Day34_Spark操作Mysql和HBase

大数据-张军锋 Day34 Spark Mysql HBase Hbase

Spark连接Mysql

JdbcRDD通过实例化JdbcRDD对象来完成对关系型数据库中的数据作为spark数据源的一种方式

实例化的过程需要

- SparkContext
- 关系型数据库的连接
- sql必须要包含2个问号
- lowbound highbound分别和sql中的2个问号对应
- 结果转换函数：`r => r`，从关系型数据库里获取是数据是ResultSet，那在读取进来的rdd中希望他是什么类型就用这个函数来完成转换

数据库中的数据

id	name	title	age	country
1	盖伦	德玛西亚之力	22	德玛西亚
2	卡特琳娜	不祥之刃	20	诺克萨斯
3	亚索	疾风剑豪	25	艾欧尼亚
4	瑞雯	放逐之刃	23	诺克萨斯

```
package com.bd14.zjf.sparktest

import java.sql.DriverManager

import org.apache.spark.rdd.JdbcRDD
import org.apache.spark.{SparkConf, SparkContext}

object MysqlTest {

  val conf = new SparkConf().setMaster("local[*]").setAppName("spar
k读写mysql")
  val sc = SparkContext.getOrCreate(conf)

  val url = "jdbc:mysql://localhost:3306/bigdata14"
  val username = "root"
  val password = "root"

  val getMysqlConnection = () => {
    Class.forName("com.mysql.jdbc.Driver")
    DriverManager.getConnection(url, username, password)
  }
  //从mysql中读取数据
  def readFromMysql() = {
    val sql = "select * from hero where age > ? and age < ?"
    val mysqlRdd = new JdbcRDD(sc, getMysqlConnection, sql, 1, 100,
2, x => x)
    mysqlRdd.foreach(x => {
      println(s"id:${x.getString("id")}\tname:${x.getString("nam
e")}\t" +
        s"title:${x.getString("title")}\tage:${x.getInt("age")}\t"
+
        s"country:${x.getString("country")}")
    })
  }

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

}
```

输出结果

```
id:1    name:盖伦 title:德玛西亚之力    age:22  country:德玛西亚
id:2    name:卡特琳娜   title:不祥之刃  age:20  country:诺克萨斯
id:3    name:亚索 title:疾风剑豪   age:25  country:艾欧尼亚
id:4    name:瑞雯 title:放逐之刃   age:23  country:诺克萨斯
```

写入数据到mysql

```
def writeToMysql() = {
  //统计每个用户在每个ip上的行为次数，结果存放到mysql表user_ip_times
  val rdd = sc.textFile("file:///D:\\svn\\user-logs-large.txt")
  val result = rdd.map(x => {
    val infos = x.split("\\s")
    ((infos(0),infos(2)),1)
  }).reduceByKey(_ + _)
  //把rdd的数据存放到mysql中
  result.foreachPartition(x => {
    //批量写入mysql
    val connection = getMysqlConnection()
    val sql = "insert into user_ip_times (user,ip,times) values
(?,?,?)"
    val preparedStatement = connection.prepareStatement(sql)
    x.foreach(record => {
      //写入到mysql中
      preparedStatement.setString(1,record._1._1)
      preparedStatement.setString(2,record._1._2)
      preparedStatement.setInt(3,record._2)
      preparedStatement.addBatch()
    })
    preparedStatement.executeBatch()
    connection.close()
  })
}
```

Spark操作HBase

写数据到hbase中

- **1.添加hbase依赖**

```
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-client</artifactId>
    <version>1.2.3</version>
</dependency>
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-server</artifactId>
    <version>1.2.3</version>
</dependency>
```

- **2.hbase配置 hbase-site.xml**
- **3.构建写入hbase的job（这个是mapreduce job）**

```
val job = Job.getInstance(configuration)
```

在job中配置往hbase中写数据的各种配置信息，与mr写hbase一样

- **4.把rdd转换成kv，必须保证v的类型是（Mutation类型）Put类型或者Delete类型**
- **5.调用saveAsNewAPIHadoopDataset，同时把job的配置信息当做参数**

```
hbaseResult.saveAsNewAPIHadoopDataset(job.getConfiguration)
```

```
//统计每个用户在每个ip上的行为次数，结果存放到hbase中，user_ip_times，rowKe
y 用户名，列簇
//列成员，ip，列，times
def writeToHbase() = {
  val configuration = HBaseConfiguration.create(sc.hadoopConfigurat
ion)
  //设置保存到hbase里面的配置信息
  configuration.set(TableOutputFormat.OUTPUT_TABLE,"user_ip_times")
  val job = Job.getInstance(configuration)
  job.setOutputKeyClass(classOf[ImmutableBytesWritable])
  job.setOutputValueClass(classOf[Put])
  job.setOutputFormatClass(classOf[TableOutputFormat[ImmutableBytes
Writable]])
  FileOutputFormat.setOutputPath(job,new Path("/hbase"))
  //计算
  val rdd = sc.textFile("file:///D:\\svn\\user-logs-large.txt")
  val result = rdd.map(x => {
    val infos = x.split("\\s")
    ((infos(0),infos(2)),1)
  }).reduceByKey(_ + _)
  //转换成能往hbase中保存的格式
  val hbaseResult = result.map(x => {
    val put = new Put(Bytes.toBytes(x._1._1))
    put.addColumn(Bytes.toBytes("i"),Bytes.toBytes(x._1._2),Bytes.t
oBytes(x._2.toString))
    (new ImmutableBytesWritable,put)
  })

  //写入hbase
  hbaseResult.saveAsNewAPIHadoopDataset(job.getConfiguration)

}
```

