resperf

The

resperf Options

Option	Description
`-a` *`local_addr`*	Specifies the local address form which to send requests. The default is the wildcard address.
`-b` *`bufsize`*	Sets the socket send/receive buffer size in kilobytes.
`-c` *`constant_traffic_time`*	Specifies the length of time, in seconds, for which to send a constant stream of traffic. The default is 0 seconds.
`-C` *`clients`*	Act as multiple clients. Requests are sent from multiple sockets. The default is to act as 1 client.
`-d` *`datafile`*	Specifies an input data file. The default is **`stdin`**.
`-D`	Sets the DNSSEC OK bit (implying EDNS).
`-e`	

`-m` *`max_qps`*
: Specifies the maximum number of queries per second.

  The default is 100000.

`-p` *`port`*
: Sets the port on which to query the server.

  The default is 53.

`-P` *`plotfile`*
: Specifies the name of the plot data file.

  The default is

Query Volume and Bandwidth Saturation

A fast caching server such as Nominum Vantio, running on a XEON® server and resolving a mix of cacheable and non-cacheable queries typical of an ISP's customer traffic, is entirely capable of resolving over 100,000 qps. In the process of resolving those queries, the server will send more than 40,000 qps 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 150 bytes, this adds up to approximately 16Mb per second (Mbps) of outbound traffic and approximately 48 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:

- There are no other applications running on the machine running perf

Timer Granularity and CPU

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

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 for the configured number of 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.

resperf

- resperf is nable to sen 0 TD [faostenoughf resperf becausemitcannoto sen sperf

Table 1-1, 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
```

PNG 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 .

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

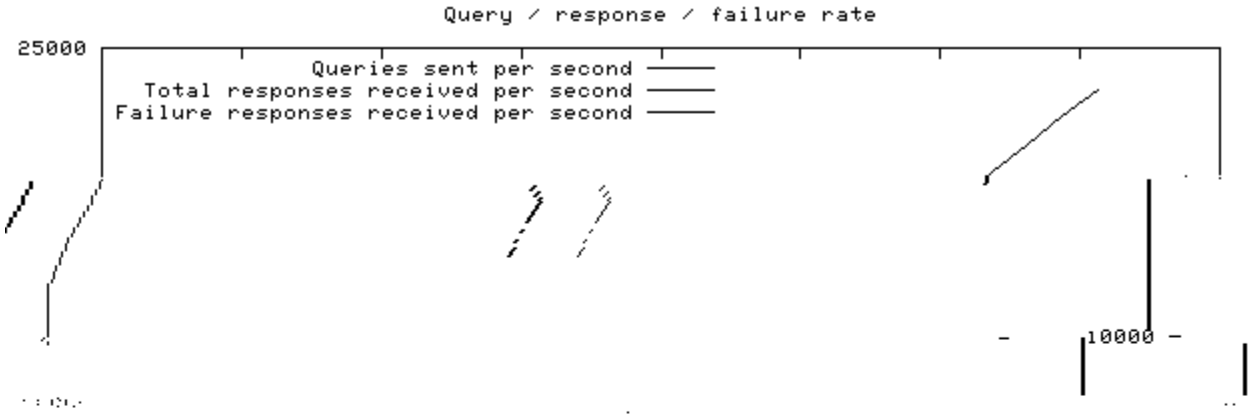
```
# scp 20060812-1550.* host:directory/
```

Resperf Plots

The resperf-report contains two plots generated by gnuplot:

- Query/response/failure rate
- Latency

Figure 1-1 Query/Response/Failure Rate Plot



Latency Plot

The Latency plot contains an “Average latency” graph that shows latency variations during the course of a test. The “Average latency” graph typically exhibits a downward trend because, during the test, the cache hit rate improves as the number of responses cached increases (and the latency for a cache hit is smaller than for a cache miss). A sharp spike in the graph indicates the point at which the server becomes overloaded.

Do not use the latency graph for absolute latency measurements or comparisons between servers; the latencies shown in the graph do not represent production latencies due to an initially empty cache, and the deliberate overloading of the server occurs towards the end of the test.

NOTE All latency measurements are displayed on the plot at the horizontal position that corresponds to the time when the query was sent (and not when the response, if any, was received). This enables the comparison of query and response rates; for example, if no queries are dropped, the query and response graphs are identical. If the plot shows 10% failure responses at 5 seconds, this indicates that 10% of the queries sent at 5 seconds

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 the following command forces `resperf` to report the highest throughput reached before the server starts dropping more than 10% of queries received:

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

When a server is driven into overload, the service it provides may deteriorate gradually, and this deterioration can manifest itself in any of the following ways:

- queries being dropped
- an increase in the number of SERVFAIL responses
- an increase in latency

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 is not 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.

Generating Constant Traffic