YARN Slider Container启动过程分析

SliderAppMaster#envVars

HADOOP\_USER\_NAME <= hadoop\_user\_name : System.getEnv

SliderAppMaster#onContainersAllocated

1) launchService <= new RoleLaunchService: //需要进一步搞清楚internal的配置作用

***launcherTmpDirPath <= internal.am.tmp.dir+***

***appMasterMasterContainerID +***

***roles***

***envVars <= HADOOP\_USER\_NAME <= hadoop\_user\_name <= System.getEnv***

***generatedConfDir <= internal.generated.conf.path***

2) launchService#launchRole

assignment : ContainerAssignment

***instanceDefinition: AggregateConf //appconfig,resources,internal***

3) new RoleLauncher

assignment

instanceDefinition : SliderAppMaster#appState.getInstanceDefinition

resources.add(roleName) roleName <= assignment.role.getName //核心作用

appconf.add(roleName)

#run

4) RoleLauncer#run

#roleLauncher <= new RoleLauncher

container <= assignment.container

roleStatus <= assignment.role

providerRole <= roleStatus.getProviderRole //providerRole的作用

***containerRole <= providerRole.name*** //role.name

role <= providerRole

resourceComponent <= RoleLaunchService

appComponent <= RoleLaunchService

***instanceDefinition: appConf <= AppState.getInstanceDefinition***

#containerLauncher <= new ContainerLauncher

***conf: getConfig() <= hadoop*** //需要确认下

***ugi <= setupUGI***

***envVars <=RoleLauchServers#envVars***

***containerTmpDirPath <= laucherTmpDirPath,container.getId***

#AgentProviderService#buildContainerLaunchContext

#providerService.initial

***appDef <= instanceDefinition.getAppConfOperations***

metainfo <= AgentUtils#getApplicationMetainfo //providerService的参数

# initialContainerLauncher :

*####setEnv####*

envVars <= instanceDefinition#getOrAddComponent : roleName

***workDir <= ApplicationConstants.Environment.PWD.$()*** //确认该PWD的值

*AGENT\_WORK\_ROOT*

***logDir <= ApplicationConstants.LOG\_DIR\_EXPANSION\_VAR <LOG\_DIR>***

AGENT\_LOG\_ROOT

***HADOOP\_USER\_NAME <= System.getenv***

***SLIDER\_PASSPHRASE <= System.getenv***

***LANG***

***LC\_ALL***

***LANGUAGE***

***scriptPath <= ./infra/agent/slider-agent/*** ***agent/main.py***

***appHome <= ${agent.package.root}/agent/main.py***

***pythonPaths***

***PYTHONPATH***

*####addLocalResource#### localResources*

***AGENT\_INSTALL\_DIR : infra/agent***

***agentImageResources <= internal.application.image.path***

***//hdfs slider-agent.tar.gz***

***APP\_DEFINITION\_DIR : app/definition***

***appDefRes <= app.def //hdfs***

***AGENT\_CONFIG\_FILE: infra/conf/agent.ini***

***agentConfRes <= agent.conf***

***AGENT\_VERSION\_FILE: infra/version***

***agentVerRes <= agent.version***

***KEYTAB\_DIR:keytab***

***keytabRes <= sliderAppMaster|+slider.hdfs.keytab.dir***

***CERT\_FILE\_LOCALIZATION\_PATH: infra/run/security/ca.cert***

***certResource <= CRT\_FILE\_NAME***

***KEYSOTRE:***

***keystoreResource***

***PROGATED\_CONF\_DIR\_NAME:*** ***propagatedconf***

***ADDON\_DEFINITON\_DIR:addon/definiton***

***addonPkgRes: addonAppDef,addonAppDefPath***

*####CommondLineBuilder,build####*

***pythonExec <= agent.python.exec.path : default ,python***

***scriptPath <=slider agent/main.py***

***ARG\_LABEL <= --label ,containerId+LABEL\_MARKER+role***

***ARG\_ZOOKEEPER\_QUORUM <= zookeeper.quorum***

***ARG\_ZOOKEEPER\_REGISTRY <= hadoop.registry.zk.root***

***ARG\_DEBUT:--debug ,debugCMD***

***LOG\_DIR\_EXPANSION\_VAR: slider-agent.out***

*####put componentStatus ####*

