

前言

上篇文章我们讲解了 Blink 源码的项目结构并分析了它的每个模块的作用以及代码量。这里我们以后文章中主要参考源码是 Blink 分支，顺带会和 Flink release-1.7.1-rc2 做一定的对比（后面就不再做重复唠叨）。这篇文章 zhisheng 我带大家来揭秘下 Flink 的启动流程。

流程

在前面的文章里也说过启动 Flink 是通过执行命令：

```
1 ./start-cluster.sh
```

那么我们先来看下这个脚本里面的东西有哪些呢？

```
1 bin=`dirname "$0"`
2 bin=`cd "$bin"; pwd`
3
4 . "$bin"/config.sh
5
6 # Start the JobManager instance(s)
7 shopt -s nocasematch
8 if [[ $HIGH_AVAILABILITY == "zookeeper" ]]; then
9     # HA Mode
10     readMasters
11
12     echo "Starting HA cluster with ${#MASTERS[@]} masters."
13
14     for ((i=0;i<${#MASTERS[@]};++i)); do
15         master=${MASTERS[i]}
16         webuiport=${WEBUIPORTS[i]}
17
18         if [ ${MASTERS_ALL_LOCALHOST} = true ] ; then
19             "${FLINK_BIN_DIR}/jobmanager.sh start "${master}"
20             "${webuiport}"
21         else
22             ssh -n $FLINK_SSH_OPTS $master -- "nohup /bin/bash -l
23             \"${FLINK_BIN_DIR}/jobmanager.sh\" start ${master} ${webuiport} &"
24         fi
25     done
26 else
27     echo "Starting cluster."
```

```
28     # Start single JobManager on this machine
29     "$FLINK_BIN_DIR"/jobmanager.sh start
30 fi
31 shopt -u nocasematch
32
33 # Start TaskManager instance(s)
34 TMSlaves start
```

执行 config.sh

可以看到首先它运行了 `./config.sh`，那么就来看看这个配置脚本吧，脚本有点长，这里我给出链接地址: https://github.com/zhisheng17/flink/blob/zhisheng_blink_1.5.1/flink-dist/src/main/flink-bin/bin/config.sh

大家可以瞧一瞧，里面其实干的事情比较简单，就是把conf目录下所有的配置文件、环境变量、读取加载，给大家截几个图就明白了。

```
zhisheng@zhisheng /usr/local/blink-1.5.1/bin$ cd ..
zhisheng@zhisheng /usr/local/blink-1.5.1$ ll conf
total 128
-rw-r--r-- 1 zhisheng staff 9.8K 1 29 11:44 flink-conf.yaml
-rw-r--r-- 1 zhisheng staff 2.1K 1 29 11:44 log4j-cli.properties
-rw-r--r-- 1 zhisheng staff 1.8K 1 28 16:51 log4j-console.properties
-rw-r--r-- 1 zhisheng staff 1.5K 1 29 11:44 log4j-kubernetes.properties
-rw-r--r-- 1 zhisheng staff 1.7K 1 28 16:51 log4j-yarn-session.properties
-rw-r--r-- 1 zhisheng staff 1.9K 1 28 16:51 log4j.properties
-rw-r--r-- 1 zhisheng staff 2.2K 1 28 16:51 logback-console.xml
-rw-r--r-- 1 zhisheng staff 1.5K 1 28 16:51 logback-yarn.xml
-rw-r--r-- 1 zhisheng staff 2.3K 1 28 16:51 logback.xml
-rw-r--r-- 1 zhisheng staff 15B 1 28 16:51 masters
-rw-r--r-- 1 zhisheng staff 10B 1 28 16:51 slaves
-rw-r--r-- 1 zhisheng staff 4.1K 1 29 11:44 sql-client-defaults.yaml
-rw-r--r-- 1 zhisheng staff 1.4K 1 28 16:51 zoo.cfg
zhisheng@zhisheng /usr/local/blink-1.5.1$
```

