WHITE PAPER

# Lies, Damn Lies and DNS Performance Statistics

How to measure the *real* performance of a DNS caching resolver

SECURE64

# Executive Summary

Service providers' DNS query loads are rising around the world, and in some industries, very dramatically. Designing and maintaining a DNS resolver infrastructure that is capable of handling this load is critical in order to ensure a positive end user experience. To do this, DNS operators must properly size their resolver infrastructure for current and anticipated future peak loads over the life of their DNS servers.

Obtaining useful information on the performance characteristics of different DNS resolvers can be difficult, however, because DNS resolver architectures exhibit dramatically different behavior under real-world loading conditions. To complicate matters further, most commercial DNS vendors publish performance statistics measured under unrealistic laboratory conditions, making initial size and capacity planning decisions difficult.

This white paper examines the primary causes of rising DNS loads, explains the pros and cons of different architectural approaches to DNS scalability, and provides actionable recommendations on how to accurately measure and compare the performance of different DNS resolvers under a variety of conditions that can be encountered in real-world deployments. The results of such benchmarks allow DNS operators to properly size competing DNS resolver infrastructures and calculate their total cost of ownership.

# Why DNS loads are rising

Historically, loads on the DNS have tracked growth in internet users fairly closely. The more internet users, the more DNS queries one could expect. More recently, however, DNS loads have been rising faster, and in some cases, dramatically faster, than internet users. There are several drivers of this increased growth:

- More devices. In the old days, a household might have a PC as its sole internet connected device. Today, there is an explosion in the number of devices that communicate over the internet, including laptop and desktop computers, smart phones, tablets, set top boxes, audio and video equipment, security systems and increasingly, everyday household appliances. Each of these devices utilizes the DNS to locate servers that it needs to communicate with.
- Mobile broadband. Mobile broadband is growing 43% per year annually, with a shift towards even faster 4G networks. Modern 4G-enabled smart devices place a much larger load on the DNS than their 3G counterparts.
- IPv6. Dual stack IPv4/IPv6 clients generally query for an AAAA record over IPv6 first and if there is no answer, fall back to an A query over IPv4. Since most of the internet still runs IPv4, this generates a substantial amount of extra traffic on the DNS servers.
- Web pages. Today's web pages are much more complex than just a few years ago, often pulling content from many web servers. The number of DNS queries generated by some popular web site pages is now in the hundreds.
- DNSSEC. Security-aware resolvers will retrieve DNSSEC signatures and DS records along with answers, even if they do not perform DNSSEC validation themselves. The additional data that must be retrieved and stored in the cache places an additional load on the resolver.

Service providers must scale their DNS infrastructure to support these rapidly increasing loads. Failure to scale the DNS properly can result in dropped queries, excessive latency and a poor end user experience.

# DNS scalability challenges

DNS operators seeking to implement an architecture that scales to these rising loads can choose from three different approaches to scalability, each having its advantages and disadvantages.

## Scale up

The simplest way to scale DNS capacity is to obtain more performance from each DNS server, typically by adding more CPUs and memory. BIND, for example, can be configured to run multiple threads, one thread per CPU core.

Unfortunately, the performance of BIND and other conventional DNS architectures do not scale very well in a multi-threaded configuration. Benchmark studies show that resolver performance does not scale well beyond the first couple of threads.

## Scale out

It seems intuitive – to add more DNS capacity, just add more servers. While this seems obvious, scaling performance by adding servers can add cost and complexity to a network, including:

- Capital cost for the server hardware

- Capital cost for load balancers. Only two resolver addresses can be published to clients, so if the load is high enough, load balancers may be required to accommodate it.
- Operating costs for rack space, power and cooling
- Deployment costs to initially set up and certify the server
- Ongoing personnel costs, including hardening, patching and configuration management

### Scale in hardware

If you can't scale up, and you can't afford to scale out, then what else can you do? Some vendors have introduced products that improve DNS appliance performance through an onboard network controller or ASIC with an embedded cache. This hardware sees all queries to and responses from the resolver and caches the responses so that subsequent queries for the same domain name and record type can be answered by the hardware rather than the software.

Although this approach seems attractive because the advertised performance numbers can be extremely high, in practice it suffers from several fatal flaws. First, this approach only provides performance for cache hits, not cache misses, which must still be resolved in software. So in a real world environment of between 70-90% cache hits, the performance will be limited by the software. Second, it adds no performance boost at startup, when the cache is empty. And finally, it does not help during a failover scenario, when the real traffic load might double.

