

实验七 Flink 部署与编程 (分布式)

7.1 实验目的

- 学习在 Yarn 平台上部署 Flink，理解 Standalone 模式与 Yarn 模式的区别
- 通过 Yarn 模式运行 Flink 程序，体会与 Standalone 模式下运行 Flink 程序的区别
- 了解 Attached 和 Detached 两种提交方式之间的差异

7.2 实验任务

- 完成 Flink Standalone 的分布式部署以及 Flink on Yarn 的分布式部署
- 在 Flink Standalone 的分布式部署方式下以 Attached 和 Detached 提交方式运行示例程序
- 在 Flink on Yarn 的分布部署方式下在 Per-Job 运行模式中分别按照 Attached 和 Detached 方式提交应用程序

7.3 实验环境

- 操作系统: Ubuntu 18.04 或 Ubuntu 20.04
- JDK 版本: 1.8
- Flink 版本: 1.12.1

7.4 实验步骤

注意: 本教程以名为 `dase-local` 或 `dase-dis` 的用户为例开展实验，读者在实验过程中需替换为自己的用户名（云主机上的默认用户名为 `ubuntu`）。

7.4.1 Flink Standalone 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 台机器之间实现免密钥登录
- 检查 IP 与主机名映射是否正确，若 IP 地址发生改变，需要重新在 4 台机器之间实现免密钥登录

(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）

```
1 ~/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 `TaskManager-Runner` 进程，则表明配置成功且启动成功。主节点存在的进程如图 7.1 所示，从节点存在的进程如图 7.2 和 7.3 所示。

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

图 7.1 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 7.2 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 7.3 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 7.4 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。

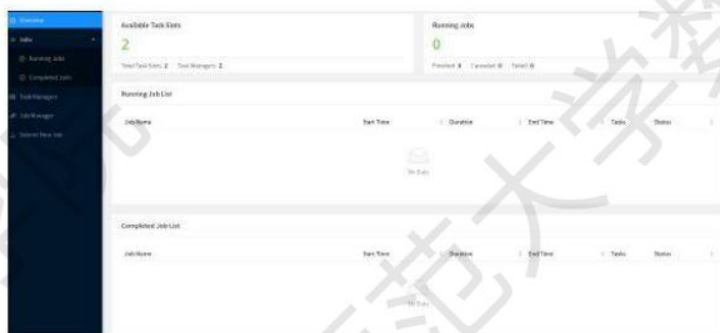


图 7.4 Flink UI 界面

(5) 通过提交 jar 包运行 Flink DataStream 程序

• Attached 方式（默认）

– 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。

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

Running Jobs

Job Name	Start Time	Duration	End Time	Tasks	Status
Socket Window WordCount	2021-02-03 21:47:26	32m 10s	-	2 / 2	RUNNING

图 7.5 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7.5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 所在的一栏，点击展开信息中的 SubTasks（如图7.6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
0	59 B	0 B	0 B	0 B	0 B	ecnu02 RUNNING

图 7.6 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7.7。终端 3.3 中会输出数据，如图7.8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
```

在客户端（ecnu04）上另起一个终端，输入 `jps` 查看进程，如图7.9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。

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

图 7.7 输入数据

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

图 7.8 Attached 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 7.9 程序运行中客户端节点存在的进程

— 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 3.1 中使用 Ctrl + C 结束 Netcat 服务。

- Detached 方式

- 启动 Netcat 服务

在终端为 3.1 中输入如下命令：

```
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

在终端为 3.2 中输入如下命令：

```
1 ~/flink-1.12.1/bin/flink run -d
  ~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar
  --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7.10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。

ID	eived	Bytes Sent	Records Sent	Attempt	Host	Status	More
0	0B	0	0	1	ecnu03	RUNNING	