上图是所有的配置文件，下面是 config.sh 的部分函数:

```

fi
}

readMasters() {
    MASTERS_FILE="${FLINK_CONF_DIR}/masters"

    if [[ ! -f "${MASTERS_FILE}" ]]; then
        echo "No masters file. Please specify masters in 'conf/masters'."
        exit 1
    fi

    MASTERS=()
    WEBUIPORTS=()

    MASTERS_ALL_LOCALHOST=true
    GOON=true
    while $GOON; do
        read line || GOON=false
        HOSTWEBUIPORT=$(extractHostName $line)

        if [ -n "$HOSTWEBUIPORT" ]; then
            HOST=$(echo $HOSTWEBUIPORT | cut -f1 -d:)
            WEBUIPORT=$(echo $HOSTWEBUIPORT | cut -s -f2 -d:)
            MASTERS+=(${HOST})

            if [ -z "$WEBUIPORT" ]; then
                WEBUIPORTS+=(0)
            else
                WEBUIPORTS+=(${WEBUIPORT})
            fi

            if [ "${HOST}" != "localhost" ] && [ "${HOST}" != "127.0.0.1" ]; then
                MASTERS_ALL_LOCALHOST=false
            fi
        fi
    done < "$MASTERS_FILE"
}

readSlaves() {
    SLAVES_FILE="${FLINK_CONF_DIR}/slaves"

    if [[ ! -f "$SLAVES_FILE" ]]; then
        echo "No slaves file. Please specify slaves in 'conf/slaves'."
        exit 1
    fi

    SLAVES=()

    SLAVES_ALL_LOCALHOST=true
    GOON=true
    while $GOON; do
        read line || GOON=false
        HOST=$(extractHostName $line)
        if [ -n "$HOST" ]; then
            SLAVES+=(${HOST})
            if [ "${HOST}" != "localhost" ] && [ "${HOST}" != "127.0.0.1" ]; then
                SLAVES_ALL_LOCALHOST=false
            fi
        fi
    done < "$SLAVES_FILE"
}

```

微信公众号:zhisheng

上图是读取 master 和 slaves 配置文件

```
#####
# DEFAULT CONFIG VALUES: These values will be used when nothing has been specified in conf/flink-conf.yaml
# -or- the respective environment variables are not set.
#####

# WARNING !!! these values are only used if there is nothing else is specified in
# conf/flink-conf.yaml

DEFAULT_ENV_PID_DIR="/tmp"                # Directory to store *.pid files to
DEFAULT_ENV_LOG_MAX=5                     # Maximum number of old log files to keep
DEFAULT_ENV_JAVA_OPTS=""                  # Optional JVM args
DEFAULT_ENV_JAVA_OPTS_JM=""               # Optional JVM args (JobManager)
DEFAULT_ENV_JAVA_OPTS_TM=""               # Optional JVM args (TaskManager)
DEFAULT_ENV_SSH_OPTS=""                   # Optional SSH parameters running in cluster mode

#####
# CONFIG KEYS: The default values can be overwritten by the following keys in conf/flink-conf.yaml
#####

KEY_JOBMAN_MEM_SIZE="jobmanager.heap.mb"
KEY_TASKMAN_MEM_SIZE="taskmanager.heap.mb"
KEY_TASKMAN_MEM_MANAGED_SIZE="taskmanager.memory.size"
KEY_TASKMAN_MEM_MANAGED_FRACTION="taskmanager.memory.fraction"
KEY_TASKMAN_OFFHEAP="taskmanager.memory.off-heap"
KEY_TASKMAN_MEM_PRE_ALLOCATE="taskmanager.memory.preallocate"

KEY_TASKMAN_NET_BUF_FRACTION="taskmanager.network.memory.fraction"
KEY_TASKMAN_NET_BUF_MIN="taskmanager.network.memory.min"
KEY_TASKMAN_NET_BUF_MAX="taskmanager.network.memory.max"
KEY_TASKMAN_NET_BUF_NR="taskmanager.network.numberOfBuffers" # fallback

KEY_TASKMAN_COMPUTE_NUMA="taskmanager.compute.numa"

KEY_ENV_PID_DIR="env.pid.dir"
KEY_ENV_LOG_DIR="env.log.dir"
KEY_ENV_LOG_MAX="env.log.max"
KEY_ENV_JAVA_HOME="env.java.home"
KEY_ENV_JAVA_OPTS="env.java.opts"
KEY_ENV_JAVA_OPTS_JM="env.java.opts.jobmanager"
KEY_ENV_JAVA_OPTS_TM="env.java.opts.taskmanager"
KEY_ENV_SSH_OPTS="env.ssh.opts"
KEY_HIGH_AVAILABILITY="high-availability"
KEY_ZK_HEAP_MB="zookeeper.heap.mb"

KEY_FLINK_MODE="mode"

#####
# PATHS AND CONFIG
#####

target="$0"
# For the case, the executable has been directly symlinked, figure out
# the correct bin path by following its symlink up to an upper bound.
# Note: we can't use the readlink utility here if we want to be POSIX
# compatible.
iteration=0
while [ -L "$target" ]; do
    if [ "$iteration" -gt 100 ]; then
        echo "Cannot resolve path: You have a cyclic symlink in $target."
        break
    fi
    ls=`ls -ld -- "$target"`
    target=`expr "$ls" : '.* -> \(.*\) '$`
    iteration=$((iteration + 1))
done

# Convert relative path to absolute path and resolve directory symlinks
bin=`dirname "$target"`
SYMLINK_RESOLVED_BIN=`cd "$bin"; pwd -P`
```

