# 实战Dns

## 理论基础

### dns的出现及演化

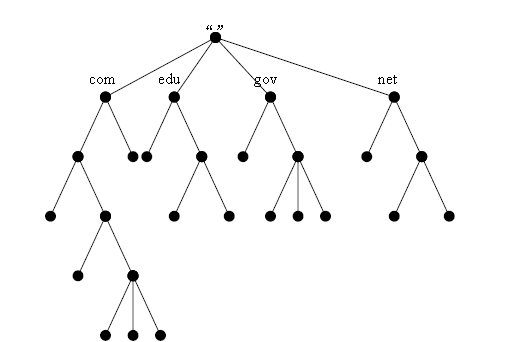
网络出现的早期是使用IP地址通讯的，那时就几台主机通讯。但是随着接入网络主机的增多，这种数字标识的地址非常不便于记忆，UNIX上就出现了建立一个叫做hosts的文件（Linux和Windows也继承保留了这个文件）。这个文件中记录着主机名称和IP地址的对应表。这样只要输入主机名称，系统就会去加载hosts文件并查找对应关系，找到对应的IP，就可以访问这个IP的主机了。

    但是后来主机太多了，无法保证所有人都能拿到统一的最新的hosts文件，就出现了在文件服务器上集中存放hosts文件，以供下载使用。互联网规模进一步扩大，这种方式也不堪重负，而且把所有地址解析记录形成的文件都同步到所有的客户机似乎也不是一个好办法。这时DNS系统出现了，随着解析规模的继续扩大，DNS系统也在不断的演化，直到现今的多层架构体系。

### 2、dns概述入门

DNS（Domain Name System，域名系统），因特网上作为域名和[IP地址](http://baike.baidu.com/view/3930.htm)相互映射的一个[分布式数据库](http://baike.baidu.com/view/68389.htm)，能够使用户更方便的访问[互联网](http://baike.baidu.com/view/6825.htm)，而不用去记住能够被机器直接读取的IP数串。通过[主机](http://baike.baidu.com/view/23880.htm" \t "_blank)名，最终得到该主机名对应的IP地址的过程叫做域名解析（或主机名解析）。DNS协议运行在[UDP](http://baike.baidu.com/view/30509.htm" \t "_blank)协议之上，使用端口号53。

DNS 的分布式数据库是以域名为索引的，每个域名实际上就是一棵很大的逆向树中路径，这棵逆向树称为域名空间（domain name space）。如图所示树的最大深度不得超过127 层，树中每个节点都有一个可以长达63 个字符的文本标号。



### 3、dns域名解析过程

首先，客户端先在本地缓存查找有没有域名缓存，如果没有，客户端发出DNS请求翻译IP地址或主机名。DNS服务器在收到客户机的请求后：

（1）检查DNS服务器的缓存，若查到请求的地址或名字，即向客户机发出应答信息；

（2）若没有查到，则在数据库中查找，若查到请求的地址或名字，即向客户机发出应答信息；

（3）若没有查到，则将请求发给根域DNS服务器，并依序从根域查找顶级域，由顶级查找二级域，二级域查找三级，直至找到要解析的地址或名字，即向客户机所在网络的DNS服务器发出应答信息，DNS服务器收到应答后现在缓存中存储，然后，将解析结果发给客户机。

（4）若没有找到，则返回错误信息。

### 4、dns的分类

**主DNS服务器：**就是一台存储着原始资料的DNS服务器。

**从DNS服务器：**使用自动更新方式从主DNS服务器同步数据的DNS服务器。也成辅助DNS服务器。

**缓存服务器：**不负责本地解析，采用递归方式转发客户机查询请求，并返回结果给客户机的DNS服务器。同时缓存查询回来的结果，也叫递归服务器。

**转发器：**这台DNS发现非本机负责的查询请求时，不再向根域发起请求，而是直接转发给指定的一台或者多台服务器。自身并不缓存查询结果。

### 5、dns中记录类型

|  |  |
| --- | --- |
| **SOA** | **可以理解为一段为自己dns做备注说明的文本,一般与ns一致**  **比如：dns.laonanhai.com sa.laonanhai.com**  **示例：**  a.shifen.com.        579    IN    SOA    dns.baidu.com. sa.baidu.com.  (                            1408010001 ; serial number                            5 ; refresh 5s                            5 ; retry 5s                            86400 ; expire 1d                            3600 ;min TTL 1h      ) |
| **NS** | **域的授权名称服务器**  NSDName：DNS的FQDN  baidu.com.        64899    IN    NS    ns2.baidu.com.      baidu.com.        64899    IN    NS    ns4.baidu.com.      baidu.com.        64899    IN    NS    dns.baidu.com.      baidu.com.        64899    IN    NS    ns7.baidu.com.      baidu.com.        64899    IN    NS    ns3.baidu.com. |
| **MX** | **域的邮件交换器，要跟着一个优先级值，越小越高**  baidu.com. 7200 IN MX 20 jpmx.baidu.com.  baidu.com. 7200 IN MX 20 mx50.baidu.com.  baidu.com. 7200 IN MX 10 mx.n.shifen.com.  baidu.com. 7200 IN MX 20 mx1.baidu.com. |
| **A** | **IPV4主机地址** |
| **AAAA** | **IPV6主机地址** |
| **PTR** | **解析IP的指针，反向记录** |
| **CNAME** | **权威（正式）名称，定义别名记录**  www.baidu.com.        1154    IN    CNAME    www.a.shifen.com. |

