

## 实验九 Flink 编程

### 9.1 实验目的

- 学习编写简单的基于 DataStream API 的 Flink 程序
- 掌握在 IDEA 中调试 Flink 相关程序，以及在单机伪分布式、分布式部署方式下提交运行 Flink 程序的方法

### 9.2 实验任务

- 完成 WordCount 示例程序的编写
- 分别在单机伪分布式和分布式环境下运行 WordCount 示例程序

### 9.3 实验环境

- 操作系统: Ubuntu 18.04
- JDK 版本: jdk 1.8
- Hadoop 版本: 2.5.1
- Flink 版本: 1.12.1
- Scala 版本: 2.11.12

### 9.4 实验步骤

#### 9.4.1 编写 Flink 应用程序

##### (1) 新建 Maven 项目并添加依赖

- 新建名为 “FlinkDemo” 的 Maven 项目
- 编辑项目根目录下的 pom.xml 文件，分别在 <build> 和 <dependencies> 标签中添加 Flink 相关的依赖包 flink-streaming-java\_2.11 和 flink-core

```
1 <dependencies>
2 <!-- https://mvnrepository.com/artifact/org.apache.flink/flink-streaming-java
   -->
3 <dependency>
4   <groupId>org.apache.flink</groupId>
5   <artifactId>flink-streaming-java_2.11</artifactId>
6   <version>1.12.1</version>
```

```

7      <scope>compile</scope>
8  </dependency>
9
10 <!-- https://mvnrepository.com/artifact/org.apache.flink/flink-core -->
11 <dependency>
12     <groupId>org.apache.flink</groupId>
13     <artifactId>flink-core</artifactId>
14     <version>1.12.1</version>
15 </dependency>
16 <dependency>
17     <groupId>org.apache.flink</groupId>
18     <artifactId>flink-clients_2.11</artifactId>
19     <version>1.12.1</version>
20 </dependency>
21 <dependency>
22     <groupId>org.slf4j</groupId>
23     <artifactId>slf4j-simple</artifactId>
24     <version>1.7.25</version>
25 </dependency>
26 </dependencies>

```

修改完成后，在菜单界面选择 View->Tool Windows->Maven，在弹出的界面中点击 Reload All Maven Projects 加载依赖文件，第一次加载此过程可能耗时较长。

## (2) 编写程序

- Java 版: 新建 Java 类 *WordCount*

在项目的 *src/main/java* 目录下，选择 New -> Package，输入名称 *cn.edu.ecnu.flink.example.java.wordcount*，之后右键单击新建好的包，选择 New -> Java Class，输入名称 *WordCount*。

```

1 package cn.edu.ecnu.flink.example.java.wordcount;
2
3 import org.apache.flink.api.common.functions.FlatMapFunction;
4 import org.apache.flink.api.common.functions.MapFunction;
5 import org.apache.flink.api.java.tuple.Tuple2;
6 import org.apache.flink.streaming.api.datastream.DataStream;
7 import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
8 import org.apache.flink.util.Collector;
9
10 public class WordCount {
11     public static void main(String[] args) throws Exception {
12         run(args);

```



```
13     }
14
15     public static void run(String[] args) throws Exception {
16         /* 步骤1: 创建StreamExecutionEnvironment对象 */
17         StreamExecutionEnvironment env =
18             StreamExecutionEnvironment.getExecutionEnvironment();
19
20         /* 步骤2: 按应用逻辑使用操作算子编写DAG, 操作算子包括数据源、转换、数据池
21            等 */
22         // 从指定的主机名和端口号接收数据, 创建名为lines的DataStream
23         DataStream<String> lines = env.socketTextStream(args[0],
24             Integer.parseInt(args[1]), "\n");
25         // 将lines中的每一个文本行按空格分割成单个单词
26         DataStream<String> words =
27             lines.flatMap(
28                 new FlatMapFunction<String, String>() {
29                     @Override
30                     public void flatMap(String value, Collector<String>
31                         out) throws Exception {
32                         for (String word : value.split(" ")) {
33                             out.collect(word);
34                         }
35                     }
36                 });
37         // 将每个单词的频数设置为1, 即将每个单词映射为[单词, 1]
38         DataStream<Tuple2<String, Integer>> pairs =
39             words.map(
40                 new MapFunction<String, Tuple2<String, Integer>>() {
41                     @Override
42                     public Tuple2<String, Integer> map(String value)
43                         throws Exception {
44                         return new Tuple2<String, Integer>(value, 1);
45                     }
46                 });
47         // 按单词聚合, 并对相同单词的频数使用sum进行累计
48         DataStream<Tuple2<String, Integer>> counts = pairs.keyBy(0).sum(1);
49         // 输出词频统计结果
50         counts.print();
51
52         /* 步骤3: 触发程序执行 */
53         env.execute("Streaming WordCount");
54     }
```

```
50 }
}
```

