



# 大数据技术基础课程实验报告

## 期末实验一：基于华为云的大数据实时数据分析综合实践

付容天

学号 2020211616

班级 2020211310

计算机学院（国家示范性软件学院）

2023 年 5 月 18 日

# 1. Local 模式部署安装

Local 模式下的 Flink 部署安装只需要使用单台机器，仅用本地线程来模拟其程序执行，不需要启动任何进程，适用于软件测试等情况，这种模式下及其不需要更改任何配置，只需要安装 JDK 8 运行环境即可。

本次实验的目的是：

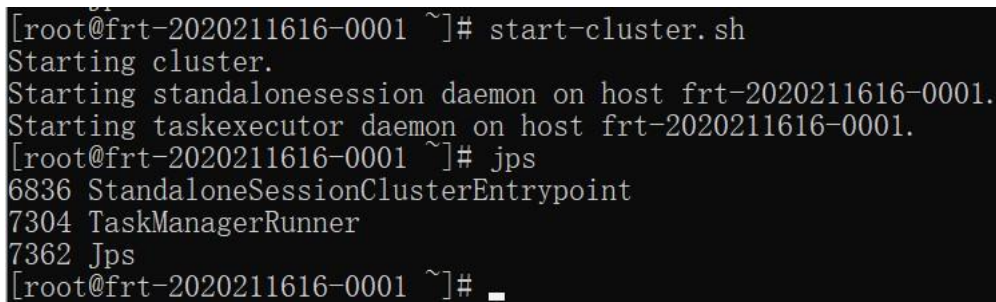
- (1) 实现 Flink 的安装；
- (2) 学会 Flink 的脚本启动；
- (3) 使用 Flink 自带的单词统计程序进行测试。

## 1.1. 安装 Flink 并进行配置

我按照实验指导书的内容，将 Flink 安装包上传到 node1 服务器中并进行安装（在目录/home/modules 下）。然后在/etc/profile 配置文件中添加 Flink 的路径，并运行指令 source /etc/profile 使新配置生效。

## 1.2. 启动 Flink 进程并查看信息

直接使用相关命令即可启动 Flink 进程，启动之后输入 jps 命令，结果截图如下所示：



```
[root@frt-2020211616-0001 ~]# start-cluster.sh
Starting cluster.
Starting standalone session daemon on host frt-2020211616-0001.
Starting taskexecutor daemon on host frt-2020211616-0001.
[root@frt-2020211616-0001 ~]# jps
6836 StandaloneSessionClusterEntrypoint
7304 TaskManagerRunner
7362 Jps
[root@frt-2020211616-0001 ~]#
```

图 1: 启动 Flink 后查看进程

此时可以在 web 端访问 Flink 的管理界面，可以看到 Task Managers、Task Slots、Available Task Slots 三个值均为 1。

### 1.3. 运行 Flink 自带测试用例

我们可以在 node1 上通过 nc 命令向 Socket 发送一些单词（需要首先安装 nc 命令），输入内容如下图所示：

```
[root@firt-2020211616-0001 ~]# nc -lk 9000
beijing university of posts and telecommunications
2020211310class 2020211616firt big data
```

图 2: 通过 nc 命令发送一些单词

同时我们需要打开另一个 node1 的 shell 界面启动 Flink 自带的单词统计程序，接收输入的 Socket 数据并进行统计，在 web 端查看结果如下：