### 6、dns命名规范

1. 26个英文字母

2. “0，1，2，3，4，5，6，7，8，9”十个数字

3. “-”（英文中的连词号）

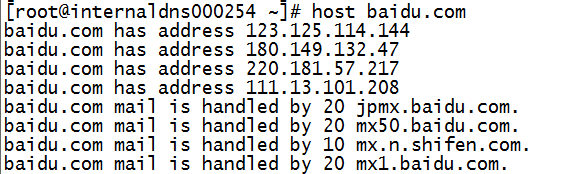
4. 最多63字节长度

要不按照这个，bind支不支持，支持，合适么，不合适。

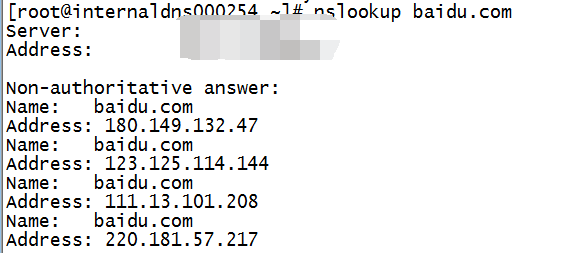
非要不按照这个，怎么办？master-view文件上配置check-names ignore;

### 7、dig、nslookup、host的介绍

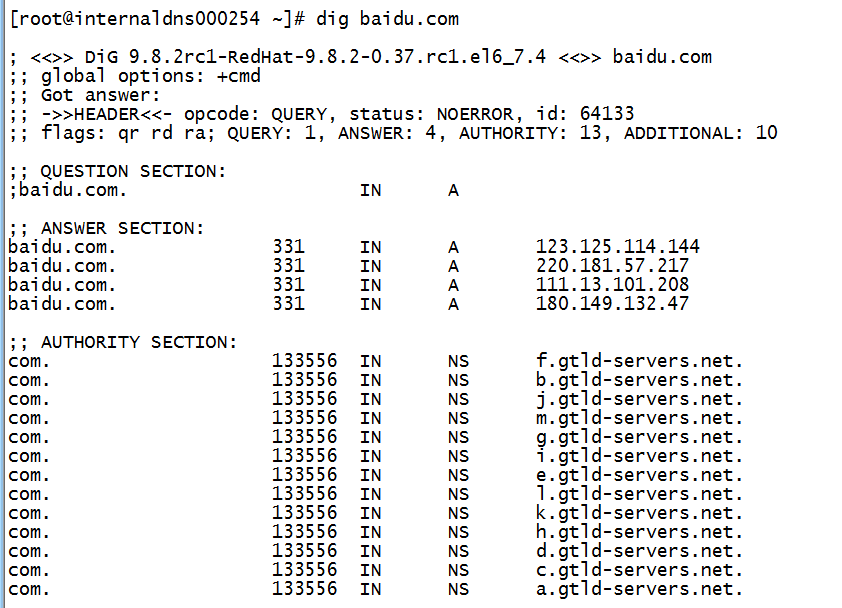
Host命令也是大多数系统、软件库调用的解析命令，比如php中。



Nslookup能够反映出是哪个dnsserver返回的结果



Dig返回整个解析过程的详细路径，类似traceroute



为什么只有13台dns根服务器

最后，让我们了解下全球DNS根服务器为什么只有13台。

DNS协议的最初定义要从20世纪80年代未期开始算起，它使用了端口上的UDP和TCP协议。

UDP通常用于查询和响应，TCP用于主服务器和从服务器之间的区传送.遗憾的是，在所有UDP实现中能保证正常工作的最大包长是512字节，对于在每个包中必须含有数字签名的一些DNS新特性（例如，DNSSEC）来说实在是太小了。

512字节的限制还影响了根服务器的数量和名字。

要让所有的根服务器数据能包含在一个512字节的UDP包中，根服务器只能限制在13个，而每个服务器要使用字母表中的单个字母命名。

以太网数据的长度必须在46-1500字节之间，这是由以太网的物理特性决定的。

事实上，这个1500字节就是网络层IP数据包的长度限制，理论上，IP数据包最大长度是65535字节。

这是由IP首部16比特总长度所限制的，去除20字节IP首部和8个字节UDP首部，UDP数据包中数据最大长度为65507字节。

在Internet数据传输中，UDP数据长度控制在576字节（Internet标准MTU值），而在许多UDP应用程序设计中数据包被限制成512字节或更小。这样可以防止数据包的丢失。

许多解析器首先发送一条UDP查询，如果它们接收到一条被截断的响应，则会用TCP重新发送该查询。

这个过程绕过了512字节的限制，但是效率不高。您或许认为DNS应该避开UDP，总是使用TCP，但是TCP连接的开销大得多。

一次UDP名字服务器交换可以短到两个包：一个查询包、一个响应包。一次TCP交换则至少包含7个包：三次握手初始化TCP会话、一个查询包、一个响应包以及最后一次握手来关闭连接。

## 学习部署bind9

Bind是一款开放源码的DNS服务器软件，Bind由美国加州大学Berkeley分校开发和维护的，全名为Berkeley Internet Name Domain它是目前世界上使用最为广泛的DNS

### 环境准备

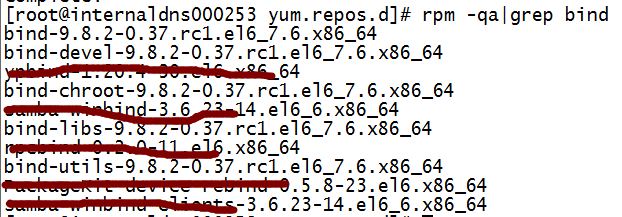
配置yum源

使用centos6自带的yum源即可，同时确保机器可以解析外网域名，并能连接公网

### 部署单台dns

**Yum安装bind**

yum install bind-utils bind bind-devel bind-chroot



**编辑/etc/named.conf**

options {

version "1.1.1";

listen-on port 53 {any;};

directory "/var/named/chroot/etc/";

pid-file "/var/named/chroot/var/run/named/named.pid";

allow-query { any; };

