Slider技术内幕

­ ——深入解析Slider架构设计与实现原理

## 第3章 Slider应用程序设计方法

YARN是一个资源管理系统，负责资源的管理和调度，如果想要将一个应用程序运行在YARN之上，通常需要编写两个组件客户端和ApplicationMaster。这两 个组件编写非常复杂，尤其是ApplicationMaster，需要考虑RPC调用、任务容错等细节。

Apache Slider将大量应用程序抽象成一种通用框架，其自带HBase、Storm和Accumulo的服务实现，如果其他应用也采用Slider在Yarn中部署，需要按照Slider的规范编写一系列的配置和脚本，然后打包文件。Application直接运行在Yarn中，每个应用都包括打包、部署及生命周期管理。下面对如何将App进行Sliderize化，进行介绍。

### 3.1 Slider应用部署

步骤1：编译Slider应用

以HBase为例，下载HBase的安装包hbase-1.1.3-bc1.3.0.tar.gz，然后进入slider 源代码目录app-packages/hbase，使用下面的命令进行应用的打包：

mvn clean package

-P hbase-app-package

-D pkg.version=1.1.3-bc1.3.0

-D pkg.name=hbase-1.1.3-bc1.3.0.tar.gz

-D pkg.src=/home/develop/gittrunck/tarball/

在../target中会编译生成hbase的应用安装包slider-hbase-app-package-1.1.3-bc1.3.0.zip

步骤2：安装应用包，命令如下：

slider package --install

--name HBASE

--package slider-hbase-app-package-1.1.3-bc1.3.0.zip

应用安装包部署在hdfs上的<User home dir>/.slider/package/<name provided in the command>目录中，该路径在应用描述文件appConfig.json中的app.def中指定。

步骤3:

slider application的配置包括两个部分：Resource描述和Application配置，在应用的安装包中有resources-default.json和appConfig.json的模板例子。resources.json描述定义部署的components、数目及资源，每个component都运行在YARN container中，该文件放置在本地文件中，appConfig.json中定义了应用的运行参数(app.def参数必须定义)。