微信公众号: zhisheng

High availability

```
if [ -z "${HIGH_AVAILABILITY}" ]; then
    HIGH_AVAILABILITY=$(readFromConfig ${KEY_HIGH_AVAILABILITY} "" "${YAML_CONF}")
    if [ -z "${HIGH_AVAILABILITY}" ]; then
        # Try deprecated value
        DEPRECATED_HA=$(readFromConfig "recovery.mode" "" "${YAML_CONF}")
        if [ -z "${DEPRECATED_HA}" ]; then
            HIGH_AVAILABILITY="none"
        elif [ "${DEPRECATED_HA}" == "standalone" ]; then
            # Standalone is now 'none'
            HIGH_AVAILABILITY="none"
        else
            HIGH_AVAILABILITY=${DEPRECATED_HA}
        fi
    fi
fi
```


上图是 flink-conf.yaml 里面的一些配置

```
##### ENVIRONMENT VARIABLES #####
#####

# read JAVA_HOME from config with no default value
MY_JAVA_HOME=$(readFromConfig ${KEY_ENV_JAVA_HOME} "" "${YAML_CONF}")
# check if config specified JAVA_HOME
if [ -z "${MY_JAVA_HOME}" ]; then
    # config did not specify JAVA_HOME. Use system JAVA_HOME
    MY_JAVA_HOME=${JAVA_HOME}
fi
# check if we have a valid JAVA_HOME and if java is not available
if [ -z "${MY_JAVA_HOME}" ] && ! type java > /dev/null 2> /dev/null; then
    echo "Please specify JAVA_HOME. Either in Flink config ./conf/flink-conf.yaml or as system-wide JAVA_HOME."
    exit 1
else
    JAVA_HOME=${MY_JAVA_HOME}
fi

UNAME=$(uname -s)
if [ "${UNAME:0:6}" == "CYGWIN" ]; then
    JAVA_RUN=java
else
    if [[ -d $JAVA_HOME ]]; then
        JAVA_RUN=$JAVA_HOME/bin/java
    else
        JAVA_RUN=java
    fi
fi

# Define HOSTNAME if it is not already set
if [ -z "${HOSTNAME}" ]; then
    HOSTNAME=`hostname`
fi

IS_NUMBER="^[0-9]+$"

# Define FLINK_JM_HEAP if it is not already set
if [ -z "${FLINK_JM_HEAP}" ]; then
    FLINK_JM_HEAP=$(readFromConfig ${KEY_JOB_M_MEM_SIZE} 0 "${YAML_CONF}")
fi

# Define FLINK_TM_HEAP if it is not already set
if [ -z "${FLINK_TM_HEAP}" ]; then
    FLINK_TM_HEAP=$(readFromConfig ${KEY_TASK_M_MEM_SIZE} 0 "${YAML_CONF}")
fi

# Define FLINK_TM_MEM_MANAGED_SIZE if it is not already set
if [ -z "${FLINK_TM_MEM_MANAGED_SIZE}" ]; then
    FLINK_TM_MEM_MANAGED_SIZE=$(readFromConfig ${KEY_TASK_M_MEM_MANAGED_SIZE} 0 "${YAML_CONF}")
fi

# Define FLINK_TM_MEM_MANAGED_FRACTION if it is not already set
if [ -z "${FLINK_TM_MEM_MANAGED_FRACTION}" ]; then
    FLINK_TM_MEM_MANAGED_FRACTION=$(readFromConfig ${KEY_TASK_M_MEM_MANAGED_FRACTION} 0.7 "${YAML_CONF}")
fi

# Define FLINK_TM_OFFHEAP if it is not already set
if [ -z "${FLINK_TM_OFFHEAP}" ]; then
    FLINK_TM_OFFHEAP=$(readFromConfig ${KEY_TASK_M_OFFHEAP} "false" "${YAML_CONF}")
fi

# Define FLINK_TM_MEM_PRE_ALLOCATE if it is not already set
if [ -z "${FLINK_TM_MEM_PRE_ALLOCATE}" ]; then
    FLINK_TM_MEM_PRE_ALLOCATE=$(readFromConfig ${KEY_TASK_M_MEM_PRE_ALLOCATE} "false" "${YAML_CONF}")
fi

# Define FLINK_TM_NET_BUF_FRACTION if it is not already set
if [ -z "${FLINK_TM_NET_BUF_FRACTION}" ]; then
```

