Slider 1.3.0 Design

——fys@cmss.com

Last Updated:2016.5.22

# Introduction

本文档讨论Slider升级到1.3.0的需求及设计。Apache Slider可以在YARN环境中部署及运行非YARN-AWARE应用，包括HBase,Tomcat等。使用Slider，用户可以在不对已有服务进行任何修改的前提下，将其部署到YARN集群中。Slider Router简化用户访问托管到YARN中服务的流程。

# Definitions

1. Service

Hadoop YARN集群中部署（可能是分布式）或者可访问的应用。例如，Apache HBase、Hcatalog、Apache Storm。服务可以是short-lived或者long-lived服务

1. Service class

Service类别的称呼，用于Registry中的路径，符合DNS的命名空间。例如，org-apache-hbase,org-apache-hcatalog

1. Component

一个Service的分布式组件，例如HBase的Master nodes，HBase Region Servers和HBase Rest Service

1. Service Instance

一个应用的Service实例。例如HBase集群的demo1实例。如果一个服务的components在运行，那么这个Service instance也在运行。

1. Component Instance

service实例的一个Component实例。例如在主机rack1serve6上的hbase master节点，或者rack3server40上的region server

1. Endpoint

Service Instance或者component Instance的binding信息。例如，HBase的Zookeeper binding，Regionserver的JMX 端口，HBase Master的Web UI，HBase Rest Component实例的REST API。

Endpoint分为两类：Internal，服务Instance内部使用，external，客户端使用的service Instance

1. Service Record

描述Service Instance及component Instance的register记录信息，包括Endpoints的列表。

# Problem Statement

YARN部署Long-lived的服务实例，例如Apache Tomcat Web Servers或者HBase集群。客户端需要访问服务instance，传统的方式是通过DNS或者配置文件中注册的IP。

但是Yarn中部署应用的主机名或者网络端口不能不对外暴露，因为客户端对这些应用的访问有难度。用户如果想访问这些应用，首先要根据启动的应用进程获取端口，然后根据端口再访问服务。这种方式不符合应用托管的特性，比较理想的情况是将应用托管到YARN中，用户可以根据自己的配置访问这些服务，而不是人为的去获取，而且一些信息不容易获取，例如Web UI，REST URI等。

YARN支持Application Master向Web URI及IPC address的基本注册。AppMaster启动的Task的注册REST URLs,ZK path及其他Endpoint；而且当App重启启动时Instance ID在不断的变化，静态的配置Named Service不可能解决binding问题，而且静态配置会导致指向不存在的service instance的问题。因此需要解决的问题：

1）提供服务器发现机制，当前是使用Nginx进行路由转发，但是仅支持Tomcat Server，而且需要暴露服务器的ip和端口（Nginx），用户跳转后会到实际的Ip和port，且不能通过{app}.yarn.domain的形式来访问。

2）App的容错机制，SliderRouterNode中实现的RouteTable(Nginx数据提供组件)在App故障时的更新，依赖Zookeeper Watcher的触发。RouteTable信息存储在HDFS中，在SliderRouterNode重启时加载，但是在目前为止的部署情况发现，经常会出现HDFS写入权限问题，而且更新依赖ZK Push机制。

3）Slider Router Node中缺少Slider App Web UI显示页面，每次访问都需要通过Slider Shell来访问，很不方便

4）Slider Router Node中App的管理都是通过Shell(目前支持部分Rest API)，不方面，后期Hcontrol集成Slider App的管理的方式也成为问题

5）其他，Slider Router Node代码中需要优化，例如冗余代码、测试代码、日志等，需要考虑扩展问题等。

6）Slider Docker的支持（需要实践，紧跟社区）

7）Slider Metrics实践，对其机制了解不够深入，将metrics和app的metrics结合

综上所述：Slider升级到1.3.0，需要做以下工作：

1）提供Slider DNS机制，无状态架构，不会持久存储App信息，将{app}.yarn.domain解释为这些服务运行的ip地址和端口(参考Mesos-DNS机制）。分成两个部分，DNS记录生成器，为所有运行的应用程序生成DNS A和SRV记录，周期性的查询YARN，得到所有运行的应用程序和任务信息；DNS解析器，负责处理DNS请求并恢复请求等。

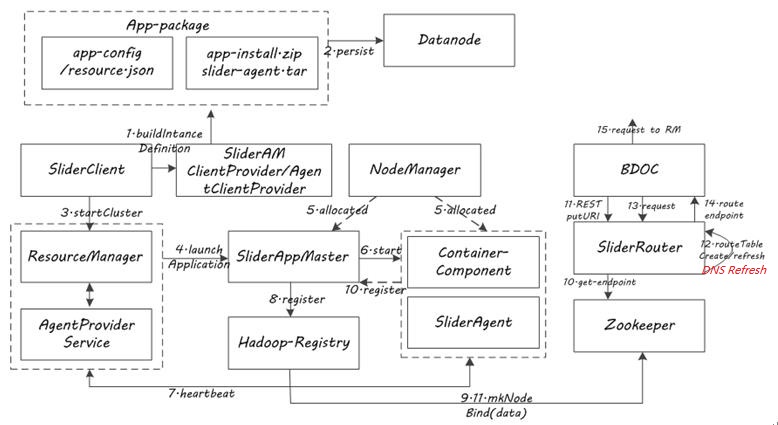
2）RouteTable实现机制，将信息存储在ZK中，虽然DNS解决了解析的问题，但是没有解析路由转发的问题。因此RouteTable的存在也是有必要的，为Nginx提供数据。这个访问并不频繁，存储在zk中，方便查询和持久化，而且不需要在hdfs中放数据（解决一致性）。RouteTable的ZK化，可以使SliderRouterNode无状态Server。Nginx 使用Lua模块，增加灵活性

