

实验十 基于 Yarn 部署 Flink*

10.1 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

10.2 实验任务

- 完成 Flink on Yarn 的单机伪分布式部署以及分布式部署
- 完成单机伪分布部署方式下在 Per-Job 运行模式中分别按照 Attached 和 Detached 方式提交应用程序
- 完成分布部署方式下在 Per-Job 运行模式中分别按照 Attached 和 Detached 方式提交应用程序

10.3 实验环境

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

10.4 实验步骤

10.4.1 单机伪分布式部署

(1) 准备工作

- 登录用户 dase-local
- 修改.bashrc 文件

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

添加如下内容：

```
1 export HADOOP_HOME=/home/dase-local/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”单机伪分布式部署）

(2) 启动 Yarn 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10.1所示。

```
dase-local@ecnu01:~$ jps
27909 NameNode
28005 ResourceManager
28597 JobHistoryServer
28186 NodeManager
28414 DataNode
29086 SecondaryNameNode
29007 Jps
dase-local@ecnu01:~$
```

图 10.1 启动服务后存在的进程

- 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://localhost:8088`，如图10.2所示。

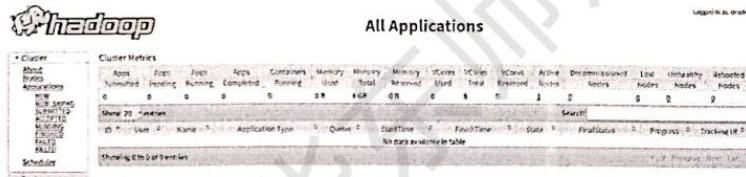


图 10.2 Yarn 界面

(4) 采用 Per-Job 运行模式执行 Flink DataStream 程序

- 以 Attached (默认) 方式提交 jar 包运行 DataStream 程序

- 启动 NetCat 服务

启动一个终端（记此终端为终端 1.1），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 1.2），以默认方式提交 jar 包。

```
1 ~/flink-1.12.1/bin/flink run -t yarn-per-job
2 ~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar --port
3 8888
```

提交 jar 包后，可以看到输出信息中的 Yarn 应用信息，如图10.3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_1619859573978_0001 的作业。

```
[...] - Deploying cluster, current state ACCEPTED
2021-05-01 22:28:22,781 INFO org.apache.flink.yarn.YarnClusterDescriptor
[...] - YARN application has been deployed successfully.
2021-05-01 22:28:22,781 INFO org.apache.flink.yarn.YarnClusterDescriptor
[...] Found Web Interface ecnu01:33129 of application 'application_1619859573978_0001'.
Job has been submitted with JobID ca18a35bc53a4382c5911a53d83484fa
```

图 10.3 提交 jar 包的部分输出信息

在终端 1.1 中输入数据，如图10.4所示。

```
dase-local@ecnu01:~$ nc -lk 8888
Hello Per-Job Mode
Hello Default
Hello flink on yarn
```

图 10.4 输入数据

打开文件管理器，定位到 /home/dase-local/hadoop-2.10.1/logs/userlogs/applica-

cation_1619859573978_0001 目录（与提交到 Yarn 上的应用程序名称对应），在这个文件夹中查看编号最大的文件夹（文件夹名称格式如 *container_**-*-*-**），查看其中的 *taskmanager.out* 文件。文件内容如图 10.5 所示。

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

图 10.5 默认方式提交示例程序的运行结果

另起一个终端，输入 *jps* 查看进程如图 10.6 所示。我们可以看到 CliFrontend 进程、YarnTaskExecutorRunner 进程和 YarnJobClusterEntrypoint 进程。

```
dase-local@ecnu01:~$ jps
27909 NameNode
28005 ResourceManager
28597 JobHistoryServer
17893 CliFrontend
22906 Jps
28186 NodeManager
18509 YarnJobClusterEntrypoint
28414 DataNode
29086 SecondaryNameNode
18719 YarnTaskExecutorRunner
dase-local@ecnu01:~$
```

图 10.6 程序运行中存在的进程

- 终止程序

在终端 1.1 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
```

