streaming接收器(Receivers) »

Apache Spark 2.3 重要特性介绍	Waterdrop：构建在Spark之上的简单高效数据处理系统	在 Apache Spark 中使用 UDF	Apache Spark SQL自适应执行实践

Apache Spark 2.3 重要特性介绍 | Waterdrop：构建在Spark之上的简单高效数 | 在 Apache Spark 中使用 UDF | Apache Spark SQL自适应执行实践

HiveServer2(Spark ThriftServer)自定义权限认证	如何在 Hadoop 2.2.0 环境下使用 Spark 2.2.x	Spark作业如何在无管理权限的集群部署Python或JDK	Spark + jupyter notebook出现图像无法显示问题解决

HiveServer2(Spark ThriftServer)自定义权限 | 如何在 Hadoop 2.2.0 环境下使用 Spark 2.2.x | Spark作业如何在无管理权限的集群部署Python | Spark + jupyter notebook出现图像无法

下面文章您可能感兴趣

- Spark 1.X 大数据平台V2百度网盘下载[完整版]
- Apache Arrow：内存列式的数据结构标准
- 在 Apache Spark 中使用 UDF
- Tunnello：免费的浏览器翻墙插件
- Linux安装软件依赖问题解决办法
- C和C++结构体的区别
- 《Apache Spark 2.0: Faster, Easier, and Smarter》ppt下载
- Akka学习笔记：ActorSystem(调度)
- SQL Joins可视化解释
- Flink是如何与YARN进行交互的
- Spark Streaming性能调优详解
- 在Tachyon运行Spark应用程序
- Spark和Hadoop作业之间的区别
- HDFS ls命令按照时间排序(sort by time)
- 数据结构：堆
- Apache Pulsar：雅虎开发的企业级发布订阅消息系统
- Spark函数讲解：aggregateByKey
- 六种使用Linux命令发送带附件的邮件
- MathJax:在浏览器上显示LaTeX等数学公式的JS引擎
- 设置SBT的日志级别

发表我的评论

写点什么吧

表情 本博客评论系统带有自动识别垃圾评论功能，请写一些有意义的评论，谢谢！ 提交评论

(16)个小伙伴在吐槽

博主 你好，请问采用DirectAPI的话 在取消息数据的时候就相当于直连broker了，并没有先去zookeeper获取我应该去连哪个broker，这样是不是会导致kafka挂了一台在用的broker的话 我就消费不到数据了，相当于抛弃了Kafka 的HA。

欲风 2017-07-05 23:59 回复

请教下博主，这两种接受方式下，不考虑wals下，从kafka读取到的数据是直接保存在executor内存中么，如果一个时间间隔下读取到的数据超过executor的内存，会怎么样?

itpudge 2016-12-13 16:12 回复

如果你是使用基于接收器的Kafka Consumer，它是把接收到的消息直接存储在executor内存的，如果超过了executor内存，executor会出现OOM而挂掉。但是使用 direct API就不会有这个问题。

w397090770 2016-12-13 17:12 回复

谢谢博主，我看sparkstreaming有三种从kafka读取的api包括：KafkaReceiver，ReliableKafkaReceiver，DirectKafka。为什么博主认为direct api的话为什么不会OOM，如果kafka的每个partition的数据量超过了executor的内存也会OOM吧？？还有，您觉得生产中如何去规避这种问题，谢谢

itpudge 2016-12-13 17:26 回复

direct api是边接收数据边处理的。如果真出现OOM也只能适当加大Exectuor的内存，避免把那些可以不放在内存的数据cache到内存。

w397090770 2016-12-13 17:36 回复

我看这文章direct api图里面写得是consume data from the caculated ranges of offsets，边接受边处理？不是一个micro batch才做为一个RDD处理么

itpudge 2016-12-14 10:40

博主 有时间http://spark.apache.org/docs/latest/streaming-kafka-0-10-integration.html 给讲讲这个spark streaming +kafka0.10的新特性呗！ 看着英文看的似懂非懂有点吃力

欢乐豆 2016-11-29 22:25 回复

好的，我找个时间翻译一下吧

w397090770 2016-11-30 08:59 回复

非常感谢博主 😺

欢乐豆 2016-11-30 10:03 回复

博主您这图是用什么工具画的?

spoofer 2016-05-30 21:37 回复

PowerPoint就可以做出来

w397090770 2016-05-31 09:59 回复

楼主你好： 关于1、不再需要Kafka接收器，Exectuor直接采用Simple Consumer API从Kafka中消费数据。 2、不再需要WAL机制，我们仍然可以从失败恢复之后从Kafka中重新消费数据； 能不能介绍一下是怎么恢复？ 我使用Kafka direct API来消费数据，运行过程中暴力kill掉任务，重启任务后数据并没有从上次kill掉的时间开始消费，而是从最后开始消费（kafka最新数据）

游乐场. 2016-03-24 14:57 回复

你需要去设置Checkpint的 ssc.checkpoint(checkpointDirectory) StreamingContext需要使用 StreamingContext.getOrCreate 创建

w397090770 2016-03-24 16:08 回复

您好，我想问一下，spark streaming如果使用checkpoint那么产生的log有没有自动清理的方法，还是需要手动或者写脚本去清理?

大志 2016-03-18 11:28 回复

spark streaming如果使用checkpoint那么产生的log只会最多保留最近的十个文件，其余的会被Spark自动清除。

w397090770 2016-03-18 15:35 回复

多谢您的回复！我的streaming程序一直在跑，checkpoint目录下生成了很多目录，但是都是空的，也没有找到您说的保留的是个文件，这里是否需要配置哪些参数？多谢啦~~

大志 2016-03-22 17:32 回复