### (3) Scala 版: 新建 Scala 类 *WordCountScala*

在项目的 *src/main* 目录下, 选择 New -> Directory, 输入 *scala*, 接着右键单击目录 *scala*, 选择 Mark Directory as -> Sources Root。在 *scala* 目录下选择 New->Package, 输入名称 *cn.edu.ecnu.flink.example.scala.wordcount*, 之后右键单击新建好的包, 选择 New -> Scala Class, 输入名称 *WordCount*。

```
1 package cn.edu.ecnu.flink.example.scala.wordcount
2
3 import org.apache.flink.streaming.api.scala.StreamExecutionEnvironment
4 import org.apache.flink.streaming.api.scala._
5 import org.apache.flink.streaming.api.windowing.assigners.{GlobalWindows,
6   TumblingEventTimeWindows, TumblingProcessingTimeWindows}
7
8 import org.apache.flink.streaming.api.windowing.time.Time
9
10 object WordCount {
11   def run(args: Array[String]): Unit = {
12     /* 步骤1: 创建StreamExecutionEnvironment对象 */
13     val env = StreamExecutionEnvironment.getExecutionEnvironment
14
15     /* 步骤2: 按应用逻辑使用操作算子编写DAG, 操作算子包括数据源、转换、数据池等 */
16     // 从指定的主机名和端口号接收数据, 创建名为lines的DataStream
17     val lines = env.socketTextStream(args(0), args(1).toInt)
18     // 将lines中的每一个文本行按空格分割成单个单词
19     val words = lines.flatMap(w => w.split(" "))
20     // 将每个单词的频数设置为1, 即将每个单词映射为[单词, 1]
21     val pairs = words.map(word => (word, 1))
22     // 按单词聚合, 并对相同单词的频数使用sum进行累计
23     val counts = pairs.keyBy(0)
24       .sum(1)
25     // 输出词频统计结果
26     counts.print()
27
28     /* 步骤3: 触发程序执行 */
29     env.execute("Streaming WordCount")
30   }
31
32   def main(args: Array[String]): Unit = {
33     run(args)
34   }
35 }
```



## 9.4.2 调试 Flink 应用程序

### (1) 启动 Netcat 服务

```
nc -lk 8888 #在 8888 端口开启Ubuntu自带的Netcat服务，之后我们会在此输入数据
```

### (2) 配置运行环境

在菜单栏点击 Run -> Edit Configuration, 在弹出的界面中点击 + 号选择 Application, 新建 Application 配置, Name 为 WordCountFlink (Scala 版则为 WordCountScala), 配置界面如图9.1所示。

- 配置 Main Class 为 `cn.edu.ecnu.flink.example.java.wordcount.WordCount` (Scala 版则配置为 `cn.edu.ecnu.flink.example.scala.wordcount.WordCount`)
- 配置 Program arguments 为 IP 或者主机名 端口号  
如 `localhost 8888`