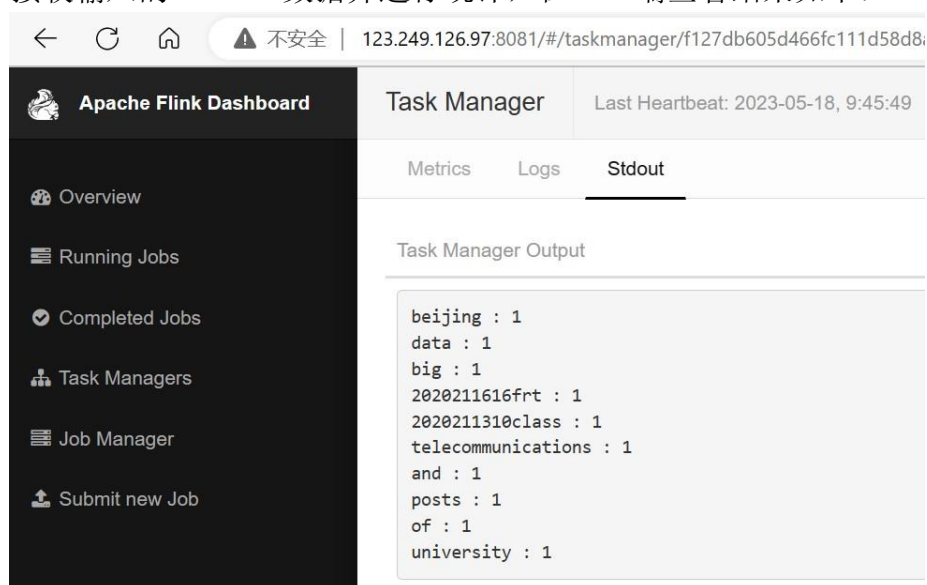


图 3: 在 web 端查看单词统计结果

我们也可以在 node1 上执行相应命令进行查看，结果如下：

```
[root@firt-2020211616-0001 ~]# cd /home/modules/flink-1.8.0/log/
[root@firt-2020211616-0001 log]# tail -200f flink-root-taskexecutor-0-firt-2020211616-0001.out
beijing : 1
data : 1
big : 1
2020211616firt : 1
2020211310class : 1
telecommunications : 1
and : 1
posts : 1
of : 1
university : 1
```

图 4: 在 node1 用命令查看统计结果

## 2. Standalone 模式部署安装

使用 standalone 模式需要启动 Flink 主节点的 JobManager 以及从节点的 TaskManager。本实验目的是：实现 standalone 模式下 Flink 进程的启动。

### 2.1. 修改配置文件并启动集群

在这部分实验中，我按照实验指导书，先后完成了：

- (1) 在 node1 节点上更改 Flink 配置文件；
- (2) 指定 JobManager 所在服务器为 node1；
- (3) 更改 slaves 配置文件；
- (4) 分发配置文件；
- (5) 启动 Flink 集群。

启动 Flink 集群后，在四个节点上输入 jps 命令，可以得到下面的效果：

```
[root@frt-2020211616-0001 modules]# start-cluster.sh
Starting cluster.
Starting standalone-session daemon on host frt-2020211616-0001.
Starting taskexecutor daemon on host frt-2020211616-0001.
Starting taskexecutor daemon on host frt-2020211616-0002.
Starting taskexecutor daemon on host frt-2020211616-0003.
Starting taskexecutor daemon on host frt-2020211616-0004.
[root@frt-2020211616-0001 modules]# jps
9486 StandaloneSessionClusterEntrypoint
10051 Jps
9988 TaskManagerRunner
```

图 5.1: node1 上查看当前进程

```
[root@frt-2020211616-0002 ~]# jps
2145 TaskManagerRunner
2219 Jps
```

图 5.2: node2 上查看当前进程

```
[root@frt-2020211616-0003 ~]# jps
2145 TaskManagerRunner
2218 Jps
```

图 5.3: node3 上查看当前进程

```
[root@frt-2020211616-0004 ~]# jps
2264 Jps
2191 TaskManagerRunner
```

