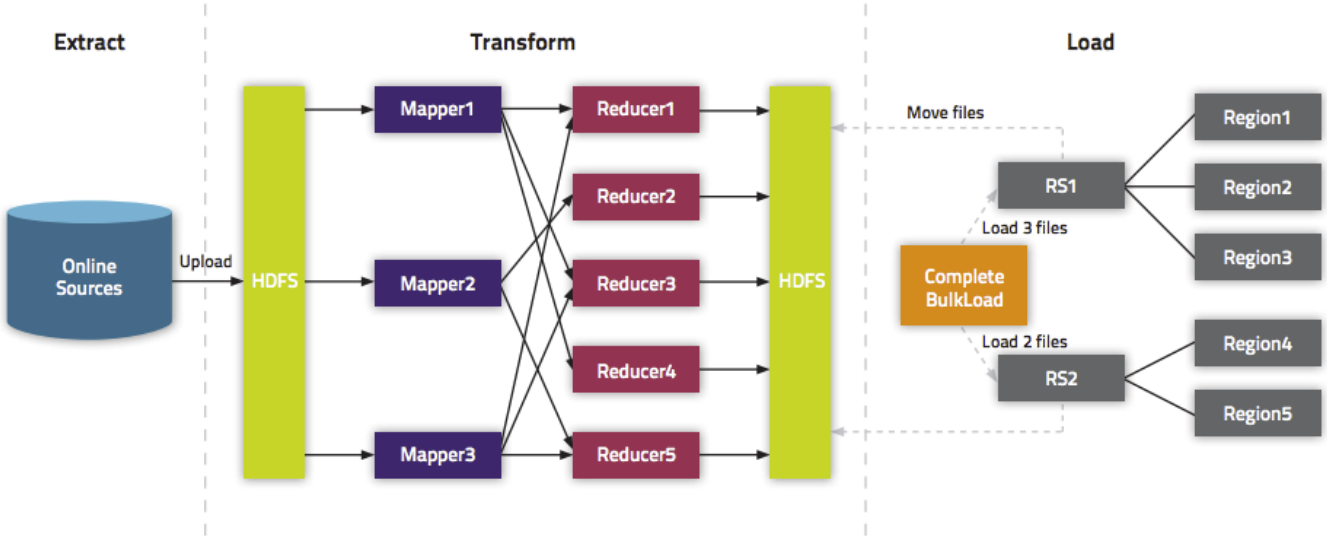
在Spark上通过BulkLoad快速将海量数据导入到Hbase

我们在《通过BulkLoad快速将海量数据导入到Hbase[Hadoop篇]》
文中介绍了一种快速将海量数据导入Hbase的一种方法，而本文将介绍如何在Spark上使用Scala编写快速导入数据到Hbase中的方法。这里将介绍两种方式：第一种使用Put普通的方法来倒数；第二种使用Bulk Load API。关于为啥需要使用Bulk Load本文就不介绍，更多的请参见《通过BulkLoad快速将海量数据导入到Hbase[Hadoop篇]》。



如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

使用org.apache.hadoop.hbase.client.Put来写数据

使用 org.apache.hadoop.hbase.client.Put 将数据一条一条写入Hbase中，但是和Bulk加载相比效率低下，仅仅作为对比。

```
import org.apache.spark._
import org.apache.spark.rdd.NewHadoopRDD
import org.apache.hadoop.hbase.{HBaseConfiguration, HTableDescriptor}
import org.apache.hadoop.hbase.client.HBaseAdmin
import org.apache.hadoop.hbase.mapreduce.TableInputFormat
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.HColumnDescriptor
import org.apache.hadoop.hbase.util.Bytes
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.HTable;

val conf = HBaseConfiguration.create()
```

```
val tableName = "/iteblog"
conf.set(TableInputFormat.INPUT_TABLE, tableName)

val myTable = new HTable(conf, tableName);
var p = new Put();
p = new Put(new String("row999").getBytes());
p.add("cf".getBytes(), "column_name".getBytes(), new String("value999").getBytes());
myTable.put(p);
myTable.flushCommits();
```

批量导数据到Hbase

批量导数据到Hbase又可以分为两种：（1）、生成Hfiles，然后批量导数据；
（2）、直接将数据批量导入到Hbase中。

批量将Hfiles导入Hbase

现在我们来介绍如何批量将数据写入到Hbase中，主要分为两步：
（1）、先生成Hfiles；
（2）、使用 org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles
将事先生成Hfiles导入到Hbase中。
实现的代码如下：

```
import org.apache.spark._
import org.apache.spark.rdd.NewHadoopRDD
import org.apache.hadoop.hbase.{HBaseConfiguration, HTableDescriptor}
import org.apache.hadoop.hbase.client.HBaseAdmin
import org.apache.hadoop.hbase.mapreduce.TableInputFormat
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.HColumnDescriptor
import org.apache.hadoop.hbase.util.Bytes
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.mapred.TableOutputFormat
import org.apache.hadoop.mapred.JobConf
import org.apache.hadoop.hbase.io.ImmutableBytesWritable
import org.apache.hadoop.mapreduce.Job
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat
import org.apache.hadoop.hbase.KeyValue
import org.apache.hadoop.hbase.mapreduce.HFileOutputFormat
import org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles
```

```
val conf = HBaseConfiguration.create()
val tableName = "iteblog"
val table = new HTable(conf, tableName)

conf.set(TableOutputFormat.OUTPUT_TABLE, tableName)
val job = Job.getInstance(conf)
job.setMapOutputKeyClass (classOf[ImmutableBytesWritable])
job.setMapOutputValueClass (classOf[KeyValue])
HFileOutputFormat.configureIncrementalLoad (job, table)

// Generate 10 sample data:
val num = sc.parallelize(1 to 10)
val rdd = num.map(x=>{
  val kv: KeyValue = new KeyValue(Bytes.toBytes(x), "cf".getBytes(), "c1".getBytes(), "value_xxx"
.getBytes() )
  (new ImmutableBytesWritable(Bytes.toBytes(x)), kv)
})

// Save Hfiles on HDFS
rdd.saveAsNewAPIHadoopFile("/tmp/iteblog", classOf[ImmutableBytesWritable], classOf[KeyV
alue], classOf[HFileOutputFormat], conf)

//Bulk load Hfiles to Hbase
val bulkLoader = new LoadIncrementalHFiles(conf)
bulkLoader.doBulkLoad(new Path("/tmp/iteblog"), table)
```