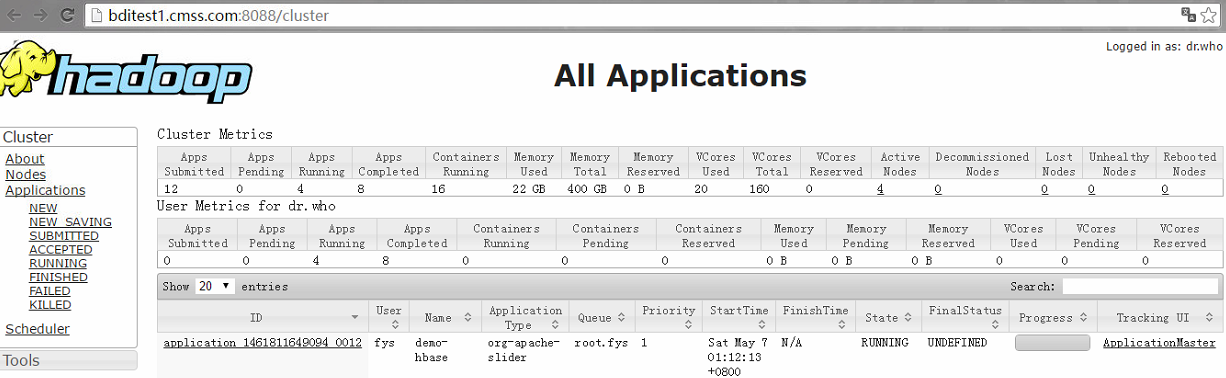
Application的启动命令如下所示：

./slider create demo-hbase

--template appConfig.json

--resources resources.json

成功启动后，可以通过YARN RM Web UI看到具体信息，如下所示：



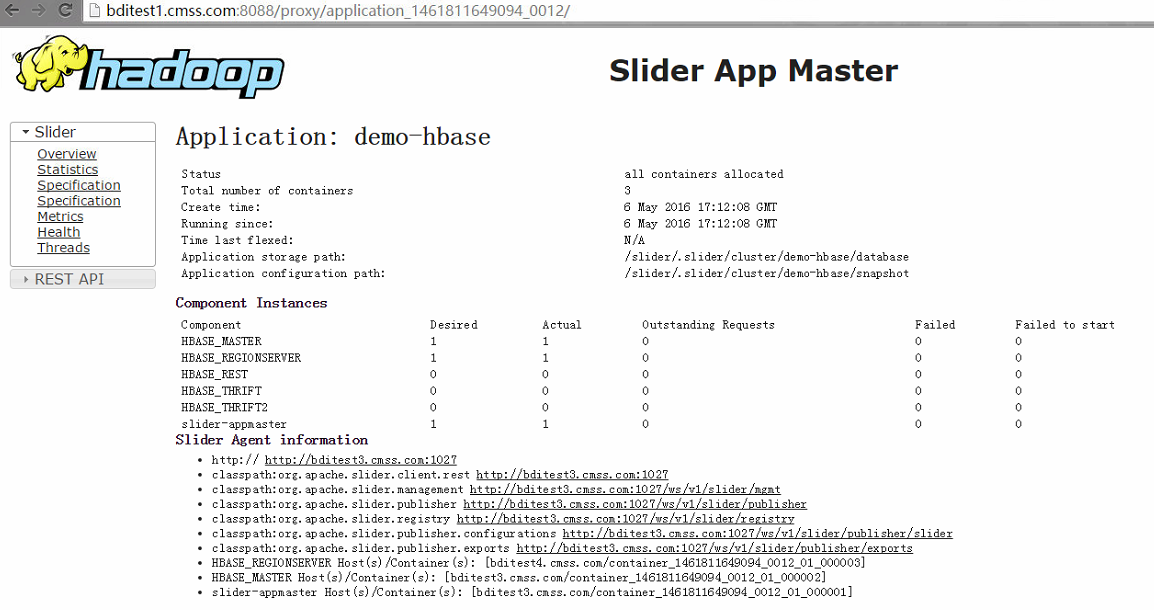


图3-1 Slider App在YARN中显示页面

步骤4：查看demo-hbase的集群信息，通过slider 命令，查看集群监控信息，命令如下：

slider registry

--name demo-hbase

--getexp quicklinks

获取信息，如下所示：

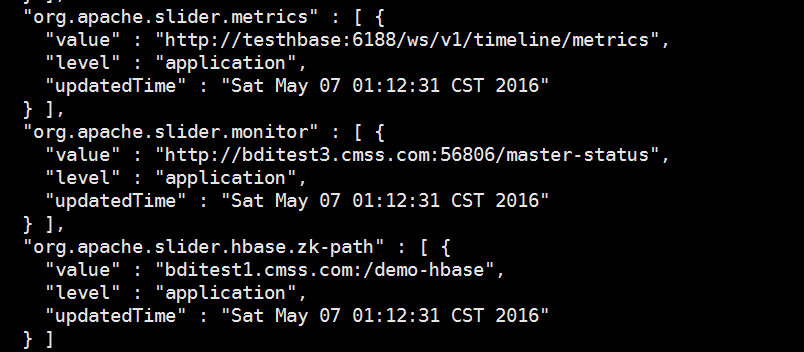


图3-2 Slider app的export信息

访问monitor，如下所示：

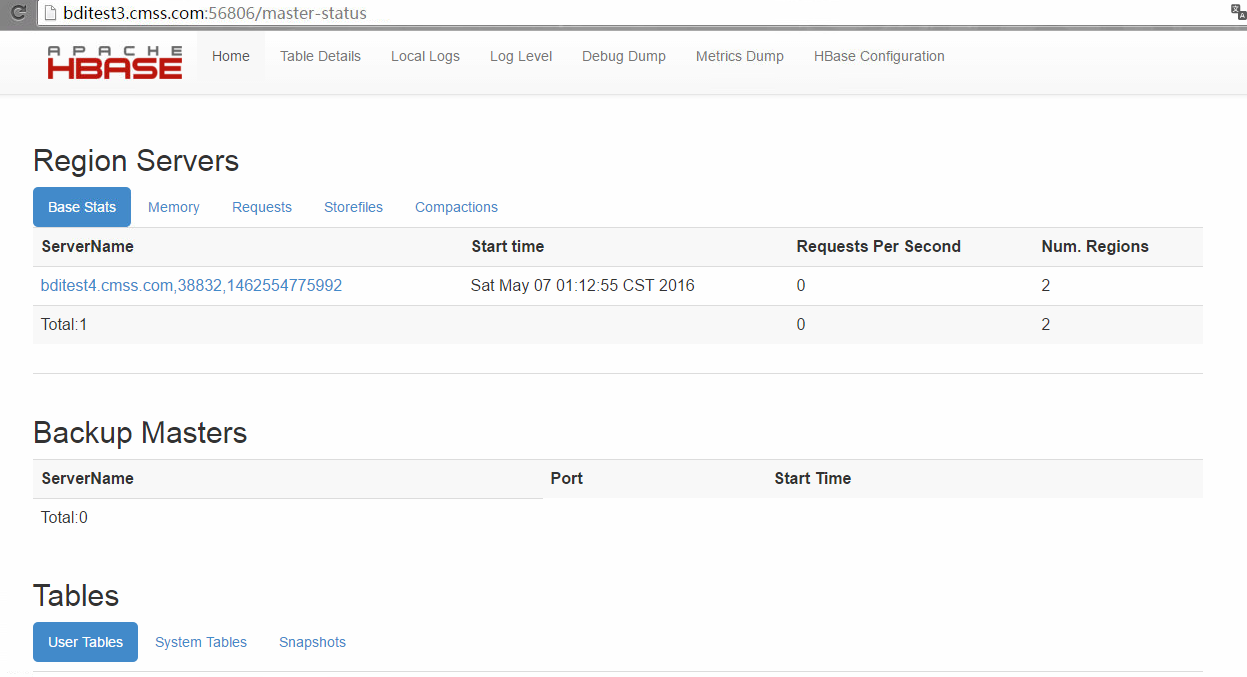


图3-3 HBase HMaster Web UI

### 3.2 Slider应用定义

创建一个Slider应用至少三个文件，应用的安装包(.zip)、应用instance配置appConfig.json,及资源配置（resources.json）。下面进行具体介绍：

#### 3.2.1 app package目录结构分析

Slider将应用安装所需要的脚本及安装tarball等打包到 zip压缩包中，具体的打包方式在上节进行了简单的介绍，将app包解压后的目录结构如下图所示：

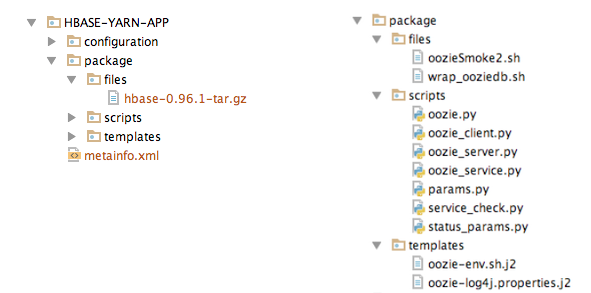


图3-4 HBASE-YARN-APP和OOZIE app package目录结构

1. app definition文件，应用的描述，包括应用的组成、定义、支持的平台、版本等
2. default configurations目录，应用相关的配置和配置文件
3. package/scripts目录，命令执行脚本，实现应用或者Component的管理操作
4. package/templates，应用的配置模板，用于根据参数生成配置文件
5. package/files，其他脚本，txt文件及tarballs等

在使用过程中，文件及目录的设置很灵活，例如metainfo.xml可以定义成metainfo.json；一般将app package放到hdfs上，但是可以将其放到本地目录，通过application.def来指定；template文件是可选项；metainfo可以通过命令行的参数来构建等。

#### 3.2.2 app definition

YARN 应用信息的描述，保存在metainfo.xml中，主要包括以下字段，其中可选字段用斜体：

* name: Application的名称
* version：应用的版本，name和version唯一的标识应用
* type：应用的类型，YARN-APP标志引用运行在YARN中
* minHadoopVersion：app可以运行在最低的Hadoop版本
* components：应用的组成component列表
* osSpecifies：应用使用的OS specific tarball信息
* *commandScript*，定义的应用component的管理脚本
* *comment:* 应用的描述
* *exportConfigs*，提供给客户端的config

