Apache Slider简介和使用

# 1、简介

Yarn的产生是为了解决Hadoop中MapReduce框架的性能瓶颈，根据思想是将JobTrack

-er和TaskTracker两个主要的功能分离成单独的组件，分别进行资源的管理和任务的调度/监控，资源管理器负责所有应用程序计算资源的分配，每一个应用的ApplicationMaster负责相应的调度和协调。同时Yarn还提供了pluggable 结构来支持其他计算框架的引入，但是需要定向开发各计算框架的AppMaster。随着Yarn的成熟，产生了Apache Tez及Slider等类似的引擎，这列引擎可以快速的集成第三方计算和服务（processing and services）。

Tez是开源的支持DAG作业的计算框架，Tez在底层提供DAG编程接口，用户编写程序直接采用这些接口进行程序设计。相类似的框架是Oozie，但是其实比较高层次，只提供一种多类型作业（MR程序、Hive和Pig）依赖关系表达式。Tez基于Yarn的，与原有的MR共存。

Apache Slider源自于Hoya（将HBase部署在Yarn上的项目），通过Slider可以在Yarn环境中运行其他的应用，这些分布式的应用可以是非Yarn-aware的，通过这种方式可以将很多的服务运行在Yarn中（不需要进行任何修改）。Slider允许这些应用使用Hadoop的数据和计算资源（同时安全，管理及Hadoop的集群的操作能力）。在Apache Slider中部署服务，和运行一个普通分布式应用程序相同，应用程序部署起来，就意味着服务部署成功。

# 2、Apache Slider Apps部署

Apache Slider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，需要安装Slider规范进行编写一系列的配置和脚本，然后定义打包文件，包括下几个文件：

1）创建meteinfo.xml文件，内容包括：Apps的名称、版本、外部端口、包含的组件及实例数限制(min/max)、组件的启停顺序以及启停的python脚本、适用的OS 以及 Apps的压缩文件路径。详细规范可参考：

<http://slider.incubator.apache.org/docs/slider_specs/application_definition.html>

2）创建Apps的压缩文件。把Apps发布的 .tar.gz 下载下来，删除掉src/ 源文件目录再打包就可以了。也可以重新编译成package。

3）创建resources.json文件，内容包括：该app中各个组件的资源需求，包括<componentsName, memory, vcores, role\_priority, #instances>。还可以配置app在某段时间内失败N次的错误检测策略。详细规范可以参考：

<http://slider.incubator.apache.org/docs/slider_specs/resource_specification.html>

4）创建appConfig.json文件，以下字段是必须要定义的：application.def：Apps打包压缩文件在HDFS上的路径；java\_home：JDK路径；package\_list：Apps原始文件的路径；config\_types：配置文件名称；site.global.app\_user：运行应用的user；site.global.app\_root：Apps压缩文件解压后的路径。另外还可以定义 site.global.\* 的应用相关的配置参数。详细规范可以参考：

<http://slider.incubator.apache.org/docs/slider_specs/application_instance_configuration.html>

5）实现metainfo.xml定义的Apps相关各组件操作的脚本。详细规范可以参考：

<http://slider.incubator.apache.org/docs/slider_specs/writing_app_command_scripts>

下面介绍Slider及HBase Apps的安装和使用：

## 2.1 Slider的编译和安装

在Hadoop 2.6之前的版本都没有针对long-live service的程序进行考虑，但是在Hadoop 2.6 Yarn中增加了几点重要的feature，包括服务的注册，NM和RM的rolling update等特性。Slider中的AppMaster根据具体的Hadoop版本进行开发，因此Slider 0.30-0.50各个版本对Long-live service的管理并不好，Slider 0.60后才逐渐的改进，这里安装Slider 0.60：

下载slider源码：

$wget http://apache.fayea.com/incubator/slider/0.60.0-incubating/src/apache-slider-0.60.0-incubating-source

-release.tar.gz

解压slider

$tar -zxvf apache-slider-0.60.0-incubating-source-release.tar.gz

编译：

$cd apache-slider-0.60.0-incubating

$mvn clean package

注意：在pom.xml中hadoop的版本号是2.6.0-snapshot，改成2.6.0

编译后，在../slider-assembly/target/目录中，生成slider-0.60.0-incubating-all.zip安装文件，解压安装文件，并配置slider-client.xml及slient-env.sh文件