- 以 Detached 方式提交 jar 包运行 DataStream 程序

- 启动 NetCat 服务

启动一个终端（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），输入如下命令：

```
1 ~/flink-1.12.1/bin/flink run -t yarn-per-job --detached
2 ~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar --port
3 8888
```

提交 jar 包后，可以看到输出信息中的 Yarn 应用信息，如图10.7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_1619859573978_0002 的作业。

```
[...] - 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_1619859573978_0002
If this should not be possible, then you can also kill Flink via YARN's web interface or
via:
$ yarn application -kill application_1619859573978_0002
Note that killing Flink might not clean up all job artifacts and temporary files.
2021-05-01 22:44:51,467 INFO org.apache.flink.yarn.YarnClusterDescriptor
[...] - Found Web Interface_ecnu01:42043 of application 'application_1619859573978_0002'.
Job has been submitted with JobID 3a313d45868729a35f33baad5703426b
dase-local@ecnu01:~$
```

图 10.7 提交 jar 包的部分输出信息

在终端 2.1 中输入数据，如图10.8所示。

```
dase-local@ecnu01:~$ nc -lk 8888
Hello Per-Job Mode
Hello Detached
Hello flink on yarn
```

图 10.8 输入数据

打开文件管理器，定位到`/home/dase-local/hadoop-2.10.1/logs/userlogs/application_1619859573978_0002`目录，在这个文件夹中查看编号最大的文件夹（文件夹名称格式如`container_*_*_*_*`），查看其中的`taskmanager.out`文件。文件内容如图10.9所示。

另起一个终端，输入`jps`查看进程如图10.10所示。我们可以看到`Yarn-TaskExecutorRunner`进程和`YarnSessionClusterEntrypoint`进程。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

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

图 10.9 默认方式提交示例程序的运行结果

现又消失的现象。

```
dase-local@ecnu01:~$ jps
14563 NameNode
7988 ResourceManager
14805 DataNode
15223 SecondaryNameNode
23400 Jps
8171 NodeManager
12619 YarnJobClusterEntrypoint
12831 YarnTaskExecutorRunner
dase-local@ecnu01:~$ █
```

图 10.10 程序运行中存在的进程

(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，可以查看应用程序的运行情况。

- 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://localhost:8088>，可以看到本次启动 Yarn 后所提交的应用程序，如

图10.11所示。



图 10.11 Yarn 界面

(6) 停止服务

- 停止 Flink DataStream 程序

在终端 2.1 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
```

- 停止 Hadoop 和 Yarn

```
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
```

10.4.2 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，参考实验一进行修改
- 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10.12、图10.13和图10.14所示。

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

图 10.12 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 10.13 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 10.14 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10.15所示。

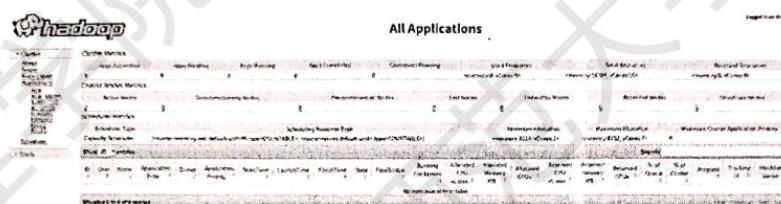


图 10.15 Yarn 界面

(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
2 ~/flink-1.12.1/examples/streaming/SocketWindowWordCount.jar
3 --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10.16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。

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10.17, 可以看到程序运行在

```
[ ] - 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 ecnu03:2385 of application 'application_1620481595386_0001'.
Job has been submitted with JobID 568bd17c2291ae604aa5d5bd5cbe6362
```

图 10.16 提交 jar 包的部分输出信息

ecnu02 上。

Detail	SubTasks	TaskManagers	Watermarks	Accumulators	BackPressure	Metrics	
ID	Received	Records Received	Bytes Sent	Records Sent	Attempt	Host	Status
0	9	0 B	0	1		ecnu02	RUNNING

图 10.17 Flink Web 页面中的 Subtasks

在终端 3.1 中输入数据，如图10.18所示。

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

图 10.18 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10.19所示。

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10.20、10.21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10.22所示，客户端节点还存在一个 `CliFrontend` 进程。

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

图 10.19 输出文件内容

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

图 10.20 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 10.21 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 10.22 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。

