1

resperf

resperf measures the resolution performance of caching DNS servers.

Overview

resperf is a caching-specific DNS testing companion to *dnsperf*, Nominum's authoritative DNS testing tool.

dnsperf was designed primarily for benchmarking authoritative DNS servers—numerous attempts to use *dnsperf* to test caching DNS servers have been made, with poor results. A tool that addressed the specific requirements inherent in the testing of caching name servers was required.

A key difference between caching and authoritative DNS servers is that caching servers may need to work on thousands of queries in parallel to achieve maximum throughput, whereas authoritative servers process queries in order and one-at-a-time.

For authoritative servers, *dnsperf* tests this "self-pacing" approach by sending a small burst of back-to-back queries to fill up network buffers (keeping the server at 100% utilization), and sending a new query when a response is received. This approach works well for authoritative servers, and even works (relatively) well for caching servers in a closed laboratory environment in which the server is "talking" to a simulated Internet on a single LAN.

For caching servers, however, this approach fails to accurately test performance when "talking" to the actual Internet—as previously mentioned, under real-world conditions a caching server may need to process thousands of queries in parallel to achieve maximum throughput.

Command Synopsis

```
resperf [-d datafile] [-s server_addr] [-p port] [-b bufsize] [-f family]
   [-e] [-D] [-y name:secret] [-v] [-A] [-h] [-i plot_interval]
   [-m max_qps] [-P plotfile] [-r rampup_time] [-L max_loss]
```

Options

Option	Description
`-d` *`datafile`*	Specifies an input data file. The default is **stdin**.
`-s` *`server_addr`*	Sets the server to query. The default is `127.0.0.1`.
`-p` *`port`*	Sets the port on which to query the server. The default is 53.
`-b` *`bufsize`*	Sets the socket send/receive buffer size in kilobytes. The default is 32 Kbps.
`-f` *`family`*	Specifies the address *`family`* of DNS transport. Options are • `any`—Uses the address type of the address as specified in `-s`. • `inet`—IPv4 transport. • `inet6`—IPv6 transport. The default is `any`.
`-e`	Enables EDNS 0.
`-D`	Sets the DNSSEC OK bit (implying EDNS).
`-y` *`name:secret`*	Specifies the TSIG `name` and `secret`. There is no default.

Table 1-1 resperf command-line options

Option	Description
`-A`	Reports command-line arguments.
`-h`	Prints a usage message and exits.
`-i` *`plot_interval`*	Specifies the time interval between plot data points, in seconds. The default is 0.5.
`-m` *`max_qps`*	Specifies the maximum number of queries per second. The default is 100000.
`-P` *`plotfile`*	Specifies the name of the plot data file. The default is *resperf.gnuplot*.
`-r` *`rampup_time`*	Specifies the ramp-up time in seconds. The default is 60.
`-L` *`max_loss`*	Specifies the maximum acceptable query loss, as a percentage. The default is100.

Table 1-1 resperf command-line options *(continued)*

Understanding resperf

resperf, unlike *dnsperf*, sends DNS queries at a controlled, steadily increasing rate—by default, *resperf* sends traffic for 60 seconds, linearly increasing the amount of traffic from zero to 100,000 queries per second (qps).

During testing, *resperf* tracks responses from the server as well as response rates, failure rates, and latencies. After *resperf* stops sending traffic, it continues to “listen” for responses for an additional 40 seconds to give the server time to respond to the last queries sent.

NOTE The 40-second time limit is longer than the overall query timeout of both Nominum CNS and current BIND versions.

Defining Success

A successful test is completed when the query rate exceeds the server's capacity and queries start to be dropped, causing the response rate to either stop or decrease as the query rate increases.

Determining Maximum Throughput

Maximum throughput is determined from the plot as either:

- The highest response rate on the plot.
- The response rate at the point where a significant number of queries begin to be dropped.

Test Results

Test results are written to a file in a tabular format as a set of measurements of the query rate, the response rate, the failure response rate and the average query latency as functions of time, and may be plotted using *gnuplot* or a similar plotting tool. See "*The Plot Data File*" on page 8.

resperf, when constructing plot data, always places each query at the point in time at which the query was sent, *not* the point in time at which the response (if any) was received. This permits easy comparison of query and response rates.

Operational Considerations

Benchmarking a live caching server can have serious operational ramifications.