微信公众号: zhisheng

上图是加载环境变量配置

是不是，这个 config.sh 就是干了这些事情，把所有需要的配置加载起来，供后面启动提供保障。分析完 config.sh，我们继续回到 start-cluster.sh，它接下来开始启动 JobManager 实例了，启动 JobManager 实例这里分两种，一种是高可用(HA)模式，另一种是单机模式。这里我们知道它是要启动 JobManager 就可以了，至于有什么区别，不是这篇文章的重点。

```

zhisheng@zhisheng: /usr/local/blink-1.5.1 cat bin/start-cluster.sh
#!/usr/bin/env bash
#####
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements. See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership. The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#####

bin=`dirname "$0"`
bin=`cd "$bin"; pwd`

. "$bin"/config.sh

# Start the JobManager instance(s)
shopt -s nocasematch
if [[ $HIGH_AVAILABILITY == "zookeeper" ]]; then
    # HA Mode
    readMasters

    echo "Starting HA cluster with ${#MASTERS[@]} masters."

    for ((i=0;i<${#MASTERS[@]};++i)); do
        master=${MASTERS[i]}
        webuiport=${WEBUIPORTS[i]}

        if [ ${MASTERS_ALL_LOCALHOST} = true ] ; then
            "${FLINK_BIN_DIR}"/jobmanager.sh start "${master}" "${webuiport}"
        else
            ssh -n $FLINK_SSH_OPTS $master -- "nohup /bin/bash -l \"${FLINK_BIN_DIR}/jobmanager.sh\" start ${master} ${webuiport} &"
        fi
    done
else
    echo "Starting cluster."

    # Start single JobManager on this machine
    "${FLINK_BIN_DIR}"/jobmanager.sh start
fi
shopt -u nocasematch

# Start TaskManager instance(s)
TMSlaves start
zhisheng@zhisheng: /usr/local/blink-1.5.1 cat conf/flink-conf.yaml | grep zookeeper
# The high-availability mode. Possible options are 'NONE', 'FILESYSTEM' or 'zookeeper'.
# high-availability: zookeeper
# high-availability.zookeeper.quorum: localhost:2181
# ACL options are based on https://zookeeper.apache.org/doc/r3.1.2/zookeeperProgrammers.html#sc_BuiltinACLschemes
# high-availability.zookeeper.client.acl: open
# zookeeper.sasl.service-name: zookeeper
# zookeeper.sasl.login-context-name: Client
zhisheng@zhisheng: /usr/local/blink-1.5.1