Dump-file "/var/named/chroot/var/log/binddump.db";

Statistics-file "/var/named/chroot/var/log/named\_stats";

zone-statistics yes;

memstatistics-file "log/mem\_stats";

empty-zones-enable no;

forwarders {202.106.196.115;8.8.8.8; };

};

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

controls {

inet 127.0.0.1 port 953

allow { 127.0.0.1; } keys { "rndc-key"; };

};

logging {

channel warning {

file "/var/named/chroot/var/log/dns\_warning" versions 10 size 10m;

severity warning;

print-category yes;

print-severity yes;

print-time yes;

};

channel general\_dns {

file "/var/named/chroot/var/log/dns\_log" versions 10 size 100m;

severity info;

print-category yes;

print-severity yes;

print-time yes;

};

category default {

warning;

};

category queries {

general\_dns;

};

};

include "/var/named/chroot/etc/view.conf";

**编辑/etc/rndc.key**

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

**编辑/etc/rndc.conf**

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

options {

default-key "rndc-key";

default-server 127.0.0.1;

default-port 953;

};

**编辑/var/named/chroot/etc/view.conf**

view "View" {

zone "lnh.com" {

type master;

file "lnh.com.zone";

allow-transfer {

10.255.253.211;

};

notify yes;

also-notify {

10.255.253.211;

};

};

};

**编辑/var/named/chroot/etc/lnh.com.zone**

$ORIGIN .

$TTL 3600 ; 1 hour

lnh.com IN SOA op.lnh.com. dns.lnh.com. (

2000 ; serial

900 ; refresh (15 minutes)

600 ; retry (10 minutes)

86400 ; expire (1 day)

3600 ; minimum (1 hour)

)

NS op.lnh.com.

$ORIGIN lnh.com.

shanks A 1.2.3.4

op A 1.2.3.4

1 **Serial**：只是一个序号，但这个序号可被用来作为 slave 与 master 更新的依据。 举例来说， master 序号为 100 但 slave 序号为 90 时，那么这个 zone file 的资料就会被传送到 slave 来更新了。由于这个序号代表新旧资料，通常我们建议你可以利用日期来设定！举例来说，上面的资料是在 2006/10/20 所写的第一次，所以用 2006102001 作为序号代表！(yyyymmddnn，nn代表这一天是第几次修改)

2 **Refresh**：除了根据 Serial 来判断新旧之外，我们可以利用这个 refresh(更新) 命令 slave 多久进行一次主动更新；

3 **Retry**：如果到了 Refresh 的时间，但是 slave 却无法连接到 master 时， 那么在多久之后，slave 会再次的主动尝试与主机连线；

4 **Expire**：如果 slave 一直无法与 master 连接上，那么经过多久的时间之后， 则命令 slave 不要再连接 master 了！ 也就是说，此时我们假设 master DNS 可能遇到重大问题而无法上线，则等待系统管理员处理完毕后， 再重新来到 slave DNS 重新启动 bind 吧！

5 **Minimun**：这个就有点象是 TTL ！

**修改目录权限，并启动服务**

cd /var && chown -R named.named named/

/etc/init.d/named start

chkconfig named on

**我们试一把能不能解析**

dig @127.0.0.1 shanks.lnh.com

### 部署主从dns

**Yum安装bind**

yum install bind-utils bind bind-devel bind-chroot

**编辑/etc/named.conf**

options {

version "1.1.1";

listen-on port 53 {any;};

directory "/var/named/chroot/etc/";

pid-file "/var/named/chroot/var/run/named/named.pid";

allow-query { any; };

Dump-file "/var/named/chroot/var/log/binddump.db";

Statistics-file "/var/named/chroot/var/log/named\_stats";

zone-statistics yes;

memstatistics-file "log/mem\_stats";

empty-zones-enable no;

forwarders {202.106.196.115;8.8.8.8; };

};

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

controls {

inet 127.0.0.1 port 953

allow { 127.0.0.1; } keys { "rndc-key"; };

};

logging {

channel warning {

file "/var/named/chroot/var/log/dns\_warning" versions 10 size 10m;

severity warning;

print-category yes;

print-severity yes;

print-time yes;

};

channel general\_dns {

file "/var/named/chroot/var/log/dns\_log" versions 10 size 100m;

severity info;

print-category yes;

print-severity yes;

print-time yes;

};

category default {

warning;

};

category queries {

general\_dns;

};

};

include "/var/named/chroot/etc/view.conf";

**编辑/etc/rndc.key**

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

**编辑/etc/rndc.conf**

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

options {

default-key "rndc-key";

default-server 127.0.0.1;

default-port 953;

};

**编辑/var/named/chroot/etc/view.conf**

view "SlaveView" {

zone "lnh.com" {

type slave;

masters {10.6.0.253; };

file "slave.lnh.com.zone";

};

};

**修改master上view.conf配置，将slave节点ip加入，**

**之后修改lnh.com.zone将serial+1**

**之后在slave上修改目录权限，并启动**

cd /var && chown -R named.named named/

/etc/init.d/named start

chkconfig named on

回到master节点，执行rndc reload

### 添加A、CNAME、MX、PTR记录

#### A记录

编辑master节点/var/named/chroot/etc/lnh.com.zone，在文件末尾添加一条记录

a A 192.168.122.100

将serial值+1

执行rndc reload命令

分别对master、slave解析下：host a.lnh.com 127.0.0.1 结果应该是192.168.122.100

#### CNAME记录

编辑master节点/var/named/chroot/etc/lnh.com.zone，在文件末尾添加一条记录

cname CNAME a.lnh.com.