运行完上面的代码之后，我们可以看到Hbase中的iteblog表已经生成了10条数据，如下：

```
hbase(main):020:0> scan 'iteblog'
ROW                                COLUMN+CELL
 ₩x00₩x00₩x00₩x01                  column=cf:c1, timestamp=1425128075586, value=
value_xxx
 ₩x00₩x00₩x00₩x02                  column=cf:c1, timestamp=1425128075586, value=
value_xxx
 ₩x00₩x00₩x00₩x03                  column=cf:c1, timestamp=1425128075586, value=
value_xxx
 ₩x00₩x00₩x00₩x04                  column=cf:c1, timestamp=1425128075586, value=
value_xxx
 ₩x00₩x00₩x00₩x05                  column=cf:c1, timestamp=1425128075586, value=
value_xxx
 ₩x00₩x00₩x00₩x06                  column=cf:c1, timestamp=1425128075675, value=
value_xxx
```

```
 ₩x00₩x00₩x00₩x07                column=cf:c1, timestamp=1425128075675, value=
value_xxx
 ₩x00₩x00₩x00₩x08                column=cf:c1, timestamp=1425128075675, value=
value_xxx
 ₩x00₩x00₩x00₩x09                column=cf:c1, timestamp=1425128075675, value=
value_xxx
 ₩x00₩x00₩x00₩x0A                column=cf:c1, timestamp=1425128075675, value=
value_xxx
```

直接Bulk Load数据到Hbase

这种方法不需要事先在HDFS上生成Hfiles，而是直接将数据批量导入到Hbase中。与上面的例子相比只有微小的差别，具体如下：
将

```
rdd.saveAsNewAPIHadoopFile("/tmp/iteblog", classOf[ImmutableBytesWritable], classOf[KeyV
alue], classOf[HFileOutputFormat], conf)
```

修改成：

```
rdd.saveAsNewAPIHadoopFile("/tmp/iteblog", classOf[ImmutableBytesWritable], classOf[KeyV
alue], classOf[HFileOutputFormat], job.getConfiguration())
```

完整的实现如下：

```
import org.apache.spark._
import org.apache.spark.rdd.NewHadoopRDD
import org.apache.hadoop.hbase.{HBaseConfiguration, HTableDescriptor}
import org.apache.hadoop.hbase.client.HBaseAdmin
import org.apache.hadoop.hbase.mapreduce.TableInputFormat
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.HColumnDescriptor
import org.apache.hadoop.hbase.util.Bytes
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.mapred.TableOutputFormat
import org.apache.hadoop.mapred.JobConf
```

```
import org.apache.hadoop.hbase.io.ImmutableBytesWritable
import org.apache.hadoop.mapreduce.Job
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat
import org.apache.hadoop.hbase.KeyValue
import org.apache.hadoop.hbase.mapreduce.HFileOutputFormat
import org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles

val conf = HBaseConfiguration.create()
val tableName = "iteblog"
val table = new HTable(conf, tableName)

conf.set(TableOutputFormat.OUTPUT_TABLE, tableName)
val job = Job.getInstance(conf)
job.setMapOutputKeyClass (classOf[ImmutableBytesWritable])
job.setMapOutputValueClass (classOf[KeyValue])
HFileOutputFormat.configureIncrementalLoad (job, table)

// Generate 10 sample data:
val num = sc.parallelize(1 to 10)
val rdd = num.map(x=>{
  val kv: KeyValue = new KeyValue(Bytes.toBytes(x), "cf".getBytes(), "c1".getBytes(), "value_xxx"
.getBytes() )
  (new ImmutableBytesWritable(Bytes.toBytes(x)), kv)
})

// Directly bulk load to Hbase/MapRDB tables.
rdd.saveAsNewAPIHadoopFile("/tmp/iteblog", classOf[ImmutableBytesWritable], classOf[KeyV
alue], classOf[HFileOutputFormat], job.getConfiguration())
```

其他

在上面的例子中我们使用了 saveAsNewAPIHadoopFile
API来将数据写到HBase中；事实上，我们还可以通过使用 saveAsNewAPIHadoopDataset
API来实现同样的目标，我们仅仅需要将下面代码

```
rdd.saveAsNewAPIHadoopFile("/tmp/iteblog", classOf[ImmutableBytesWritable], classOf[KeyV
alue], classOf[HFileOutputFormat], job.getConfiguration())
```

修改成

```
job.getConfiguration.set("mapred.output.dir", "/tmp/iteblog")
rdd.saveAsNewAPIHadoopDataset(job.getConfiguration)
```

剩下的和和之前完全一致。