图 5.4: node4 上查看当前进程

这时再进入前面所说的 web 页面，可以看到：

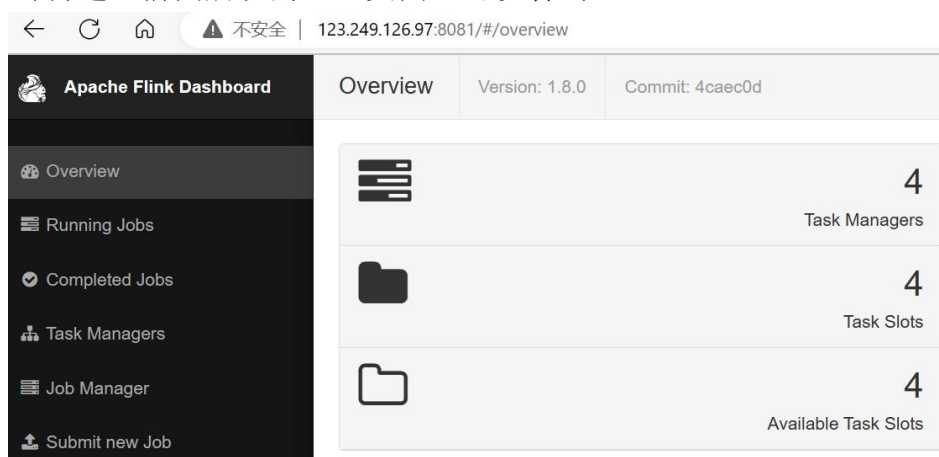


图 6: web 页面查看集群信息

## 2.2. 运行 Flink 自带测试用例

现在可以在 node1 中启动 Socket 服务并输入单词，输入单词如下所示：

```
[root@firt-2020211616-0001 hadoop]# nc -lk 9000
flink flink flink flink hadoop hadoop zookeeper
123 123 123 2020211616 firt firt firt
cmd
```

图 7: 在 node1 中通过 nc 命令输入单词

现在可以再启动一个 node1 的 shell，启动 Flink 自带的单词统计程序，对 Socket 中的单词进行计数，并在 web 页面中查看得到如下结果：

Path, ID	Data Port	Last Heartbeat	All Slots	Free Slots	CPU Cores	Physical Memory	JVM Heap Size	Flink Managed Memory
akka.tcp://flink@192.168.0.34:34695/user/taskmanager_0 F6419443D275FF90F6A82ADEFAA892E7	37701	2023-05-18, 10:30:03	1	0	2	3.40 GB	922 MB	553 MB
akka.tcp://flink@192.168.0.114:42101/user/taskmanager_0 174494E6C395977303659F000861496E	41909	2023-05-18, 10:30:03	1	1	2	3.40 GB	922 MB	553 MB
akka.tcp://flink@192.168.0.41:38031/user/taskmanager_0 3868A3534BE0A1A9DEC092BFE42D7019	38243	2023-05-18, 10:30:03	1	1	2	3.40 GB	922 MB	553 MB
akka.tcp://flink@192.168.0.167:43007/user/taskmanager_0 93A53473CE04D4E8BA2AC5827587DD93	45177	2023-05-18, 10:30:03	1	1	2	3.40 GB	922 MB	553 MB

