Spark 学习之提交任务（六）

作者：Calon
原文链接：https://ld246.com/article/1545293971507
来源网站：链滴

本篇文章主要记录Spark的任务提交到集群上的过程

在 http://itechor.top/solo/articles/2018/12/17/1545016407680.html 这篇文章搭建好的集群环上,进行任务的提交运行。

新建一个maven项目，以统计用户身高性别等为主，pom.xml添加以下依赖:

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <java.version>1.8></java.version>
  <spark.version>2.4.0></spark.version>
</properties>

<dependencies>
  <dependency>
    <groupId>com.thoughtworks.paranamer</groupId>
    <artifactId>paranamer</artifactId>
    <version>2.8</version>
  </dependency>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.12</artifactId>
    <version>${spark.version}</version>
  </dependency>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-sql_2.12</artifactId>
    <version>${spark.version}</version>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
```

```
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.13</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-assembly-plugin</artifactId>
      <version>3.1.0</version>
      <configuration>
        <descriptorRefs>
          <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
        <archive>
          <manifest>
            <mainClass>xxx.yyy</mainClass>
          </manifest>
        </archive>
      </configuration>
      <executions>
        <execution>
          <id>make-assembly</id>
          <phase>package</phase>
          <goals>
            <goal>single</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.0</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

配置数据库的配置信息，application.xml:

```
mysql.datasource.url=jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding
utf8&autoReconnect=true&useSSL=false
mysql.datasource.username=root
mysql.datasource.password=root
mysql.datasource.driverClassName=com.mysql.cj.jdbc.Driver
```

读取数据库配置文件DataSourceUtil.java：

```
public class DataSourceUtil {
```

```
    public static String url(){
        return PropertyUtil.getInstance().getString("mysql.datasource.url");
    }
    public static String userName(){
        return PropertyUtil.getInstance().getString("mysql.datasource.username");
    }
    public static String passWord(){
        return PropertyUtil.getInstance().getString("mysql.datasource.password");
    }
    public static String driverClassName(){
        return PropertyUtil.getInstance().getString("mysql.datasource.driverClassName");
    }
}
```

PropertyUtil.java：

```
public class PropertyUtil {
    private PropertyUtil() {
    }
    private static class SingleTonHoler {
        private static ResourceBundle INSTANCE = ResourceBundle.getBundle("application");
    }
    public static ResourceBundle getInstance() {
        return SingleTonHoler.INSTANCE;
    }
}
```

MySQLService.java：

```
import cn.grgpay.analyze.util.DataSourceUtil;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.*;
import org.apache.spark.rdd.RDD;
import org.apache.spark.sql.*;

import java.io.Serializable;
import java.util.*;

public class MySQLService implements Serializable {
    private static final long serialVersionUID = 396720719322480114L;

    public static void main(String[] args) {
        readMySQL();

    }

    private static void readMySQL() {
        SparkSession session = SparkSession.builder().master("local[*]").appName("readMySQLT
Day").config("spark.sql.warehouse.dir", "./spark-warehouse").getOrCreate();
    SQLContext sqlContext = session.sqlContext();

    Properties connectionProperties = new Properties();
    connectionProperties.put("user", DataSourceUtil.userName());
```

```
 connectionProperties.put("password", DataSourceUtil.passWord());
 connectionProperties.put("driver", DataSourceUtil.driverClassName());
long start = System.currentTimeMillis();

 // 读取person表中所有数据
 Dataset data = sqlContext.read().jdbc(DataSourceUtil.url(), "person", connectionProperties).s
lect("*");
long end = System.currentTimeMillis();
 System.out.println("读取数据库数据【"+data.count()+"】条，耗时："+((end-start)/1000));
 // 过滤出性别为男的数据
 Dataset maleData = data.filter(new FilterFunction() {
        private static final long serialVersionUID = -6182357065815734414L;

 @Override
 public boolean call(Row value) {
            String sex = value.getAs("sex");
return sex.equals("男");
 }
    });
// 得到性别为男的身高数据
 Dataset maleHeightData = maleData.map(new MapFunction, Integer>() {
        private static final long serialVersionUID = -78816638100036826 51L;

 @Override
 public Integer call(Row value) {
            return value.getAs("height");
 }
    }, Encoders.INT());

 // 全部男性身高相加
 Integer maleReduce = maleHeightData.reduce(new ReduceFunction() {
        private static final long serialVersionUID = -7419948477276929434L;

 @Override
 public Integer call(Integer v1, Integer v2) {
            return v1 + v2;
 }
    });

 Dataset maleHeight = maleData.sort(maleData.col("height").desc());//男性身高倒序排序
 Dataset lowerMaleHeight = maleData.sort(maleData.col("height").asc());//男性身高升序排序
 System.out.println("男性平均身高："+(maleReduce/maleHeightData.count())+"，最高的男性
高为：" + maleHeight.first() + "，最矮：" + lowerMaleHeight.first());

// 过滤出性别为女的数据
 Dataset feMaleData = data.filter(new FilterFunction() {
        private static final long serialVersionUID = 6593222075687505570L;

 @Override
 public boolean call(Row value) {
            String sex = value.getAs("sex");
return sex.equals("女");
 }
    });
```

```
// 得到性别为女的身高数据
  Dataset femaleHeightData = feMaleData.map(new MapFunction, Integer>() {
          private static final long serialVersionUID = -7881663810003682651L;

  @Override
  public Integer call(Row value) {
              return value.getAs("height");
  }
      }, Encoders.INT());

  // 全部女性身高相加
  Integer femaleReduce = femaleHeightData.reduce(new ReduceFunction() {
          private static final long serialVersionUID = -7419948477276929434L;

  @Override
  public Integer call(Integer v1, Integer v2) {
              return v1 + v2;
  }
      });

  Dataset femaleHeight = feMaleData.sort(feMaleData.col("height").desc());//女性身高倒序排序
  Dataset lowerFemaleHeight = feMaleData.sort(feMaleData.col("height").asc());//女性身高升
排序
  System.out.println("女性平均身高："+(femaleReduce/femaleHeightData.count())+"，最高的
性身高为：" + femaleHeight.first() + ",最矮：" + lowerFemaleHeight.first());

  System.out.println("计算耗时："+((System.currentTimeMillis()-end)/1000));

  }
}
```

如果本地有安装Spark服务，可以直接右键Run这个main函数即可计算出结果。

下面介绍一下提交任务到Spark集群中运行。

其实提交任务到Spark集群也很简单，先maven打包出jar，把jar包上传到Spark的Master节点的任意录，执行命令：

```
spark-submit --master spark://spark1:7077 --class xxx.yyy.MySQLService /usr/local/apps/test-
.0.jar

--master spark://spark1:7077
  这个是指定master节点的地址
--class xxx.yyy.MySQLService
  这个是指定执行那个类的main函数
/usr/local/apps/test-1.0.jar
  这个是指定jar包的路径
```

这样就可以提交任务到Spark集群里了。

```
<br/>
<br/>
<br/>
<br/>
<br/>
```

扫一扫有惊喜：
[![imagepng](http://itechor.top/solo/upload/bb791a58c3a84193b7f643b6849482c5_image.png)](http://ym0214.com)