spark从hbase中读取数据加载成RDD

使用sparkContext的newAPIHadoopRDD方法，以mr的InputFormat的方式从hbase中加载数据

- 需要**指定读取数据的inputformat**为：TableInputFormat
- **key的类型**：ImmutableBytesWritable
- **value的类型**：Result
- rdd从hbase读取的数据每一行会作为rdd的一个元素，类型为Result
- 可以使用result的cellScanner对数据进行取值
- 也可以使用getColumn等hbase的api方法从result中取值

```
def readFromHBase() = {
  val configuration = HBaseConfiguration.create(sc.hadoopConfigurat
ion)
  //在configuration中配置读取hbase所需的参数
  configuration.set(TableInputFormat.INPUT_TABLE,"user_ip_times")
//    val job = Job.getInstance(configuration)
//    job.setInputFormatClass()
  val hbaseRdd = sc.newAPIHadoopRDD(configuration,classOf[TableInpu
tFormat],classOf[ImmutableBytesWritable],classOf[Result])
  val resultRdd = hbaseRdd.map(x => {
    val result = x._2
    val user = Bytes.toString(result.getRow)
    val cellScanner = result.cellScanner()
    val values = new ArrayBuffer[String]()
    while(cellScanner.advance()){
      val cell = cellScanner.current()
      val ip = Bytes.toString(CellUtil.cloneQualifier(cell))
      val times = Bytes.toString(CellUtil.cloneValue(cell))
      values += s"$ip $times"
      //values += s"$user $ip $times"
    }
    (user,values.toList)
    //values
  })
  resultRdd.foreach(x => {
    x._2.foreach(z => println(s"${x._1} ${z}"))
  })
}
```

spark程序发布运行模式

hadoop发布运行（hadoop jar）
hive发布程序运行（hive -f hive脚本位置）
发布程序需要使用**spark-submit**指令

```
spark-submit [options] <app jar | python file> [app arguments]
```

- –class 应用程序主函数入口类的指定
- –jars 指定应用程序的运行的依赖jar包的位置

> 一般情况下我们可以在应用程序里面使用sc.addJar方法指定hdfs上的jar加载应用程序的依赖jar

- –master 发布运行模式
- local —本地模式运行 local[n],local[*]

standalone

```
--deploy-model client/cluster
```

- **client模式**，driver程序是运行在client上，哪里submit的spark程序，哪里就是client，要等集群中所有的计算都结束spark-submit才会结束
- **cluster模式**，是driver程序运行在集群中的某个工作节点上，客户端那里只要提交任务成功spark-submit就已经结束了

yarn

- 部署模式也是分成两种client和cluster
- 运行结构也和standalone一样

mesos

- –deploy-mode 部署模式，除了local之外standalone，yarn，mesos都有两种部署模式：cluster、client
- –conf 运行时环境参数 比方说运行时内存设置，运行时executor数量设置等

示例代码

本地模式运行

```
spark-submit --class com.bd14.zjf.WordCount --master local sparkTes
t.jar
```

standalone模式运行

```
spark-submit --master spark://master:7077 --deploy-mode client --cl
ass com.bd14.zjf.WordCount sparkTest.jar
spark-submit --master spark://slaver1:7077 --deploy-mode cluster --
class com.bd14.zjf.WordCount sparkTest.jar
```

yarn运行模式

```
spark-submit --master yarn --deploy-mode client --class com.bd14.zjf.WordCount sparkTest.jar
spark-submit --master yarn --deploy-mode cluster --class com.bd14.zjf.WordCount sparkTest.jar
```

yarn的client模式下如果因为内存不够抛异常

在yarn-site.xml把yarn.nodemanager.vmem-check-enabled设置为false重启yarn，在submit-spark程序

```
<property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
</property>
```

java开发spark

1. 构建maven项目
2. 添加spark依赖
3. 构建JavaSparkContext
4. 使用JavaSparkContext构建JavaRDD
5. 对JavaRDD调用transformation和action 操作数据（大量的内部类写法实例化算子）
6. 计算结束，停用sparkcontext：sc.stop()

```java
package com.bd14.zjf;

import java.util.Arrays;
import java.util.Iterator;

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.FlatMapFunction;
import org.apache.spark.api.java.function.Function2;
import org.apache.spark.api.java.function.PairFunction;

import scala.Tuple2;

public class WordCount {

    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setMaster("local[*]").setA
ppName("Java Spark");
        JavaSparkContext sc = new JavaSparkContext(conf);
        // 构建rdd
        JavaRDD<String> jRdd = sc.textFile("/user_info.txt");
        // 把一行转换成单词
        JavaRDD<String> wordRdd = jRdd.flatMap(new FlatMapFunctio
n<String, String>() {

            @Override
            public Iterator<String> call(String t) throws Exception
{
                return Arrays.asList(t.split("\\s")).iterator();
            }
        });

        // 把单词的rdd转换成kv的rdd
        JavaPairRDD<String, Integer> javaPairRdd = wordRdd.mapToPai
r(new PairFunction<String, String, Integer>() {

            @Override
            public Tuple2<String, Integer> call(String t) throws Ex
ception {
                return new Tuple2<String, Integer>(t, 1);
            }
        });
        JavaPairRDD<String, Integer> result = javaPairRdd.reduceByK
ey(new Function2<Integer, Integer, Integer>() {

            @Override
```

```
            public Integer call(Integer v1, Integer v2) throws Exce
ption {
                return v1 + v2;
            }
        });

        result.saveAsTextFile("/java-spark");
        sc.stop();
    }

}
```