使用Apache Spark将数据写入ElasticSearch

ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

本文并不打算介绍ElasticSearch的概念，安装部署等知识，而是直接介绍如何使用Apache Spark将数据写入到ElasticSearch中。本文使用的是类库是elasticsearch-hadoop，其从2.1版本开始提供了内置 支持Apache Spark的功能，在使用elasticsearch-hadoop之前，我们需要引入依赖：

```
<dependency>
   <groupId>org.elasticsearch</groupId>
   <artifactId>elasticsearch-hadoop</artifactId>
   <version>2.3.4</version>
</dependency>
```

为了方便，本文直接在spark-shell中操作ElasticSearch。在此之前，我们需要在$SPARK_HOME/conf/spark-default.conf文件中加入以下配置：

```
spark.es.nodes       www.iteblog.com
spark.es.port        9200
```

其中spark.es.nodes指定你es集群的机器列表，但是不需要把你集群所有的节点都列在里面；spark.es.port表示集群HTTP端口。之所以要加上spark前缀是因为Spark通过从文件里面或者命令行里面读取配置参数只会加载spark开头的，其他的参数将会被忽略。之后elasticsearch-hadoop会把spark前缀去掉。

如果你直接将代码写入文件，那么你可以在初始化SparkContext之前设置好ElasticSearch相关的参数，如下：

```
import org.apache.spark.SparkConf

val conf = new SparkConf().setAppName("iteblog").setMaster(master)
conf.set("es.nodes", "www.iteblog.com")
conf.set("es.port", "9200")
```

```
conf.set("es.index.auto.create", "true")
```

在写入数据之前，先导入org.elasticsearch.spark._包，这将使得所有的RDD拥有saveToEs方法。下面我将一一介绍将不同类型的数据写入ElasticSearch中。

将Map对象写入ElasticSearch

```
scala> import org.elasticsearch.spark._
import org.elasticsearch.spark._

scala> val numbers = Map("one" -> 1, "two" -> 2, "three" -> 3)
numbers: scala.collection.immutable.Map[String,Int] = Map(one -> 1, two -> 2, three -> 3)

scala> val airports = Map("OTP" -> "Otopeni", "SFO" -> "San Fran")
airports: scala.collection.immutable.Map[String,String] = Map(OTP -> Otopeni, SFO -> San Fran
)

scala> sc.makeRDD(Seq(numbers, airports)).saveToEs("iteblog/docs")
```

上面构建了两个Map对象，然后将它们写入到ElasticSearch中；其中saveToEs里面参数的iteblog表示索引(indexes)，而docs表示type。然后我们可以通过下面URL查看iteblog这个index的属性：

```
curl -XGET :9200/iteblog

{
   "iteblog": {
      "aliases": { },
      "mappings": {
         "docs": {
            "properties": {
               "SFO": {
                  "type": "string"
               },
               "arrival": {
                  "type": "string"
               },
               "one": {
                  "type": "long"
               },
```

```
            "three": {
               "type": "long"
            },
            "two": {
               "type": "long"
            }
         }
      }
   },
   "settings": {
      "index": {
         "creation_date": "1470805957888",
         "uuid": "HNIcGZ69Tf6qX3XVccwKUg",
         "number_of_replicas": "1",
         "number_of_shards": "5",
         "version": {
            "created": "2030499"
         }
      }
   },
   "warmers": { }
}
}
```


同时使用下面URL搜索出所有的documents：

:9200/iteblog/docs/_search


```
{
   "took": 2,
   "timed_out": false,
   "_shards": {
      "total": 5,
      "successful": 5,
      "failed": 0
   },
   "hits": {
      "total": 2,
      "max_score": 1,
      "hits": [
         {
            "_index": "iteblog",
            "_type": "docs",
```

```
            "_id": "AVZy3d5sJfxPRwCjtWM9",
            "_score": 1,
            "_source": {
               "one": 1,
               "two": 2,
               "three": 3
            }
         },
         {
            "_index": "iteblog",
            "_type": "docs",
            "_id": "AVZy3d5sJfxPRwCjtWM-",
            "_score": 1,
            "_source": {
               "arrival": "Otopeni",
               "SFO": "San Fran"
            }
         }
      ]
   }
}
```


将case class对象写入ElasticSearch

我们还可以将Scala中的case
class对象写入到ElasticSearch；Java中可以写入JavaBean对象，如下：

```
scala> case class Trip(departure: String, arrival: String)
defined class Trip

scala> val upcomingTrip = Trip("OTP", "SFO")
upcomingTrip: Trip = Trip(OTP,SFO)

scala> val lastWeekTrip = Trip("MUC", "OTP")
lastWeekTrip: Trip = Trip(MUC,OTP)

scala> val rdd = sc.makeRDD(Seq(upcomingTrip, lastWeekTrip))
rdd: org.apache.spark.rdd.RDD[Trip] = ParallelCollectionRDD[1] at makeRDD at <console>:37

scala> rdd.saveToEs("iteblog/class")
```

上面的代码片段将upcomingTrip和lastWeekTrip写入到名为iteblog的_index中，type是class。上面都是通过隐式转换才使得rdd拥有saveToEs方法。elasticsearch-hadoop还提供显式方法来把RDD写入到ElasticSearch中，如下：

```
scala> import org.elasticsearch.spark.rdd.EsSpark
import org.elasticsearch.spark.rdd.EsSpark

scala> val rdd = sc.makeRDD(Seq(upcomingTrip, lastWeekTrip))
rdd: org.apache.spark.rdd.RDD[Trip] = ParallelCollectionRDD[0] at makeRDD at <console>:34

scala> EsSpark.saveToEs(rdd, "spark/docs")
```