图 7.10 Subtasks

在终端 3.3 中输入如下命令等待查看运行结果。

```
1 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7.11，终端 3.3 中会输出数据，如图7.12所示。

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

图 7.11 输入数据

此时，不会出现 CliFrontend 进程。

(6) 查看 Flink 程序运行信息

- 实时查看应用运行情况


```

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

```

图 7.12 Detached 方式提交示例程序的部分运行结果

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 7.13 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。

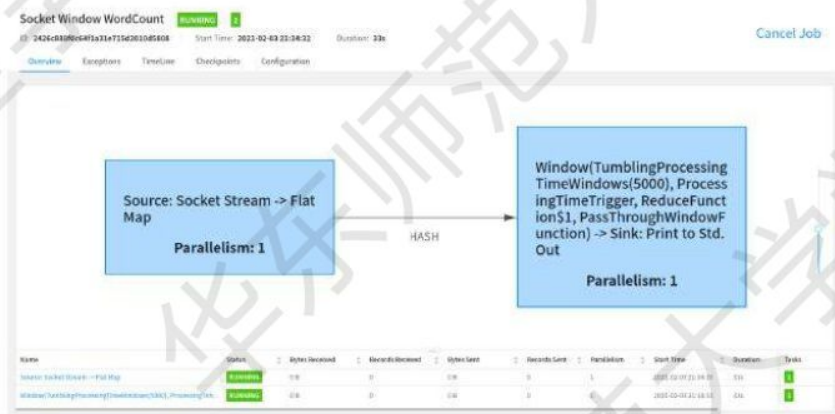


图 7.13 Flink 应用程序界面

• 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

在应用运行结束后，访问 <http://ecnu01:8081>，点击左侧 Dashboard 中的 Completed Jobs 可查看应用提交历史记录。Web 界面如图 7.14 所示。

Completed Jobs					
Job Name	Start Time	Duration	End Time	Tasks	Status
Socket Window WordCount	2021-02-09 21:34:32	7m 15s	2021-02-09 21:41:48	1	FINISHED
Socket Window WordCount	2021-02-09 21:37:34	16m 30s	2021-02-09 21:53:59	1	FINISHED

图 7.14 Flink 已完成应用界面

(7) 运行 WordCount 示例程序

- 将之前打好的 WordCount jar 包上传至 `~/flink-1.12.1/myApp` 路径中
- 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 UI 定位输出文件路径，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

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。

(8) 停止服务

在主节点进行如下操作：

- 停止 DataStream 程序
如有正在运行的 DataStream 程序，参照前述方法停止 DataStream 程序。
- 停止 Flink

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

7.4.2 Flink on Yarn 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 台机器之间实现免密钥登录
- 检查 IP 与主机名映射是否正确，若 IP 地址发生改变，需要重新在 4 台机器之间实现免密钥登录
- 修改 .bashrc 文件 (各节点都需要此操作)

```
1 vi ~/.bashrc
```


添加如下内容：

```
1 export HADOOP_HOME=/home/dase-dis/hadoop-2.10.1
2 export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
3 export HADOOP_CLASSPATH=$(hadoop classpath)
```

使.bashrc 配置生效：

```
1 source ~/.bashrc
```

- 修改 Hadoop 和 Yarn 的配置（参考实验“Hadoop 2.x 部署”分布式部署）

(2) 启动服务

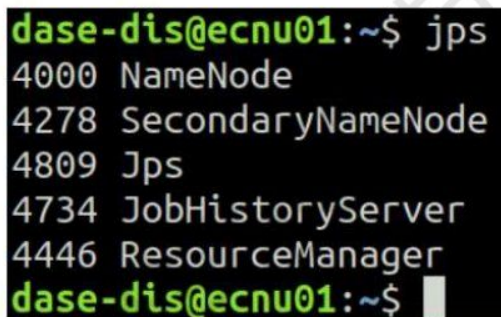
- 启动命令

