单点服务的HA方案调研文档

Hadoop集群里面涉及到了多个单点服务，分成几类：

SliderRouter

Ranger Admin

HS

Spark-HS

Spark-ThriftServer

HBase-ThriftServer

Ldap

NameNode 50070 webhdfs

这类服务为无状态的服务，启动在任何节点上都可以正常运行

以下有几种方式保证服务的可用性

# 1.DNS负载均衡

实现原理是在DNS服务器中为同一个主机名配置多个IP地址，在应答DNS查询时，DNS服务器对每个查询将以DNS文件中的主机记录IP地址顺序返回不同的解析结果，将客户端的访问引导到不同的机器上去，使得不同的客户端访问不同的服务器，从而达到负载均衡的目的。访问架构图如下：



客户端发送请求时，使用主机名进行发送请求，DNS根据主机名随机选择一个IP地址发给客户端，从而实现负载均衡。

DNS系统本身是一个分布式网络，相对可靠。在zone中配置如下：

$TTL 1D

@ IN SOA @ rname.invalid. (

0 ; serial

1D ; refresh

1H ; retry

1W ; expire

3H ) ; minimum

NS dns.fys.com.

dns A 10.133.47.5

bbs CNAME www.fys.com.

www IN A 10.133.47.16

www IN A 10.133.47.5

www IN A 10.133.47.54

重启后dns后，

*[root@fys1 named]# ping www.fys.com*

*PING www.fys.com (10.133.47.5) 56(84) bytes of data.*

*64 bytes from fys1.cmss.com (10.133.47.5): icmp\_seq=1 ttl=64 time=0.017 ms*

*[root@fys1 named]# ping www.fys.com*

*PING www.fys.com (10.133.47.54) 56(84) bytes of data.*

*64 bytes from fys2.cmss.com (10.133.47.54): icmp\_seq=1 ttl=64 time=0.760 ms*

*[root@fys1 named]# ping www.fys.com*

*PING www.fys.com (10.133.47.16) 56(84) bytes of data.*

*64 bytes from fys3.cmss.com (10.133.47.16): icmp\_seq=1 ttl=64 time=0.849 ms*

DNS采用轮询负载算法返回不同的IP地址，不能区分服务器的差异，不能反映服务器的当前运行涨停，不能做到为性能较好的服务器多分配请求，甚至会出现客户请求集中在某台服务器上的情况。但是使用该方法实现HA有问题，缺点：

1）不支持高可靠性，发送请求至随机的服务器上，当某台服务器宕机时，会造成请求失败的情况。

2）额外的网络问题，dns内部数据交互

3）负载均衡策略单一，仅有轮询负载算法

4）需要考虑dns服务器本身的高可用性

参考文献：

http://www.cnblogs.com/cuihongyu3503319/archive/2012/07/09/2583129.html

# 2.Nginx负载均衡

Nginx是一款轻量级的Web服务器/反向代理服务器，作为一个反向代理服务器时Nginx可以安装调度规则实现动态、静态页面的分离，可以安装轮询、ip哈希、权重等多种方式对后端服务器做负载均衡，同时还支持后端服务器的健康检查。访问架构图如下：



客户端访问Nginx服务器，由Nginx反向代理模块转发请求，反向代理中配置接收请求的服务器，一般多个。Nginx根据配置的转发策略，随机选择某个提供服务的服务器转发请求。Nginx的负载均衡测试如下：

在nginx.conf文件中添加配置：

*keepalive\_timeout 65;*

*upstream myproject {*

*server 10.133.47.5:8080 ; //测试端口为hcontrol页面*

*server 10.133.47.54:8080 ; //测试端口为启动的tomcat主页面*

*}*

修改path:

*location / {*

*#root html;*

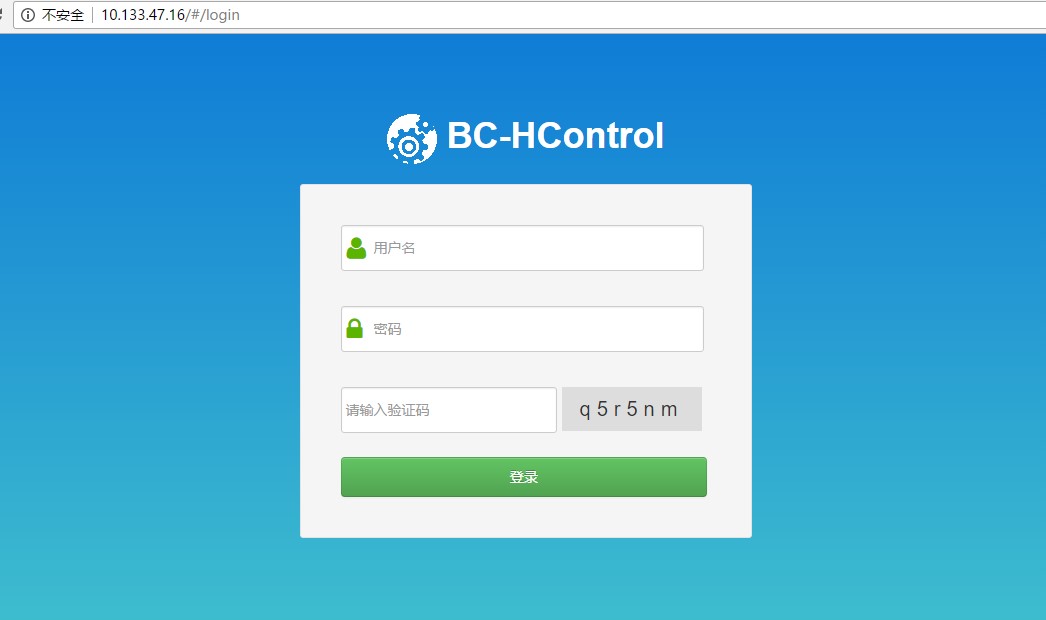
*#index index.html index.htm;*

*proxy\_pass http://myproject;*

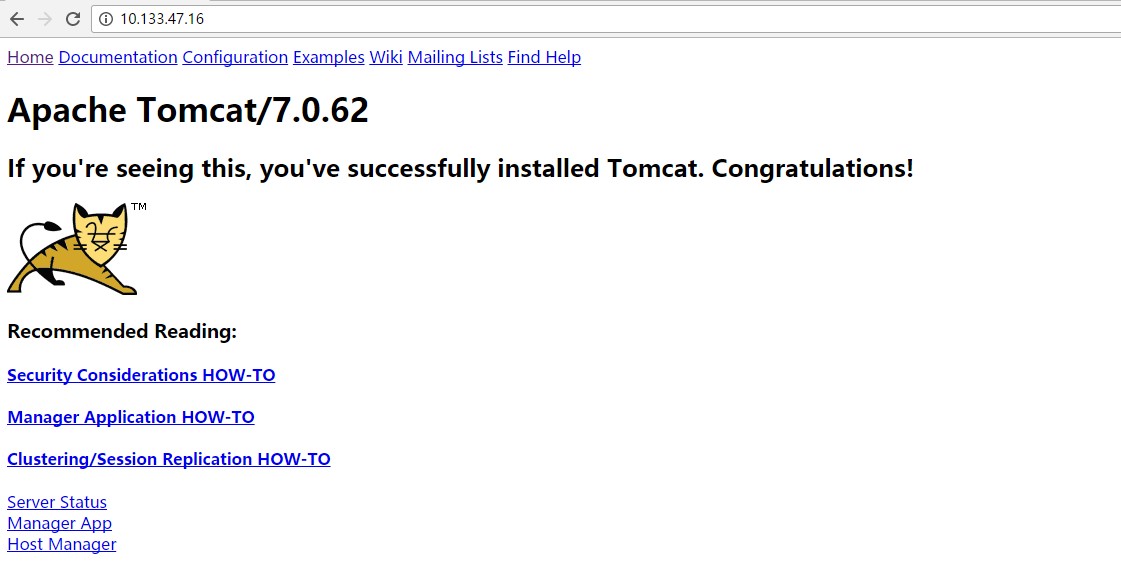
*}*

重新启动Nginx后，打开<http://10.133.47.16:80/>后：

第一次页面如下：



第二次打开如下：



Nginx会将浏览器的请求，根据myproject定义的服务器列表转发到不同的Server中。当某个Server的服务宕机后，不会将该转发发送过去。Nginx的负载均衡和DNS的负载均衡对比，最大的好处是在转发请求时会将宕机的服务从机器列表中剔除出列表。但是缺点如下：

