> Flink 是一个面向分布式数据流处理和批量数据处理的开源计算平台，提供支持流处理和批处理两种类型应用的功能。主要用于实时计算。

# Flink安装

## 安装Flink

将 Flink 上传至新建的目录 `/opt/software/` 目录下，并解压至 `/usr/local/src/` 目录下。

## 配置环境变量

```
export FLINK_HOME=usr/local/src/flink
export PATH=$PATH:$FLINK_HOME/bin
```

# Flink配置

## 修改配置文件

Flink 配置目录：

```
$FLINK_HOME/conf
```

修改 `flink-conf.yaml` 文件，以设置主节点：

找到以下设置，将值设置为master

**（注意在冒号后面一定要添加空格）**

```
jobmanager.rpc.address: master
```

修改 `slaves` 文件添加集群：

```
master
slave1
slave2
```

## 拷贝Mysql驱动包

将JDBC驱动包拷贝至 `$FLINK_HOME/lib/` 下。

## 拷贝至其他集群

将安装好的Flink与环境变量拷贝至其他的集群。

```
scp -r /usr/local/src/flink slave1:/usr/local/src
scp -r /usr/local/src/flink slave2:/usr/local/src
scp -r /root/.bash_profile slave1:/root/.bash_profile
```

并在其他集群使用 `source` 重新加载集群的环境变量 `.bash_profile` 。

## 启动 Flink

```
start-cluster.sh
```

在启动成功后，在master主机上将会运行以下进程：

- TaskManagerRunner
- StandaloneSessionClusterEntrypoint

在slave主机上将会运行以下进程

- TaskManagerRunner

## 进入WebUI

安装配置完毕后，在实体机的浏览器地址栏输入 `虚拟机IP:8081` 即可进入 Flink 的WebUI。

# Flink使用

## 运行自带案例

Flink自带案例存放目录：

`flink/examples`

在该目录下：

- batch 目录下存放批处理案例
- streaming 目录下存放流式处理案例

**1. 运行批离线案例 WordCount**

在 `/batch` 目录下使用 flink 直接运行 `WordCount.jar` 即可

```
flink run WordCount.jar
```

**2. 运行流式处理案例 WordCount**

新打开一个 master 终端 ， 并设置监听端口

```
nc -l 12600
```

在 `/streaming` 目录下使用 flink 运行 `SocketWindowWordCount.jar` 并设置将要连接的主机与端口。

```
flink run SocketWindowWordCount.jar --hostname master --port 12600
```

在WebUI中可以查看执行结果。

## 运行jar打包文件

**方式一：Flink集群运行**

（需要启动flink集群，并且在8081上能够看到运行的任何和状态）

```
flink run -c com.xxx.demo.XXXXDemo /opt/jars/xxxx.jar
```

在开始任务后，进入WebUI可以看到当前正在运行的任务，并且通过点击Cancel Job取消任务。

**方式二：调用yarn运行**

（无需启动flink集群，但是需要保证yarn是正常的，hadoop需要启动）

```
flink run -m yarn-cluster -c com.xxx.demo.XXXXDemo /opt/jars/xxxx.jar
```

**如果使用yarn方式运行提示hadoop-classpath错误则执行下面的命令：*

```
export HADOOP_CLASSPATH=`hadoop classpath`
```

停止任务的命令：

```
yarn application -kill application_xxxxx
# plication_xxxxx为任务ID，在任务提交结束后可以找终端中找到。
```

# 阶段四 —— 介绍

## 技能点

1. 使用Flume监听端口发送的数据，将数据存至Kafka中对应主题下
2. 使用Flink实时消费Kafka中指定主题数据
3. 对数据进行过滤
4. 对数据进行分组、聚合、开窗计算
5. 将结果存储至MySql、Redis中

## Flink 任务开发步骤

1. 构建Flink运行环境
2. 确认数据来源（测试数据、文件、网络端口、Kafka等）
3. 对数据进行清洗（过滤掉不需要的数据）
4. 确认是否对数据分组（按某个类别进行汇总时需要分组）
5. 确认是否需要开窗（按时间统计时，如统计每10秒内的滚动窗口，或每5秒统计近10秒内的滑动窗口数据）
6. 对结果进行聚合计算（内置的聚合函数、或者自定义聚合函数）
7. 将最终结果存储到指定位置（打印、存储至Redis、存储至MySql等）
8. 提交执行任务

# 阶段四 —— 步骤

## 构建Flink运行环境

```
// 构建flink运行环境对象
val env = StreamExecutionEnvironment.getExecutionEnvironment
//设置并行度
env.setParallelism(1)

// 业务代码块...

// 提交执行任务
env.execute("类名")
```

## 获取数据流

数据来源可能包含文件、网络端口、Kafka等。

```
// 从本地文件获取获取
val inputStream = env.readTextFile("文件路径")
// 从网络端口获取数据
val inputStream = env.socketTextStream("localhost",26001)
// 从kafka获取数据
val prop = new Properties()
prop.setProperty("bootstrap.servers","kafka集群ip:9092","kafka集群ip:9092","kafka集群ip:9092")
prop.setProperty("group.id","test")
val inputStream3 = env.addSource(new FlinkKafkaConsumer[String]("kafka中主题名",new SimpleStringSchema(),prop))
```

