Nominum

WHITEPAPER

How to Measure the Performance of a Caching DNS Server

Introduction

Nominum Vantio® is a high-performance caching DNS server. Nominum customers who evaluate Vantio are encouraged to run their own performance tests to compare it against competing DNS servers or against their existing DNS infrastructure, but those who attempt this are sometimes surprised to find that the results of their measurements differ significantly from what was expected or from Nominum's published performance numbers.

This document attempts to explain some of the issues involved in measuring the performance of caching servers, why some measurement techniques can yield widely varying results, and how to construct performance tests to get results that are as accurate and reproducible as possible.

What Are We Measuring, Anyway?

The first thing we need to know is what aspect of server performance we are trying to measure. This document discusses the measurement of throughput and latency; there are other things that could be measured such as memory consumption or network bandwidth consumption, but those will not be discussed here.

The throughput of a caching DNS server is *the number of queries per second it is capable of handling*. This number is of interest to administrators of high-volume servers who need to ensure sufficient capacity, for example at an ISP with a large number of customers or a search engine operator who needs to look up a large number of web server addresses.

The latency of a caching DNS server is the time it takes to answer any particular query. This is the aspect of server performance that is most visible to the individual end user – whenever you find yourself waiting for a web browser with a status line saying "Looking up host www.something.com...", you are experiencing the latency of a caching DNS server.

One might think that latency and throughput are related – if a server can answer *N* queries per second, doesn't that mean answering each query must take 1/*N* seconds on average? In fact, there is no such simple or obvious mathematical relationship between throughput and latency - it is possible for a caching server to have excellent latency characteristics but abysmal throughput or vice versa. This is because caching servers can and do work on multiple DNS queries simultaneously. In practice, busy servers can be working on several hundred queries or even thousands of queries at any one time, handling thousands of queries per second even though some of them take many seconds to complete.

Measuring Latency

In theory, measuring latency is easy: you send a query to a caching server, wait for a response, and see how long it took. You can even do this without any specialized measurements tool by using "dig", which conveniently prints the "Total query time" as part of its output.

Latency depends in part on the quality of the caching server implementation, such as its ability to effectively use cached information instead of performing repeated queries, to consistently select the fastest of multiple authoritative servers, and to avoid waiting for unresponsive servers for a long time when alternative servers are available. Unfortunately, it also depends heavily on a number of factors unrelated to the caching server itself, such as the following:

- The amount of repetition among the queries. Repeated queries answered from the cache will show a very small latency compared to ones where the server has to contact one or more authoritative servers to find the answer.
- Whether the cache already contains the same data used in the test (if it does, the test will run very fast indeed).
- Whether the cache already contains data unrelated to the test.
- The quality of the network connection. Latency measurements from a server that talks to authoritative servers over a 28 kbps dialup connection on another continent will look very different from those of a server with an OC-3 in the city next door.
- The percentage of queries that fail with a timeout due to unresponsive authoritative servers. These can take hundreds of times longer to complete than the average successful query: in a typical case, a successful query might take 0.1 seconds, but query that fails with a timeout might take 30 seconds. Therefore, even a small percentage of failed queries can easily dominate the overall average latency.

- The client's overall query timeout (how long the client will keep trying before giving up on a query). In practice, the latency of a failed query will be determined by the shorter of the caching server's internal query timeout and the client's query timeout; in other words, by whichever party is the first to give up.
- The client's retransmission timeout. This is of particular interest when measuring the performance of BIND version 8 or older because of a peculiarity in its lookup algorithm. In some circumstances, BIND 8 will find that it is missing some name server addresses needed to answer the query and will initiate a lookup for the missing addresses, but will not automatically resume processing the query when they arrive. Instead, it relies on the client to retransmit the query, which makes the overall latency dependent on how long the client waits to retransmit.

Note that the latency will not significantly depend on the speed of the machine the caching server is running on, since it is dominated by external network delays and timeouts rather than by processing time. For the same reason, measuring latency in a closed laboratory environment using artificial authoritative servers makes little sense; the latencies shown by such a measurement will bear little resemblance to those experienced when the server is talking to the live Internet.

DNSPerf measures Authoritative Domain Name services and is designed to simulate network conditions by "self-pacing" the query load.

In practice, getting repeatable latency numbers that can be meaningfully compared between different servers is very hard. Probably the best you can do is to measure the latency by sending a large number of queries similar to those a production server in your particular environment would see, with similar timeouts to what your production clients use, using the same network connection you would in production, and calculate an average. If you do this using multiple server implementations, the numbers should be meaningful for comparing one server against another, but they cannot be directly compared against numbers obtained in another environment.

One way of performing the latency measurement is using "dnsperf" (see Appendix A). Although dnsperf is primarily intended for measuring throughput, it can also be used for measuring latency. The trick is to limit the number of concurrent queries (specified by the "-q" option) to a small number to ensure that that neither the server nor the network link becomes overloaded at any point during the test. Values of 2 to 10 are reasonable choices.

You also need to specify the -H option, which makes dnsperf display latency statistics, including a histogram showing the latency distribution. The argument to the -H option is the number of buckets in this histogram; 20 is a reasonable value. Of the latency statistics printed, the average (Avg) is usually the one of most interest.

To summarize, running dnsperf with the following options should provide reasonable latency statistics:

dnsperf -q 10 -H 20

Measuring Throughput

The throughput of a caching DNS server is usually limited either by the amount of CPU time needed to process each query, the number of packets the network interface on the server machine can process per second, or by network bandwidth.