每个应用都有多个component，相关的字段如下：

* name：component的名称
* category: component的类别，MASTER,SLAVE及CLIENT
* *minInstanceCount*:该component需要的最少的数目
* *maxInstanceCount*:该component可以启动的最多的instances数目
* *appExports*：该component需要向client提供的application level的export
* *compExports*：component向client提供的component level的export

Component包括的export的定义，包括以下字段：

* name：export的名称
* value:由slider解析的值模板，在运行中替换成具体值，然后提供给客户端

Component中定义命令的执行脚本，模块为commandScript，包括以下字段：

* script：script的位置，在app package root中的相对目录
* scriptType：script的类型
* timeout：脚本的执行timeout
* *customCommands*:组件的附加命令

应用需要定义组件在activation过程中相互之间的命令执行顺序：

* command:以{component}-{command}的模式定义组件和命令，但是当前START是目录支持的有效命令
* requires:定义component执行某命令时需要的状态，模式为{component}-{state}，当前STARTED是有效的状态

应用可以定义某个component Active时向client提供的properties列表（特别是urls），可以是jmx endpoint，properties组成成export groups（exportGroup），这些值也可以通过registry service中获取

* name：export group中的名称，每个exportGroup包含一个或者多个export

export group中包括的export如下：

* name：export的名称
* value：由slider生成内容的模板

metainfo.xml的一个例子如下：

<metainfo>

<schemaVersion>2.0</schemaVersion>

<application>

<name>HBASE</name>

<version>0.96.0.2.1.1</version>

<type>YARN-APP</type>

<minHadoopVersion>2.1.0</minHadoopVersion>

<components>

<component>

<name>HBASE\_MASTER</name>

<category>MASTER</category>

<minInstanceCount>1</minInstanceCount>

<maxInstanceCount>2</maxInstanceCount>

<exportedConfigs>hbase-site</exportedConfigs>

<commandScript>

<script>scripts/hbase\_master.py</script>

<scriptType>PYTHON</scriptType>

<timeout>600</timeout>

</commandScript>

</component>

<component>

<name>HBASE\_REGIONSERVER</name>

<category>SLAVE</category>

<minInstanceCount>1</minInstanceCount>

...

</component>

</components>

<osSpecifics>

<osSpecific>

<osType>any</osType>

<packages>

<package>

<type>tarball</type>

<name>hbase-0.96.1-tar.gz</name>

<location>package/files</location>

</package>

</packages>

</osSpecific>

</osSpecifics>

<commandOrders>

<commandOrder>

<command>HBASE\_REGIONSERVER-START</command>

<requires>HBASE\_MASTER-STARTED</requires>

</commandOrder>

</commandOrders>

<exportGroups>

<exportGroup>

<name>QuickLinks</name>

<exports>

<export>

<name>JMX\_Endpoint</name>

<value>http://${HBASE\_MASTER\_HOST}:${site.hbase-site.hbase.master.info.port}/jmx</value>

</export>

<export>

<name>Master\_Status</name>

<value>http://${HBASE\_MASTER\_HOST}:${site.hbase-site.hbase.master.info.port}/master-status</value>

</export>

</exports>

</exportGroup>

</exportGroups>

</application>

</metainfo>

#### 3.2.3 app configuration

该小节用于描述应用的默认配置，该配置会基于用户提供的instance configuration、集群信息（例如HDFS root，local dir root等）、container分配的资源（端口和hostname）及依赖（如zk quorum hosts）等进行默认配置的更新。

App Configuration目录中包含了应用所需要的所有配置，config文件包括site.xml、log4j properties文件等。另外，有一些参数不需要放到config文件中，例如可以替换配置模板中的参数（evn.sh文件中的参数，环境变量，如JAVA\_HOME,用户名等），这些参数可以通过用户提供，或者由环境提供。这些参数定义在app\_config.xml中。



图3-5 app conf的目录

Config文件的格式如下：

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

...