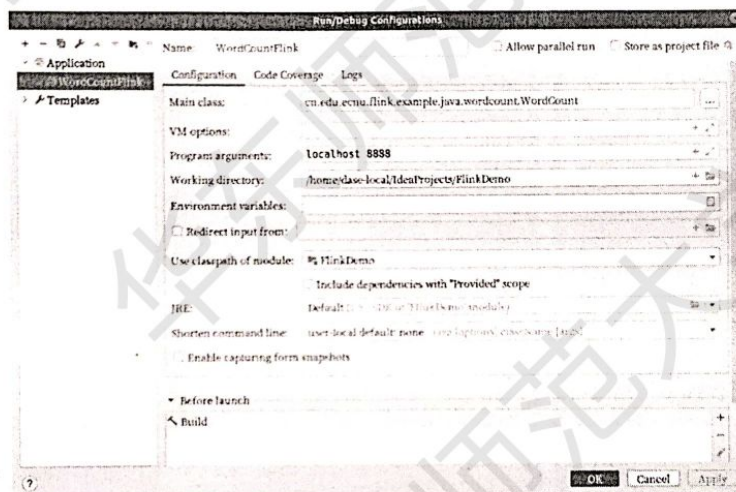


图 9.1 运行应用程序配置界面

### (3) 调试程序

- 在菜单栏点击 Run->Run 'WordCountFlink'(Scala 版为 Run 'WordCountScala')
- 在启动 Netcat 服务的窗口中输入数据，如图9.2所示

```
dase-local@ecnu01:~$ nc -lk 8888
hello dase
hello ecnu
hello flink
```

图 9.2 输入数据

- 在 IDEA 中查看处理结果，如图9.3所示

### (4) 停止调试

```

4> (dase,1)
2> (hello,1)
1> (ecnu,1)
2> (hello,2)
2> (hello,3)
4> (flink,1)

```

图 9.3 在 IDEA 中的运行结果

- 在启动 Netcat 服务的窗口中使用 Ctrl + C 停止服务
- 在 IDEA 中使用 Ctrl + F2 停止程序运行

### 9.4.3 运行 Flink 应用程序

以 Java 版本为例。

#### (1) 准备工作

以下操作在各节点均以 dase-local 用户身份进行。

- 使用 IDEA 将程序打成 jar 包

jar 包名称为 FlinkWordCount.jar, 打包路径为 `/home/dase-local/IdeaProjects/FlinkDemo/out/artifacts/FlinkWordCount`, 配置界面如图 9.4 所示。

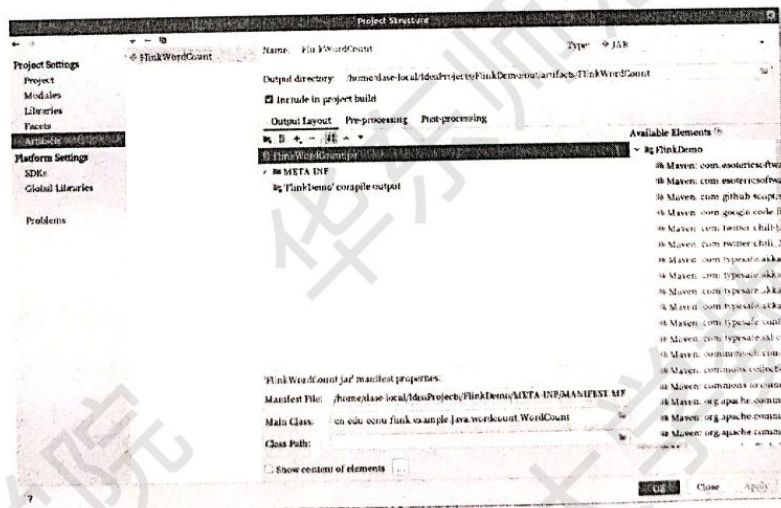


图 9.4 jar 包配置界面

- 复制 jar 包到指定目录

```

1 su dase-local
2 mkdir ~/flink-1.12.1/myApp
3 cp ~/IdeaProjects/FlinkDemo/out/artifacts/FlinkWordCount/FlinkWordCount.jar