```
1 ~/hadoop-2.10.1/sbin/start-dfs.sh
2 ~/hadoop-2.10.1/sbin/start-yarn.sh
3 ~/hadoop-2.10.1/sbin/mr-jobhistory-daemon.sh start historyserver
```

(3) 查看 Yarn 服务信息

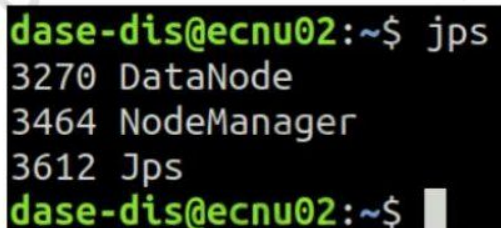
- 使用 jps 查看进程, 验证是否成功启动服务

对比验证正常出现的进程，如图7.15、图7.16和图7.17所示。



```
dase-dis@ecnu01:~$ jps
4000 NameNode
4278 SecondaryNameNode
4809 Jps
4734 JobHistoryServer
4446 ResourceManager
dase-dis@ecnu01:~$
```

图 7.15 启动 Yarn 服务后主节点存在的进程



```
dase-dis@ecnu02:~$ jps
3270 DataNode
3464 NodeManager
3612 Jps
dase-dis@ecnu02:~$
```

图 7.16 启动 Yarn 服务后从节点存在的进程

```
dase-dis@ecnu03:~$ jps
3105 DataNode
3301 NodeManager
3449 Jps
dase-dis@ecnu03:~$
```

图 7.17 启动 Yarn 服务后从节点存在的进程

- 查看 Yarn 进程日志

本实验 Yarn 相关的进程日志记录在 `~/hadoop-2.10.1/logs/`，前缀为 `yarn`，后缀为 `.out` 的文件中。

- ResourceManager 进程日志：
默认位置：`~/hadoop-2.10.1/logs/yarn-*-resourcemanager-*.log`
- NodeManager 进程日志：
默认位置：`~/hadoop-2.10.1/logs/yarn-*-nodemanager-*.log`

- 访问 Yarn Web 界面

通过 `http://ecnu01:8088`，如图 7.18 所示。



图 7.18 Yarn 界面

(4) Per-Job 运行模式

在客户端 (ecnu04) 执行如下操作。

- 以 Attached (默认) 方式通过提交 jar 包运行 DataStream 程序

- 启动 NetCat 服务

在客户端 (ecnu04) 启动一个终端 (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)，以 Attached 方式提交 jar 包。

```
1 ~/flink-1.12.1/bin/flink run -t yarn-per-job
   ~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar
   --hostname ecnu04 --port 8888
```

提交 jar 包后，可以看到输出信息中的 Yarn 应用信息，如图7.19所示。可以看到为了执行这个单独的作业，在 Yarn 中提交了一个名称为 application_1620481595386_0001 的作业。

```
[*] - Submitted application application_1620481595386_0001
2021-05-08 22:01:57,429 INFO org.apache.flink.yarn.YarnClusterDescriptor
[*] - Waiting for the cluster to be allocated
2021-05-08 22:01:57,431 INFO org.apache.flink.yarn.YarnClusterDescriptor
[*] - Deploying cluster, current state ACCEPTED
2021-05-08 22:02:01,213 INFO org.apache.flink.yarn.YarnClusterDescriptor
[*] - YARN application has been deployed successfully.
2021-05-08 22:02:01,214 INFO org.apache.flink.yarn.YarnClusterDescriptor
[*] - Found Web Interface ecnu03:23385 of application 'application_1620481595386_0001'.
Job has been submitted with JobID 568bd17c2291ae604aa5d5bd5cbe6362
```

图 7.19 提交 jar 包的部分输出信息

访问 <http://ecnu01:8088>，点击名称为 application_1620481595386_0001 的应用记录的 Tracking UI 列中的 ApplicationMaster，跳转至 Flink Web 界面，点击 Subtasks 中包含 Sink: Print to Std. 的任务，如图7.20，可以看到程序运行在 ecnu02 上。

Detail	SubTasks	TaskManagers	Watermarks	Accumulators	BackPressure	Metrics		
ID	ceived	Records Received	Bytes Sent	Records Sent	Attempt	Host	Status	More
0		0	0 B	0	1	ecnu02	RUNNING	...

图 7.20 Flink Web 页面中的 Subtasks

在终端 3.1 中输入数据，如图7.21所示。