图 8: 在 web 页面查看状态

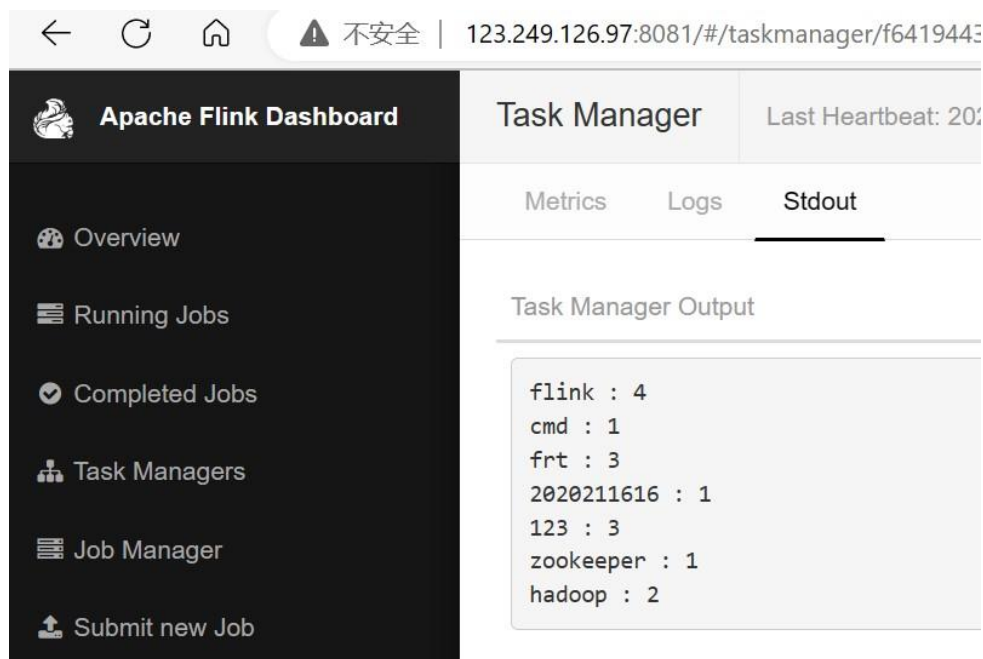


图 9: 在 web 页面查看单词统计结果

图 8 中显示执行的是 node1，所以可以在 node1 中的 log 中查看统计结果如下所示：

```
[root@frt-2020211616-0001 log]# tail -200f flink-root-taskexecutor-0-frt-2020211616-0001.out
flink : 4
cmd : 1
frt : 3
2020211616 : 1
123 : 3
zookeeper : 1
hadoop : 2
```

图 10: 在 node1 中查看统计结果

### 3. Flink on Yarn 模式

Flink 任务也可以运行在 Yarn 上，将 Flink 任务提交到 Yarn 平台可以实现统一的任务资源调度管理，方便开发人员管理集群中的 CPU 和内存等资源。本模式需要先启动集群，然后再提交作业，接着会向 Yarn 申请资源空间，之后资源保证不变。如果资源不足，下一个作业就无法提交，需要等到 Yarn 中的一个作业执行完成后释放资源。

本实验的目的如下：

- (1) 完成 Flink on Yarn 模式的配置；
- (2) 在 Yarn 中启动 Flink 集群；
- (3) 以文件的形式进行任务提交。

### 3.1. 修改配置与启动集群

根据实验指导书的内容，在这部分的实验中，我首先对配置文件进行了修改，包括以下配置文件：

- (1) 修改 node1 的 yarn-site.xml 文件；
- (2) 修改 node1 的 flink-conf.yaml 文件；
- (3) 修改 node1 的 masters 文件；
- (4) 在 node1 的指定目录下新增 jar 包。

并将上述修改分发到其余节点上。

现在可以按照实验三的相关内容完成 ZooKeeper 在各个节点上的启动。

### 3.2. 在 Yarn 中启动集群并查看

我在完成了指导书所述的检查工作后，按照指定步骤启动了 Flink 集群，并可以访问 Yarn 的 8088 管理界面，可以查看得到下面的结果：

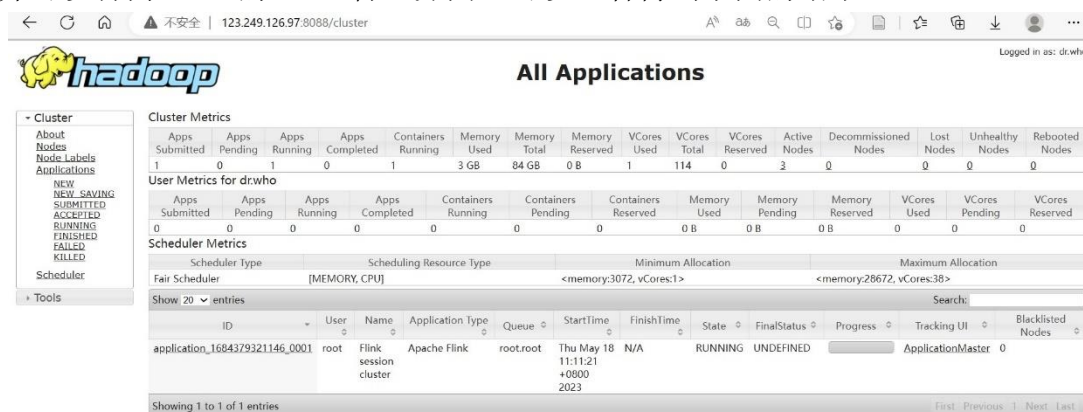


图 11: 查看 Yarn 管理页面

### 3.3. 提交任务并查看运行结果

首先在 node1 上准备单词文件 wordcount.txt，其内容如下所示：

```
[root@firt-2020211616-0001 ~]# cat wordcount.txt
hello world
flink hadoop
hive spark
bupt firt 2020211616
```