Query Volume and Bandwidth Saturation

A fast caching server such as Nominum CNS, running on an Opteron™ server and resolving a mix of cacheable and non-cacheable queries typical of an ISP's customer traffic, is entirely capable of resolving over 50,000 qps. In the process of resolving those queries, the server will send more than 20,000qps to authoritative Internet servers, and receive answers to most of those queries.

If you assume an average request size of 50 bytes and a response size of 100 bytes, this adds up to approximately 8 Mb per second (Mbps) of outbound traffic and approximately 16 Mbps of inbound traffic.

Ensure that your internet connection can handle this amount of bandwidth with room to spare. If you fail to do so:

1. You may saturate your connection, causing a service degradation for your users.

2. More seriously (in the context of testing accuracy), you may wind up measuring the speed of the *connection* instead of the speed of the *server*.

Firewalls

Ensure that no stateful firewalls exist between the caching server and the Internet. As a rule, stateful firewalls can't handle the amount of UDP traffic generated by *resperf*, and may skew test results by:

- Dropping packets.
- Locking up.
- Crashing.

Separate But Equal

Performance testing at the traffic levels involved is essentially a hard real-time application. At a query rate of 100,000 qps, a 1/100s delay translates into 1000 incoming UDP packets—this is far more than most operating systems can buffer.

Therefore—on the same LAN as the server under test—run *resperf* on *its own machine*, ensuring that:

- The machine running *resperf* is *at least* as fast as the server being tested—otherwise, it may become a performance bottleneck.
- There are no other applications running on the machine running *resperf*.

Timer Granularity and CPU

Most operating system timers are of too coarse a granularity to schedule packet transmissions at sub-millisecond intervals. As a result, *resperf* busy-waits between packet transmissions, constantly polling for responses. It's therefore normal for *resperf* to consume 100% CPU during the whole test run, even during periods where query rates are relatively low.

What You Will Need

A Query Input File

Each test requires a set of test queries in the *dnsperf* file format.

The input file contains one line per query, consisting of a domain name and an RR type name separated by a space. Query classes are implicitly IN.

To make the test as realistic as possible, these should be derived from recorded production client DNS traffic, without removing duplicate queries or other filtering. With the default settings, *resperf* will use up to 3 million queries in each test run.

Servers that serve non-terminal zones such as the root zone or top-level domains (TLDs) spend most of their time providing referral responses, not authoritative answers, and a realistic input file, therefore, should:

- Consist primarily of "A" queries for names *below*, not names *at*, the delegations present in the zone.
- Contain a realistic proportion of queries for nonexistent domains to queries for existing domains.
- Randomly order the lines of the input file.

For example, when testing the performance of a server configured to be authoritative for the top-level domain *fi.*, which contains delegations for the *helsinki.fi* and *turku.fi* domains, the input file should contain lines similar to:

```
www.turku.fi A
www.helsinki.fi A
```

In these lines, the `www` prefix ensures that the server will respond with a referral.

Configuration

In *resperf* plots, limits on the number of simultaneous resolutions (like the `max-recursive-clients` statement in Nominum CNS or the `recursive-clients` option in BIND 9) show up as increases in the number of failure responses.

To avoid this, increase these limits.

Nominum's recommended settings are:

- CNS—Set `max-recursive-clients` to 10000 (ten thousand).
- BIND 9—Set `recursive-clients` to 100000 (one hundred thousand).

NOTE You may be tempted to "prime" the server's cache by having the server resolve typical query traffic for a period of time prior to testing, thus ensuring that the cache isn't empty when the test starts.

In practical terms, this isn't necessary, as the server reaches a reasonable cache hit rate (60% or more) during the test, even when starting with an empty cache. Furthermore, you run the risk of accidentally using the same query set for both "priming" and testing—if that happens, when you start testing, the server will answer almost all queries from the "primed" cache, and yield inflated performance numbers.

Running Tests

To run *resperf*, providing the minimal requirements of a server IP address and the query data file, issue a command like:

```
# resperf -s 10.0.0.2 -d queryfile
```

As *resperf* runs, some status messages and summary statistics are written to **stdout**, and plot file data is written to *resperf.gnuplot* in the current directory (unless another plot file name has been specified with the `-P` option).

Test Duration

A test run, using the default settings, takes 100 seconds at most, comprised of 60 seconds of traffic ramp-up followed by 40 seconds of waiting for responses. However, in practice, the 60-second traffic phase is usually curtailed.

There are several different conditions under which *resperf* will transition from the traffic-sending phase to the waiting-for-responses phase:

- *resperf exceeds 65,536 outstanding queries*—This is the most frequent reason *resperf* stops sending queries before the 60 seconds has finished, and this occurs because *resperf* has exceeded the capacity of the server being tested.

