链滴

Spark Streaming 管理 Kafka 偏移量

作者：UFO

原文链接：https://ld246.com/article/1506871057309

来源网站：链滴

前言

为了让Spark Streaming消费kafka的数据不丢数据，可以创建Kafka Direct DStream，由Spark Streaming自己管理offset，并不是存到zookeeper。启用Spark Streaming的 checkpoints是存储偏移量最简单方法，因为它可以在Spark的框架内轻松获得。 checkpoints将应用程序的状态保存到HDFS，便在故障时可以恢复。如果发生故障，Spark Streaming应用程序可以从checkpoints偏移范围读取息。 但是，Spark Streaming checkpoints在应用程序挂掉或者重启无法恢复，因此不是非常可靠特别是如果您将此机制用于关键生产应用程序，另外，基于zookeeper的offset可视化工具将无法使。我们不建议通过Spark checkpoints来管理偏移量。因此本文将手动存储offset到zookeeper，完自我掌控offset。

从ZK获取offset

● 创建ZKClient，API有好几个，最后用带序列化参数的，不然保存offset的时候容易出现乱码。

```
 val zkClient = new ZkClient("192.168.1.225:2181", 60000, 60000, new ZkSerializer {
    override def serialize(data: Object): Array[Byte] = {
      try {
        return data.toString().getBytes("UTF-8")
      } catch {
        case e: ZkMarshallingError => return null
      }
    }
    override def deserialize(bytes: Array[Byte]): Object = {
      try {
        return new String(bytes, "UTF-8")
      } catch {
        case e: ZkMarshallingError => return null
      }
    }
  })
```

● 查看该groupId在该topic下是否有消费记录，如果有，肯定在对应目录下会有分区数，children大0则有记录。

```
  val topicDirs = new ZKGroupTopicDirs(groupId, topic)
  val zkTopicPath = s"${topicDirs.consumerOffsetDir}"
  val topics = Set(topic)
  val children = zkClient.countChildren(s"${topicDirs.consumerOffsetDir}")
```

在有记录的情况下，去拿具体的offset

```
 if (children > 0) {
    var fromOffsets: Map[TopicAndPartition, Long] = Map()
    //---get partition leader begin----
    val topicList = List(topic)
    val req = new TopicMetadataRequest(topicList, 0)
    //得到该topic的一些信息，比如broker,partition分布情况
    val getLeaderConsumer = new SimpleConsumer("192.168.1.225", 9092, 10000, 10000, "Off
etLookup")
    // brokerList的host 、brokerList的port、过期时间、过期时间
```

```
    val res = getLeaderConsumer.send(req)
    //TopicMetadataRequest   topic broker partition 的一些信息
    val topicMetaOption = res.topicsMetadata.headOption
    val partitions = topicMetaOption match {
      case Some(tm) =>
        tm.partitionsMetadata.map(pm => (pm.partitionId, pm.leader.get.host)).toMap[Int, Stri
g]
      case None =>
        Map[Int, String]()
    }
    for (i <- 0 until children) {
      val partitionOffset = zkClient.readData[String](s"${topicDirs.consumerOffsetDir}/${i}")
      val tp = TopicAndPartition(topic, i)
      //---additional begin-----
      val requestMin = OffsetRequest(Map(tp -> PartitionOffsetRequestInfo(OffsetRequest.Earl
estTime, 1)))
      // -2,1
      val consumerMin = new SimpleConsumer(partitions(i), 9092, 10000, 10000, "getMinOffse
")
      val curOffsets = consumerMin.getOffsetsBefore(requestMin).partitionErrorAndOffsets(tp)
offsets
      var nextOffset = partitionOffset.toLong
      if (curOffsets.length > 0 && nextOffset < curOffsets.head) {
        //如果下一个offset小于当前的offset
        nextOffset = curOffsets.head
      }
      //---additional end-----
      fromOffsets += (tp -> nextOffset) //将不同 partition 对应的 offset 增加到 fromOffsets 中
    }//这个会将 kafka 的消息进行 transform，最终 kafak 的数据都会变成 (topic_name, message)
这样的 tuple
    val messageHandler = (mmd: MessageAndMetadata[String, String]) => (mmd.topic, mmd
message())
```

● 注意：在zookeeper里存储的offset有可能在kafka里过期了，所以要拿kafka最小的offset和zooke per里的offset比较一下。

创建DStream

● 接下来就可以创建Kafka Direct DStream了，前者是从zookeeper拿的offset，后者是直接从最新开始（第一次消费）。

```
    kafkaStream = KafkaUtils.createDirectStream[String, String, StringDecoder, StringDecoder,
(String, String)](ssc, kafkaParams, fromOffsets, messageHandler)
  } else {
    kafkaStream = KafkaUtils.createDirectStream[String, String, StringDecoder, StringDecoder]
ssc, kafkaParams, topics)
  }
```

● 最后就是处理RDD，保存Offset。

```
  kafkaStream.foreachRDD(rdd => {
    if (!rdd.isEmpty) {
```

```
        doSomething....
          saveOffset(path,edd)
      }
    })

  private def saveOffset(path:String,rdd: RDD[(String, String)]) = {
    val offsetRanges = rdd.asInstanceOf[HasOffsetRanges].offsetRanges
    for (o <- offsetRanges) {
      ZkUtils.updatePersistentPath(zkClient, s"${path}/${o.partition}", String.valueOf(o.untilOffse
))
    }
  }
```

原文链接：Spark Streaming 管理 Kafka 偏移量