图 12: 待统计单词文件的内容



然后将其提交到 HDFS 文件夹（需要提前创建），之后在 node1 上执行指定命令，提交任务到 Flink 集群，结果如下所示：

```
root@firt-2020211616-0001:/home/modules/flink-1.8.0
[root@firt-2020211616-0001 ~]# hdfs dfs -mkdir -p /flink_input
23/05/18 11:16:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-jav
a classes where applicable
[root@firt-2020211616-0001 ~]# hdfs dfs -put wordcount.txt /flink_input
23/05/18 11:17:09 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-jav
a classes where applicable
[root@firt-2020211616-0001 flink-1.8.0]# bin/flink run examples/batch/WordCount.jar -input hdfs://node1:8020/flink_input -
output hdfs://node1:8020/flink_output/wordcount-result.txt
2023-05-18 11:19:13,172 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - Found Yarn properties file
under /tmp/.yarn-properties-root.
2023-05-18 11:19:13,172 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - Found Yarn properties file
under /tmp/.yarn-properties-root.
2023-05-18 11:19:13,497 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - YARN properties set default
parallelism to 2
2023-05-18 11:19:13,497 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - YARN properties set default
parallelism to 2
YARN properties set default parallelism to 2
2023-05-18 11:19:13,552 INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManag
er at node1/192.168.0.34:8032
2023-05-18 11:19:13,656 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - No path for the flink jar p
assed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
2023-05-18 11:19:13,656 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - No path for the flink jar p
assed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
2023-05-18 11:19:13,709 INFO org.apache.flink.yarn.AbstractYarnClusterDescriptor - Found application JobManage
r host name 'node2' and port '33231' from supplied application id 'application_1684379321146_0001'
Starting execution of program
Program execution finished
Job with JobID 9747d3b047880974c3c0a2ce51d2bea9 has finished.
Job Runtime: 12767 ms
root@firt-2020211616-0001 flink-1.8.0]#
```

图 13：单词统计任务运行完成

```
[root@firt-2020211616-0001 flink-1.8.0]# hdfs dfs -cat /flink_output/wordcount-result.txt
2020211616 1
bupt 1
flink 1
firt 1
hadoop 1
hello 1
hive 1
spark 1
world 1
```

图 14：单词统计任务运行结果

## 4. Flink 消费 Kafka 数据

在实际工作中，实时数据的处理一般是使用 Kafka 实现的。Flink 提供了一个特有的 Kafka 连接器去读写 Kafka topic 的数据。本实验通过本地打包 jar 包上传到 Flink 集群去处理终端 Kafka 输入的数据。

实验目的包括：

- (1) 安装 Kafka；
- (2) 本地编辑代码读取 Kafka 数据，并打包成 jar 包；
- (3) 将 jar 包上传到 Flink 集群运行。



## 4. 1. Kafka 安装与启动

首先将实验相关资源中的 Kafka 安装包上传到 node1 中，并安装到指定目录下，然后分发到各个节点，之后修改四个节点上的配置文件：

- (1) /etc/profile 文件；
- (2) Kafka 安装目录中的 server.properties 文件。

接下来在四个节点上启动 Kafka，各节点输入 jps 命令得到下面的结果：

```
[root@firt-2020211616-0001 config]# jps
QuorumPeerMain
ResourceManager
Jps
Kafka
SecondaryNameNode
NameNode
```

图 15.1: node1 使用 jps

```
[root@firt-2020211616-0002 config]# jps
2086 Kafka
2035 QuorumPeerMain
1732 DataNode
1858 NodeManager
2132 Jps
[root@firt-2020211616-0002 config]#
```

图 15.2: node2 使用 jps

```
[root@firt-2020211616-0003 config]# jps
1724 DataNode
2029 QuorumPeerMain
2126 Jps
1845 NodeManager
2077 Kafka
[root@firt-2020211616-0003 config]#
```

图 15.3: node3 使用 jps

```
[root@firt-2020211616-0004 config]# jps
2135 Jps
1856 NodeManager
1730 DataNode
2038 QuorumPeerMain
2082 Kafka
[root@firt-2020211616-0004 config]#
```

