Kafka and Spark Streaming in practice @TalkingData

@chico2011

Architecture

- http://spark.apache.org/docs/latest/streaming-programming-guide.html

Lessons learned from operating Kafka

- choose the Kafka_2.10-0.8.1.1 or above, because of Kafka-0.7 has only Scala_2.8 package, hasn't Scala_2.10 version
- DON'T delete topic under Kafka-0.8.1.1
- Kafka-0.8.2 has topic delete script
- bin/kafka-producer-perf-test.sh for performance testing

MessageSizeTooLarge Exception

- $ bin/kafka-topics.sh --zookeeper 10.10.0.56:2181,10.10.0.57:2181,10.10.0.58:2181/kafka/product --alter --topic trackinfo --config max.message.bytes=10000000

- vim config/producer.properties

```
# specifies whether the messages are sent asynchronously (async) or synchronously (sync)
producer.type=sync
# specify the compression codec for all data generated: none , gzip, snappy.
# the old config values work as well: 0, 1, 2 for none, gzip, snappy, respectivally
#compression.codec=none
compression.codec=1
# allow topic level compression
#compressed.topics=
compressed.topics=ipinfo
```

- split the large message by a setting size

Spark Streaming tips

- ssc.textFileStream(path)
  - just supports file moving, don't supports file copying
- received massage is empty
  - need a DStream output operation: print(), saveAsXX, foreachRDD

- foreachRDD pattern

```
class KafkaWriter[T: ClassTag](@transient rdd: RDD[T]) extends Serializable with Logging {

  def write2Kafka[K, V](serializerFunc: T => KeyedMessage[K, V]) {
    val props = new Properties()
    props.load(this.getClass.getResourceAsStream("/producer-defaults.properties"))
    val broadcastedConfig = rdd.sparkContext.broadcast(props)

    rdd.foreachPartition(events => {
      val producer: Producer[K, V] = new Producer[K, V](new ProducerConfig(broadcastedConfig.value))
      try {
        producer.send(events.map(serializerFunc).toArray: _*)
        logDebug("Data sent successfully to Kafka")
      } catch {
        case e: Exception =>
          logError("Failed to send data to Kafka", e)
          throw e
      } finally {
        producer.close()
        logDebug("Kafka Producer closed successfully")
      }
    })

  }
}
```

- foreachRDD + GenericObjectPool, but failed by serializer exception

```
// Define the actual data flow of the streaming job
kafkaStream.map { case bytes =>
  numInputMessages += 1
  converter.value.invert(bytes) match {
    case Success(tweet) => tweet
    case Failure(e) =>
  }
}.foreachRDD(rdd => {
  rdd.foreachPartition(partitionOfRecords => {
    val p = producerPool.value.borrowObject()
    partitionOfRecords.foreach { case tweet: Tweet =>
      val bytes = converter.value.apply(tweet)
      p.send(bytes)
      numOutputMessages += 1
    }
    producerPool.value.returnObject(p)
  })
})
```

```
private def createKafkaProducerPool(brokerList: String, topic: String): GenericObjectPool[KafkaProducerApp] = {
  val producerFactory = new BaseKafkaProducerAppFactory(brokerList, defaultTopic = Option(topic))
  val pooledProducerFactory = new PooledKafkaProducerAppFactory(producerFactory)
  val poolConfig = {
    val c = new GenericObjectPoolConfig
    val maxNumProducers = 10
    c.setMaxTotal(maxNumProducers)
    c.setMaxIdle(maxNumProducers)
    c
  }
  new GenericObjectPool[KafkaProducerApp](pooledProducerFactory, poolConfig)
}
```

- DStream to parquet
  - DStream window operation
  - DStream + sqlContext

```
case class Gis(lat: Double, long: Double)
val sqlContext = new SQLContext(ssc.sparkContext)
import sqlContext._
kafkaStream.foreachRDD(rdd => {
  if (rdd.count == 0) println("### RDD is empty! ###")
  if (rdd.count > 0) {
    val outrdd = rdd.map{bytes =>
      val bytearray = bytes.split("\t")
      val lat = bytearray(2)
      val lng = bytearray(3)
      println(s"### print bytes $lat and $lng ###")
      Gis(bytearray(2).toDouble, bytearray(3).toDouble)
    }
    outrdd.saveAsParquetFile("/extract/chico/stream/stream2parquet/" + folderTimestamp + ".parquet")
  }
})
```

References

- http://www.michael-noll.com/blog/2014/10/01/kafka-spark-streaming-integration-example-tutorial/#kafka-integration
- https://github.com/harishreedharan/spark/blob/Kafka-output/external/kafka/src/main/scala/org/apache/spark/streaming/kafka/KafkaOutputWriter.scala