最近，忙于开发完善flink平台，并且使用我们的平台去支持一些复杂的业务，比如用户画像处理等。遇见了很多bug和性能点，后面陆续出文章给大家解析。

书归正传，大家都知道spark streaming是微批批处理，而Structured streaming在2.3以前也是批处理，在2.3引入了连续处理的概念，延迟大幅度降低值～1ms，但是还有诸多限制，这点比flink差了许多。

至于低延迟的测试，建议本文使用本文代码去测试，kafka source->kafka sink，这样便于观察延迟。

连续处理是Spark 2.3中引入的一种新的实验版本流执行模式，可实现极低（~1 ms）端到端延迟，并且具有至少一次处理容错保证。 structured streaming的连续处理模式与微批处理模式进行比较，微批处理引擎可以实现一次性保证，但微批处理最好仅可实现约100ms的延迟。 对于某些类型的查询（在下面讨论），可以选择在不修改应用代码的情况下运行该模式（即，不更改DataFrame / Dataset操作）。

要在连续处理模式下运行支持的查询，您只需指定一个连续触发器，并将所需的checkpoint间隔作为参数。 例如浪尖的demo如下：

object ContinuousProcessing {

 def main(args: Array[String]): Unit = {

   val sparkConf = new SparkConf().setAppName(this.getClass.getName).setMaster("yarn-client")

     .set("yarn.resourcemanager.hostname", "mt-mdh.local")

     .set("spark.executor.instances","2")

     .set("spark.default.parallelism","4")

     .set("spark.sql.shuffle.partitions","4")

     .setJars(List("/Users/meitu/Desktop/sparkjar/bigdata.jar"

       ,"/opt/jars/spark-streaming-kafka-0-10\_2.11-2.3.1.jar"

       ,"/opt/jars/kafka-clients-0.10.2.2.jar"

       ,"/opt/jars/kafka\_2.11-0.10.2.2.jar"

       ,"/opt/jars/spark-sql-kafka-0-10\_2.11-2.0.2.jar"))

   val spark = SparkSession

     .builder

     .appName("StructuredKafkaWordCount")

     .config(sparkConf)

     .getOrCreate()

   spark

     .readStream

     .format("kafka")

     .option("kafka.bootstrap.servers", "mt-mdh.local:9093")

     .option("subscribe", "StructuredSource")

     .load()

     .selectExpr("CAST(key AS STRING)", "CAST(value AS STRING)")

     .writeStream

     .format("kafka")

     .option("kafka.bootstrap.servers", "mt-mdh.local:9093")

     .option("topic", "StructuredSink")

     .option("checkpointLocation","/sql/checkpoint")

**.trigger(Trigger.Continuous("1 second"))  // only change in query**

     .start()

     .awaitTermination()

 }

}

checkpoint 间隔为1秒意味着连续处理引擎将每秒记录查询的进度。 生成的checkpoint采用与微批处理引擎兼容的格式，因此可以使用任何触发器重新启动任何查询。 例如，假如查询支持微批处理和连续处理，那么实际上也可以用连续处理触发器去启动微批处理触发器，反之亦然。

请注意，无论何时切换到连续模式，都将获得至少一次的容错保证。

支持的查询

从Spark 2.3开始，连续处理模式仅支持以下类型的查询。

Operations：在连续模式下仅支持dataset/dataframe的类似于map的操作，即支持projection（select，map，flatMap，mapPartitions等）和selection（where，filter等）。

除了聚合函数（因为尚不支持聚合），current\_timestamp（）和current\_date（）（使用时间的确定性计算具有挑战性）之外，支持所有SQL函数。

Sources

Kafka Source：支持所有操作。

Rate source：适合测试。只有连续模式支持的选项是numPartitions和rowsPerSecond。

Sinks

Kafka sink：支持所有选项。

Memory sink：适合调试。

Console sink：适合调试。支持所有操作。请注意，控制台将打印你在连续触发器中指定的每个checkpoint间隔。

更详细的关于sink和source信息，请参阅输入源和输出接收器部分的官网。虽然控制台接收器非常适合测试，但是使用Kafka作为源和接收器可以最好地观察到端到端的低延迟处理。

注意事项

连续处理引擎启动多个长时间运行的任务，这些任务不断从源中读取数据，处理数据并连续写入接收器。 查询所需的任务数取决于查询可以并行从源读取的分区数。 因此，在开始连续处理查询之前，必须确保群集中有足够的核心并行执行所有任务。 例如，如果您正在读取具有10个分区的Kafka主题，则群集必须至少具有10个核心才能使查询正常执行。

停止连续处理流可能会产生虚假的任务终止警告。 这些可以安全地忽略。

目前没有自动重试失败的任务。 任何失败都将导致查询停止，并且需要从检查点手动重新启动。(深受其害，kafka topic没数据流入也会挂掉的)

---------------------

作者：Spark高级玩法

来源：CSDN

原文：https://blog.csdn.net/rlnLo2pNEfx9c/article/details/83155380

版权声明：本文为博主原创文章，转载请附上博文链接！