$unzip slider-0.60.0-incubating-all.zip

$cd slider-0.60-incubating/conf

$vim slider-client.xml

slider-client.xml的配置如下：

1. <property>
2. <name>yarn.resourcemanager.address</name>
3. <value>ambarimaster:8032</value>
4. </property>
5. <property>
6. <name>yarn.resourcemanager.scheduler.address</name>
7. <value>ambarimaster:8030</value>
8. </property>
9. <property>
10. <name>fs.defaultFS</name>
11. <value>hdfs://ambarimaster:8020</value>
12. </property>
13. <property>
14. <name>hadoop.registry.zk.quorum</name>
15. <value>ambarimaster:2181</value>
16. </property>

slider-env.sh的配置如下：

export JAVA\_HOME=${JAVA\_HOME}

export HADOOP\_CONF\_DIR=${HADOOP\_CONF\_DIR}

运行slider命令：

$bin/slider -version

结果如下：

1. 2015-01-10 20:42:26,846 [main] INFO client.SliderClient - Slider Core-0.60.0-incubating Built against commit# ${buildNumber} on Java 1.7.0\_60 by yarn
2. 2015-01-10 20:42:26,850 [main] INFO client.SliderClient - Compiled against Hadoop 2.6.0
3. 2015-01-10 20:42:26,861 [main] INFO client.SliderClient - Hadoop runtime version branch-2.6.0 with source checksum 18e43357c8f927c0695f1e9522859d6a and build date 2014-11-13T22:35Z
4. 2015-01-10 20:42:26,864 [main] INFO util.ExitUtil - Exiting with status 0

则安装成功

## 2.2 HBase的App的编译和安装

可以根据需要，选择HBase的安装版本，但是在slider-xxx的pom.xml中有hbase的版本，默认是hbase-0.98.4，对应的Apache的hbase-0.98.4-hadoop2-bin.tar.gz。

1）HBase安装文件的打包

在app-package/hbase的metainfo.xml和pom.xml中可以进行配置，不再介绍，本处使用默认的hbase的版本：

$cd ../apache-slider-0.60.0-incubating/app-package/hbase

$ mvn clean package -Phbase-app-package -Dpkg.version=0.98.4-hadoop2

-Dpkg.name=hbase-0.98.4-hadoop2.tar.gz

-Dpkg.src=/Users/user1/Downloads/2.2

2）安装应用包

在app-package/hbase/target中可以找到编译好的安装文件slider-hbase-app-package

-0.60.0-incubating.zip。使用命令将zip文件安装到hdfs中（安装后，slider可以自动找到该安装文件）

$bin/slider install-package --name HBASE --package ../slider-hbase-app-package-0.60.0-incubating.zip

默认的安装路径为/user/yarn/.slider/package/中，通过hdfs命令查看

$hdfs dfs -ls /user/yarn/.slider/package/HBASE

可以看到slider-hbase-app-package-0.60.0-incubating.zip

3）配置服务

根据具体需要，定义appConfig.json和resources.json。

appConfig.json用于配置HBase的基本参数，这部分参数在hbase-site.xml中已经进行了配置，例如hbase.rootdir,zookeeper.quorum等。必须配置的参数为application.def，其配置了HBase App安装包的所在位置

application.def": ".slider/package/HBASE/slider-hbase-app-package-0.60.0-incubating.zip

resource.json中定义了HBase需要启动的实例参数，包括HMaster,RegionServer等的组件个数，及各组件需要的memory（以后会扩展到cpu等资源）。下面是Region\_server组件的配置：

"HBASE\_REGIONSERVER": {

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

"yarn.component.instances": "3",

"yarn.memory": "1500"

}

4）创建HBase的集群实例

bin/slider create hbasetest --template ../sliderapp/appConfig.json --resources ../sliderapp/resources.json

通过该命令，会在Yarn中启动一个SliderAppMaster，根据需要从RM中获取资源，并启动Container，并在Container中启动HBase服务的各个组件。通过http://rm-host:8088，端口可以看到服务的运行状况。通过下面的命令可以看到已经启动的服务：