```
dase-dis@ecnu04:~$ nc -lk 8888
Hello Per-Job Mode
Hello Dis
Hello flink on yarn
```

图 7.21 输入数据

— 查看输出结果

在 ecnu02 节点（JobManager 所在节点）执行如下操作：

```
1 cd ~/hadoop-2.10.1/logs/userlogs/application_1620481595386_0001 #
   与提交到Yarn上的应用程序名称对应
2 ls #查看应用文件夹中的内容
3 cat container_WW_XX_YY_ZZ/taskmanager.out #
   container_WW_XX_YY_ZZ为其中编号最大的文件夹
```

上述命令代表：在 ecnu02 节点上进入 `/home/dase-dis/hadoop-2.10.1/logs/userlogs`

目录，打开文件夹 `application_1620481595386_0001`，再在这个文件夹中查看编号最大的文件其中的 `taskmanager.out` 文件。文件内容如图7.22所示。

```
Hello : 1
Mode : 1
Per-Job : 1
Hello : 1
Dis : 1
Hello : 1
yarn : 1
on : 1
flink : 1
```

图 7.22 输出文件内容

在各个节点另起一个终端，输入 `jps` 查看进程。我们可以看到 `ecnu01`、`ecnu04` 节点无特殊进程存在。如图7.23、7.24所示，我们可以看到 `ecnu02` 节点上存在一个 `YarnTaskExecutorRunner` 进程，`ecnu03` 节点上存在一个 `YarnJobClusterEntrypoint` 进程。此外，如图7.25所示，客户端节点还存在一个 `CliFrontend` 进程。

```
dase-dis@ecnu02:~$ jps
4005 Jps
3270 DataNode
3464 NodeManager
3935 YarnTaskExecutorRunner
dase-dis@ecnu02:~$
```

图 7.23 程序运行中从节点（`ecnu02`）存在的进程

```
dase-dis@ecnu03:~$ jps
3105 DataNode
3301 NodeManager
3785 YarnJobClusterEntrypoint
3881 Jps
dase-dis@ecnu03:~$
```

图 7.24 程序运行中从节点（`ecnu03`）存在的进程

```
dase-dis@ecnu04:~$ jps
3399 CliFrontend
3743 Jps
dase-dis@ecnu04:~$
```

图 7.25 程序运行中客户端存在的进程

– 终止程序

在终端 3.1 中，使用 Ctrl + C 结束 NetCat 服务以停止正在运行的 Flink 程序。

如程序的输入源无法停止，则可以使用如下命令终止程序运行。其中，‘application_XXXX_YY’ 需替换为提交到 Yarn 上的应用程序名称。

```
1 yarn application -kill application_XXXX_YY
```

• 以 Detached 方式通过提交 jar 包运行 DataStream 程序

– 启动 NetCat 服务

在客户端 (ecnu04) 启动一个终端 (记此终端为终端 4.1)，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 4.2)，以 Detached 方式提交 jar 包。

```
1 ~/flink-1.12.1/bin/flink run -t yarn-per-job --detached
  ~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar
  --hostname ecnu04 --port 8888
```

提交 jar 包后，可以看到输出信息中的 Yarn 应用信息，如图 7.26 所示。可以看到为了执行这个单独的作业，在 Yarn 中提交了一个名称为 application_1620481595386_0002 的作业。

访问 <http://ecnu01:8088>，点击名称为 application_1620481595386_0002 的应用记录的 Tracking UI 列中的 ApplicationMaster，跳转至 Flink Web 界面，点击 Subtasks 中包含 Sink: Print to Std. 的任务，如图 7.27，可以看到程序运行在 ecnu03 上。

在终端 4.1 中输入数据，如图 7.28 所示。

– 查看输出结果