将serial值+1

执行rndc reload命令

分别对master、slave解析下：host cname.lnh.com 127.0.0.1结果应该是192.168.122.101

#### MX记录

编辑master节点/var/named/chroot/etc/lnh.com.zone，在文件末尾添加一条记录

mx MX 5 192.168.122.101

mx MX 10 192.168.123.101

将serial值+1

执行rndc reload命令

分别对master、slave解析下：host mx.lnh.com 127.0.0.1 结果应该是192.168.122.101

#### PTR记录

编辑master节点/var/named/chroot/etc/view.conf，加入ptr的zone配置

zone "168.192.in-addr.arpa" {

type master;

file "168.192.zone";

allow-transfer {

10.6.0.254;

};

notify yes;

also-notify {

10.6.0.254;

};

};

编辑master节点/var/named/chroot/etc/168.192.zone

$TTL 3600 ; 1 hour

@ IN SOA op.lnh.com. dns.lnh.com. (

2004 ; serial

900 ; refresh (15 minutes)

600 ; retry (10 minutes)

86400 ; expire (1 day)

3600 ; minimum (1 hour)

)

NS op.lnh.com.

102.122 IN PTR a.lnh.com.

修改168.192.zone文件的属性

chown named.named 168.192.zone

rndc reload

编辑slave节点/var/named/chroot/etc/view.conf，加入ptr的zone配置

zone "168.192.in-addr.arpa" {

type slave;

masters {10.5.35.14; };

file "slave.168.192.zone";

};

使slave配置生效，slave上执行

rndc reload

分别对两台机器做ptr记录的解析

host 192.168.122.102 127.0.0.1 结果应为a.lnh.com

### 通过dns实现服务的负载均衡

编辑master节点/var/named/chroot/etc/lnh.com.zone，在文件末尾添加一条记录

a A 192.168.122.102

将serial值+1

执行rndc reload命令

分别对master、slave解析下：host a.lnh.com 127.0.0.1 结果应该是192.168.122.100和192.168.122.102

### 配置dns视图（智能dns）

编辑master节点vim /var/named/chroot/etc/named.conf，在include上面添加

acl group1 {

10.5.35.14;

};

acl group2 {

10.5.35.15;

};

编辑master节点vim /var/named/chroot/etc/view.conf为

view "GROUP1" {

match-clients { group1; };

zone "viewlnh.com" {

type master;

file "group1.viewlnh.com.zone";

};

};

view "GROUP2" {

match-clients { group2; };

zone "viewlnh.com" {

type master;

file "group2.viewlnh.com.zone";

};

};

编辑master节点vim /var/named/chroot/etc/group1.viewlnh.com.zone

$ORIGIN .

$TTL 3600 ; 1 hour

viewlnh.com IN SOA op.viewlnh.com. dns.viewlnh.com. (

2005 ; serial

900 ; refresh (15 minutes)

600 ; retry (10 minutes)

86400 ; expire (1 day)

3600 ; minimum (1 hour)

)

NS op.viewlnh.com.

$ORIGIN viewlnh.com.

op A 192.168.122.1

view A 192.168.122.1

编辑master节点vim /var/named/chroot/etc/group2.viewlnh.com.zone

$ORIGIN .

$TTL 3600 ; 1 hour

viewlnh.com IN SOA op.viewlnh.com. dns.viewlnh.com. (

2005 ; serial

900 ; refresh (15 minutes)

600 ; retry (10 minutes)

86400 ; expire (1 day)

3600 ; minimum (1 hour)

)

NS op.viewlnh.com.

$ORIGIN viewlnh.com.

op A 192.168.122.2

view A 192.168.122.2

修改文件所属，加载配置

chown named.named /var/named/chroot/etc/group\*.zone

rndc reload

分别在主机A与B上做view.viewlnh.com的域名解析测试

结果应为主机A上是192.168.122.1主机B上是192.168.122.2

## 三、构建企业级dns

### 1、硬件选型

**Cpu**：12c以上配置

**内存**：16G

**网络**：千兆

### 初始化系统

**关闭selinux**: /etc/selinux/config

**关闭iptables**: chkconfig iptables off

**调整ulimit限制**: echo -e '\* soft nproc 65536\n\* hard nproc 65536\n\* soft nofile 65536\n\* hard nofile 65536\n' >> /etc/security/limits.conf