将Json字符串写入ElasticSearch

我们可以直接将Json字符串写入到ElasticSearch中，如下：

```
scala> val json1 = """{"id" : 1, "blog" : "www.iteblog.com", "weixin" : "iteblog_hadoop"}"""
json1: String = {"id" : 1, "blog" : "www.iteblog.com", "weixin" : "iteblog_hadoop"}

scala> val json2 = """{"id" : 2, "blog" : "books.iteblog.com", "weixin" : "iteblog_hadoop"}"""
json2: String = {"id" : 2, "blog" : "books.iteblog.com", "weixin" : "iteblog_hadoop"}

scala> sc.makeRDD(Seq(json1, json2)).saveJsonToEs("iteblog/json")
```

动态设置插入的type

上面的示例都是将写入的type写死。有很多场景下同一个Job中有很多类型的数据，我们希望一次就可以将不同的数据写入到不同的type中，比如属于book的信息全部写入到type为book里面；而属于cd的信息全部写入到type为cd里面。很高兴的是elasticsearch-hadoop为我们提供了这个功能，如下：

```
scala> val game = Map("media_type"->"game","title" -> "FF VI","year" -> "1994")
game: scala.collection.immutable.Map[String,String] = Map(media_type -> game, title -> FF VI, year -> 1994)

scala> val book = Map("media_type" -> "book","title" -> "Harry Potter","year" -> "2010")
book: scala.collection.immutable.Map[String,String] = Map(media_type -> book, title -> Harry Potter, year -> 2010)

scala> val cd = Map("media_type" -> "music","title" -> "Surfing With The Alien")
```

```
cd: scala.collection.immutable.Map[String,String] = Map(media_type -> music, title -> Surfing With The Alien)

scala> sc.makeRDD(Seq(game, book, cd)).saveToEs("iteblog/{media_type}")
```

type是通过{media_type}通配符设置的，这个在写入的时候可以获取到，然后将不同类型的数据写入到不同的type中。

自定义id

在ElasticSearch中，_index/_type/_id的组合可以唯一确定一个Document。如果我们不指定id的话，ElasticSearch将会自动为我们生产全局唯一的id，自动生成的ID有20个字符长如下：

```
{
   "_index": "iteblog",
   "_type": "docs",
   "_id": "AVZy3d5sJfxPRwCjtWM-",
   "_score": 1,
   "_source": {
      "arrival": "Otopeni",
      "SFO": "San Fran"
   }
}
```

很显然，这么长的字符串没啥意义，而且也不便于我们记忆使用。不过我们可以在插入数据的时候手动指定id的值，如下：

```
scala> val otp = Map("iata" -> "OTP", "name" -> "Otopeni")
otp: scala.collection.immutable.Map[String,String] = Map(iata -> OTP, name -> Otopeni)

scala> val muc = Map("iata" -> "MUC", "name" -> "Munich")
muc: scala.collection.immutable.Map[String,String] = Map(iata -> MUC, name -> Munich)

scala> val sfo = Map("iata" -> "SFO", "name" -> "San Fran")
sfo: scala.collection.immutable.Map[String,String] = Map(iata -> SFO, name -> San Fran)

scala> val airportsRDD = sc.makeRDD(Seq((1, otp), (2, muc), (3, sfo)))

scala> airportsRDD.saveToEsWithMeta("iteblog/2015")
```

上面的Seq((1, otp), (2, muc), (3,
sfo))语句指定为各个对象指定了id值，分别为1、2、3。然后你可以通过/iteblog/2015/1
URL搜索到otp对象的值。我们还可以如下方式指定id：

```
scala> val json1 = """{"id" : 1, "blog" : "www.iteblog.com", "weixin" : "iteblog_hadoop"}"""
json1: String = {"id" : 1, "blog" : "www.iteblog.com", "weixin" : "iteblog_hadoop"}

scala> val json2 = """{"id" : 2, "blog" : "books.iteblog.com", "weixin" : "iteblog_hadoop"}"""
json2: String = {"id" : 2, "blog" : "books.iteblog.com", "weixin" : "iteblog_hadoop"}

scala> val rdd = sc.makeRDD(Seq(json1, json2))

scala> EsSpark.saveToEs(rdd, "iteblog/docs", Map("es.mapping.id" -> "id"))
```

上面通过es.mapping.id参数将对象中的id字段映射为每条记录的id。

自定义记录的元数据

我们甚至可以在写入数据的时候自定义记录的元数据，如下：

```
scala> import org.elasticsearch.spark.rdd.Metadata._
import org.elasticsearch.spark.rdd.Metadata._

scala> val otp = Map("iata" -> "OTP", "name" -> "Otopeni")
otp: scala.collection.immutable.Map[String,String] = Map(iata -> OTP, name -> Otopeni)

scala> val muc = Map("iata" -> "MUC", "name" -> "Munich")
muc: scala.collection.immutable.Map[String,String] = Map(iata -> MUC, name -> Munich)

scala> val sfo = Map("iata" -> "SFO", "name" -> "San Fran")
sfo: scala.collection.immutable.Map[String,String] = Map(iata -> SFO, name -> San Fran)

scala> val otpMeta = Map(ID -> 1, TTL -> "3h")

scala> val mucMeta = Map(ID -> 2, VERSION -> "23")

scala> val sfoMeta = Map(ID -> 3)
```

```
scala> val airportsRDD = sc.makeRDD(Seq((otpMeta, otp), (mucMeta, muc), (sfoMeta, sfo)))

scala> airportsRDD.saveToEsWithMeta("iteblog/2015")
```

上面代码片段分别为otp、muc和sfo设置了不同的元数据，这在很多场景下是非常有用的。

好了不早了，该洗洗睡，后面我将介绍如何使用Apache Spark读取ElasticSearch中的数据。