```

[ ] - Submitted application application_1620481595386_0002
2021-05-08 22:20:25,327 INFO org.apache.flink.yarn.YarnClusterDescriptor
[ ] - Waiting for the cluster to be allocated
2021-05-08 22:20:25,329 INFO org.apache.flink.yarn.YarnClusterDescriptor
[ ] - Deploying cluster, current state ACCEPTED
2021-05-08 22:20:28,600 INFO org.apache.flink.yarn.YarnClusterDescriptor
[ ] - YARN application has been deployed successfully.
2021-05-08 22:20:28,601 INFO org.apache.flink.yarn.YarnClusterDescriptor
[ ] - The Flink YARN session cluster has been started in detached mode. In order to stop
Flink gracefully, use the following command:
$ echo "stop" | ./bin/yarn-session.sh -id application_1620481595386_0002
If this should not be possible, then you can also kill Flink via YARN's web interface or
via:
$ yarn application -kill application_1620481595386_0002
Note that killing Flink might not clean up all job artifacts and temporary files.
2021-05-08 22:20:28,602 INFO org.apache.flink.yarn.YarnClusterDescriptor
[ ] - Found Web Interface ecnu03:31085 of application 'application_1620481595386_0002'.
Job has been submitted with JobID c9eae56af5d56638b3df354e6dbdb2b
dase-dis@ecnu04:~$

```

图 7.26 提交 jar 包的部分输出信息

Detail	Subtasks	TaskManagers	Watermarks	Accumulators	BackPressure	Metrics
ID	Received	Records Received	Bytes Sent	Records Sent	Attempt	Hook
0	10	0.0	0.0	0	1	ecnu03
						RUNNING

图 7.27 Flink Web 页面中的 Subtasks

在 ecnu03 节点 (JobManager 所在节点) 执行如下操作:

```

1 cd ~/hadoop-2.10.1/logs/userlogs/application_1620481595386_0001 #
   与提交到Yarn上的应用程序名称对应
2 ls #查看应用文件夹中的内容
3 cat container_WW_XX_YY_ZZ/taskmanager.out #
   container_WW_XX_YY_ZZ为其中编号最大的文件夹

```

上述命令代表:在 ecnu03 节点上进入 `/home/dase-dis/hadoop-2.10.1/logs/userlogs` 目录, 打开文件夹 `application_1620481595386_0002`, 再在这个文件夹中查看编号最大的文件夹中的 `taskmanager.out` 文件。文件内容如图7.29所示。

```

Hello : 1
Mode : 1
Per-Job : 1
Hello : 1
Detached : 1
Dis : 1
Hello : 1
yarn : 1
on : 1
flink : 1

```

图 7.29 输出文件内容

在各个节点另起一个终端, 输入 `jps` 查看进程。我们可以看到 ecnu01、ecnu02 和 ecnu04 节点无特殊进程存在。如图7.30、7.31所示, 我们可以看到 ecnu03 节


```
dase-dis@ecnu04:~$ nc -lk 8888
Hello Per-Job Mode
Hello Dis Detached
Hello flink on yarn
```

图 7.28 输入数据

点上存在一个 `YarnJobClusterEntrypoint` 进程和 `YarnTaskExecutorRunner` 进程，程序运行结束后这些进程将消失。在 `Detached` 提交方式下，`CliFrontend` 进程一旦成功提交 jar 包，`CliFrontend` 进程就会退出，不保持客户端与集群的连接。因此，我们可以观察到 `CliFrontend` 进程短暂出现又消失的现象。

```
dase-dis@ecnu02:~$ jps
3270 DataNode
3464 NodeManager
4139 Jps
dase-dis@ecnu02:~$
```

图 7.30 程序运行中从节点 (ecnu02) 存在的进程

