YARN Fast\_launch

YARN Server启动命令中有enableFastLaunch标识：

*-enableFastLaunch <Destination Folder>*

*Uploads AM dependencies to HDFS to make future launches faster.Optionally a destination folder for the tarball can be specified.*

可以将AM依赖jar包预先上传到HDFS上，一般以TARBALL的形式提供，加快AM的启动速度，如果不配置目录，则默认通过下面的参数指定：

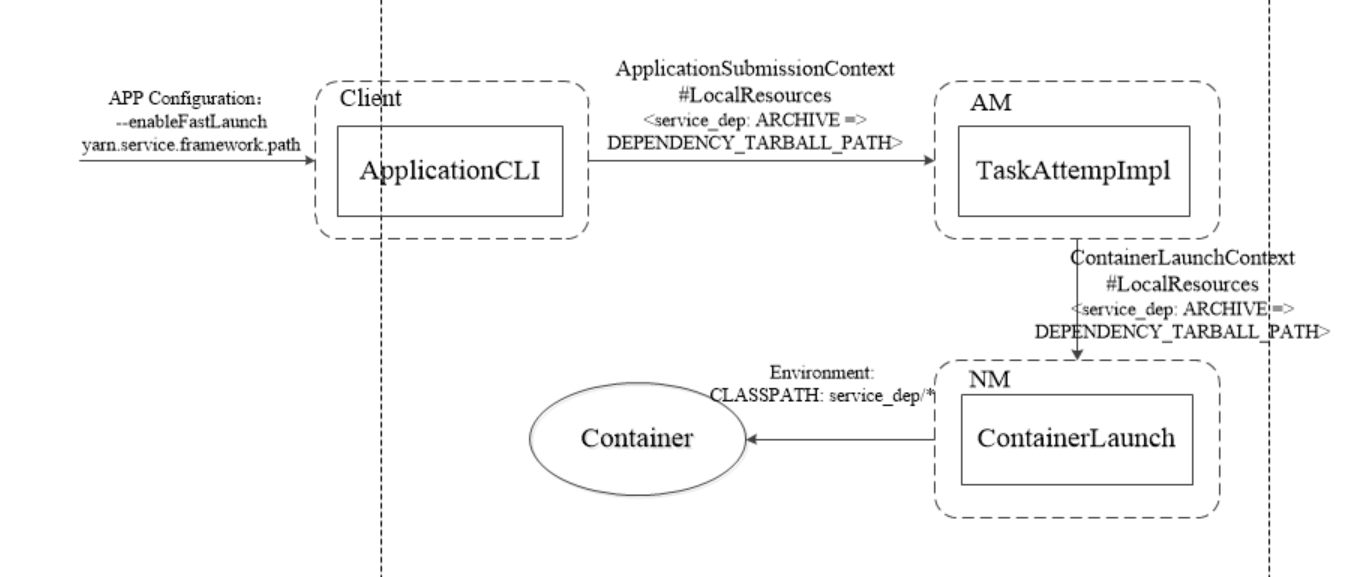
*<property>*

*<name>yarn.service.framework.path</name>*

*<value>/bch/apps/3.0.0/yarn/service-dep.tar.gz</value>*

*</property>*

其执行流程图如下所示：



1. 通过参数，设置将yarn.service.framework.path中的资源添加到LocalResource中

*Path dependencyLibTarGzip = fs.getDependencyTarGzip();  
if (fs.isFile(dependencyLibTarGzip)) {  
 fs.submitTarGzipAndUpdate(localResources);  
}*

1. 通常配置依赖资源为ARCHIVE

*public enum LocalResourceType {*

*/\*\**

*\* Archive, which is automatically unarchived by the <code>NodeManager</code>.*

*\*/*

*ARCHIVE,*

*FILE,*

*PATTERN)*

上述示例中是LLAP的配置示例，其通过LocalResource将Tarball资源从HDFS下载到本地，并且解压，其执行如下：

FSDownload#unpack

*case ARCHIVE:*

*if (lowerDst.endsWith(".jar")) {*

*RunJar.unJar(inputStream, dst, RunJar.MATCH\_ANY);*

*} else if (lowerDst.endsWith(".zip")) {*

*FileUtil.unZip(inputStream, dst);*

*} else if (lowerDst.endsWith(".tar.gz") ||*

*lowerDst.endsWith(".tgz") ||*

*lowerDst.endsWith(".tar")) {*

*FileUtil.unTar(inputStream, dst, lowerDst.endsWith("gz"));*

*}*

*......*

*}*

1. 将依赖的JAR(LocalsResource)包添加到luanch\_contianer.sh中的CLASSPATH中，其执行如下：

*ClasspathConstructor classpath =  
 buildClasspath(YarnServiceConstants.SUBMITTED\_CONF\_DIR, "lib", fs, getConfig()  
 .getBoolean(YarnConfiguration.IS\_MINI\_YARN\_CLUSTER, false));*

*if (sliderFileSystem.isFile(sliderFileSystem.getDependencyTarGzip())) {*

*//service\_dep  
 classpath.addLibDir(YarnServiceConstants.DEPENDENCY\_LOCALIZED\_DIR\_LINK);  
 }  
env.put("CLASSPATH", classpath.buildClasspath());*

在脚本中生成的环境变量如下所示：

*ln -sf "..../filecache/10/service-dep.tar.gz" "service\_dep"*

*export CLASSPATH="......:lib/\*:service\_dep/\*:$CLASSPATH:$HADOOP\_CONF\_DIR"*

附录：MapReduce Framework Archive，配置参数如下所示：

*public static final String MAPREDUCE\_APPLICATION\_FRAMEWORK\_PATH =*

*"mapreduce.application.framework.path";*

将该参数指定的配置，配置到CACHE\_ARCHIVES中

*String framework =  
 conf.get(MRJobConfig.MAPREDUCE\_APPLICATION\_FRAMEWORK\_PATH, "");*

*Path frameworkPath = fs.makeQualified(*

*new Path(uri.getScheme(), uri.getAuthority(), uri.getPath()));*

*FileContext fc = FileContext.getFileContext(frameworkPath.toUri(), conf);*

*frameworkPath = fc.resolvePath(frameworkPath);*

*uri = frameworkPath.toUri();*

*try {*

*uri = new URI(uri.getScheme(), uri.getAuthority(), uri.getPath(),*

*null, linkedName);*

*} ......*

*DistributedCache.addCacheArchive(uri, conf);*

具体使用不再详述，目前默认参数为：

*/bch/apps/3.0.0/mapreduce/mapreduce.tar.gz*

tar.gz中包含了Hadoop的安装目录，

*hadoop*

├── *bin*

├── *etc*

├── *include*

├── *lib*

├── *libexec*

├── *LICENSE.txt*

├── *NOTICE.txt*

├── *README.txt*

├── *sbin*

└── *share*

MR Job的launch\_container.sh中启动中CLASSPATH，配置如下：

*export CLASSPATH="......;mapreduce.tar.gz:job.jar/\*:job.jar/classes/:job.jar/lib/\*:$PWD/\*"*

*export TIMELINE\_FLOW\_RUN\_ID\_TAG="1536941449941"*

*echo "Setting up job resources"*

*ln -sf "/hadoop/yarn/local/filecache/11/mapreduce.tar.gz" "mapreduce.tar.gz"*

*ln -sf "/hadoop/yarn/local/usercache/hdfs/appcache/application\_1536914037863\_0016/filecache/11/job.jar" "job.jar"*