Debian Linux Set BIND 9 Caching DNS Server

The resolving name servers directly finds out information about the root servers, top level domains and authoritative name servers. It also speed up queries by caching results locallly as configured by hostmater in their domains' TTL field. You can setup BIND 9 as caching name server to speed up dns caching for the rest of all clients. This tutorial explains:

1. How to install BIND 9 caching only dns server under Debian Linux
2. BIND 9 troubleshooting tips
3. Iptables firewall configuration

Install BIND 9

The Berkeley Internet Name Domain (BIND) implements an Internet domain name server. BIND is the most widely-used name server software on the Internet, and is supported by the Internet Software Consortium. Debian Linux come with BIND 9. Type the following command to update package database i.e. resynchronize the package index files from their sources:

```
# apt-get update
```

To install BIND 9 under Debian Linux, enter:

```
# apt-get install bind9
```

Sample outputs:

```
Reading package lists... Done
```

```
Building dependency tree

Reading state information... Done

The following extra packages will be installed:

  bind9utils libbind9-40 libcap2 libdns45 libisc45
libisccc40 libisccfg40 liblwres40

Suggested packages:

  dnsutils bind9-doc resolvconf ufw

The following NEW packages will be installed:

  bind9 bind9utils libbind9-40 libcap2 libdns45 libisc45
libisccc40 libisccfg40 liblwres40

0 upgraded, 9 newly installed, 0 to remove and 0 not
upgraded.

Need to get 1292kB of archives.

After this operation, 3498kB of additional disk space will
be used.

Do you want to continue [Y/n]? y

Get:1 http://mirrors.kernel.org lenny/main libcap2 2.11-2
[11.8kB]
```

```
Get:2 http://mirrors.kernel.org lenny/main libisc45
1:9.5.1.dfsg.P3-1+lenny1 [163kB]

Get:3 http://mirrors.kernel.org lenny/main libdns45
1:9.5.1.dfsg.P3-1+lenny1 [603kB]

Get:4 http://mirrors.kernel.org lenny/main libisccc40
1:9.5.1.dfsg.P3-1+lenny1 [28.9kB]

Get:5 http://mirrors.kernel.org lenny/main libisccfg40
1:9.5.1.dfsg.P3-1+lenny1 [50.4kB]

Get:6 http://mirrors.kernel.org lenny/main libbind9-40
1:9.5.1.dfsg.P3-1+lenny1 [32.1kB]

Get:7 http://mirrors.kernel.org lenny/main liblwres40
1:9.5.1.dfsg.P3-1+lenny1 [47.7kB]

Get:8 http://mirrors.kernel.org lenny/main bind9utils
1:9.5.1.dfsg.P3-1+lenny1 [99.5kB]

Get:9 http://mirrors.kernel.org lenny/main bind9
1:9.5.1.dfsg.P3-1+lenny1 [255kB]

Fetched 1292kB in 1min9s (18.7kB/s)

Preconfiguring packages ...

Selecting previously deselected package libcap2.

(Reading database ... 21297 files and directories
currently installed.)
```

```
Unpacking libcap2 (from .../libcap2_2.11-2_amd64.deb) ...

Selecting previously deselected package libisc45.

Unpacking libisc45 (from .../libisc45_1%3a9.5.1.dfsg.P3-
1+lenny1_amd64.deb) ...

Selecting previously deselected package libdns45.

Unpacking libdns45 (from .../libdns45_1%3a9.5.1.dfsg.P3-
1+lenny1_amd64.deb) ...

Selecting previously deselected package libisccc40.

Unpacking libisccc40 (from
.../libisccc40_1%3a9.5.1.dfsg.P3-1+lenny1_amd64.deb) ...

Selecting previously deselected package libisccfg40.

Unpacking libisccfg40 (from
.../libisccfg40_1%3a9.5.1.dfsg.P3-1+lenny1_amd64.deb) ...

Selecting previously deselected package libbind9-40.

Unpacking libbind9-40 (from .../libbind9-
40_1%3a9.5.1.dfsg.P3-1+lenny1_amd64.deb) ...

Selecting previously deselected package liblwres40.

Unpacking liblwres40 (from
.../liblwres40_1%3a9.5.1.dfsg.P3-1+lenny1_amd64.deb) ...
```

```
Selecting previously deselected package bind9utils.

Unpacking bind9utils (from
.../bind9utils_1%3a9.5.1.dfsg.P3-1+lenny1_amd64.deb) ...

Selecting previously deselected package bind9.

Unpacking bind9 (from .../bind9_1%3a9.5.1.dfsg.P3-
1+lenny1_amd64.deb) ...

Processing triggers for man-db ...

Setting up libcap2 (2.11-2) ...

Setting up libisc45 (1:9.5.1.dfsg.P3-1+lenny1) ...

Setting up libdns45 (1:9.5.1.dfsg.P3-1+lenny1) ...

Setting up libisccc40 (1:9.5.1.dfsg.P3-1+lenny1) ...

Setting up libisccfg40 (1:9.5.1.dfsg.P3-1+lenny1) ...

Setting up libbind9-40 (1:9.5.1.dfsg.P3-1+lenny1) ...

Setting up liblwres40 (1:9.5.1.dfsg.P3-1+lenny1) ...

Setting up bind9utils (1:9.5.1.dfsg.P3-1+lenny1) ...

Setting up bind9 (1:9.5.1.dfsg.P3-1+lenny1) ...

Adding group `bind' (GID 107) ...
```

```
Done.

Adding system user `bind' (UID 104) ...

Adding new user `bind' (UID 104) with group `bind' ...

Not creating home directory `/var/cache/bind'.

wrote key file "/etc/bind/rndc.key"

#

Starting domain name service...: bind9.
```