```
dase-dis@ecnu03:~$ jps
3105 DataNode
3301 NodeManager
4456 Jps
4249 YarnJobClusterEntrypoint
4364 YarnTaskExecutorRunner
dase-dis@ecnu03:~$
```

图 7.31 程序运行中从节点 (ecnu03) 存在的进程

(5) 查看 Yarn 程序运行信息

- 实时查看应用运行情况

访问 <http://ecnu01:8088>，点击对应名称的应用记录的 Tracking UI 列中的 `ApplicationMaster`，跳转至 Flink Web 界面，如图 7.32 所示。

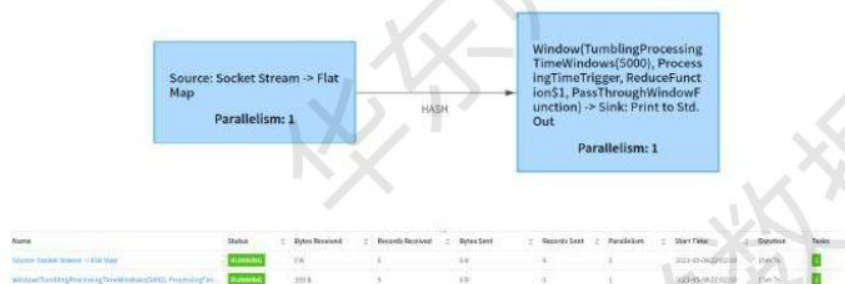


图 7.32 Flink 应用程序界面

- 查看 Yarn 应用程序日志

在提交一个应用程序后，在 `~/hadoop-2.10.1/logs/userlogs/` 下会出现应用程序日志文件夹（文件夹名称格式如 `application_*`）。在应用程序日志文件夹下又会有多个 Container 文件夹（文件夹名称格式如 `container_***`）。每个 Container 文件夹中包含以下日志：

- * 任务输出信息: `taskmanager.out`
- * 任务服务信息日志: `taskmanager.log`
- * 任务错误信息日志: `taskmanager.err`

- 访问 Yarn Web 界面

通过 `http://ecnu01:8088`，可以看到本次启动 Yarn 后所提交的应用程序，如图 7.33 所示。

ID	User	Name	Application Type	Queue	Application Priority	Start Time	Launch Time	Finalized Time	State	Final Status	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Allocated GPUs	Reserved CPU Vcores
application_1612450151718_8005	data	flink	Application	flink	default	0	Thu Feb 9 18:54:03 +0800 2023	Thu Feb 9 18:54:03 +0800 2023	Thu Feb 9 18:54:03 +0800 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A
application_1612450151718_8006	data	flink	per-job	flink	default	0	Thu Feb 9 18:55:25 +0800 2023	Thu Feb 9 18:55:25 +0800 2023	Thu Feb 9 18:54:06 +0800 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A
application_1612450151718_8007	data	flink	session	flink	default	0	Thu Feb 9 18:54:08 +0800 2023	Thu Feb 9 18:54:08 +0800 2023	Thu Feb 9 18:17:31 +0800 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A

图 7.33 Yarn All Applications

(6) 停止服务

在主节点进行如下操作：

- 停止 Flink DataStream 程序

在终端 4.1 中，使用 `Ctrl + C` 结束 NetCat 服务以停止正在运行的 Flink 程序。

如程序的输入源无法停止，则可以使用如下命令终止程序运行。其中，‘`application_XXXX_YY`’需替换为提交到 Yarn 上的应用程序名称。

```
yarn application -kill application_XXXX_YY
```

- 停止命令

```
1 ~/hadoop-2.10.1/sbin/stop-dfs.sh
2 ~/hadoop-2.10.1/sbin/stop-yarn.sh
3 ~/hadoop-2.10.1/sbin/mr-jobhistory-daemon.sh stop historyserver
```

7.5 思考题

1 除了 Per-Job 运行模式之外，Flink on Yarn 还支持 Application 运行模式，请采用 Application 运行模式来运行 Flink 程序，并通过 jps 命令观察 Application 运行模式与 Per-Job 运行模式下的进程有何不同。使用 Application 运行模式来运行 Flink 程序的命令如下所示：