### 3、构建高性能、高可用dns

采用lvs-dr模式负载均衡，多idc，多套dns集群，通过master-slave技术保证dns配置的一致性。

1. 高可用

物理层：

首先确保两台lvs不在同一机柜、同一物理交换机接入；

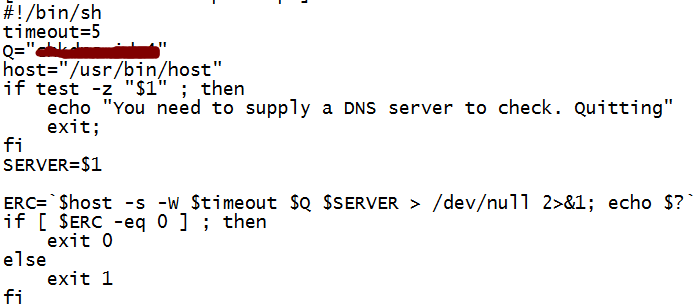
其次确保将所有dns服务器也做到不在同一机柜、同一物理交换机接入。

同时，在不同的idc构建多套dns集群，为客户端提供可切换的配置。

服务层：

坚决摒弃lvs上端口检测这种方式，采用自定义脚本检测，为dns的健康检测单独设置一个域名，就为了lvs检测dns是否存活而设计。

脚本示例：



客户端层：

多idc之间的流量切换是通过客户端的健康检测cron实现的，脚本每分钟运行一次，分别检测每个dns集群虚地址的可用性。

1. 高性能

通过lvs可以对每个集群做横向扩容，是否需要扩容的依据是对现有系统的压测结果，以及实时的监控数据。

亦或者可以在最靠近应用层处，加上一层cache-only集群，但前提是你的线上环境中，没有任何系统依赖于dns负载均衡。

### 4、压测

安装queryperf

**下载bind源码**：wget <http://ftp.isc.org/isc/bind9/9.7.3/bind-9.7.3.tar.gz>

**解压bind源码**：tar zxf bind-9.7.3.tar.gz

**进入解压后bind源码目录**：cd bind-9.7.3/contrib/queryperf/

**编译**：./configure

make

会在当前目录下出现queryperf，可以将它拷贝至/usr/bin/下

编辑域名记录文件(test.txt)，格式如下：

www.baidu.com A

www.baidu.com A

www.baidu.com A

www.baidu.com A

www.baidu.com A

**压测命令**：queryperf -d test.txt -s 8.8.8.8

按照3.1中的硬件选型，也做了3.2的初始化，这样在相同idc的压测下，单台dns-server的qps可以达到35~40k

### 5、监控

监控结合zabbix实现

#### 系统基础性能

使用zabbix自带模板即可。Cpu、内存、主机存活、磁盘空间、主机运行时间、系统load

#### Loopback地址绑定状态监控

该架构中，dnsserver在集群中充当realserver的角色，在dr中，需要绑定loopback地址方能通信，因此当loopback地址没有绑定上时，lvs健康检测通过，但是当请求到达dnsserver时，请求被拒绝，dns集群会出现异常。

#### Dns数据与master一致性监控

此处我的方案是分为两部分：

一是通过写zabbix自定义discovery，扫出dns配置中所有zone，然后分别对比slave和master每个zone的serial值，当slave与master的值持续5分钟不一致时报警

二是写脚本，每15分钟扫一遍master上所有域名解析结果，与每个slave的结果做对比，当出现结果不一致情况时，报警

#### Dns响应时间监控

远端一组主机跑在fullnat下（提供高可用），通过dig命令检测dnsserver的响应时间。

#### Dns每秒请求数监控

在每台dns主机上，编写zabbix脚本，分析named\_stats文件，获取每秒请求数

#!/bin/bash

#rndc stats

STATS='/var/named/chroot/var/log/named\_stats'

if [[ $# -ne 1 ]]

then

echo "$0 [querys]"

exit 2

else

which=$1

fi

if [[ -f "${STATS}" ]]

then

echo > ${STATS}

rndc stats >/dev/null 2>&1

else

echo "${STATS} not found."

exit 2

fi

case ${which} in

querys)

RESULT=`awk '{if ($2=="QUERY") {print $1}}' ${STATS}`

;;

\*)

echo "$0 [querys]"

exit 2

;;

esac

echo ${RESULT}

##### 

#### Dns可用性监控

远端一组主机跑在fullnat下（提供高可用），通过host命令检测dnsserver的可用性，脚本与lvs健康检测脚本类似。

### 6、自动化

#### saltstack安装、部署

通过定制saltstack配置，实现自动、批量安装、部署dns

#### 配置管理自动化

业界最多的是bind-dlz，dlz是指将所有的配置都存在mysql表中，对bind做特殊配置，使得每次bind接受的请求都去mysql中查询数据之后返回给用户。