图 15.4: node4 使用 jps

## 4. 1. 创建 maven 工程并编写代码

根据实验指导书的内容，我创建了 maven 项目，并在修改 pom.xml 文件后编写了项目代码（命名为 WordCount），代码如下：

```
package org.example;
import java.util.Properties;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer08;

public class WordCount {
    public static void main(String[] args) throws Exception {
        // 获取Flink 运行环境
        StreamExecutionEnvironment env
            = StreamExecutionEnvironment.getExecutionEnvironment();
```

```

// 配置Kafka连接属性
Properties properties = new Properties();
// TODO: node1换成主节点名称
properties.setProperty("bootstrap.servers", "node1:9092");
properties.setProperty("zookeeper.connect", "node1:2181");
properties.setProperty("group.id", "1");

FlinkKafkaConsumer08<String> myconsumer = new FlinkKafkaConsumer08<>
("test", new SimpleStringSchema(), properties);

// 默认消费策略
myconsumer.setStartFromGroupOffsets();

DataStream<String> dataStream = env.addSource(myconsumer);

DataStream<Tuple2<String, Integer>> result = dataStream.flatMap
(new MyFlatMapper()).keyBy(0).sum(1);

result.print().setParallelism(1);
env.execute();
}
}

```

还有一个 MyFlatMapper 类，源代码如下所示：

```

package org.example;

import org.apache.flink.api.common.functions.FlatMapFunction;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.util.Collector;

//import scala.Tuple2;

public class MyFlatMapper implements FlatMapFunction<String, Tuple2<String, Integer>> {
    @Override
    public void flatMap(String s, Collector<Tuple2<String, Integer>> out)
throws Exception {
        // 按空格分词
        String[] words = s.split(" ");
        for (String word : words) {
            out.collect(new Tuple2<>(word, 1));
        }
    }
}
}

```

并且将上面的代码进行打包，得到 WordCount.jar 后上传到 node1 中。

### 4.3. 运行 jar 包

该任务需要在前面的 Flink 集群环境下运行，我按照实验指导书的内容，分别启动四个节点的 Kafka，并启动一个生产者，如下图所示：

```
[root@firt-2020211616-0001 kafka_2.10-0.8.2.1]# ./bin/kafka-topics.sh --create --zookeeper node1:2181 --replication-factor 1 --partitions 1 --topic wordsendtest
Created topic "wordsendtest".
[root@firt-2020211616-0001 kafka_2.10-0.8.2.1]# kafka-console-producer.sh --broker-list node1:9092 --topic test
[2023-05-18 15:28:54,852] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
```

图 16: 启动生产者

然后再启动一个 node1 终端运行 jar 包，运行结果如下图所示：

```
[root@firt-2020211616-0001 ~]# flink run -c org.example.WordCount WordCount.jar
2023-05-18 16:00:06,766 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - Found Yarn properties file
under /tmp/.yarn-properties-root.
2023-05-18 16:00:06,766 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - Found Yarn properties file
under /tmp/.yarn-properties-root.
2023-05-18 16:00:07,223 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - YARN properties set default
parallelism to 2
2023-05-18 16:00:07,223 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - YARN properties set default
parallelism to 2
YARN properties set default parallelism to 2
2023-05-18 16:00:07,276 INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager
at node1/192.168.0.34:8032
2023-05-18 16:00:07,383 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - No path for the flink jar
passed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
2023-05-18 16:00:07,383 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - No path for the flink jar
passed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
2023-05-18 16:00:07,438 INFO org.apache.flink.yarn.AbstractYarnClusterDescriptor - Found application JobManager
er host name 'node4' and port '36707' from supplied application id 'application_1684394586393_0001'
Starting execution of program
```