```
1 ~/flink-1.12.1/bin/flink run-application -t yarn-application
  ~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar --port 8888
  #以Application运行模式运行Flink程序，监听8888端口的数据
```

2 上述 Flink 程序中仅包含一个 Flink 作业，然而一个 Flink 程序中允许包含多个 Flink 作业。附录A是一个包含 2 个作业的 Flink 程序，请分别使用 Per-Job 运行模式和 Application 运行模式来运行该 Flink 程序，并结合 jps 命令显示的进程名称、Yarn 和 Flink 两者的 Web UI 的主界面，观察两种运行模式分别启动了多少个 Flink 集群（一个 Flink 集群包含一个 JobManager 和若干个 TaskManager，当然，JobManager 和 TaskManager 具体的进程名在不同的运行模式下可能是不同的）？（额外提示：请注意流计算程序是长时运行的。）

附录

A 包含两个作业的 Flink 示例程序

```
1 import org.apache.flink.api.common.functions.FlatMapFunction;
2 import org.apache.flink.api.common.functions.ReduceFunction;
3 import org.apache.flink.streaming.api.datastream.DataStream;
4 import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
5 import org.apache.flink.util.Collector;
6
7 import org.apache.flink.api.common.functions.FlatMapFunction;
8 import org.apache.flink.api.common.functions.MapFunction;
9 import org.apache.flink.api.java.tuple.Tuple2;
10 import org.apache.flink.streaming.api.datastream.DataStream;
11 import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
12 import org.apache.flink.util.Collector;
13
14 /**
15  * 该 Flink 程序包含 2 个 StreamExecutionEnvironment，每个 StreamExecutionEnvironment
16  * 上各调用 1 次 execute() 方法
17  */
18 public class DemoWith2Env2Exec {
19     public static void main(String[] args) throws Exception {
20         /* 步骤1: 创建2个StreamExecutionEnvironment对象 */
21         StreamExecutionEnvironment env1 =
22             StreamExecutionEnvironment.getExecutionEnvironment();
23         StreamExecutionEnvironment env2 =
24             StreamExecutionEnvironment.getExecutionEnvironment();
25         run(env1, "ecnu01", 8888);
26         run(env2, "ecnu01", 8889);
27     }
28
29     public static void run(StreamExecutionEnvironment env, String hostname, int port)
30         throws Exception {
31         /* 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 */
32         // 从指定主机名和端口的套接字获取输入数据，创建名为lines的DataStream
33         DataStream<String> lines = env.socketTextStream(hostname, port, "\n");
34         // 将lines中的每一个文本行按空格分割成单个单词
35         DataStream<String> words =
36             lines.flatMap(
```

```
34     new FlatMapFunction<String, String>() {
35         @Override
36         public void flatMap(String value, Collector<String> out) throws Exception {
37             for (String word : value.split(" ")) {
38                 out.collect(word);
39             }
40         }
41     });
42     // 将每个单词的频数设置为1, 即将每个单词映射为[单词, 1]
43     DataStream<Tuple2<String, Integer>> pairs =
44         words.map(
45             new MapFunction<String, Tuple2<String, Integer>>() {
46                 @Override
47                 public Tuple2<String, Integer> map(String value) throws Exception {
48                     return new Tuple2<String, Integer>(value, 1);
49                 }
50             });
51     // 按单词聚合, 并对相同单词的频数使用sum进行累计
52     DataStream<Tuple2<String, Integer>> counts = pairs.keyBy(0).sum(1);
53     // 输出词频统计结果
54     counts.print();
55
56     /* 步骤3: 触发程序执行 */
57     env.execute("DemoWith2Env2Exec:" + port);
58 }
59 }
```