```

微信公众号: zhisheng

执行 jobmanager.sh

启动 JobManager 最终是执行了 bin/jobmanager.sh 脚本, 在该脚本里面执行如下(传递了一个参数: JOBMANAGER_TYPE):

```

# Start/stop a Flink JobManager.
USAGE="Usage: jobmanager.sh ((start|start-foreground) [host] [webui-port])|stop|stop-all"

STARTSTOP=$1
HOST=$2 # optional when starting multiple instances
WEBUIPORT=$3 # optional when starting multiple instances

if [[ $STARTSTOP != "start" ]] && [[ $STARTSTOP != "start-foreground" ]] && [[ $STARTSTOP != "stop" ]] && [[ $STARTSTOP != "stop-all" ]]; then
    echo $USAGE
    exit 1
fi

bin=`dirname "$0"`
bin=`cd "$bin"; pwd`

. "$bin"/config.sh

JOBMANAGER_TYPE=jobmanager

if [[ "${FLINK_MODE}" == "new" ]]; then
    JOBMANAGER_TYPE=standalonesession
fi

if [[ $STARTSTOP == "start" ]] || [[ $STARTSTOP == "start-foreground" ]]; then
    if [[ ! ${FLINK_JM_HEAP} =~ $IS_NUMBER ]] || [[ "${FLINK_JM_HEAP}" -lt "0" ]]; then
        echo "[ERROR] Configured JobManager memory size is not a valid value. Please set '${KEY_JOB_M_MEM_SIZE}' in
        "
        exit 1
    fi

    if [ "${FLINK_JM_HEAP}" -gt "0" ]; then
        export JVM_ARGS="$JVM_ARGS -Xms"${FLINK_JM_HEAP}"m -Xmx"${FLINK_JM_HEAP}"m"
    fi

    # Add JobManager-specific JVM options
    export FLINK_ENV_JAVA_OPTS="${FLINK_ENV_JAVA_OPTS} ${FLINK_ENV_JAVA_OPTS_JM}"

    # Startup parameters
    args="--configDir" "${FLINK_CONF_DIR}" "--executionMode" "cluster"
    if [ ! -z $HOST ]; then
        args+=("--host")
        args+=("${HOST}")
    fi

    if [ ! -z $WEBUIPORT ]; then
        args+=("--webui-port")
        args+=("${WEBUIPORT}")
    fi
fi

if [[ $STARTSTOP == "start-foreground" ]]; then
    exec "${FLINK_BIN_DIR}/flink-console.sh $JOBMANAGER_TYPE "${args[@]}"
else
    "${FLINK_BIN_DIR}/flink-daemon.sh $STARTSTOP $JOBMANAGER_TYPE "${args[@]}"
fi

```

微信公众号: zhisheng

那我们接下来看下 bin/flink-daemon.sh 这个脚本：

```

# Start a Flink service as a console application. Must be stopped with Ctrl-C
# or with SIGTERM by kill or the controlling process.
USAGE="Usage: flink-console.sh (jobmanager|taskmanager|historyserver|zookeeper) [args]"
SERVICE=$1
ARGS=("${@:2}") # get remaining arguments as array

bin=`dirname "$0"`
bin=`cd "$bin"; pwd`

. "$bin"/config.sh

case $SERVICE in
    (jobmanager)
        CLASS_TO_RUN=org.apache.flink.runtime.jobmanager.JobManager
        ;;
    (taskmanager)
        CLASS_TO_RUN=org.apache.flink.runtime.taskmanager.TaskManager
        ;;
    (taskexecutor)
        CLASS_TO_RUN=org.apache.flink.runtime.taskexecutor.TaskManagerRunner
        ;;
    (historyserver)
        CLASS_TO_RUN=org.apache.flink.runtime.webmonitor.history.HistoryServer
        ;;
    (zookeeper)
        CLASS_TO_RUN=org.apache.flink.runtime.zookeeper.FlinkZooKeeperQuorumPeer
        ;;
    (standalonesession)
        CLASS_TO_RUN=org.apache.flink.runtime.entrypoint.StandaloneSessionClusterEntrypoint
        ;;
    (*)
        echo "Unknown service '${SERVICE}'. $USAGE."
        exit 1
        ;;
esac

```

微信公众号: zhisheng

该脚本就利用了传递的参数 JOBMANAGER_TYPE，这里有个 case 场景，不同的服务就会启动不一样的类，如果是 jobmanager 的话，启动的类是 `org.apache.flink.runtime.jobmanager.JobManager`，如果是 standalonesession，执行 `org.apache.flink.runtime.entrypoint.StandaloneSessionClusterEntrypoint` 类。

这里的结果是 standalonesession，所以运行 `org.apache.flink.runtime.entrypoint.StandaloneSessionClusterEntrypoint` 类。

那么我们来看下该类的 main 方法：


```
StandaloneSessionClusterEntrypoint.java x SessionClusterEntrypoint.java x JobClusterEntrypoint.java x
81      return new StandaloneResourceManager(
82          rpcService,
83          FlinkResourceManager.RESOURCE_MANAGER_NAME,
84          resourceId,
85          configuration,
86          resourceManagerConfiguration,
87          highAvailabilityServices,
88          heartbeatServices,
89          slotManager,
90          metricRegistry,
91          resourceManagerRuntimeServices.getJobLeaderIdService(),
92          clusterInformation,
93          fatalErrorHandler);
94    }
95
96    public static void main(String[] args) {
97        // startup checks and logging
98        EnvironmentInformation.logEnvironmentInfo(LOG, StandaloneSessionClusterEntrypoint.class.getSimpleName(), args);
99        SignalHandler.register(LOG);
100        JvmShutdownSafeguard.installAsShutdownHook(LOG);
101
102        Configuration configuration = loadConfiguration(parseArguments(args));
103
104        StandaloneSessionClusterEntrypoint entrypoint = new StandaloneSessionClusterEntrypoint(configuration);
105
106        entrypoint.startCluster();
107    }
108
109
```