***componentStatus#***

***addOn pkgStatus <= applicationPackage : metainfo***

#RoleInstance initial => instance

#command : ***commandAsString <= containerLauncher.getCommandAsString***  //需要查看

#environment: ***envDescription <= containerLauncher.dumpEnvToString***  //需要查看

#role : ***containerRole***

#roleId: ***role.id <= ProviderRole***

#appVersion: ***<=global :site.global.app\_version***

#delay : ***container.launch.delay.sec //container start delay time***

5) ContainerLaunchContext initial

AbstractLauncer => ContainerLauncher

#compleContainerLaunch

***#setCommands <= commands***

***#setEnvironment <= env***

***#setServiceData <= serviceData***

***#setLocalResources <= localResources***

***#setTokens: tokenBuffer***

6) startContainer

Container的启动命令是由各个ApplicationMaster通过RPC函数

ContainerManagementProtocol#startContainers向NodeManager发起的，NodeManager中的ContainerManager（组件实现为ContainerManagerImpl）负责接收并处理该请求。

Container的启动过程主要经历三个阶段：资源本地化、启动并运行Container和资源清理

NMClientAsyncImpl#startContainerAsync

Send => StartContainersEvent: container,containerLaunchContext

StartContainerRequest#setContainerLauncherContext //set containerLauncherContext

ContainerManagerImpl#startContainerInternal

1.资源本地化

创建工作目录，及从HDFS下载各类文件资源

核心实现类：

—> ResourceLocalizationService, 接收INIT\_CONTAINER\_RESOURCES事件

—> LocalResourcesTrackerImpl, 将为对应的资源创建LocalizedResource以跟踪资源的生命周期

—>LocalizedResource，资源执行状态的检查

—>ResourceLocalizationService，调用ContainerExecutor.startLocalizer下载资源

ContainerLocalizer

—> LinuxContainerExecutor#startLocalizer

localDirs

logDirs

localResources

2.Container的运行

ContainerImpl向ContainersLauncher服务发送LAUNCH\_CONTAINER事件，请求启动Container

ContainersLauncher收到事件后，将为该Container创建一个Callable类型的对应ContainerLaunch,，并创建Token文件和执行脚本launch\_container.sh，并保存至NodeManager目录../nmPrivate中，其中launch\_container.sh中包含执行container所需要的环境变量、运行命令等全部信息

ContainerLaunch#call

—> command <= ***ContainerLaunchContext.getCommands***

—> localResources <= ***ContainerLaunchContext.getLocecalizedResoures***

—> environment <= ***ContainerLaunchContext.getEnvironment***

appDirs <=

containerLogDirs <=

—>sanitizeEnv: ApplicationConstants#Environment

Map<String,String> environment:

CONTAINER\_ID <= ***container.getContainerId***

NM\_PORT <= ***context.getNodeId.getPort***

NM\_HOST <= ***context.getNodeId.getHost***

NM\_HTTP\_PORT <= ***context.getHttpPort***

LOCAL\_DIRS <= ***appDirs***

LOG\_DIRS <= ***containerLogDirs***

USER <= ***container.getUser***

LOGNAME <= ***container.getUser***

HOME <= ***yarn.nodemanager.user-home-dir***

PWD <= ***pwd.name : PATH***

HADOOP\_CONF\_HOME <= ***system.getEnv(HADOOP\_CONF\_DIR)***

……

—>CONTAINER\_SCRIPT <= Shell.appendScriptExtension(launch\_container)

—>nmPrivateContainerScriptPath

—>exec :LinuxContainerExecutor

#writeLaunchEnv : environment,localResources,launchContext.getCommands

env(key,value): line(“export ”,key,”=\”,”key”) //UnixShellScriptBuilder

link(src,dst): line(“ln –sf \”,src,dst)

command(command:List<String>): line(“exec /bin/bash –c”,StringUtils.join)

#activateContainer

pidFiles <= ***containerID,pidFilePath***

#launchContainer:

