

实验八 Flink 部署

8.1 实验目的

- 学习 Flink 的部署，简单使用 Scala Shell
- 查看 Flink 的运行日志，体会与 Storm 运行过程中日志的区别
- 通过系统部署理解体系架构，体会流计算系统与批处理系统之间的区别

8.2 实验任务

- 完成 Flink 的单机伪分布式部署以及分布式部署
- 两种部署方式下分别以 Attached 和 Detached 提交方式运行示例程序

8.3 实验环境

- 操作系统：Ubuntu 18.04
- JDK 版本：1.8
- Flink 版本：1.12.1

8.4 实验步骤

8.4.1 单机集中式部署

(1) 准备工作

- 登录用户 dase-local
- 下载并安装 Flink

```
1 wget http://archive.apache.org/dist/flink/flink-1.12.1  
2   /flink-1.12.1-bin-scala_2.11.tgz  
3 tar -zxvf flink-1.12.1-bin-scala_2.11.tgz
```

(2) 使用 Shell 运行 DataStream 程序

- 启动 Scala Shell

```
~/flink-1.12.1/bin/start-scala-shell.sh local # 连接到本地Flink集群
```

- 在 `scala>` 后输入 Scala 代码

```

1 val textstreaming=senv.fromElements("a a b b c")
2 val countsstreaming=textstreaming.flatMap { _.toLowerCase.split("\\\\W+") }
3 .map { (_, 1) }.keyBy(0).sum(1)
4 countsstreaming.print()
senv.execute() // 提交作业

```

运行结果如图8.1所示。

```

scala> senv.execute()
(a,1)
(a,2)
(b,1)
(b,2)
(c,1)

```

图 8.1 Shell 运行示例程序的结果

- 退出 Scala Shell，在 *scala>* 后输入:*q*

8.4.2 单机伪分布式部署

a) 准备工作

- 登录用户 *dase-local*

b) 修改配置

配置文件位于 *~/link-1.12.1/conf/* 目录下，修改其中的 *flink-conf.yaml* 文件。

注意：配置 *key/value* 时候在 “*:*” 后面需要有一个空格，否则配置不会生效。此外，不要将注释与配置项写在同一行。

```

1 jobmanager.rpc.address: localhost
2 # 配置 JobManager 进行 RPC 通信的地址，使用默认值即可
3
4 jobmanager.rpc.port: 6123
5 # 配置 JobManager 进行 RPC 通信的端口，使用默认值即可
6
7 rest.port: 8081
8 # 客户端访问端口与可视化端口，使用默认值即可
9
10 taskmanager.numberOfTaskSlots: 2

```

```

11 # 配置TaskManager 提供的任务 slots 数量大小，默认为1
12
13 parallelism.default: 1
14 # 配置程序默认并行计算的个数，使用默认值即可

```

以下列举一些重要的配置值（需要调节时更改，本例中不做更改）：

```

1 jobmanager.heap.mb: 每个JobManager的可用内存量
2
3 taskmanager.heap.mb: 每个TaskManager的可用内存量
4
5 taskmanager.numberOfTaskSlots: 每台机器的可用CPU数量
6
7 parallelism.default: 集群中的CPU总数
8
9 taskmanager.tmp.dirs: 临时目录
10 #内存不够用时，写入到taskmanager.tmp.dirs指定的目录中。如果未显式指定参数，  
Flink会将临时数据写入操作系统的临时目录。

```

c) 启动服务

- 启动命令

```
1 ~/flink-1.12.1/bin/start-cluster.sh
```

d) 查看 Flink 服务信息

- 使用 jps 查看进程，验证是否成功启动服务

在单机伪分布式部署方式下，该节点既充当 JobManager 角色，又充当 TaskManager 角色，故该节点上会出现两个进程：一个 JobManager 进程和一个 TaskManager 进程。若同时出现 JobManager 进程和 TaskManager 进程，则表明配置成功以及启动成功。注意，在 Standalone 模式下，JobManager 的进程名为 StandaloneSessionClusterEntrypoint，如图8.2所示。

```

dase-local@ecnu01:~$ jps
22534 StandaloneSessionClusterEntrypoint
22968 Jps
22808 TaskManagerRunner
dase-local@ecnu01:~$ █