微信公众号: zhisheng

方法里面的执行流程下篇文章好好写下。

执行 taskmanager.sh

启动完 JobManager 后，它接下来就是启动 TaskManager 了：

```
1 # Start TaskManager instance(s)
2 TMSlaves start
```

执行了上面命令，那这个命令到底干了啥呢？其实上面我给大家说的那个 config.sh 里面就有：

```
1 # starts or stops TMs on all slaves
2 # TMSlaves start|stop
3 TMSlaves() {
4     CMD=$1
5
6     readSlaves
7
8     if [ ${SLAVES_ALL_LOCALHOST} = true ] ; then
9         # all-local setup
10        for slave in ${SLAVES[@]}; do
11            "${FLINK_BIN_DIR}"/taskmanager.sh "${CMD}"
12        done
13    else
```



```

. "$bin"/config.sh
TYPE=taskmanager
if [[ "${FLINK_MODE}" == "new" ]]; then
    TYPE=taskexecutor
fi

if [[ $STARTSTOP == "start" ]] || [[ $STARTSTOP == "start-foreground" ]]; then

    # if memory allocation mode is lazy and no other JVM options are set,
    # set the 'Concurrent Mark Sweep GC'
    if [[ $FLINK_TM_MEM_PRE_ALLOCATE == "false" ]] && [ -z "${FLINK_ENV_JAVA_OPTS}" ] && [ -z "${FLINK_ENV_JAVA_OPTS_TM}" ]; then
        export JVM_ARGS="${JVM_ARGS} -XX:+UseG1GC"
    fi

    if [[ ! ${FLINK_TM_HEAP} =~ ^[0-9]+$ ]] || [[ "${FLINK_TM_HEAP}" -lt "0" ]]; then
        echo "[ERROR] Configured TaskManager JVM heap size is not a number. Please set '${KEY_TASKM_MEM_SIZE}' in ${FLINK_CONF_FILE}."
        exit 1
    fi

    if [ "${FLINK_TM_HEAP}" -gt "0" ]; then
        TASK_MANAGER_RESOURCE=$(CalculateTaskManagerResource)

        TM_HEAP_SIZE=$(echo ${TASK_MANAGER_RESOURCE} | sed 's/.*TotalHeapMemory:\([0-9]*\).*$/\1/g')
        TM_YOUNG_HEAP_SIZE=$(echo ${TASK_MANAGER_RESOURCE} | sed 's/.*YoungHeapMemory:\([0-9]*\).*$/\1/g')
        TM_MAX_OFFHEAP_SIZE=$(echo ${TASK_MANAGER_RESOURCE} | sed 's/.*TotalDirectMemory:\([0-9]*\).*$/\1/g')

        export JVM_ARGS="${JVM_ARGS} -Xms${TM_HEAP_SIZE}M -Xmx${TM_HEAP_SIZE}M -Xmn${TM_YOUNG_HEAP_SIZE}M -XX:MaxDirectMemorySize=${TM_MAX_OFFHEAP_SIZE}M"
    fi

    # Add TaskManager-specific JVM options
    export FLINK_ENV_JAVA_OPTS="${FLINK_ENV_JAVA_OPTS} ${FLINK_ENV_JAVA_OPTS_TM}"

    # Startup parameters
    args=(--configDir "${FLINK_CONF_DIR}")
fi

if [[ $STARTSTOP == "start-foreground" ]]; then
    exec "${FLINK_BIN_DIR}"/flink-console.sh $TYPE "${args[@]}"
else
    if [[ $FLINK_TM_COMPUTE_NUMA == "false" ]]; then
        # Start a single TaskManager
        "${FLINK_BIN_DIR}"/flink-daemon.sh $STARTSTOP $TYPE "${args[@]}"
    else
        # Example output from 'numactl --show' on an AWS c4.8xlarge:
        # policy: default
        # preferred node: current
        # physcpubind: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
        # cpubind: 0 1
        # nodebind: 0 1
        # membind: 0 1
        read -ra NODE_LIST <<< $(numactl --show | grep "^nodebind: ")
        for NODE_ID in "${NODE_LIST[@]:1}"; do
            # Start a TaskManager for each NUMA node
            numactl --membind=$NODE_ID --cpunodebind=$NODE_ID -- "${FLINK_BIN_DIR}"/flink-daemon.sh $STARTSTOP $TYPE "${args[@]}"
        done
    fi
fi