  The limit of 65,536 queries originates with the number of possible ID field values in the DNS packet—*resperf* allocates a unique ID for each outstanding query, and is therefore unable to send further queries if the set of possible IDs is exhausted.

- *resperf is unable to send queries fast enough*—*resperf* may fall behind in query transmission because it cannot send queries quickly enough. If so, once the backlog reaches 1000 queries, *resperf* prints a message describing the number of backlogged queries and stops sending traffic.

If this message appears, ensure that the machine running *resperf* is sufficiently fast and has no other applications running, and monitor the CPU usage of the server under test.

- *resperf successfully reaches the maximum query rate—resperf* may run for its full allotted time and successfully reach the maximum query rate (by default, 60 seconds and 100,000qps, respectively). With today's hardware, this is unlikely.

Troubleshooting

Server response rates, regardless of what causes the test to end, should flatten towards the end of the test. If this trend doesn't show in the result plot, the server isn't loaded heavily enough.

If the CPU usage of the server under test doesn't reach 100% (or close to it) at the point of maximum traffic, you likely have a bottleneck in some other part of the test harness. For example, your external Internet connection may be saturated.

As previously mentioned, if *resperf* is unable to send queries quickly enough, ensure that the machine upon which it is running is sufficient for the purposes of testing—ensure that it's fast enough, and that there are no other applications running on the machine.

The Plot Data File

Test runs are divided by default into 0.5-second time intervals for the purposes of generating the plot data file (alternative intervals may be specified using `-i`). Each line in the plot data file corresponds to one interval, and contains the following values specified as floating-point integers:

Time

The midpoint of this time interval in elapsed seconds since the beginning of the test run.

Target queries per second

The number of queries per second *scheduled* to be sent in this time interval.

Actual queries per second

The number of queries per second *actually* sent in this time interval.

Responses per second

For this time interval, the number of responses received to queries sent, divided by the interval length.

Failures per second

For this time interval, the number of non-NXDOMAIN and non-NOERROR responses received to queries sent, divided by the interval length.

Average latency

For queries sent in this time interval, the average time between the sending of a query and the receiving of a response.

Measurements for any given query always apply to the time interval within which the query was sent, not the time interval within which the response (if any) was received. For example, if no queries are dropped, the query and response curves are identical. However, if a plot shows 10% failure responses at t=5 seconds, it means that 10% of the *queries sent* at t=5 seconds eventually failed, not that 10% of the *responses received* at t=5 seconds were failures.

Plotting Test Results

The *resperf-report* shell script runs *resperf* with its output redirected to a file, from which an illustrated report in HTML format is automatically generated. *resperf-report* accepts the command-line arguments in Table 1-1 on page 2, *"resperf command-line options"*, passing those arguments unchanged to *resperf*.

Reports are stored with a unique filename based on the current date and time. For example:

```
20060812-1550.html
```

GIF images of the plots and other auxiliary files are stored in separate files beginning with the same date-time string.

For example, to benchmark a server running on `10.0.0.2`, you could run

```
# resperf-report -s 10.0.0.2 -d queryfile
```

and then open the resulting *20060812-1550.html* file in a web browser.

NOTE *resperf-report* uses *gnuplot* to generate plots—ensure that *gnuplot* is installed, and that it supports the *gif* terminal driver. *gnuplot* may be obtained at http://www.gnuplot.info.

To copy the report to a separate machine for viewing, copy the *.gif* files along with the *.html* file, or copy all of the files using

```
# scp 20060812-1550.* host:directory/
```

Interpreting Results

Summary statistics are printed to standard output at the end of the test, and include the server's measured maximum throughput.

By default, the maximum throughput is the highest point on the response rate plot, without regard to the number of queries dropped or failing at that point. If you want *resperf* to report throughput at the point in the test where the percentage of queries dropped exceeds a given limit, which may be a more realistic indication of how much the server can be loaded while still providing an acceptable level of service, use the `-L` command-line option.

For example, specifying

```
# resperf -s 10.0.0.2 -L 10 -d queryfile
```

forces *resperf* to report the highest throughput reached before the server starts dropping more than 10% of queries received.

A Note On Failed Queries

All plots should be manually inspected to ensure that they don't contain an abnormal number of failed queries. It isn't possible to automatically constrain results based upon the number of failed queries, because failed queries (unlike dropped queries) occur even when the server is not overloaded. Additionally, the number of failed queries is heavily dependent upon both query data and network conditions.