</property>

</configuration>

每个配置的项描述如下所示：

<property>

<name>storm.zookeeper.session.timeout</name>

<value>20000</value>

<description>The session timeout for clients to ZooKeeper.</description>

<required>false</required>

<valueRestriction>0-30000</valueRestriction>

</property>

<property>

<name>storm.zookeeper.root</name>

<value>/storm</value>

<description>The root location at which Storm stores data in ZK.</description>

<required>true</required>

</property>

<property>

<name>jvm.heapsize</name>

<value>256</value>

<description>The default JVM heap size for any component instance.</description>

<required>true</required>

</property>

<property>

<name>nimbus.host</name>

<value>localhost</value>

<description>The host that the master server is running on.</description>

<required>true</required>

<clientVisible>true</clientVisible>

</property>

* name ：参数的名称
* value：参数的默认值
* description：参数的描述
* required：是否必须定义，默认是false
* clientVisible：客户端是否可以查看到

#### 3.2.4 Package folder

该package包括应用的管理操作，目录scripts、templates和files，作用如下所示：

1. script folder

该目录用于应用的操作的实现，包括5个默认操作和一个组合操作，restart 可以定制实现。默认操作包括install、configure、start、stop、status和restart(stop+start)。

脚本的使用在metainfo.xml中，脚本的定义如下所示：

class HbaseMaster(Script):

def install(self, env):

self.install\_packages(env)

def configure(self, env):

import params

env.set\_params(params)

hbase(name='master')

def start(self, env):

import params

env.set\_params(params)

self.configure(env) # for security

hbase\_service( 'master',

action = 'start'

)

1. templates folder

配置模板文件（非常规的config文件）。在slider 中定义了一个lib用于将property bag的值实例化模板，生成配置文件，其他文件如env sh文件、log4j等也可以通过模板生成。

模板文件，例如dfs.exclude文件，用于列举excluded/decommission hosts，定义在JSON\_FILE的模板如下：

{% if hdfs\_exclude\_file %}

{% for host in hdfs\_exclude\_file %}

{{host}}

{% endfor %}

{% endif %}

在params.py中定义的hdfs\_excluder\_files。

1. files folder

存放管理操作所需要的文件，包括安装所需要的tarballs及操作使用的shell脚本

#### 3.2.5 App Instance Configuration

上面介绍了应用的安装包(.zip)的目录结构，在创建应用时，需要定义应用实例的配置，在appConfig.json文件中，定义文件中配置会覆盖config文件中的默认配置。

appConfig.json中格式如下：

{

"schema": "http://example.org/specification/v2.0.0",

"metadata": {

},

"global": {

},

}

在应用启动后，会将appConfig中配置的参数传送这具体启动的component。

1. 变量命名约定

Slider Agent接收命令（例如START、INSTALL及CONFIGURE等）的执行数据包，命令数据包的configuration部分的变量site.xx.yy解释为配置组xx，参数名为yy，将参数yy会写入配置文件xx中。例如,site.hbase-site.hbase.regionserver.port，发送到slider agent后，hbase.regionserver.port写入配置文件hbase-site中。类似core-site.fs.defaultFS，命令执行时变量fs.defaultFs放入模块core-site中。格式如site.global.zz的变量，以global方式传递的变量使用在模板替换或者其他过滤条件中。

Slider variables，以site.xx.yy为格式传递的配置是slider自身运行所需要的配置，例如必须配置的参数：

application.def：应用定义包的位置，例如/slider/hbase\_v096.zip

system\_config：传送至containers中的container type的列表（例如core-site,hdfs-site等）

java\_home：java home路径，例如/usr/jdk64/jdk1.7.0\_45

OPTIONAL：create.default.zookeeper.node，应用是否使用默认的ZK node

Dynamicall allocated ports：Slider需要动态分配端口给应用，或者设置为0。hbase master info 端口及JMX endpoint等。

动态分配的端口格式：${COMPONENT\_NAME.ALLOCATED\_PORT}，例如:site.hbase-site.hbase.master.info.port设置为${HBASE\_MASTER.ALLOCATED\_PORT}