$slider list

5）Slider其他命令

这些服务启动后，可以进行停止，启动（重启）及销毁等操作

停止：

$slider stop hbasetest

启动：

$slider start hbasetest

销毁：

$slider destroy hbasetest

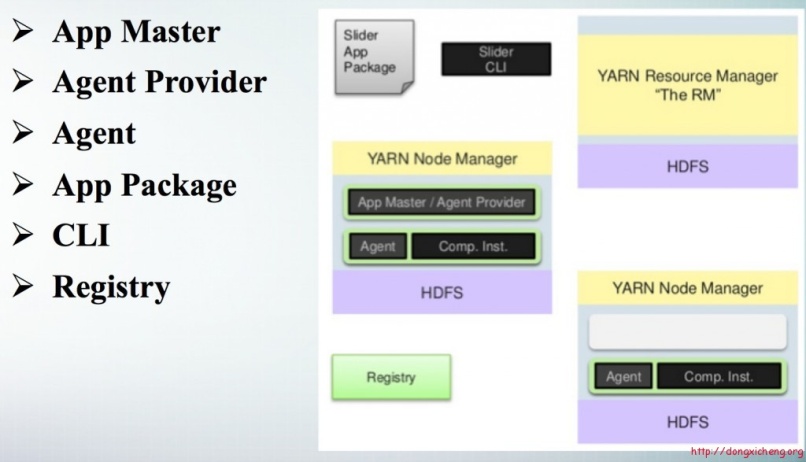
查看状态：

$slider status hbasetest

还有其他命令如，kill-container(停掉应用的某个container)，lookup（查找app）等命令，这些命令可以通过slider help进行查看

# 3、Apache Slider组件

Slider的基本组成模块及关系如下图所示：



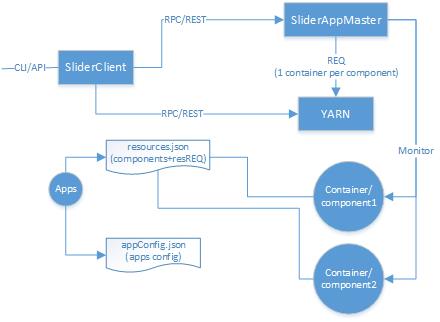
如上图，Slider的基本模块为App Master、Agent Provider、Slider Agent、CLI、Application Registry和App Package，下面进行详细介绍：

Slider AppMaster:根据App配置向Yarn申请Container，每个Component对应一个Container，并监控Container状态

Slider Client:与SlientAppMaster和Yarn通过REST/RPC通信，对外提供CLI和底层API

App Package: App应用程序安装包，需要提供resource.json（各个components以及mem/cpu资源需求），appConfig（各个components的配置信息）及.zip（app应用的压缩文件）

这三个组件的应用关系，如下图所示：



其中Slider AppMaster又由以下几个模块组成：

1）AM Engine: SliderClient和Yarn的接口。与RM交互请求和释放container，与NM交互启停Container。当集群变化时，AM Engine将该信息发送给Apps State，后者返回相应的action，例如请求释放container或者请求在某个特定节点上的资源。之后，AM Engine调用provider来创建App启动环境并启动

2）Agent Provider: 用户App的部署和启动，provider会在container中首先部署一个agent，该agent调用Provider的Rest API来执行返回的命令。具体包括，从HDFS上下载并解压app压缩文档，运行python脚本来启动服务

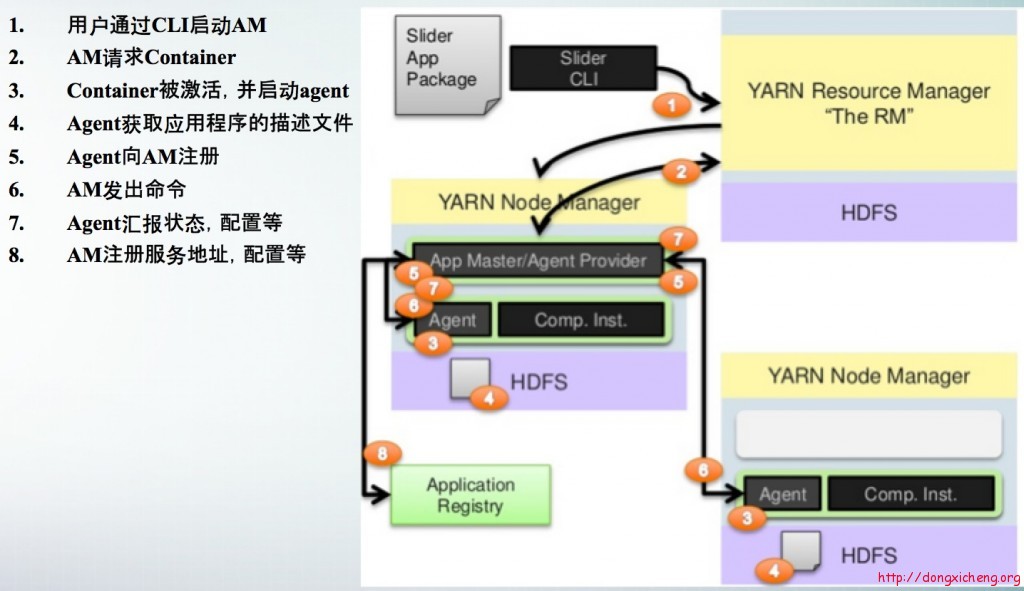
3）Apps State: 应用实例的信息，包括resource需求说明，当前各role运行哪些实例Instance的对应关系信息，Role History用以记录哪些Role部署在哪些nodes上的信息，该信息持久存储到磁盘中，以便后续快速启动服务。