```



```
~/flink-1.12.1/myApp/
```

## (2) 伪分布部署方式下运行应用程序

以下操作以 dase-local 用户身份进行。

- 启动 Flink 服务

```
1 su dase-local
2 ~/flink-1.12.1/bin/start-cluster.sh
```

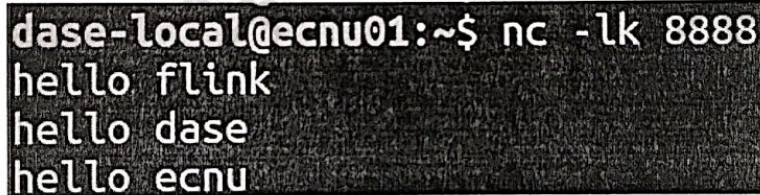
- 启动 Netcat 服务

```
1 nc -lk 8888 #在 8888 端口开启Ubuntu自带的Netcat服务，之后我们会在此输入数据
```

- 通过提交 jar 包运行应用程序

```
1 ~/flink-1.12.1/bin/flink run -c
  cn.edu.ecnu.flink.example.java.wordcount.WordCount
  ~/flink-1.12.1/myApp/FlinkWordCount.jar localhost 8888
```

- 在启动 Netcat 服务的终端中输入数据，如图9.5所示。

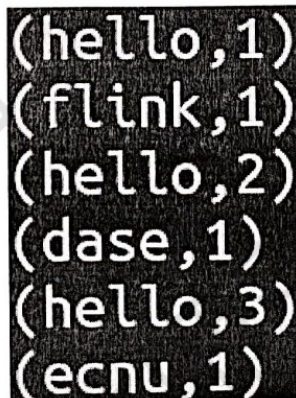


```
dase-local@ecnu01:~$ nc -lk 8888
hello flink
hello dase
hello ecnu
```

图 9.5 为程序提供输入数据

- 另起一个终端查看输出结果，如图9.6所示

```
1 tail -f ~/flink-1.12.1/log/flink-dase-local-taskexecutor-0-ecnu01.out
```



```
(hello,1)
(flink,1)
(hello,2)
(dase,1)
(hello,3)
(ecnu,1)
```

图 9.6 程序的输出结果

- 停止应用程序和服务

- 终止应用程序

在启动 Netcat 服务的窗口使用 Ctrl + C，停止 Netcat 服务。应用程序状态会变为 Finished。

- 停止 Flink 服务

```
~/flink-1.12.1/bin/stop-cluster.sh
```

### (3) 分布式部署方式下运行应用程序

以下操作在各节点均以 dase-dis 用户身份进行。

- 在客户端节点，复制 jar 包到指定目录

```
1 su dase-dis
2 mkdir ~/flink-1.12.1/myApp
3 scp
   dase-local@localhost:/home/dase-local/flink-1.12.1/myApp/FlinkWordCount.jar
   r ~/flink-1.12.1/myApp
```

- 启动 Flink 服务

在主节点执行如下命令：

```
1 su dase-dis
2 ~/flink-1.12.1/bin/start-cluster.sh
```

- 启动 Netcat 服务

在客户端节点执行如下命令：

```
1 nc -lk 8888 #开启Netcat服务，之后在此输入数据
```

- 通过提交 jar 包运行应用程序

在客户端节点执行如下命令：

```
1 ~/flink-1.12.1/bin/flink run -c
   cn.edu.ecnu.flink.example.java.wordcount.WordCount
   ~/flink-1.12.1/myApp/FlinkWordCount.jar ecnu04 8888
```

- 在启动 Netcat 服务的终端中输入数据

• 根据实验八所述的定位输出文件路径的方法，在对应节点输入如下命令

```
1 tail -f ~/flink-1.12.1/log/flink-dase-dis-taskexecutor-*-ecnu02.out
2 # “*” 号内容与具体对应的文件名称一致
```



- 停止应用程序和服务

在主节点执行如下操作：

- 终止应用程序

在启动 Netcat 服务的窗口使用 Ctrl + C，停止 Netcat 服务。应用程序状态会变为 Finished。

- 停止 Flink 服务

```
~/flink-1.12.1/bin/stop-cluster.sh
```

## 9.5 思考题

1 如何查看一个 Flink 流计算应用程序启动了多少个任务，以及每个任务的并行度是多少？请结合 Web UI 来说明。

2 一个 Flink 应用程序中的算子会发生合并吗？哪些算子会合并？并结合上述编程示例进行说明。