1. Available environment variable

|  |  |
| --- | --- |
| 参数 | 描述 |
| AGENT\_LOG\_ROOT | Log的目录 |
| AGENT\_WORK\_ROOT | 工作目录 |
| NN\_URI | 提供给Slider的NN URI |
| NN\_HOST | NN的host |
| COMPONENT\_NAME.ALLOCATED\_PORT | 让Slider分配端口 |
| {COMPONENT\_NAME.ALLOCATED\_PORT}  {PER\_CONTAINER} | 每个Container分配的端口 |
| DEFAULT\_ZK\_PATH | 如果create.default.zookeeper.node设置true，Slider创建默认的ZK路径 |

#### 3.2.6 Resource Specification

资源定义文件(resoures.json)，定义了每个component类型的YARN资源。包括container的cpu、内存 、container分配策略（包括YARN Label）、failure policy用于处理component的failing及logs的产生等。

使用的应用资源需求有，两个组件master和worker。Slider会默认的增加AppMaster，组件为slider-appmaster。组件的instances的一些参数如下：

yarn.component.instances，component类型的instances的数目

yarn.memory，component instance的分配的内存

yarn.vcores，需求的virtuals数目

yarn.role.priority，组件的唯一标识

对于Java应用，需要设置JVM堆大小，yarn.memory的设置必须比JVM大，JVM实际使用的堆一般使用的必设置的大，一般yarn.memory至少比jvm大50%。

{

"schema": "http://example.org/specification/v2.0.0",

"metadata": {

},

"global": {

},

"components": {

"HBASE\_MASTER": {

"yarn.role.priority": "1",

"yarn.component.instances": "1",

"yarn.placement.escalate.seconds": "10"

},

"HBASE\_REGIONSERVER": {

"yarn.role.priority": "2",

"yarn.component.instances": "10",

"yarn.placement.escalate.seconds": "600"

},

"slider-appmaster": {

}

}

}

更详细的信息：

http://slider.incubator.apache.org/docs/configuration/resources.html

### 3.3 slider应用设计

下面以jmemcached为例，介绍Slider application package的定义。Jmemcached是Memcached（分布式缓存系统）的Java实现。Slider Jmemcached App需要提供启动一个或者多个memcached进程(需要配置使用内存)的基本功能，其使用的端口需要定制。

App Package的目录结构在上节已经介绍，Jmemcached app package的目录结构如下图所示：

jmemcached-1.0.0.zip

-------- ---- ---- ----

|--appConfig-default.json

|--1673 07-15-14 17:58 metainfo.xml

|--package/

|--package/files/

|--package/files/jmemcached-1.0.0.tar

|--package/scripts/

|--package/scripts/memcached.py

|--package/scripts/params.py

|--README.txt

|--resources-default.json

**步骤1：创建metainfo.xml**

Application的Metainfo包含的最小元素组合包 括name、comment、version，至少定义一个component type，该例子中定义一个component为MEMCACHED，同时定义tarball的信息，如下所示：

<metainfo>

<schemaVersion>2.0</schemaVersion>

<application>

<name>MEMCACHED</name>

<comment>Memcache is a network accessible key/value storage system, often used as a distributed cache.</comment>

<version>1.0.0</version>

<exportedConfigs>None</exportedConfigs>

<exportGroups>

<exportGroup>

<name>Servers</name>

<exports>

<export>

<name>host\_port</name>

<value>${MEMCACHED\_HOST}:${site.global.listen\_port}</value>

</export>

</exports>

</exportGroup>

</exportGroups>

<components>

<component>

<name>MEMCACHED</name>

<category>MASTER</category>

<compExports>Servers-host\_port</compExports>

<commandScript>

<script>scripts/memcached.py</script>

<scriptType>PYTHON</scriptType>

</commandScript>

</component>

</components>

<osSpecifics>

<osSpecific>

<osType>any</osType>

<packages>

<package>

<type>tarball</type>

<name>files/jmemcached-1.0.0.tar</name>

</package>

</packages>

</osSpecific></osSpecifics>

</application>

</metainfo>

**步骤2：Ensure Application tarball**

大多数应用的都有发布的tarball，否则需要手动的创建，本例子中，创建jmemcached的tarball包含cli和core jar，如下所示：

tar tvf jmemcached-1.0.0.tar

drwxr-xr-x 0 smohanty staff 0 Nov 5 20:22 ./

-rw-r--r-- 0 yarn hadoop 13537 Jul 15 17:51 jmemcached-cli-1.0.0.jar

-rwxr-xr-x 0 yarn hadoop 101467 Jul 15 17:51 jmemcached-core-1.0.0.ja

**步骤3：创建default的resources file(resources.json)**

默认的resources.json中必须包含slider-appmaster、MEMCACHED组件，同时为组件赋予唯一的priority标识、默认的instances的数目及yarn.memory的合适的值：