1. 一定程度上可以nginx来解决服务的单点问题，但是引入了Nginx单点问题。解决Nginx的单点问题，一般也是使用keepalive（虚拟IP）的方式。
2. 无法选择active及standby

2）发送的请求，需要通过Nginx来完成转发，带来一定开销

参考链接：

<http://nginx.org/cn/>

# 3.Keepalived 实现HA

Keepalived是一个类似于layer 3,4&7交换机制的软件，其作用是检测服务器的状态，如果一台web服务器死机，或工作出现故障，Keepalived将检测到，并将有故障的服务器从系统中剔除，同时使用其他服务器替代该服务器的工作；当服务器工作正常后Keepalved自动将服务器加入到服务器集群中，这些工作自动完成，不需要人工干涉。



客户端使用虚拟IP发送请求，虚拟IP属于Keepalive某个节点。正常情况下客户端访问主节点对应的WebServer提供服务。当主节点不可用时，备节点接管虚拟IP，并提供正常服务。这种方式是最常用的保证服务的高可用性方案。

# 4.服务注册与发现

实现服务的高可用性可以利用服务发现技术，例如ZK、Eureka、Consul、Etcd等：



服务启动后，向服务注册中心注册服务的API，服务终止后会从该注册中心删除该服务注册信息。客户端访问服务时，首先向服务注册中心获取服务实例的网络地址，客户端决定可以的服务实例。客户端使用获取的服务实例信息，向服务发起请求。使用这种方式，实现服务的HA，缺点是需要客户端做服务访问的决策。

对这种模式的改进，如下所示：



客户端向负载均衡器发送请求，这个负载均衡器会查询服务注册表，并将请求路由到可用的服务实例上。类似Nginx的负载均衡器都可以用作服务器端发现的负载均衡器。最大的好处是服务发现的细节被从客户端中抽象出来。

# 5.DNS实现HA的改进方案

方案1和方案2本质上是负载均衡的作用，DNS负载均衡不保证可用性，Nginx负载均衡保证可用性，但是要经过一次反向代理的转发，有一定的代价。

方案3，Keepalived是比较经典的HA方案，但是每个服务都需要申请一个虚拟机IP地址。

方案4：为了实现Server的HA，要修改客户端和服务端的代码，代价太大

结合前几种方案，折中的方案，如下：



用户通过hostname来访问服务，hostname经过dns转化为可用服务的IP地址，HA watchDog实时监控Server的服务状态，将可用的服务器的IP写入到DNS zone文件中。

# 6.SliderRouter的HA实现方案

SliderRouter启动两个或者多个SliderRouter实例，然后使用上述中方案实现HA，上述四种方式：

方式1：DNS负载均衡，客户端访问时仅需要指定hostname，但是不能保证请求转发的到的SliderRouter可用。

方式2：可保证可用，但是需要实现Nginx的高可用性，其中常用的方式Keepalive（虚拟机IP）

方式3：使用这种方式，比较简单，也是常规做法

方式4：这种方式中第一种方法，需要客户端改变代码，BDOC端需要改代码，从ZK中获取可用的SliderRouter的服务实例地址。第二种方法是方式2的复杂版。

因此，实现SliderRouter HA的建议做法是方式3。方式4中的第一种方法可以考虑。

SliderRouter HA的方案1：



方案1：

需要申请虚拟IP并进行配置，部署是完成。由于Hadoop中单点的Server有多个，因此建议Hcontrol实现Keepalive的安装及虚拟IP的管理

SliderRouter HA方案2：



方案2：

1. SliderRouter中增加服务注册逻辑
2. BDOC中增加SliderRouter的服务发现逻辑