```

微信公众号: zhisheng

最后还是执行了 flink-daemon.sh 脚本:

```
#####

# Start/stop a Flink daemon.
USAGE="Usage: flink-daemon.sh (start|stop|stop-all) (jobmanager|taskmanager|zookeeper|historyserver) [args]"

STARTSTOP=$1
DAEMON=$2
ARGS=("${@:3}") # get remaining arguments as array

bin=`dirname "$0"`
bin=`cd "$bin"; pwd`

. "$bin"/config.sh

case $DAEMON in
    (jobmanager)
        CLASS_TO_RUN=org.apache.flink.runtime.jobmanager.JobManager
        ;;
    (taskmanager)
        CLASS_TO_RUN=org.apache.flink.runtime.taskmanager.TaskManager
        ;;
    (taskexecutor)
        CLASS_TO_RUN=org.apache.flink.runtime.taskexecutor.TaskManagerRunner
        ;;
    (zookeeper)
        CLASS_TO_RUN=org.apache.flink.runtime.zookeeper.FlinkZooKeeperQuorumPeer
        ;;
    (historyserver)
        CLASS_TO_RUN=org.apache.flink.runtime.webmonitor.history.HistoryServer
        ;;
    (standalonesession)
        CLASS_TO_RUN=org.apache.flink.runtime.entrypoint.StandaloneSessionClusterEntrypoint
        ;;
    (*)
        echo "Unknown daemon '${DAEMON}'. $USAGE."
        exit 1
        ;;
esac
```

微信公众号: zhisheng

在这里执行了 `org.apache.flink.runtime.taskexecutor.TaskManagerRunner` 类，同样，我们先看一眼这个类的 `main` 方法长啥样：


```
main > java > org > apache > flink > runtime > taskexecutor > TaskManagerRunner
StandaloneSessionClusterEntrypoint.java x TaskManagerRunner.java x SessionClusterEntrypoint.java x

264 public static void main(String[] args) throws Exception {
265     // startup checks and logging
266     EnvironmentInformation.logEnvironmentInfo(LOG, componentName: "TaskManager", args);
267     SignalHandler.register(LOG);
268     JvmShutdownSafeguard.installAsShutdownHook(LOG);
269
270     long maxOpenFileHandles = EnvironmentInformation.getOpenFileHandlesLimit();
271
272     if (maxOpenFileHandles != -1L) {
273         LOG.info("Maximum number of open file descriptors is {}.", maxOpenFileHandles);
274     } else {
275         LOG.info("Cannot determine the maximum number of open file descriptors");
276     }
277
278     ParameterTool parameterTool = ParameterTool.fromArgs(args);
279
280     final String configDir = parameterTool.get("configDir");
281
282     final Configuration configuration = GlobalConfiguration.loadConfiguration(configDir);
283
284     try {
285         FileSystem.initialize(configuration);
286     } catch (IOException e) {
287         throw new IOException("Error while setting the default " +
288             "filesystem scheme from configuration.", e);
289     }
290
291     SecurityUtils.install(new SecurityConfiguration(configuration));
292
293     try {
294         SecurityUtils.getInstalledContext().runSecured(new Callable<Void>() {
295             @Override
296             public Void call() throws Exception {
297
TaskManagerRunner > main()
```

微信公众号: zhisheng

大概看了下该类后，接着回到 flink-daemon.sh 脚本，该脚本后面配置了对应的 TaskManager 的运行日志和 GC 日志，这里 Flink 是没有单独的 GC 日志打印出来。