Default Configuration Files

The default configuration files are as follows:

1. **/etc/bind** – This is the default bind 9 configuration directory under Debian Linux. Zone data files for the root servers, and the forward and reverse localhost zones are also provided in /etc/bind.
2. **/etc/bind/named.conf** – This is the primary configuration file for the BIND DNS server named.
3. **/var/cache/bind** – This is the working directory for named. Thus, any transient files generated by named, such as database files for zones the daemon is secondary for, will be written to the /var filesystem, where they belong.

4. **/var/log/syslog** – This the default log file for BIND 9 and you need to use this file to debug problems.

Running Caching Only Name Server

By default Bind 9 under Debian Linux runs as a “caching only” name server. You don’t have to make any changes and you can run the provided configuration as-is.

Service Configuration

To start, stop, and restart bind 9 server use the following command respectively:

```
# /etc/init.d/bind9 start
# /etc/init.d/bind9 stop
# /etc/init.d/bind9 restart
```

Make sure bind9 service star on boot time (this is set by installer):

```
# update-rc.d bind9 defaults
```

Testing Your Caching Name Server

Edit your /etc/resolv.conf file, type:

```
# vi /etc/resolv.conf
```

Update it as follows:

```
nameserver 127.0.0.1
```

Save and close the file. Now, type the following command:

```
# dig yahoo.com
```

Sample outputs:

```
; <<>> DiG 9.5.1-P3 <<>> yahoo.com

;; global options:  printcmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1676

;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 7,
ADDITIONAL: 2



;; QUESTION SECTION:

;yahoo.com.            IN    A



;; ANSWER SECTION:

yahoo.com.       21600 IN    A     69.147.125.65

yahoo.com.       21600 IN    A     72.30.2.43

yahoo.com.       21600 IN    A     98.137.149.56

yahoo.com.       21600 IN    A     209.191.122.70

yahoo.com.       21600 IN    A     67.195.160.76
```

```
;; AUTHORITY SECTION:

yahoo.com.       172800     IN    NS    ns5.yahoo.com.

yahoo.com.       172800     IN    NS    ns3.yahoo.com.

yahoo.com.       172800     IN    NS    ns2.yahoo.com.

yahoo.com.       172800     IN    NS    ns8.yahoo.com.

yahoo.com.       172800     IN    NS    ns4.yahoo.com.

yahoo.com.       172800     IN    NS    ns1.yahoo.com.

yahoo.com.       172800     IN    NS    ns6.yahoo.com.



;; ADDITIONAL SECTION:

ns6.yahoo.com.         172800     IN    A
      202.43.223.170

ns8.yahoo.com.         172800     IN    A
      202.165.104.22



;; Query time: 433 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)

;; WHEN: Mon May 24 05:18:21 2010

;; MSG SIZE  rcvd: 265
```

The query took 433 msec, but if you run the same query again, you will get extremely fast response as query is now cached by your own dns server:

```
# dig yahoo.com | grep time
```

Sample outputs:

```
;; Query time: 1 msec
```

This query will be cached as per domains TTL field (in this case it is set to 21438 seconds).