## The Secure64 approach

Secure64 has developed the Capacity Expansion Module (CEM), a simple add-on to its DNS Cache product that allows performance to be scaled in software. By leveraging proprietary queuing and query processing algorithms, CEM doubles the performance of DNS Cache under real-world traffic. This allows DNS performance to be easily scaled up in software without the cost and complexity of additional servers or the performance limitations of hardware-based approaches to scalability.

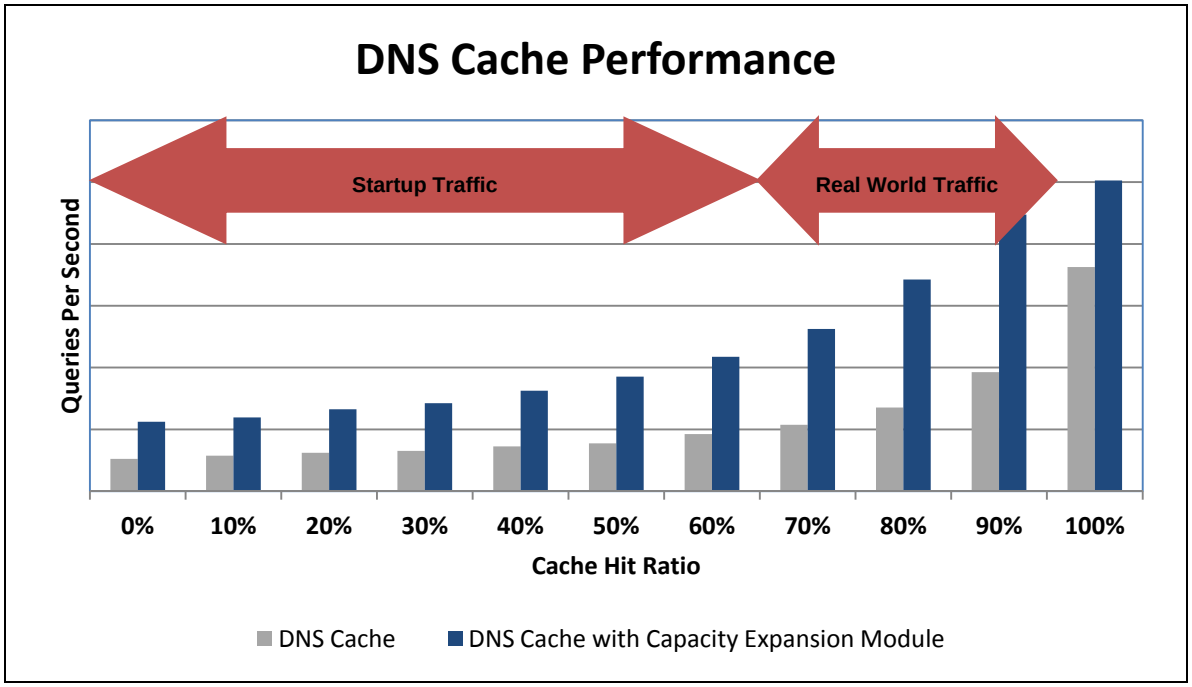


Fig 1 – CEM performance results under varying traffic loads

The chart above illustrates the performance gains that the Capacity Expansion Module provides. From startup through the top of the real-world traffic range of 90% cache hits, the Capacity Expansion module doubles the performance of the DNS server.

## Performance measurement challenges

Given the very different performance and cost characteristics of the three different scalability architectures described above, it is important for a DNS operator to understand performance of a resolver in real-world traffic before sizing the DNS infrastructure. However, measuring the performance of a DNS resolver in real world traffic can be difficult, as the performance of the resolver depends on a number of variables, including:

- Cache hit/miss ratio. Cache misses must be resolved by querying authoritative servers, sometimes multiple times, over the internet. Cache misses take significantly more time to process than hits.
- Network latency. Resolvers will timeout and retry authoritative servers if they have not received an answer in a certain amount of time. If there is high latency on the network, the resolver will end up issuing multiple authoritative queries in order to resolve a domain name, consuming system resources and slowing down the server.
- Internet unpredictability. Even though the internet is built for resiliency and redundancy, nameservers can be down and networks congested. When this happens, a DNS resolver must work harder to obtain an answer to a particular query. This is also why the same test run at two different times of day can yield different results, as external conditions may have changed.
- Configuration settings. Resolver performance depends to some degree on available system resources such as cache memory (so that more cache hits can be stored) and recursive contexts (how many recursive queries are allowed to be in flight at any given time) as well as timeout and retry settings. Different settings may be required to optimize performance for different DNS software architectures and different networks.
- DNSSEC. While adding important security to the DNS, DNSSEC puts an additional load on the resolver as it must issue additional queries to obtain public keys and perform additional cryptographic operations to check digital signatures. Even when DNSSEC validation is turned off, a security-aware resolver must request and store DNSSEC information from authoritative servers in case a validating, security-aware client requests this DNSSEC information in order to perform validation itself. These additional queries and data impact the maximum throughput of the resolver.
- Other loads on DNS server. Non-DNS activity on the server can impact its performance, as the available CPU cycles must be shared among all running processes. Essential services such as syslog, SNMP, NTP, and BGP and optional services such as query logging will impact DNS performance, sometimes dramatically.