```

图 8.2 启动 Flink 服务后存在的进程

- 查看 Flink 进程日志

本实验的进程日志记录在 `~/flink-1.12.1/log/` 路径下，后缀为.log 的文件中。

- Client 进程日志: *flink-*-client-*.**log*
- Scala Shell 进程日志: *flink-*-scala-shell-local-*.**log*
- StandaloneSession 进程日志: *flink-*-standalonesession-*-*.**log*
- TaskExecutor 进程日志: *flink-*-taskexecutor-*-*.**log*
- 访问 Flink UI 界面

通过访问 <http://localhost:8081>(该端口号为 flink-conf.yaml 中的 rest.port), Flink 集群信息, 如图8.3所示。

因为之前在 flink-conf.yaml 中设置 taskmanager.numberOfTaskSlots 的值为 2, 故每个 TaskManager 有 2 个 slot。



图 8.3 Flink UI 界面

e) 运行 Flink DataStream 程序

- 使用 Shell 运行 DataStream 程序

- 启动 Scala Shell

```
1 ~/flink-1.12.1/bin/start-scala-shell.sh remote localhost 8081 #
连接本地单机伪分布式集群
```

- 在 *scala>* 后输入 Scala 代码

```
1 val textstreaming=senv.fromElements("a a b b c")
2 val countsstreaming=textstreaming.flatMap {
3   _.toLowerCase.split("\\W+") }.map { (_, 1) }.keyBy(0).sum(1)
4 countsstreaming.print()
5 senv.execute() // 提交作业
```

另起一个终端, 输入如下命令:

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

运行结果如图8.4所示。

- 退出 Scala Shell, 在 *scala>* 后输入:*q*

```
(a,1)
(a,2)
(b,1)
(b,2)
(c,1)
```

图 8.4 Shell 运行示例程序的结果

- [以 Attached 方式] 提交 jar 包运行 DataStream 程序

- 启动 Netcat 服务

另起一个终端（记此终端为终端 2.1），输入如下命令：

```
1 nc -lk 8888
2 #监听8888端口，之后我们会在此输入数据
```

- 提交 jar 包

另起一个终端（记此终端为终端 2.2），并在该终端中输入相应提交命令来提交 jar 包。注意，不同的提交方式所对应的提交命名有所不同。

```
1 ~/flink-1.12.1/bin/flink run
  ~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar
  --port 8888
```

在终端 2.1 中输入如图 8.5 所示的数据。另起一个终端，输入如下命令查看运行结果，如图 8.6 所示，可以看到这次的程序运行结果也追加到了 flink-dase-local-taskexecutor-0-ecnu01.out 文件中。注意，若实验过程中 Flink 集群出现过故障，则可能会产生如 flink-dase-local-taskexecutor-1-ecnu01.out、flink-dase-local-taskexecutor-1-ecnu01.out 之类的文件，同时运行结果也会输出到最新的文件中，因此需要将命令中的 flink-dase-local-taskexecutor-0-ecnu01.out 替换为对应的最新的文件即可。

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

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

图 8.5 输入数据

```
(a,1)
(a,2)
(b,1)
(b,2)
(c,1)
hello : 2
ecnu : 1
dase : 1
```

图 8.6 默认方式提交示例程序的部分运行结果

```
dase-local@ecnu01:~$ jps
11762 StandaloneSessionClusterEntrypoint
3155 CliFrontend
4762 Jps
12045 TaskManagerRunner
dase-local@ecnu01:~$
```

图 8.7 程序运行中存在的进程

另起一个终端，输入 jps 查看进程，如图8.7所示。我们可以看到一个 CliFrontend 进程。

- 停止 DataStream 程序

- * 方法 1：终止数据源

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。

- * 方法 2：在 Flink UI 中停止应用程序

访问 <http://localhost:8081>，选中 Running Jobs 标题下对应的应用程序进入详情页面，点击右上角的 Cancel 停止 DataStream 程序。应用程序状态会变为 Canceled。

- * 方法 3：命令行停止应用程序

另起一个终端，使用 list 命令列出正在运行的 Job(其中的 JobID 为一串哈希值)，随后根据 JobID 使用 cancel 命令终止应用程序。终止后，应用程序状态会变为 Canceled。

```
1 ~/flink-1.12.1/bin/flink list
2 ~/flink-1.12.1/bin/flink cancel JobID #用查询到的哈希值替换JobID
```

此外，若未结束 Netcat 服务，则在终端 2.2 中使用 Ctrl + C 结束 Netcat 服

务。

• [以 Detached 方式提交 jar 包运行 DataStream 程序]