##### bind-dlz的优缺点

优势：将数据全部存在数据库，符合运维开发的理念。

劣势：每次解析都要select数据库，性能低下；

增加了系统的藕合性，还需要保证mysql的高可用。

##### 我的方案

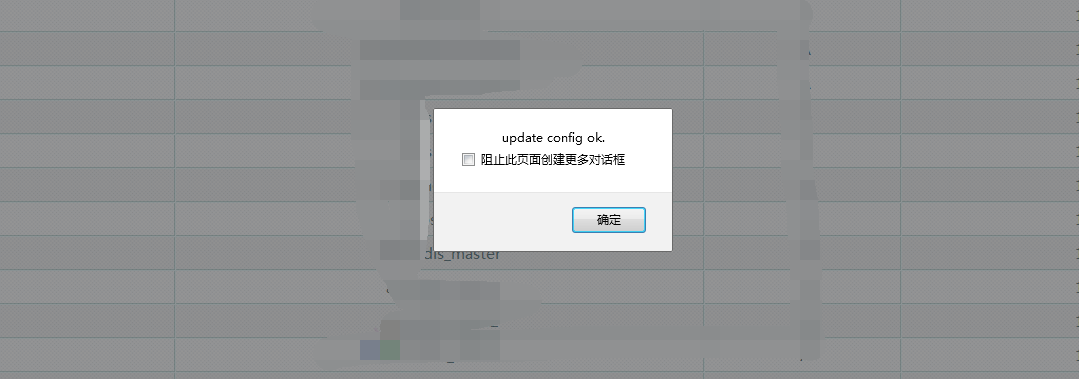
采用dlz的数据库部分表结构；

用thinkphp实现对mysql的增删改查，和一些权限控制的页面，在该页面，用户可以完成对域名的增删改查操作，数据源在MySQL中；

通过saltstack+py实现从mysql中调数据，生成bind的配置文件，并检测文件格式，之后reload；







为什么做这些，我要将dns做成可交付、已维护的系统，交付给应用运维同学使用，我只负责dns架构的server端；

<https://github.com/shanks1127/dns>

### 7、安全

时刻关注dns相关的漏洞、补丁；

选用稍大些的厂商作为域名服务商，我们是万网；

对服务器的登录日志做监控分析；

### 8、日常运维规范

Dns作为基础服务，在做好高可用、高性能、好扩容的基础上，任何时刻都不能掉以轻心！！！

确保所有监控均处于生效状态；

所有新机器，均在saltstack上完成初始化和安装、部署操作，不能单独操作；

所有针对dns架构调整的操作，均需在流量低谷时操作；

对集群扩容操作时，务必对新加入节点做压测，同时重启服务器并检测重启后各项指标是否正常；

关注dns相关新闻，时刻跟进；

故障演练；

## dns的发展趋势

#### Dnsmasq

DNSmasq是一个小巧且方便地用于配置DNS和DHCP的工具，适用于小型网络，它提供了DNS功能和可选择的DHCP功能。它服务那些只在本地适用的域名，这些域名是不会在全球的DNS服务器中出现的。DHCP服务器和DNS服务器结合，并且允许DHCP分配的地址能在DNS中正常解析，而这些DHCP分配的地址和相关命令可以配置到每台主机中，也可以配置到一台核心设备中（比如路由器），DNSmasq支持静态和动态两种DHCP配置方式。

有一些公司在每台服务器上都起着dnsmasq，充当本地dns缓存服务，来提高dns解析性能同时减轻dnsserver的压力。

#### Httpdns

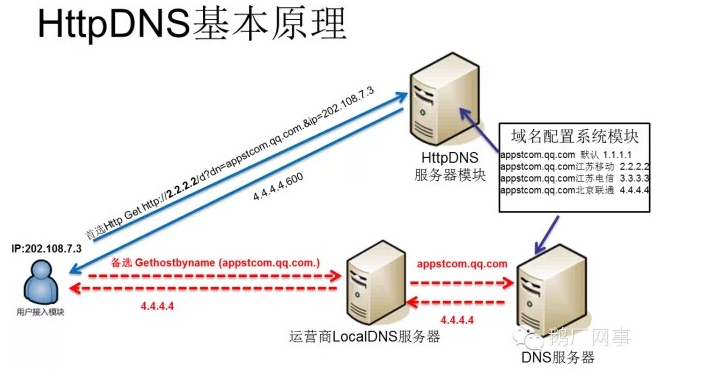
但凡使用域名来给用户提供服务的互联网企业，都或多或少地无法避免在有中国特色的互联网环境中遭遇到各种域名被缓存、用户跨网访问缓慢等问题。

首先是域名缓存，不同运营商，不同节点的缓存时间设置的差别较大。这样在流量切换时，就会产生新、旧应用数据不一致的现象。

其次就是域名解析过了太多层的nat，这就导致dns获取客户端地址时很难准确定位，从而智能dns的准确度大打折扣。

**最近两年，httpdns出现了**

用户明确的知道我在访问某厂的服务时，应该去找哪个ip要对应的域名，实现这个的前提是你可以左右用户的访问习惯，目前应用最适合的场景是app



在可预见的未来，不论在公司内网还是公网环境，dns在整个架构中，都将会以一个基础服务的位置存在，而目前最为稳定、高效的，依然是bind。

## 生产常见dns问题以及排障方法

#### 运维同学突然告诉我，你的dns不能用了

记得那时我刚来公司还没转正的时候。有个哥们问我，dns现在没什么事吧？卧槽，dns？什么dns？

记得那会线上跑着多少dns，每台dns配置一不一样，没人说得清楚。于是我说兄弟，你什么dns，怎么了？他告诉我一个ip，说用的这个dns，现在解析不了了。我说我看下，你先改个别的dns，这种时候一定要先恢复业务，然后再排查、解决问题。

我尝试用那台dns解析域名，确实不行了，登上去看，dns进程还在，想来是有什么瓶颈了，看日志吧，在日志中发现有报netfilter满了的日志，怎么tm会报这个呢，难道机器起着iptables？卧槽，还真是起着呢，这不蛋逼呢么。赶紧把iptables停掉，问题恢复了。

在iptable运行时，即便是没有任何规则，所有的报文也要过netfilter表，而dns这种高并发的业务很容易把netfilter表填满，从而影响服务。

后续的工作就是整理线上dns，同一配置，做好监控，做好高可用和负载均衡

#### 突然收到报警，提示dns虚地址不能正常解析了

那天带孩子体检，请了一天假，还在医院的时候，收到的报警，赶紧电话到公司，询问怎么回事，原来是有同事将zabbix的一个域名删除了，而这个域名恰恰是我lvs健康检测dns服务是否可以提供解析的域名，域名都没了，那肯定解析失败，于是lvs把所有dns都剔除了。

虽然恢复起来很快，但是这也提醒我，系统之间，一定要尽量减小藕合性，避免雪崩。

#### 开发来找，说调用微信支付接口有超时的现象，错误日志中报解析微信域名失败

接到这个case，首先是确认问题是否出现在域名解析的环节，我登录到服务器上，用nslookup解析对应的域名，连续解析1000次均没问题，但开发小妹依然喋喋不休，我按耐住tmd心情。突然想起也许她php中调用的解析函数与nslookup不同，于是我用host解析，这时发现问题了。每次解析微信支付这个域名的时候，都要等10s左右才出结果。这时，我又找其他机器做测试，发现只有这个微信支付的域名有这种现象，这不由得让我对大qq产生了膜拜的情绪。