Slider Agent，用户App的部署和启动是通过Agent来完成，Agent使用python程序来完成其任务，这点类似于Ambari agent

App Registry: 用于App的注册过程，与AM通信

# 4、Slider 运行流程及源码分析

使用Slider部署服务时，用户需要编写一系列的配置和脚本，并需要自己尝试编写ApplicationMaster和Clients实现，Slider自带HBase On Yarn、Storm On Yarn和Accumulo On Yarn，可以仿照着三个服务将自己的服务部署到Yarn中。将一个服务部署到Yarn中的流程如下图所示：



4.1 CLI启动AM

用户通过CLI启动AM，例如通过命令slider create,slider start 等方式，在Slider中客户端是slider.py脚本，在脚本中根据配置的JAVA\_HOME及HADOOP\_CONF等目录，创建客户端的虚拟机，通过jps命令可以看到名为slider（org.apache.slider.Slider）这个进程。 命令执行时，Slider由 ServiceLauncher#serviceMain来启动，Slider继承SliderClient，以App的create为例，分析其执行流程，如下图所



AM的启动

通过上述流程，SliderClient，通过SliderYarnClientImpl（继承YarnClientImpl）向RM提交App，RM根据配置分配Container给SlientAppMaster，并启动AppMaster。

4.2 AM请求Container

SliderClient通过submitApplication方法，向RM提交应用程序，首先是创建并启动AM服务。

4.3 Container激活及Agent启动

4.4 Agent获取应用程序的描述文件

4.5 Agent的注册

4.6 Service启动，及AM发出命令

4.7 Agent汇报状态及配置等

4.8 AM注册服务地址及配置等

# 5、Apache Slider Apps定义

Apache Slider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，需要安装Slider规范进行编写一系列的配置和脚本，然后定义打包文件。

下面介绍jmemcached的Slider app安装包的定义，一个App package的目录格式应该如下所示：

unzip -l "$@" jmemcached-1.0.0.zip

Archive: /jmemcached-1.0.0.zip

Length Date Time Name

637 07-15-14 19:17 appConfig-default.json

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

0 07-15-14 17:54 package/

0 07-15-14 18:03 package/files/

122880 07-15-14 18:03 package/files/jmemcached-1.0.0.tar

0 07-15-14 19:31 package/scripts/

1530 07-15-14 19:31 package/scripts/memcached.py

1287 07-15-14 18:46 package/scripts/params.py

1581 07-15-14 19:16 README.txt

252 07-15-14 17:58 resources-default.json

在安装包中，必须的文件为metainfo.xml，package目录（包括软件安装包，安装执行脚本），及默认的appconfig、resource定义文件。下面进行介绍：

## 5.1 metainfo.xml

该文件中至少要定义application，包括名字、描述及版本信息，在application中要至少要有一个components类型。下面是MEMCACHED的例子的定义文件，其中也包括了tarball的描述：

<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>

在这个定义文件中，首先定义了Memcached的名称及版本号，<exportGroups>定义了使用的端口。<components>中定义了一个类型为MEMCACHED MASTER，同时指定了执行脚本memcached.py和执行环境。在osSpeccifics中定义了Memcached所使用的tar包，指定了其位置。如果没有安装tar包，需要自己构建一个。

## 5.2 resource-default.json

resource.json文件，定义了服务中有哪些components。每个应用都必须有一个slider-appmaster，用于container的申请和管理。增加MEMCACHED组件，可以指定优先级、实例的数目及yarn.memory。下面是一个实例：

{

"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"

}

}

}

## 5.3 appConfig.json

该文件用于定义应用的配置信息，其中有几项是必须要定义：

1）application.def，应用安装包所在的位置，默认是HDFS中

2）java\_home，目标主机的jdk所放路径

3）package\_list，app tarball在app package中的位置，以files开始的相对路径（该项在0.60中没有，可能使用了默认的位置）

4）config\_types:一般是空值

5）site.global.app\_user，运行app的用户

6）site.global.app\_root，app root的位置，Slider将tarball解压缩到${AGENT\_WORK\_ROO

T}/app/install目录，因此根据application tarball的结构附件一个安装包的相对路径，例如jmemcached-1.0.0,（这个一定是解压后的目录名）

可以为应用添加其他的参数，这个根据需要进行定义，格式一般为：site.global.xxx

下面是一个例子：

{

"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"