- 启动 Netcat 服务

在终端为 2.1 中输入如下命令：

```
1 nc -l k 8888  
2 #监听8888端口，之后我们会在此输入数据
```

- 提交 jar 包

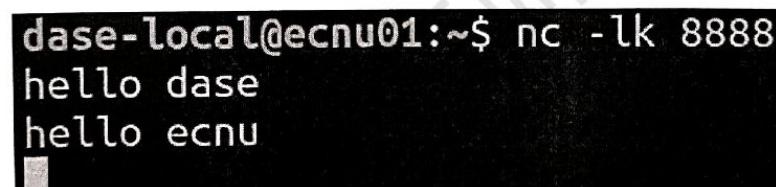
在终端为 2.2 中输入如下命令：

```
1 ~/flink-1.12.1/bin/flink run -d  
~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar  
--port 8888
```

在终端 2.1 中输入数据，如图8.8所示。

[在终端 2.2 中]输入如下命令查看运行结果。如图8.9所示，可以看到这次的程序运行结果也追加到了 flink-dase-local-taskexecutor-0-ecnu01.out 文件中。[值得注意的是，图8.9所显示的运行结果会随着图8.8中输入速度的不同而有所差异。]

```
1 tail -f ~/flink-1.12.1/log/flink-dase-local-taskexecutor-0-ecnu01.out  
# tail -f 默认显示文件最后10行内容
```



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

图 8.8 输入数据



```
(b,2)
(c,1)
hello : 1
dase : 1
hello : 1
ecnu : 1
hello : 1
dase : 1
hello : 1
ecnu : 1
```

图 8.9 detached 方式提交示例程序的部分运行结果

[另起一个终端，输入 jps 查看进程。]此时，不会出现 CliFrontend 进程。

- 停止 DataStream 程序 在完成下一步中的实时查看应用运行情况之后，参照前述方法停止 DataStream 程序。

(3) 查看 Flink 程序运行信息

- #### • 实时查看应用运行情况

在应用运行过程中，访问 <http://localhost:8081>，选中 Running Jobs 标题下对应的应用程序进入详情页面。如图8.10所示，可以从图中看到应用程序的 DAG 图和 Subtasks 信息；还可切换到 Timeline、Exceptions 或 Configuration 等标签页查看相关信息。

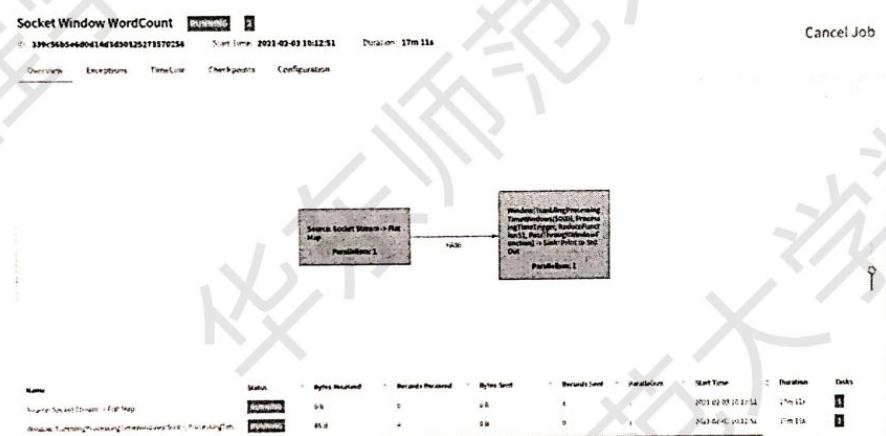


图 8.10 Flink 应用程序界面

- 查看 Flink 应用程序日志

在提交一个应用程序后，在`~/flink-1.12.1/log`下会出现应用程序运行日志（注意日志名称格式为`flink-*taskexecutor-*-.log`, `.out`结尾的文件为应用程序输出结果）。

- #### • 查看应用历史记录



图 8.11 Flink 已完成应用界面

在应用运行结束后，访问 <http://localhost:8081>，点击左侧 Dashboard 的中的 Completed Jobs 可查看应用提交历史记录。Web 界面如图8.11所示。