Answering a query from the cache requires much less CPU time and fewer packets of network traffic than answering a query where the server needs to perform a recursive lookup by querying authoritative servers. Therefore, it is useful to characterize the throughput of a caching server by two numbers:

- The number of queries it can answer per second when the answers are already in the cache (cache hits)
- The number of queries it can answer per second when the answers are not in the cache and the server must perform a recursive lookup, contacting one or more authoritative servers (cache misses)

The actual throughput with a mixed production load will be somewhere between these two numbers, closer to one or the other depending on the cache hit rate.

The throughput for cache misses will depend on how complicated the recursive lookup is - is it enough to query a single authoritative server, or does the caching server have to follow multiple referrals and look up missing name server addresses before reaching the answer? When the testing is done by doing lookups against the Internet, it is a good idea to use test queries derived from actual recorded caching name server traffic to make sure that the mix of queries is similar to that which occurs in production use (see Appendix B).

Measuring Throughput Querying the Internet

One way of measuring throughput is to have the caching server query actual authoritative servers on the Internet. This will give you realistic numbers, but for them to actually show actual maximum throughput of the server, you need a very fast Internet connection so that it does not become the bottleneck. It is very easy to end up measuring the speed of your Internet connection rather than that of the caching server. Preferably the connection should not be behind a firewall - we have found that some stateful firewalls do not take kindly to the deluge a DNS traffic generated by this kind of test and lock

Throughput can be measured by querying authoritative servers on the Internet or by testing it in a closed lab environment.

up, crash, or drop packets as a result of running out of state storage space.

You will also need an appropriate testing tool. The current tool of choice is "resperf", which was specifically designed for measuring caching server throughput and works much better than dnsperf for this purpose. The resperf program is distributed with dnsperf (see Appendix A). Please read the resperf(8) man page for detailed instructions.

Measuring Throughput in a Closed Lab Environment

As an alternative to running tests against authoritative servers on the Internet, throughput can be measured in a closed lab environment against a limited set of authoritative server machines, or even a set of authoritative servers running on virtual IP addresses on a single machine.

This is likely to yield slightly less realistic but more reproducible results than tests involving the Internet. It also requires much less parallelism from the test clients because the round-trip times on a laboratory LAN are shorter – each query will be answered more quickly, which lets each client send more queries per unit of time. Finally, it has the advantage of not overloading one's Internet connection with huge amounts of UDP traffic.

The downside is that you need to do the extra work of designing and setting up a set of authoritative servers to simulate those on the Internet, generating and loading authoritative zone data, and generating a set of queries that can be resolved using those servers. The authoritative servers need to be fast enough not to become the bottleneck in the test.

In our own laboratory measurements, we use a standardized test designed to approximate the behavior of typical web surfing clients. It involves queries of the form "www.domain.com" for a large number of different values of "domain", where each "domain" is separately delegated from a simulated "com" server using two NS records and two glue A records. This means that each query to the caching server will require two queries to the authoritative servers: one to the "com" server returning a referral, and a second one to the "domain.com" server returning the answer, just like a typical lookup for an uncached web server address would. To avoid the authoritative servers becoming a bottleneck, we use a simple authoritative server specifically written for the purpose of this test; this server only knows how to answer the types of queries that occur in the tests but will do it much faster than a general purpose authoritative server could.

Appendix A: Obtaining *dnsperf* and *resperf*

The "dnsperf" and "resperf" programs were written by Nominum and are distributed in source code form as well as installable packages for Red Hat Linux and Solaris. These packages can be downloaded from:

http://www.nominum.com/resources/measurement-tools

Appendix B: Obtaining Test Data

The dnsperf distribution contains a sample query file with 100,000 queries. A larger sample query file with 3,000,000 queries is also available from the measurement tools web page. These files contain queries that were gathered from the Internet. However, to run a latency or throughput test that provides performance numbers most representative of a particular network, you will need to collect a set of test queries gathered from that site. One way of collecting such queries is to log the queries received by a caching DNS server at that site, either using tcpdump or using the server's query logging function.

Obtaining Test Data Using *tcpdump*

To make a packet capture of the incoming DNS traffic at a caching server, you can run tcpdump on the server machine. For example, if the machine's IP address is 10.0.0.1, you can use the command:

```
tcpdump -w queries.pcap -p -s 1500 dst 10.0.0.1 and dst port 53
```

to save the incoming DNS traffic to the file "queries.pcap". The purpose of the "dst 10.0.0.1" clause is to make sure that only incoming queries are saved, not the outgoing queries sent to authoritative servers by the caching server itself.

This saves the queries in the binary "pcap" file format. This can be converted into the query file format accepted by dnsperf and resperf using the "queryparse" program in the contrib/queryparse directory of the dnsperf source distribution.

Obtaining Test Data Using Query Logging

Another way of saving DNS queries for use as test data is to enable query logging on the server. For example, if you are running BIND 9, you can add these lines to named.conf:

```
logging {
        channel queries {
                file "query.log";
```

```
        };
        category queries { queries; };
};
```

and all queries will be saved in the file "query.log". To convert this log file into a format suitable as input for dnsperf or resperf, you can use a shell command like

```
perl -n -e 'print "$1 $2\n" if /query: ([^ ]+) IN ([^ ]+);/' query.log >queries
```

If you are running Nominum Vantio, you can use the global configuration statements:

```
server.update use-querylog-file=true;
server.update querylog-format="{name} {type}"
```

and the view or resolver configuration updates:

```
view.update name=world log-queries=true

resolver.update name=res1 log-queries=true
```

The querylog-format statement used in the above example will make Nominum Vantio write a query log file that is compatible with dnsperf and resperf as-is, without requiring a separate conversion step.

About Nominum

Nominum's network naming and addressing solutions power the world's largest always-on networks. Nominum is a global provider of DNS, and DHCP solutions that enable communication providers to deliver high quality always-on broadband internet and innovative services to their customers, including VoIP, push to talk, fixed-mobile convergence, and IPTV.

For further information, visit www.nominum.com.