```
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。

```
~/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10.23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。

```
[ ] - 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 c9eale56af5d56638b3df354e60db2b
dase-ds@ecnu04:~$
```

图 10.23 提交 jar 包的部分输出信息

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10.24，可以看到程序运行在 ecnu03 上。

	Detail	SubTasks	TaskManagers	Watermarks	Accumulators	BackPressure	Metrics
ID	Received	Records Received	Bytes Sent	Records Sent	Attempt	Host	Status
0	20	0 B	0	1	ecnu03	RUNNING	...

图 10.24 Flink Web 页面中的 Subtasks

在终端 4.1 中输入数据，如图10.25所示。

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

图 10.25 输入数据

- 查看输出结果

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10.26所示。

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

图 10.26 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 10.27、10.28 所示，我们可以看到 ecnu03 节点上存在一个 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 10.27 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 10.28 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 10.29 所示。

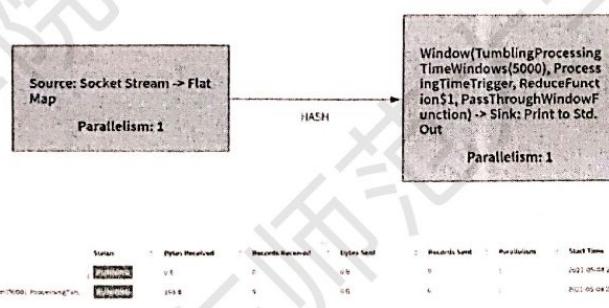


图 10.29 Flink 应用程序界面

- 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10.30所示。

ID	User	Name	Application Type	Owner	Priority	Start Time	Launch Time	Final Time	State	Final Status	Running Containers	Allocated Vcores	Allocated Memory	Allocated CPU	Allocated GPUs	Allocated HDFS	Allocated S3
APPLICATION_1512425747716_0003	datanode	Flink Application Cluster	Apache Flink	default	0	Thu Feb 4 18:56:10 2021	Thu Feb 4 19:36:45 2021	Thu Feb 4 19:36:45 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	
APPLICATION_1512425747716_0004	datanode	Flink per-job cluster	Apache Flink	default	0	Thu Feb 4 18:56:25 2021	Thu Feb 4 19:26:25 2021	Thu Feb 4 19:26:25 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	
APPLICATION_1512425747716_0001	datanode	Flink Application Cluster	Apache Flink	default	0	Thu Feb 4 18:56:30 2021	Thu Feb 4 18:56:30 2021	Thu Feb 4 18:56:30 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	

图 10.30 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。

```
1 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
```

10.5 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：

```
~/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
36         @Override
37         public void flatMap(String value, Collector<String> out) throws Exception {
38             for (String word : value.split(" ")) {
39                 out.collect(word);
40             }
41         }
42     });
43
44     // 将每个单词的频数设置为1，即将每个单词映射为[单词, 1]
45     DataStream<Tuple2<String, Integer>> pairs =
46         words.map(
47             new MapFunction<String, Tuple2<String, Integer>>() {
48
49                 @Override
50                 public Tuple2<String, Integer> map(String value) throws Exception {
51                     return new Tuple2<String, Integer>(value, 1);
52                 }
53             });
54
55     // 按单词聚合，并对相同单词的频数使用sum进行累计
56     DataStream<Tuple2<String, Integer>> counts = pairs.keyBy(0).sum(1);
57
58     // 输出词频统计结果
59     counts.print();
60
61     /* 步骤3：触发程序执行 */
62     env.execute("DemoWith2Env2Exec:" + port);
63 }
64 }
```