(4) 停止服务

- ### • 停止命令

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

8.4.3 分布式部署

(1) 准备工作

- 准备 4 台机器，其中包括 1 个主节点（主机名：ecnu01）、2 个从节点（主机名：ecnu02 和 ecnu03）、1 个客户端（主机名：ecnu04）
- 4 台机器均已创建用户 dase-dis
- 4 台机器之间实现免密钥登录
- 登录 dase-dis 用户
- 检查 IP 与主机名映射是否正确，若 IP 地址发生改变，参考第??节进行修改
- 在主节点上将之前下载的 flink-1.12.1-bin-scala_2.11.tgz 拷贝至当前用户目录下并解压

```
1 scp dase-local@localhost:~/flink-1.12.1-bin-scala_2.11.tgz ~/
2 tar -zxvf flink-1.12.1-bin-scala_2.11.tgz
```

(2) 修改配置

以下操作在主节点执行：

配置文件位于 `~/flink-1.12.1/conf/` 目录下，修改其中的 `flink-conf.yaml` 和 `workers` 这 2 个文件。

- 修改 `flink-conf.yaml` 文件

只需修改如下配置项：

```
1 jobmanager.rpc.address: ecnu01
2 #配置JobManager进行RPC通信的地址，修改为主节点主机名
```

- 修改 `workers` 文件

将以下内容添加至文件中（注意注释 `localhost` 一行）：

```
1 # localhost
2 ecnu02
3 ecnu03
```

- 将配置好的 Flink 拷贝到其它节点

```
1 scp -r ~/flink-1.12.1 dase-dis@ecnu02:~
2 scp -r ~/flink-1.12.1 dase-dis@ecnu03:~
3 scp -r ~/flink-1.12.1 dase-dis@ecnu04:~/
```

(3) 启动服务

- 启动命令（在主节点上执行）

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

(4) 查看 Flink 服务信息

- 使用 jps 查看进程，验证是否成功启动服务

在此分布式部署方式下，[主节点充当 JobManager 角色，各从节点充当 TaskManager 角色。分别在主节点和从节点上使用 jps 命令，若在主节点上出现 StandaloneSessionClusterEntrypoint 进程，且在从节点上出现 TaskManagerRunner 进程，则表明配置成功且启动成功。]主节点存在的进程如图 8.12 所示，从节点存在的进程如图 8.13 和 8.14 所示。

```
dase-dis@ecnu01:~$ jps
8252 StandaloneSessionClusterEntrypoint
9535 Jps
dase-dis@ecnu01:~$
```

图 8.12 启动 Flink 服务后主节点存在的进程

```
dase-dis@ecnu02:~$ jps
14786 Jps
11446 TaskManagerRunner
dase-dis@ecnu02:~$
```

图 8.13 启动 Flink 服务后从节点存在的进程

```
dase-dis@ecnu03:~$ jps
699454 TaskManagerRunner
699820 Jps
dase-dis@ecnu03:~$
```

图 8.14 启动 Flink 服务后从节点存在的进程

- 查看 Flink 进程日志

本实验的进程日志记录在 `~/flink-1.12.1/log/` 路径下，后缀为 `.log` 的文件中。

- Client 进程日志: `flink-*-client-* .log`

- Scala Shell 进程日志: *flink-*-scala-shell-local-* .log*
- StandaloneSession 进程日志: *flink-*-standalonesession-*-* .log*
- TaskExecutor 进程日志: *flink-*-taskexecutor-*-* .log*
- 访问 Flink UI 界面

通过访问 <http://ecnu01:8081>(该端口号为 flink-conf.yaml 配置文件中的 rest.port), 查看 Flink 集群信息, 如图8.15所示。

当前有 2 个 TaskManager (ecnu02、ecnu03), 因为没有对配置文件 taskmanager.numberOfTaskSlots 项进行更改, 其默认为 1, 故 Task Slots 的总数为 2。

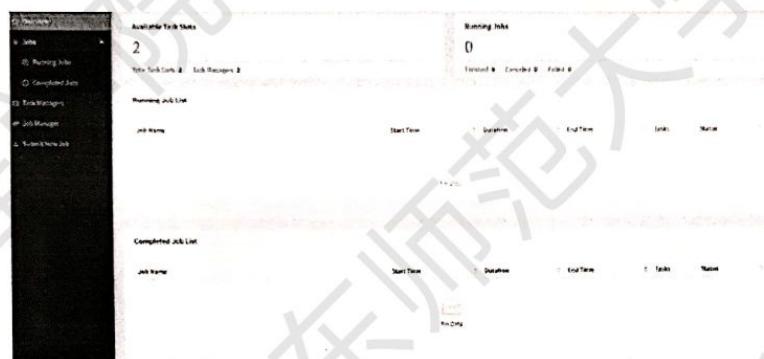


图 8.15 Flink UI 界面

(5) 运行 Flink DataStream 程序

在客户端执行如下操作:

- 通过提交 jar 包运行 DataStream 程序

注意: [通过 2 次独立的实验来尝试两种提交 jar 包的方式, 每次实验重复以下步骤。每次实验需要在“提交 jar 包”步骤中选择一种提交 jar 包的方式来完成。]

- 启动 Netcat 服务

在客户端 (ecnu04) 另起一个终端 (记此终端为终端 3.1), 输入如下命令

```
1 nc -lk 8888
2 #监听8888端口, 之后我们会在此输入数据
```

- 提交 jar 包

在客户端 (ecnu04) 另起一个终端 (记此终端为终端 3.2), 并在该终端中输入相应提交命令来提交 jar 包。注意, 不同的提交方式所对应的提交命名有所不同。

* Attached 提交方式

```
1 ~/flink-1.12.1/bin/flink run
2 ~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar
3 --hostname ecnu04 --port 8888
```

Running Jobs							
Job Name	Start Time	Duration	End Time	Tasks	Status		
Socket Window WordCount	2021-02-03 21:47:25	32m 10s	-	2	RUNNING		

图 8.16 Running Jobs

通过访问 <http://ecnu01:8081>，选中对应的正在运行中的应用程序（如图8.16所示）。进入到应用程序详情页，点击 DAG 图下方 Name 中的 Sink: Print to Std. Out 所在一栏，点击展开信息中的 SubTasks（如图8.17所示），我们得知应用程序运行在 ecnu02 上。这次的程序运行结果会追加到 flink-dase-dis-taskexecutor-0-ecnu02.out 文件中。

Detail	SubTasks	TaskManagers	Watermarks	Accumulators	BackPressure	Metrics
ID	Bytes Received	Records Received	Bytes Sent	Records Sent	Attempt	Host
0	598	3	98	0	1	ecnu02

图 8.17 Subtasks

在 ecnu02 节点上启动一个终端（记此终端为终端 3.3），输入如下命令等待查看运行结果。在终端 3.1 中输入数据，如图8.18。终端 3.3 中会输出数据，如图8.19所示。

```
1 tail -f ~/flink-1.12.1/log/flink-dase-dis-taskexecutor-0-ecnu02.out
#注意将文件名中的主机名要与刚才在UI中获得的Host相对应
```

```
dase-dis@ecnu04:~$ nc -lk 8888
hello dase
hello ecnu
flink flink
```

图 8.18 输入数据

在客户端（ecnu04）上另起一个终端，输入 jps 查看进程，如图8.20所示。我们可以看到一个 CliFrontend 进程，程序运行结束后该进程会消失。其他节点上的进程没有变化。

```
hello : 1
dase : 1
hello : 1
flink : 2
ecnu : 1
```

图 8.19 默认方式提交示例程序的部分运行结果

```
dase-dis@ecnu04:~$ jps
16978 Jps
16659 CliFrontend
dase-dis@ecnu04:~$
```

图 8.20 程序运行中客户端节点存在的进程

* Detached 提交方式

```
~/flink-1.12.1/bin/flink run -d
~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar
--hostname ecnu04 --port 8888
```

我们仍然通过默认提交方式中的方法查看程序输出在哪个节点上，根据展开信息中的 Host（如图8.21所示），我们得知应用程序运行在 ecnu03 上。这次的程序运行结果会追加到 flink-dase-dis-taskexecutor-0-ecnu03.out 文件中。在 ecnu03 节点上启动一个终端（记此终端为终端 3.3），输入如下

	Detail	SubTasks	TaskManagers	Watermarks	Accumulators	BackPressure	>
ID	Received	Bytes Sent	Records Sent	Attempt	Host	Status	More
0		0 B	0	1	ecnu03	RUNNING	...