{

"schema" : "http://example.org/specification/v2.0.0",

"metadata" : {

},

"global" : {

},

"components": {

"slider-appmaster": {

},

"MEMCACHED": {

"yarn.role.priority": "1",

"yarn.component.instances": "1",

"yarn.memory": "256"

}

}

}

**步骤4：创建默认的配置模板（appConfig.json）**

配置文件中必须包含以下几项：

|  |  |
| --- | --- |
| 配置 | 描述 |
| application.def | 应用定义包的hdfs目录 |
| java\_home | 目标主机的JAVA目录 |
| site.global.additional\_cp | 该参数用于提供辅助Jars(hadoop client jars的目录) |
| site.global.xmx\_val | Xmx的值 |
| site.global.xms\_val | Xms的值 |
| site.global.memory\_val | 内存的值 |
| site.global.listen\_port | Slider需要知道的分配的ports |

根据需要将参数添加到appConfig.json中，例子如下所示：

{

"schema": "http://example.org/specification/v2.0.0",

"metadata": {

},

"global": {

"application.def": ".slider/package/MEMCACHED/jmemcached-1.0.0.zip",

"java\_home": "/usr/jdk64/jdk1.7.0\_67",

"site.global.additional\_cp": "/usr/lib/hadoop/lib/\*",

"site.global.xmx\_val": "256m",

"site.global.xms\_val": "128m",

"site.global.memory\_val": "200M",

"site.global.listen\_port": "${MEMCACHED.ALLOCATED\_PORT}{PER\_CONTAINER}"

},

"components": {

"slider-appmaster": {

"jvm.heapsize": "256M"

}

}

}

**步骤5：实现app的基本操作命令**

所有的Application都需要实现INSTALL/CONFIGURE/START/STOP/STATUS命令，一些实现可以为NOP。在本节的例子中，仅实现INSTAL用默认实现L、START命令，其他命令使用默认实现。

parameters的执行脚本parameters.py，如下：

from resource\_management import \*

config = Script.get\_config()

app\_root = config['configurations']['global']['app\_root']

java64\_home = config['hostLevelParams']['java\_home']

pid\_file = config['configurations']['global']['pid\_file']

additional\_cp = config['configurations']['global']['additional\_cp']

xmx\_val = config['configurations']['global']['xmx\_val']

xms\_val = config['configurations']['global']['xms\_val']

memory\_val = config['configurations']['global']['memory\_val']

port = config['configurations']['global']['listen\_port']

parameters.py文件仅读取command实现所需要的参数，命令的执行脚本如下：

import sys

from resource\_management import \*

class Memcached(Script):

def install(self, env):

self.install\_packages(env)

def configure(self, env):

import params

env.set\_params(params)

def start(self, env):

import params

env.set\_params(params)

self.configure(env)

process\_cmd = format("{java64\_home}/bin/java -Xmx{xmx\_val} -Xms{xms\_val} -classpath {app\_root}/\*:{additional\_cp} com.thimbleware.jmemcached.Main --memory={memory\_val} --port={port}")

Execute(process\_cmd,

logoutput=False,

wait\_for\_finish=False,

pid\_file=params.pid\_file

)

def stop(self, env):

import params

env.set\_params(params)

def status(self, env):

import params

env.set\_params(params)

check\_process\_status(params.pid\_file)

if \_\_name\_\_ == "\_\_main\_\_":

Memcached().execute()

实现以下操作：

* 解压应用的tarball
* 读取配置及创建command字符串
* 执行启动jmemcached的命令
* 将pid写入文件，检查进程的执行状况

以上步骤实现了基本的应用定义，将这些文件压缩后，上传到hdfs目录后，使用slider create命令创建服务。