<Container,

nmPrivateContainerScriptPath,

nmPrivateTokensPath

,user,

appIdStr,

containerWorkDir,

localDirs,

logDirs>

—> resourcesHandler#preExecute: containerID,container.getResource

—>ShellCommandExecutor#execute

launch\_container.sh脚本如下：

#!/bin/bash

export JAVA\_HOME="/usr/jdk64/jdk1.7.0\_67"

export NM\_AUX\_SERVICE\_mapreduce\_shuffle="AAA0+gAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=^M"

export HADOOP\_USER\_NAME="fys"

export NM\_HOST="bditest3.cmss.com"

export HADOOP\_YARN\_HOME="/usr/lib/hadoop-yarn"

export AGENT\_LOG\_ROOT="/data11/hadoop/yarn/log/application\_1461811649094\_0010/container\_1461811649094\_0010\_01\_000002"

export PYTHONPATH="./infra/agent/slider-agent/"

export JVM\_PID="$$"

exportPWD="/data10/hadoop/yarn/local/usercache/fys/appcache/application\_1461811649094\_0010/container\_1461811649094\_0010\_01\_000002"

export NM\_PORT="45454"

export LOGNAME="fys"

export MALLOC\_ARENA\_MAX="4"

export NM\_HTTP\_PORT="8042"

export USER="fys"

export CONTAINER\_ID="container\_1461811649094\_0010\_01\_000002"

export HOME="/home/"

export AGENT\_WORK\_ROOT="$PWD"

export HADOOP\_CONF\_DIR="/etc/hadoop/conf"

exportSLIDER\_PASSPHRASE="EAFsDSUExeLJSMg44L0aMHLHgVPe3pfozXEyATx6YXbmZx9QiR"

export LANG="en\_US.UTF-8"

mkdir -p app

hadoop\_shell\_errorcode=$?

if [ $hadoop\_shell\_errorcode -ne 0 ]

then

exit $hadoop\_shell\_errorcode

fi

ln -sf "/data15/hadoop/yarn/local/usercache/fys/appcache/application\_1461811649094\_0010/filecache/11/slider-kafka-app-package-0.80.0-bc1.3.0.zip" "app/definition"

hadoop\_shell\_errorcode=$?

if [ $hadoop\_shell\_errorcode -ne 0 ]

then

exit $hadoop\_shell\_errorcode

fi

mkdir -p infra

hadoop\_shell\_errorcode=$?

if [ $hadoop\_shell\_errorcode -ne 0 ]

then

exit $hadoop\_shell\_errorcode

fi

ln -sf "/data14/hadoop/yarn/local/usercache/fys/appcache/application\_1461811649094\_0010/filecache/10/slider-agent.tar.gz" "infra/agent"

hadoop\_shell\_errorcode=$?

if [ $hadoop\_shell\_errorcode -ne 0 ]

then

exit $hadoop\_shell\_errorcode

fi

exec /bin/bash -c "python ./infra/agent/slider-agent/agent/main.py --label container\_1461811649094\_0010\_01\_000002\_\_\_KAFKA\_BROKER --zk-quorum bditest3.cmss.com:2181,bditest2.cmss.com:2181,bditest1.cmss.com:2181 --zk-reg-path /registry/users/fys/services/org-apache-slider/kafka210 > /data11/hadoop/yarn/log/application\_1461811649094\_0010/container\_1461811649094\_0010\_01\_000002/slider-agent.out 2>&1 "

hadoop\_shell\_errorcode=$?

if [ $hadoop\_shell\_errorcode -ne 0 ]

then

exit $hadoop\_shell\_errorcode

fi

注：

通过EXPORT 将一些配置加载到环境变量中，包括JAVA\_HOME/HADOOP\_CONF等

启动slider agent的命令如下：

exec /bin/bash -c

"python ./infra/agent/slider-agent/agent/main.py

--label container\_1461811649094\_0010\_01\_000002\_\_\_KAFKA\_BROKER

--zk-quorum bditest3.cmss.com:2181,bditest2.cmss.com:2181,bditest1.cmss.com:2181

--zk-reg-path /registry/users/fys/services/org-apache-slider/kafka210

>/data11/hadoop/yarn/log/application\_1461811649094\_0010/container\_1461811649094\_0010\_01\_000002/slider-agent.out 2>&1 "

启动slider-agent的main.py进程