注意：new SimpleStringSchema 导包时会出现两个包，此时需要选择 `flink.api.common` 下的包。

## 时间窗口

Flink 中的时间概念一般有三种：

- 事件时间（一般在数据中会自带，在需要在代码中指定）
- 处理时间（默认，进入flink的处理时间）
- 数据进入flink的时间

***注意：数据中自带的事件时间戳，如果是10位，则是秒，需要乘以1000转换为毫秒。如果是13秒则为毫秒时间戳，不需要转换。**

```
// 时间的设定
env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime)
dataStream.assignAscendingTimestamps((x) => {
    // 提取x中某列数据作为事件时间，并转为毫秒Long型
    val listen_time = xxxx.toLong*1000
    listen_time
})
```

## 开窗操作

Flink 可以按照时间统计某一时间段内的数据，窗口最为常见有：

- 滚动窗口（数据不会重叠多个窗口），如：统计每8分钟内的信息
- 滑动窗口（同一数据会出现在多个窗口），如每3分钟统计近8分钟内的数据

```
// 开滚动窗口简写，8代表8分钟窗口
dataStream.timeWindow(Time.minutes(8))
// 开滑动窗口简写，8代表窗口为8分钟，3代表每3分钟滑动一次
dataStream.timeWindow(Time.minutes(8),Time.minutes(3))
```

## 数据清洗

```
val dataStream = inputStream.filter(x=>{
    // 将数据文本开头不为0的过滤掉
    if(x.startsWith("0")){
        true
    }
    else{
        false
    }
})
```

## 确认是否对数据分组

```
// 如果使用map需要导包，只需要将以上导包的scala.后面改为下划线
import org.apache.flink.streaming.api.scala._
```

```
// 计算过滤后的数据总条数
// 将dataStream进行map，每条数据以对偶元组的方式进行返回以便统计总条数
dataStream.map((x) => {
    ("key",1)
    // 将数据进行分组，分组的Key为元组的第一个元素
    // 将分组后的结果进行求和，求和的位置为1（位置0为Key，所以取Value的位置）
}).keyBy(_._1).sum(1)
```

## 将数据保存或打印

### 打印数据

```
resultStream.print()
```

### 存储到 Redis

```
// 创建Redis配置对象
val conf = new FlinkJedisPoolConfig.Builder().setHost("Redis服务器
ip").setPort(6379).build()

// 设置resultStream的输出流，指定为Redis
resultStream.addSink(new RedisSink[结果流的类型对应](conf,new RedisMapper[(String,
Double)] {
    override def getCommandDescription: RedisCommandDescription =   new
RedisCommandDescription(RedisCommand.命令（选择存储到Redis的数据类型）)
    override def getKeyFromData(t: (String, Double)): String = 存入redis中的key
    override def getValueFromData(t: (String, Double)): String = 存入redis中的value（必须
是字符串，如果不是则需要转换为字符串类型）
}))
```

```
// 说明：使用DataStream的addSink方法，为其添加输出源，输出源指定为RedisSink对象，该对象传递两个参数，分别是conf与ReidsMapper对象，ReidsMapper对象需要提供三个参数，第一个参数是Reids命令，也就是需要构建一个RedisCommandDescription对象，这个对象需要传递的参数为RedisCommand.命令。

// 存储为字符串格式
override def getCommandDescription: RedisCommandDescription = new
RedisCommandDescription(RedisCommand.SET)

// 存储为Hash格式
override def getCommandDescription: RedisCommandDescription = new
RedisCommandDescription(RedisCommand.HSET,"user_borrow_totals") // Hash表的key在此处传递，下面参数的Key就是Hash表中的filed
```

## 存储到 Mysql

由于Flink没有自带的Sinik到Mysql的方法，需要自己定义一个Sink方法。

```
class MySqlSink extends RichSinkFunction[(结果流的类型)]{
    // 先定义好JDBC连接对象与执行对象
    var conn:Connection = _
    var pstmt:PreparedStatement = _
    override def open(parameters: Configuration): Unit = {
        conn = DriverManager.getConnection("jdbc:mysql://192.168.6.86:3306/bike_db?
characterEncoding=utf-8","root","123456")
        pstmt = conn.prepareStatement("插入表的sql语句，带占位符？的")
    }
    override def invoke(value: (String, Double), context: SinkFunction.Context[_]):
Unit = {
        //将每行结果的字段取出对上sql的问号，最后执行
        pstmt.setString(1,value._1)
        pstmt.setDouble(2,value._2)
        pstmt.executeUpdate()
    }
    override def close(): Unit = {
        pstmt.close()
        conn.close()
    }
}
```

```
// 使用时指定Sink方式为自定义的Sink对象即可
resultStream.addSink(new MySqlSink())
```

## 注意事项

*注意：Flink连接Mysql或Redis前，需要将提供的放入flink下lib的架包全部拷贝进去，集群其他机器也要拷贝，否则将会提示找不到对应的类。