上篇文章也说过可以不断的执行 ./start-cluster.sh 可以启动多个 TaskManager，下面这图就可以作出解释了：

```
case $STARTSTOP in
  (start)
    # Rotate log files
    rotateLogFilesWithPrefix "$FLINK_LOG_DIR" "$FLINK_LOG_PREFIX"

    # Print a warning if daemons are already running on host
    if [ -f "$pid" ]; then
      active=()
      while IFS= read -r p || [[ -n "$p" ]]; do
        kill -0 $p >/dev/null 2>&1
        if [ $? -eq 0 ]; then
          active+=($p)
        fi
      done < "$pid"

      count="${#active[@]}"

      if [ $count -gt 0 ]; then
        echo "[INFO] $count instance(s) of $SDAEMON are already running on $HOSTNAME."
      fi
    fi

    # Evaluate user options for local variable expansion
    FLINK_ENV_JAVA_OPTS=$(eval echo ${FLINK_ENV_JAVA_OPTS})

    echo "Starting $SDAEMON daemon on host $HOSTNAME."
    $JAVA_RUN $JVM_ARGS ${FLINK_ENV_JAVA_OPTS} "${log_setting[@]}" -classpath "`manglePathList "$FLINK_TM_CLASSPATH:$INTERNAL_HADOOP_CLASSPATHS" "$CLASS_TO_RUN" "${ARGS[@]}"`" > "$out" 200<&- 2>&1 < /dev/null &

    mypid=$!

    # Add to pid file if successful start
    if [[ $mypid =~ ${IS_NUMBER} ]] && kill -0 $mypid > /dev/null 2>&1; then
      echo $mypid >> "$pid"
    else
      echo "Error starting $SDAEMON daemon."
      exit 1
    fi
  ;;
;;
```

微信公众号: zhisheng

这也可以解释启动的时候打印出来的日志了：

```
1 Starting cluster.
2 [INFO] 1 instance(s) of standalone-session are already running on
  zhisheng.
3 Starting standalone-session daemon on host zhisheng.
4 log4j:WARN No appenders could be found for logger
  (org.apache.flink.configuration.GlobalConfiguration).
5 log4j:WARN Please initialize the log4j system properly.
6 log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for
  more info.
7 [INFO] 2 instance(s) of taskexecutor are already running on zhisheng.
8 Starting taskexecutor daemon on host zhisheng.
```

中文源码分析

<https://github.com/zhisheng17/flink>

相关

更多私密文章和资料请加知识星球！



Flink 精进学习

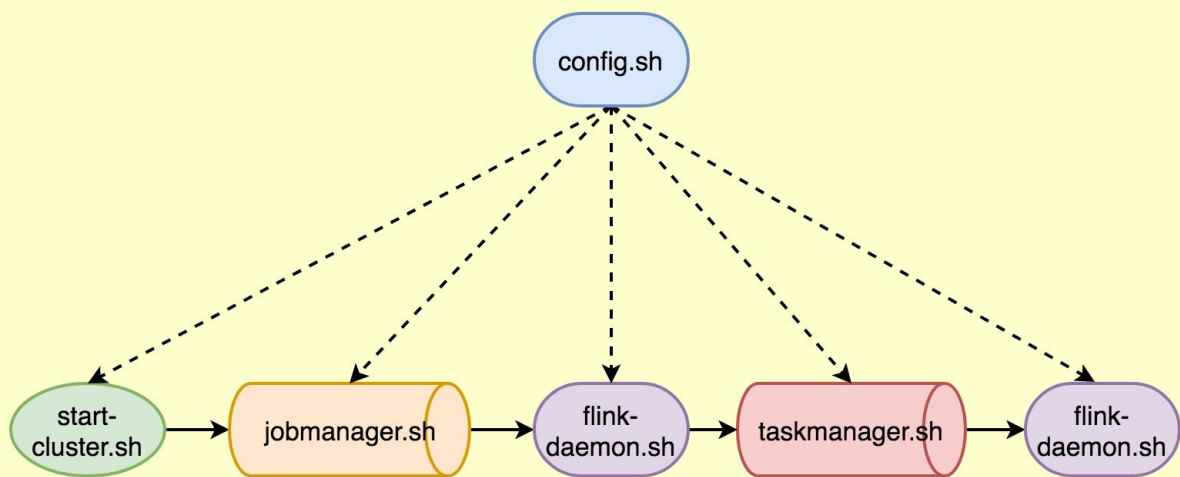
星主: zhisheng



知识星球

微信扫描预览星球详情

总结



微信公众号：zhisheng

本文先根据 Blink 的启动方式来查看脚本的执行流程，并跟随脚本探究下它里面最终是先读取配置文件，然后启动 jobmanager，完成后接着启动 taskmanager，并找到单机集群模式下启动 jobmanager 和 taskmanager 对应的 Java 启动类。下篇文章我们就可以来具体的分析下这两个类。