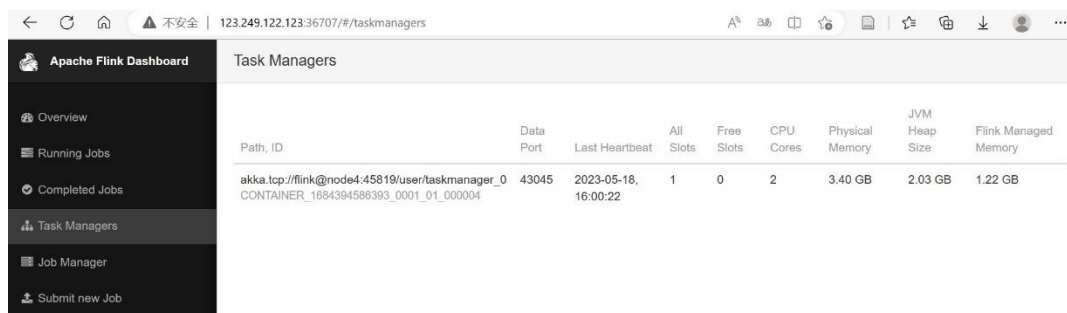
图 17: 运行 jar 包得到的结果

然后可以在生产者终端输入单词进行计数，我输入了下面的内容：

```
[root@firt-2020211616-0001 kafka_2.10-0.8.2.1]# kafka-console-producer.sh --broker-list node1:9092 --topic test
[2023-05-18 15:58:11,039] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
123 123 123 456 456
firt firt firt asd asd asd
bupt bupt
2020211616
```

图 18: 在生产者终端输入指定内容

之后进入图 17 所示的节点和端口（需要打开该端口）的 web 页面，得到如下图所示的结果：



Path, ID	Data Port	Last Heartbeat	All Slots	Free Slots	CPU Cores	Physical Memory	JVM Heap Size	Flink Managed Memory
akka.tcp://flink@node4:45819/user/taskmanager_0 CONTAINER_1684394586393_0001_01_000004	43045	2023-05-18, 16:00:22	1	0	2	3.40 GB	2.03 GB	1.22 GB

图 19: web 管理页面下的 Task Manager 页面

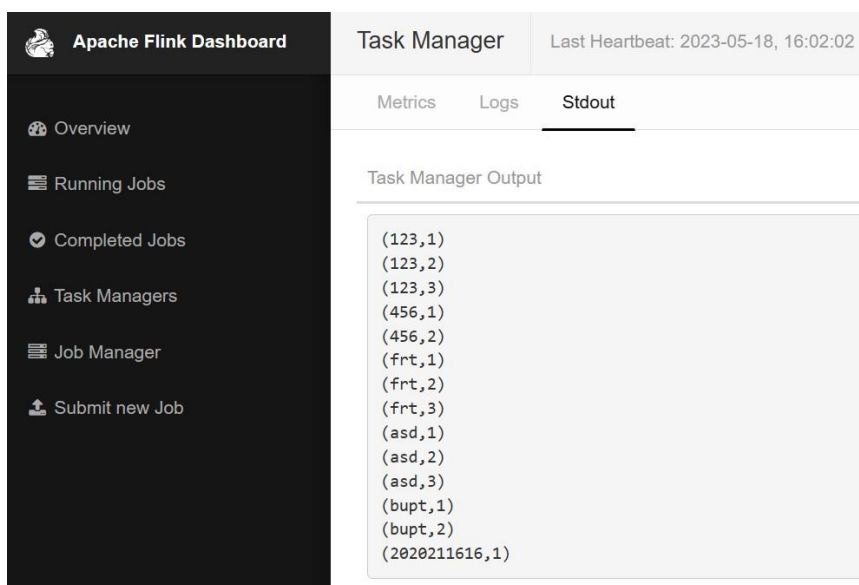


图 20: 单词计数结果

## 5. 实验问题与实验总结

在本次实验中，我也遇到了一些问题，现记录如下：

- (1) 命名问题：一开始我按照实验指导书的内容进行试验，但出现了“节点找不到”的问题，我查看配置文件，想起来在之前的实验中，为简便起见，把节点名称进行了映射（例如“frt-2020211616-0001”映射到了“node1”等）；
- (2) 集群启动问题：由于到目前为止，我们已经给实验环境配置了许多组件，启动步骤（尤其是启动顺序）变得很重要，我通过梳理组件之间的依赖关系，得到了正确的启动顺序。

在本次实验中，我在实验一、二、三、四的基础上安装了 Flink 和 Kafka 两个组件，并进行了相应的配置。我使用 Flink 自带的单词计数程序和自编的单词计数程序在多种环境下完成了单词计数功能，通过耐心的调试解决了所有出现的问题，得到了预期的正确结果。总的来说，我在本次实验中复习了理论知识、进行了工程实践，圆满完成了实验任务，收获颇丰！