出现这种情况让人很头大，而旁边开发小妹还在殷切的看着你，你背后还有领导不时的喘着粗气，而恰恰此时你对这问题一点头绪也没有，怎么办？

越是这种时候，越要淡定，不然怎么办？！这玩意不是谁教的，是你多遇到几次这种情况，每次都要提醒自己，冷静、冷静、tmd冷静

既然只有解析微信这个接口有问题，那我想可能是微信对dns这块有什么限制，那我可不可以换个forward的公网dns试下呢?分两头，一头是我去测试，另外一头让同事联系微信，给提供一个qq那边的权威dnsip，最终我将qq权威dns的ip加到我dns的forward列表中解决了这个case。

## 六、部分dns相关的面试题

### dns中递归查询与迭代查询的区别

1.递归查询:

一般客户机和服务器之间属递归查询，即当客户机向DNS服务器发出请求后,若DNS服务器本身不能解析,则会向另外的DNS服务器发出查询请求，得到结果后转交给客户机；

2.迭代查询(反复查询):

一般DNS服务器之间属迭代查询，如：若DNS2不能响应DNS1的请求，则它会将DNS3的IP给DNS2，以便其再向DNS3发出请求；

我们目前大部分查询都属于递归查询。

### Dns都会用到哪些网络协议，分别在什么时候用到

Tcp：主从同步数据时会用到

Udp：域名解析时用到

### 简述DNS进行域名解析的过程

首先，客户端发出DNS请求翻译IP地址或主机名。DNS服务器在收到客户机的请求后：

（1）检查DNS服务器的缓存，若查到请求的地址或名字，即向客户机发出应答信息；

（2）若没有查到，则在自身配置中查找，若查到请求的地址或名字，即向客户机发出应答信息；

（3）若没有查到，则将请求发给根域DNS服务器，并依序从根域查找顶级域，由顶级查找二级域，二级域查找三级，直至找到要解析的地址或名字，即向客户机所在网络的DNS服务器发出应答信息，DNS服务器收到应答后现在缓存中存储，然后，将解析结果发给客户机。

（4）若没有找到，则返回错误信息。

## 附录

### Dns安装脚本yum版

#!/bin/bash

####################################################################

# Auto install bind

# Create Date : 2012-11-28

# Written by :shanks

# Organization: DangDang

####################################################################

IN\_Face=`route -n |awk '{if($4~/UG/){print $8}}'|head -n 1`

Local\_IP=`/sbin/ifconfig|grep -B1 -C1 -w "${IN\_Face}"|grep -w 'inet addr'|awk -F: '{print $2}'|awk '{print $1}'`

cd /usr/local/src/

yum -y install bind-utils bind bind-devel bind-chroot bind-libs >>/tmp/init\_sn.log -y && rndc-confgen -r /dev/urandom -a || exit 1

# \*\*\*config /etc/named.conf\*\*\*

cat << shanks1 > /etc/named.conf

options {

version "1.1.1";

listen-on port 53 {any;};

directory "/var/named/chroot/etc/";

pid-file "/var/named/chroot/var/run/named/named.pid";

allow-query { any; };

Dump-file "/var/named/chroot/var/log/binddump.db";

Statistics-file "/var/named/chroot/var/log/named\_stats";

zone-statistics yes;

memstatistics-file "log/mem\_stats";

empty-zones-enable no;

forwarders {202.106.196.115;8.8.8.8; };

};

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

controls {

inet 127.0.0.1 port 953

allow { 127.0.0.1; } keys { "rndc-key"; };

};

logging {

channel warning {

file "/var/named/chroot/var/log/dns\_warning" versions 10 size 10m;

severity warning;

print-category yes;

print-severity yes;

print-time yes;

};

channel general\_dns {

file "/var/named/chroot/var/log/dns\_log" versions 10 size 100m;

severity info;

print-category yes;

print-severity yes;

print-time yes;

};

category default {

warning;

};

category queries {

general\_dns;

};

};

include "/var/named/chroot/etc/view.conf";

shanks1

# \*\*\*config /etc/rndc.key\*\*\*

cat << shanks2 > /etc/rndc.key

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

shanks2

# \*\*\*config /etc/rndc.conf\*\*\*

cat << shanks3 > /etc/rndc.conf

# Start of rndc.conf

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

options {

default-key "rndc-key";

default-server 127.0.0.1;

default-port 953;

};

shanks3

# \*\*\*config /var/named/chroot/etc/view.conf\*\*\*

cat << shanks4 > /var/named/chroot/etc/view.conf

view "View" {

allow-transfer {

#dns-ip-list;

};

notify yes;

also-notify {

#dns-ip-list;

};

# ixfr-from-differences yes;

zone "com" {

type master;

file "com.zone";

allow-transfer {

10.255.253.211;

};

notify yes;

also-notify {

10.255.253.211;

};

};

zone "forward.com" {

type forward;

forwarders { 10.255.253.220; };

};

};

shanks4

# \*\*\*config /var/named/chroot/etc/com.zone\*\*\*

cat << shanks5 > /var/named/chroot/etc/com.zone

\$ORIGIN .

\$TTL 3600 ; 1 hour

com IN SOA op.shanks.com. dns.shanks.com. (

2000 ; serial

900 ; refresh (15 minutes)

600 ; retry (10 minutes)

86400 ; expire (1 day)

3600 ; minimum (1 hour)

)

NS op.shanks.com.