图 8.21 Subtasks

命令等待查看运行结果。

```
tail -f ~/flink-1.12.1/log/flink-dase-dis-taskexecutor-0-ecnu03.out
#注意将文件名中的主机名要与刚才在UI中获得的Host相对应
```

在终端 3.1 中输入数据，如图8.22，终端 3.3 中会输出数据，如图8.23所示。

此时，不会出现 CliFrontend 进程。

- 停止 DataStream 程序

* 方法 1：在启动 Netcat 服务的窗口使用 Ctrl + C，停止 Netcat 服务。应

```
dase-dis@ecnu04:~$ nc -lk 8888
hi dase
hi ecnu
flink flink
```

图 8.22 输入数据

```
hi : 2
ecnu : 1
dase : 1
flink : 2
```

图 8.23 Detached 方式提交示例程序的部分运行结果

用程序状态会变为 Finished。

- * 方法 2：在 Flink UI 中停止应用程序

访问 <http://ecnu01:8081>，选中 Running Jobs 标题下对应的应用程序进入详情页面，点击右上角的 Cancel 停止 DataStream 程序。应用程序状态会变为 Canceled。

- * 方法 3：命令行停止应用程序

使用如下命令列出正在运行的 Job(其中的 JobID 为一串哈希值)，随后根据 JobID 终止应用程序。应用程序状态会变为 Canceled。

```
1 ~/flink-1.12.1/bin/flink list
2 ~/flink-1.12.1/bin/flink cancel JobID #用查询到的哈希值替换JobID
```

- 若未结束 Netcat 服务，则在终端 3.2 中使用 Ctrl + C 结束 Netcat 服务

(6) 查看 Flink 程序运行信息

- 实时查看应用运行情况

在应用运行过程中，访问 <http://ecnu01:8081>，选中 Running Jobs 标题下对应的应用程序进入详情页面。如图8.24所示，可以从图中看到应用程序的 DAG 图和 Subtasks 信息，还可切换到 Timeline、Exceptions 或 Configuration 等标签页查看相关信息。

- 查看 Flink 应用程序日志

在提交一个应用程序后，在 `~/flink-1.12.1/log` 下会出现应用程序运行日志（注意日志名称格式为 `flink-*taskexecutor-*-.log`, `.out` 结尾的文件为应用程序输出结果）。

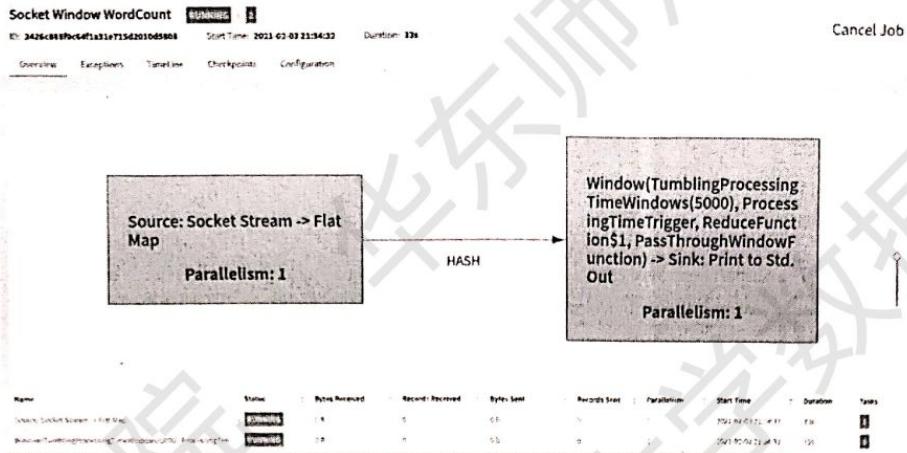


图 8.24 Flink 应用程序界面

- 查看应用历史记录

在应用运行结束后，访问 <http://ecnu01:8081>，点击左侧 Dashboard 的中的 Completed Jobs 可查看应用提交历史记录。Web 界面如图8.25所示。

Completed Jobs						
Job Name	Start Time	Duration	End Time	Tasks	Status	
Socket Window WordCount	2021-02-03 21:43:32	7m 13s	2021-02-03 21:41:48	2 / 2	FINISHED	
Socket Window WordCount	2021-02-03 21:17:24	16m 34s	2021-02-03 21:33:59	2 / 2	FINISHED	

图 8.25 Flink 已完成应用界面

(7) 停止服务

在主节点执行如下命令：

- 停止命令

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

8.5 思考题

1 在部署 Flink 时，具体是通过 flink.yaml 配置文件中的什么参数来指定每个 TaskManager 中的 slots 数量的？

2 一个 Flink 流计算应用程序何时停止、如何停止？请通过与 Spark 批处理应用程序进行对比来阐述。