How Do I See Current Cache?

Type the following command to dump cache(s) to the dump file called /var/cache/bind/named_dump.db:

```
# rndc dumpdb
# less /var/cache/bind/named_dump.db
# grep 'yahoo.com' /var/cache/named/named_dump.db
```

Sample outputs:

```
yahoo.com.        172051      NS    ns1.yahoo.com.

                  172051      NS    ns2.yahoo.com.

                  172051      NS    ns3.yahoo.com.
```

```
                    172051      NS    ns4.yahoo.com.

                    172051      NS    ns5.yahoo.com.

                    172051      NS    ns6.yahoo.com.

                    172051      NS    ns8.yahoo.com.

ns1.yahoo.com.          172050      A     68.180.131.16

ns2.yahoo.com.          172050      A     68.142.255.16

ns3.yahoo.com.          172050      A     121.101.152.99

ns4.yahoo.com.          172050      A     68.142.196.63

ns5.yahoo.com.          172050      A     119.160.247.124

ns6.yahoo.com.          172051      A     202.43.223.170

ns8.yahoo.com.          172051      A     202.165.104.22
```

How Do I Debug BIND 9 Caching Server Problems?

The first place is to look error or warnings in /var/log/syslog file using the grep, cat, more, less or awk commands:

```
# tail -f /var/log/syslog
```

Sample outputs:

```
May 24 05:30:18 debian named[2165]: too many timeouts
resolving 'ns2.google.com/AAAA' (in 'google.com'?):
disabling EDNS

May 24 05:30:18 debian named[2165]: too many timeouts
resolving 'ns3.google.com/AAAA' (in 'google.com'?):
disabling EDNS

May 24 05:30:18 debian named[2165]: network unreachable
resolving 'ns4.google.com/AAAA/IN': 2001:503:a83e::2:30#53

May 24 05:30:19 debian named[2165]: too many timeouts
resolving 'ns1.google.com/AAAA' (in 'google.com'?):
reducing the advertised EDNS UDP packet size to 512 octets

May 24 05:30:20 debian named[2165]: too many timeouts
resolving 'ns4.google.com/AAAA' (in 'google.com'?):
reducing the advertised EDNS UDP packet size to 512 octets

May 24 05:30:47 debian named[2165]: received control
channel command 'dumdb'

May 24 05:30:47 debian named[2165]: unknown control
channel command 'dumdb'

May 24 05:30:50 debian named[2165]: received control
channel command 'dumpdb'

May 24 05:30:50 debian named[2165]: dumpdb started

May 24 05:30:50 debian named[2165]: dumpdb complete
```

Use the grep command to search log file:

```
# grep 'something' /var/log/syslog
# grep 'ns2.google.com' /var/log/syslog
# egrep -i 'example.com|example.org' /var/log/syslog
```

Is Port 53 Open?

Next make sure BIND 9 caching server is running on default port 53, run:

```
# netstat -tulpn | grep :53
```

Sample outputs:

```
tcp        0      0 192.168.1.10:53         0.0.0.0:*
LISTEN      2165/named

tcp        0      0 127.0.0.1:53            0.0.0.0:*
LISTEN      2165/named

tcp6       0      0 :::53                   :::*
LISTEN      2165/named

udp        0      0 192.168.1.10:53         0.0.0.0:*
2165/named

udp        0      0 127.0.0.1:53            0.0.0.0:*
2165/named

udp6       0      0 :::53                   :::*
2165/named
```

If port 53 is not open, start the bind 9 as follows:

```
# /etc/init.d/bind9 start
```

Iptables: Open Port 53

Finally, make sure iptables (Linux firewall) is allowing access to your dns server. In other words open TCP /

UDP port 53 as follows:

```
iptables -L -n
```

First, save existing iptables rules:

```
# iptables-save > /root/working.iptables.rules
```

Open both TCP and UDP port 53 for subnet 192.168.1.0/24, enter:

```
# iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW
-p tcp --dport 53 -j ACCEPT
# iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW
-p udp --dport 53 -j ACCEPT
# iptables-save > /root/new.iptables.rules
```

The above commands are necessary in order to allow LAN machines to contact the DNS server. You may need to adjust the rules as per your subnet and NAT setup; please refer to the iptables man page for further information.