}

}

}

## 5.4 基本命令（脚本）

所有的Appliation都需要实现以下几个命令INSTALL/CONFIGURE/START/STOP/STA

-TUS，一些命令可以使空的。这些命令都是通过py脚本来实现。

在slider agent/resource\_management中，已经实现了公用的执行脚本，如script.py,confi

g-dictionary.py等。在MEMCACHED的例子中，实现了两个脚本：parameter.py和Memcached.py：

1）parameter.py，该脚本用于从配置文件中获取应用参数，该参数可以定义在appConfig.json或者自定义xml文件中。

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']

2）memcached.py，该脚本用于实现应用的启动等各个操作，脚本要实现的功能基本上要有展开给定的tarball，读取配置及生成命令字符串，执行启动命令，将PID写入文件中（用于检测daemon的状态信息）。下面是Jmemcached的例子：

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,

user=params.app\_user,

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)

# 6、Slider 应用总结

尽管Slider项目的动机是将已存在的应用程序或者服务部署到Yarn上，但是就Yarn本身而言，还不能够很好的驾驭服务这种特殊的应用程序，这主要是由于Yarn的以下几个Feature尚未发布，包括：

1）RM/NM upgrade：RM和NM在升级时，正在运行的服务/Task不能受干扰，当升级完毕后，再重新向他们注册。（Yarn 666,Support rolling upgrades in YARN）

2）Logs management：服务的log是不断增加的，应该支持滚动递增,（Yarn 896）

3）服务注册：服务启动后，应该有一个中央化的服务注册组件，以供服务注册地址，从而让其他访问者发现服务的位置。（Yarn 913）

4）Container资源可动态伸缩。服务运行过程中，对资源的要求可能会改变，因为，应支持在不停止服务的前提下，对服务的资源动态调整。（yarn 1197）

以上只是列举了几个重要的feature，还有其他很多feature待开发。总之，让服务能够灵活的运行在YARN上，让YARN成为一个数据中心的资源管理系统，还有很长一段路要走。

# 7、Slider应用实践

使用Slider 0.60进行HBase的管理，Slider 0.60的本质上实现了SliderAppMaster及SliderAgent。其中SliderAppMaster是实现了Hadoop 2.6 Yarn接口的AppMaster，根据配置文件申请container，并在container中启动Slider agent。Slider agent实现了HMaster和HRegionServer等服务的启动。

## 7.1 Hadoop 2.6上运行HBase CDH 5.1.2

兼容性有问题

## 7.2 Hadoop bc 1.1.0上运行HBase CDH 5.1.2

1）由于Slider 0.60运行在Hadoop 2.6上，为了实现bc1.1.0对Slider 0.60的支持，增加yarn-registry子项目，实现的功能是将运行的service（例如HBase）注册到zookeeper中。

2）Slider 0.60实现的是Hadoop 2.6的接口，因此要改变0.60中对bc 1.1.0接口不兼容的问题，包括以下几个方面：

1. ApplicationSubmissionContext

#setLogAggregationContext，LogAggregation实现了NodeManager对日志上传至HDFS等，方便日后分析。LogAggregationContext，定义了IncludePattern和ExcludePattern，选择性的上传日志文件。hadoop 2.3中没有IncludePattern和ExcludePattern的设置。

#setNodeLabelExpression，提供更灵活的Applications控制，Hadoop 2.6中为了支持hadoop多租户集群管理，用户可以通过NodeLabel来控制job任务运行在哪些node中。Host,Queue,Job都赋予Label。Fair-scheduler等根据Label及资源情况调度Job。

1. CredentialProvider

证书管理机制，在Hadoop2.6中增加了UserProvider及JavakeyStoreProvider，但是Hadoop 2.3中没有这个权限管理机制。

1. 其他

在Hadoop 2.6中，增加了ApplicationAttemptNodeFoundException,ForbiddenException等异常类。

一些常量的增加和更改，如DFSConfigKeys.DFS\_NAMENODE\_KERBE..等

# 8、Slider REST接口

Slider 0.60使用Hadoop IPC进行Slider Client和Application Master的通信，之间的通信安全包括Authorization,authentication及encryption。但是仅通过Web Service URL提供read-only views。所有的services都是通过Jersey JAX-RS引擎，在Jetty Web Engine中托管，其实现是通过AMwebFilter将非RM Proxy IP address的请求转移都RM Proxy(302 redirect-as-GET)。下面介绍几个REST API:

1）/ws/v1/slider/registry：/v1/slider/registry，列举registry的根路径，其中包含了所有的实体