\$ORIGIN com.

shanks A 1.2.3.4

shanks5

cd /var && chown -R named.named named/

/etc/init.d/named start

chkconfig named on

#check install status.

check\_cmd=`host -s -W 0.5 shanks.com 127.0.0.1|grep "1.2.3.4"`

if [ -z "${check\_cmd}" ]

then

echo "<ERROR!> hey,man.install bind --- ERROR!"

exit 5

else

echo "<OK> hey,man.install bind --- ok."

rndc stats

fi

if [ -f /tmp/Install\_bind.sh ]

then

rm -rf /tmp/Install\_bind.sh

fi

### dns安装脚本9.9版（编译版本）

#!/bin/bash

####################################################################

# Auto install bind

# Create Date : 2012-11-28

# Written by :shanks

# Organization: DangDang

####################################################################

IN\_Face=`route -n |awk '{if($4~/UG/){print $8}}'|head -n 1`

Local\_IP=`/sbin/ifconfig|grep -B1 -C1 -w "${IN\_Face}"|grep -w 'inet addr'|awk -F: '{print $2}'|awk '{print $1}'`

prefix='/usr/local/bind'

cd /usr/local/src/ && wget http://192.168.1.9/soft/dns/9.9/bind-9.9.7-P2.tar.gz && tar zxf bind-9.9.7-P2.tar.gz

if [ -d '/usr/local/src/bind-9.9.7-P2' ]

then

cd /usr/local/src/bind-9.9.7-P2 && ./configure --prefix=/usr/local/bind --enable-threads --with-libtool && make && make install

REAV=$?

if [ ${REAV} != 0 ]

then

echo 'bind make faild!!!'

exit 2

fi

else

echo 'bind src get filed!!!'

exit 1

fi

# \*\*\*config /etc/named.conf\*\*\*

cat << shanks1 > ${prefix}/etc/named.conf

options {

version "1.1.1";

listen-on port 53 {any;};

directory "${prefix}/etc/";

pid-file "${prefix}/var/run/named.pid";

allow-query { any; };

Dump-file "${prefix}/var/binddump.db";

Statistics-file "${prefix}/var/named\_stats";

zone-statistics yes;

memstatistics-file "var/mem\_stats";

empty-zones-enable no;

masterfile-format text;

# allow-update {none;};

# allow-recursion {any;};

# serial-query-rate 100;

# recursion no;

# dnssec-enable yes;

};

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

controls {

inet 127.0.0.1 port 953

allow { 127.0.0.1; } keys { "rndc-key"; };

};

logging {

channel warning {

file "${prefix}/var/dns\_warning" versions 10 size 10m;

#file "${prefix}/var/dns\_warning";

severity warning;

print-category yes;

print-severity yes;

print-time yes;

};

channel general\_dns {

file "${prefix}/var/dns\_log" versions 10 size 50m;

#file "${prefix}/var/dns\_log";

severity info;

print-category yes;

print-severity yes;

print-time yes;

};

category default {

warning;

};

category queries {

general\_dns;

};

};

include "${prefix}/etc/view.conf";

shanks1

# \*\*\*config /etc/rndc.key\*\*\*

cat << shanks2 > /etc/rndc.key

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

shanks2

# \*\*\*config /etc/rndc.conf\*\*\*

cat << shanks3 > ${prefix}/etc/rndc.conf

# Start of rndc.conf

key "rndc-key" {

algorithm hmac-md5;

secret "Eqw4hClGExUWeDkKBX/pBg==";

};

options {

default-key "rndc-key";

default-server 127.0.0.1;

default-port 953;

};

shanks3

# \*\*\*config ${prefix}/etc/view.conf\*\*\*

cat << shanks4 > ${prefix}/etc/view.conf

view "View" {

allow-transfer {

#dns-ip-list;

};

notify yes;

also-notify {

#dns-ip-list;

};

# ixfr-from-differences yes;

zone "com" {

type master;

file "com.zone";

allow-transfer {

10.255.253.211;

};

notify yes;

also-notify {

10.255.253.211;

};

};

zone "forward.com" {

type forward;

forwarders { 10.255.253.220; };

};

};

shanks4

# \*\*\*config ${prefix}/etc/com.zone\*\*\*

cat << shanks5 > ${prefix}/etc/com.zone

\$ORIGIN .

\$TTL 3600 ; 1 hour

com IN SOA op.shanks.com. dns.shanks.com. (

2000 ; serial

900 ; refresh (15 minutes)

600 ; retry (10 minutes)

86400 ; expire (1 day)

3600 ; minimum (1 hour)

)

NS dns.shanks.com.

\$ORIGIN com.

shanks A 1.2.3.4

shanks5

useradd named -s /sbin/nologin

cd /usr/local && chown -R named.named bind/

if [ -f /etc/init.d/named ]

then

rm -rf /etc/init.d/named

fi

wget -q http://192.168.1.9/soft/dns/9.9/named -O /etc/init.d/named && chmod +x /etc/init.d/named

/etc/init.d/named start

ln -s ${prefix}/sbin/rndc /usr/bin/rndc

ln -s ${prefix}/bin/host /usr/bin/host

ln -s ${prefix}/bin/dig /usr/bin/dig

chkconfig named on

#check install status.

check\_cmd=`host -s -W 0.5 shanks.com 127.0.0.1|grep "1.2.3.4"`

if [ -z "${check\_cmd}" ]

then

echo "<ERROR!> hey,man.install bind --- ERROR!"

else

echo "<OK> hey,man.install bind --- ok."

fi

if [ -f /tmp/Install\_bind.sh ]

then

rm -rf /tmp/Install\_bind.sh

fi