3）增加Slider Web UI和Manager Web UI，直接SliderClient或者REST API。需要考虑扩展性，将以后的SliderResource的实习实现增加扩展性，方便快捷的增加其他App的REST 开发。（这个是否需要与Hcontrol结合考虑）

4）Slider Docker和Metrics的支持工作

# Design

Slider Router在Slider中作用如下图所示：



Slider创建及使用Service的流程如下：

1）Slider Client执行Service Create命令 //或者Slider Rest api请求（例如BDOC）

2）通过SliderAMClientProvider及AgentClientProvider将安装包及配置文件写入HDFS

3）启动Service

4）创建SliderAppMaster，NM分配Container

5）SliderAppMaster启动component Container，初始化。启动SliderAgent

6）Slider Agent与SliderAppMaster以心跳信息交互

7）SliderAppMaster将自己及Component注册到Hadoop Registry中

注：以上是Service的启动过程，下面是Slider Router的处理过程及用户访问流程，但是下面的仅是大概的执行，需要细化

8）BDOC发起服务的路由注册，将创建服务的URL写入Slider Router中

9）Slider Router根据URL及Endpoint，在反向代理服务器中写入相应的配置

10）用户访问URL，通过Slider Router访问已创建的服务

注：需要调研下Mesos-DNS的处理流程，分析DNS的使用过程，包括app.mesos.domain配置及写入过程

# Architecture

SliderRouter的总体框架图如下所示：



为了实现slider 创建应用的托管，需要实现以上几个部分，下面对每个部分进行详细介绍。

1）Slider Router 配置

在slider-server.xml中增加slider -router配置项：

|  |  |  |
| --- | --- | --- |
| 配置 | 描述 | 默认值 |
| slider.router.address | IPC端口，SliderRouter IPC Port | 60888 |
| slider.router.proxy.address | Slider代理端口 | 60889 |
| slider.router.webapp.address | REST端口 | 60875 |
| slider.router.webapp.https.address | REST的HTTPS端口 | 60876 |
| *slider.routetable.base.path* | *routeTable的hdfs路径(修改为zk)* | *.slider/routetable* |
| slider.nginx.conf\_file | Nginx Server的默认路径 | /usr/local/openresty  /nginx/conf/nginx.conf |
| slider.router.bind-host | 双网卡配置项 | {HOST\_NAME} |

注：DNS的配置暂时还不确定

2）路由表及Nginx配置文件

RouteTable在ZK的目录结构如下：

|—{ROOT\_PATH}

|—{USER-1}

|—{service\_type} //当前仅有services/org-apache-slider

|— {APP-1}

|—{KEY-VALUE}

|— {APP-2}

|—{USER-2}

RouteTable用于缓存已创建服务的URL和Endpoint的映射关系。

eg:创建应用demo-tomcat，以http://myapp.bdoc.com访问该Tomcat container，映射关系为：<key,value>

key： *myapp.bdoc.com@domo-tomcat //需要变动*

value：

Endpoint {

"api" : "server-xml.http.port",

"addressType" : "host/port",

"protocolType" : "tcp",

"addresses" : [ {

"port" : "35504",

"host" : "bdi219" }

]

}

Nginx配置文件用于根据RouteTable保存的映射关系，生成反向代理，在Slider Router中将反向代理直接写入配置文件，*整个流程需要变动*

eg:

server {

listen 80;

server\_name \*.bdoc.com;

if ( $host ~\* (\b(?!www\b)\w+)\.\w+\.\w+){

set $subdomain $1;

}

location /myapp {

proxy\_pass http://bdi219:60531/myapp;

}

}

3) Registry Info Manager

注册信息管理管理模块分为两个部分：服务注册信息查询模块和服务注册信息异常处理模块。

服务注册信息查询模块：用于从服务注册路径下查询服务对外提供的Endpoint，查询参数为Service-Name

服务注册信息异常处理模块：启动Service的Container可能会因为异常导致停止，ZK中的信息会更新，该模块监听zk信息改变事件。根据事件的类型进行RouteTable及代理信息的更新（*增加DNS server中的信息更新）*

4）代理信息管理

*服务创建后，通过Client或REST将该服务注册到Router中，信息存储到RouteTable，由于Slider Router可能会停止运行，因此需要进行信息的持久化。*

*Proxy服务管理，反向代理使用nginx，服务创建时需要将对外的Endpoint及使用的访问URL注册到nginx中，目前考虑方案是在nginx conf文件中添加一条location记录。*

服务创建后，生成服务信息 //自动或者用户触发，信息存储到RouteTable中

Proxy服务管理，反向代理使用nginx，服务创建时需要将对外的Endpoint及使用的访问URL注册到nginx中，方案应该使用Nginx-lua模块

5）DNS 管理模块

Record Manager ：记录生成器，负责为所有运行着的应用程序生成DNS A和SRV记录，周期性的查询master（是否由zk信息监听完成触发），记录生成器根据触发知道任务的状态，比如启动、完成、失败或者重启

DNS解析器，负责处理DNS请求并回复请求，解析器直接回复YARN上运行着的任务，围外部请求选择DNS服务器

6）APIs

API分为两类，REST API和Java API。JAVA API为SliderRouterClient，实现接口SliderRouterClientApi，当前实现方法register和unregister；REST API，使用SliderRouterClient提供的方法，向外提供服务。

为了及时访问Router信息，提供Web UI，方便管理者查看信息。

# Change histroy