For these reasons, most vendors do not attempt to publish performance numbers except as 100% cache hits, which is the only number that is reasonable easy to generate and reproduce. Unfortunately, 100% cache hit performance is not a particularly useful metric for most organizations.

# How to measure *real* DNS performance

What matters the most is how a DNS server will perform in *your* network, not how it is characterized in a vendor's sales material. Here are some recommendations for conducting your own benchmarks that will allow you to determine and compare the performance of different DNS servers against one another in meaningful ways.

## Lessons learned

First, a note about how we came to these conclusions. During the development of the Capacity Expansion Module, Secure64 engineers had a need to benchmark performance under a variety of loading conditions. Thinking that this would be a fairly straightforward exercise, we set up a private network, configured the hardware and software, and began testing. Many months later, after numerous restarts and much head scratching, we had learned some very important lessons about testing high performance resolvers:

- You need a really fast test server to generate enough queries.
- Resperf, which is really the only freely available tool available that was designed to test a resolver, can be inconsistent from one run to another, especially at high query rates. For this reason, we had to run each test multiple times and take the median or mode result to dampen the variability.
- You cannot just run multiple instances of resperf to scale to high loads, because of the way resperf decides when maximum resolver capacity has been reached.
- The switch matters.
- The NICs on the test server and resolver matter.

The following recommendations are made as a direct result of our own experiments and learning.

## Lab testing

Despite the fact that they are not "real-world" tests, lab tests have several advantages when comparing different products against one another. First, because they can be conducted on a closed network, lab test results are consistent, as they eliminate variables such as network latency, internet unpredictability, nameserver availability, and other factors that are outside the operator's control but can impact results significantly. Second, lab tests are repeatable, regardless of when they are executed, as long as the test environment is well controlled. Finally, lab tests are comparable across products, as each product is being tested under exactly the same test conditions.

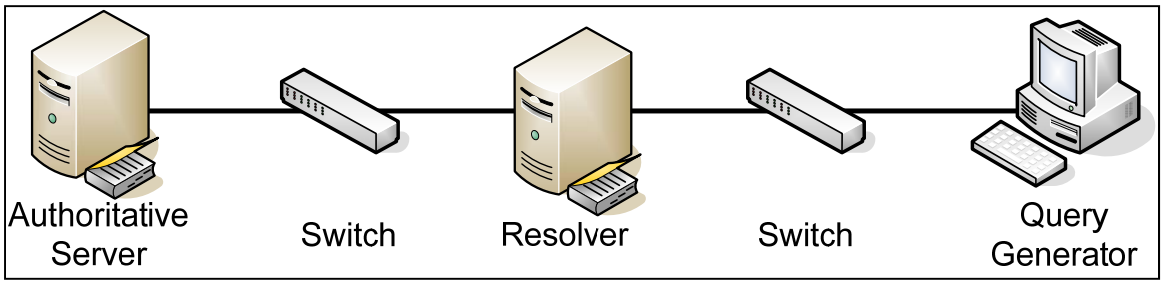


Fig 2 – An ideal test environment

The figure above illustrates the ideal lab test environment. If you use this infrastructure and follow the testing recommendations below, you should be able to obtain test results for different DNS resolvers that can be compared against one another with confidence:

1. Create one or more query files that you will use during the performance test. You may wish to generate one file for 0% cache hits (simulating server startup), one file matching your average cache hit rate and another file for 100% cache hits (so you can see if you can duplicate the vendor's performance claims). Make sure the query files have enough data in them so that the test can be completed before running out of queries.
2. Choose your domains carefully. To mimic the real world, use reasonable referrals (you can use a distribution of domain names across .com, .org, .net or any other top level domains frequently queried in your network) and reasonable domain name lengths (the median length of domain names underneath the gTLDs is eleven characters). The average name length used for cache hits and cache misses should be the same or else the test may favor one over the other.
3. Set up the authoritative server(s) to be authoritative for the domains being queried. In the simplest test, a single authoritative server can be configured to be authoritative for the root, top level domains and all lower level domains used by the queries.
4. Ensure the authoritative server is not a bottleneck in the test. The easiest way to do this is to run a 0% cache hit test to make sure the authoritative server keeps up with the resolver without dropping packets. Run the same test directly against the authoritative server to verify that it maintains the required rate without the cache server interceding.
5. Standardize DNS and system configurations across all platforms and tests. Make sure that you are using the same configuration files (or the equivalents, if you are testing different software products) and ensure that the same system services are running on each server during the test.
6. Make sure you have a fast enough switch. For each test run, ensure that you are measuring the performance of the resolver, not the query driver or switch or NIC. You can do this by checking the number of sent and unanswered packets reported by resperf versus the number of packets received and dropped on the resolver by checking the netstat "UDP packet receive errors" if it is running on Linux or the netstat "UDP packets dropped when queue was full" if it is a Secure64 server. If there were more unanswered packets reported by resperf than dropped on the target server, then the switch or other network hardware is impacting the test and the result does not reflect a valid performance limit.
7. Use two switches if possible. One switch may be sufficient, but we found that two switches were better able to isolate traffic to the authoritative servers from traffic from the client and more closely mimics how the resolver would be configured on a real network.
8. Use a powerful server to generate queries. In our own testing, we found that CPU is the most important criteria (we needed to utilize a 3 Ghz or faster Xeon server) and we needed a minimum of four cores to generate sufficient query volume.
9. Tune resperf to get reliable and accurate results. You should set the maximum query rate that is just a little bit higher than the maximum rate you expect in any test, and use that value across all tests. This value can affect the results reported by resperf, so it is important to keep it consistent.
10. Run resperf multiple times (3-7) and take the mode or median result, especially at high query rates. We found a 14% variance from one test to another using resperf at high query rates.
11. Verify that there are no errors in any of the query responses. Because this is a lab test, all queries should be answered with a NOERROR response code. The existence of any other response code (e.g., SERVFAIL, NXDOMAIN or REFUSED) is an indication that something is wrong with the test setup.

## Real world testing

Real world testing is important, even if it is more difficult to get repeatable results, because it tells you what you can expect in your own network. It also tests other capabilities of the resolver that may not be exercised in the

lab test, like its timeout and retry algorithms, it's ability to handle large packets through EDNS(0) and TCP fallback, and its ability to resolve domains even when authoritative servers are poorly configured or lame. Here are some recommendations to conduct a controlled, real world test:

1. Capture real world DNS traffic from your network for some reasonable period of time (make sure you have enough data to run the test – 15 minutes or so should suffice)
2. Configure all target resolvers identically. This includes configuration settings and the system services that are running.
3. Configure DNSSEC identically. This can be tricky, because different resolvers behave differently with regard to DNSSEC. If you want to run with a security-aware, validating resolver, make sure your trust anchor and your EDNS(0) buffer size are set properly. If you want to run a security-aware, non-validating resolver, make sure that the resolver is properly retrieving and caching DNSSEC signatures and DS records. This is an important step, as retrieving DNSSEC information can put a significant load on the resolver.
4. Use resperf to run each test, interleaving one server under test at a time if you are benchmarking multiple products (e.g., test A, B, C, A, B, C…). This is important to minimize external changes on the results.
5. For all tests, capture both the performance statistics and the statistics on response codes (how many NOERROR codes, how many SERVFAILs, etc.)
6. Make sure that the servers are all resolving approximately the same number of domains. If one product is getting notably more SERVFAILS or other RCODES than another, in the same scenario and timeframe, then it indicates something is inconsistent either in the DNS server configuration or in the test environment setup.
7. Use the mode or median result for each server, as resperf results may vary significantly from one run to the next. The most frequent value (the mode) is the most accurate measure, as the outliers indicate random variances in the environment.

# Conclusion

DNS performance is increasingly important in today's networks to ensure a good end user experience. However, there are large differences in the capacity of different DNS architectures, which can be disguised by the way in which vendor's report the performance of their own products. This makes it virtually impossible to predict the performance of a given DNS resolver without performing your own benchmarks.

The recommendations provided in this white paper have been informed by Secure64's own experience in measuring the performance of our own products as well as others, By following these recommendations, you should be able to measure and compare different products in a consistent and repeatable way. More importantly, you will know how the resolver will behave on your network, and can make more informed decisions regarding the architecture required to handle your network load, and the total cost of ownership of that solution.

## About Secure64

Headquartered in Greenwood Village, Colorado, Secure64 is a software company providing the most secure DNS products available to their customers in the government and telecommunications industry. Partially funded by a grant awarded by the Department of Homeland Security, Secure64's patented technology has been proven to be immune to compromise from rootkits and malware and resistant to network attacks that are the source of today's most serious security threats. The company offers a suite of trusted and secure DNS software appliances for caching, signing and authoritative use. Secure64's products are sold and serviced worldwide by both Hewlett-Packard and Secure64. Notable customers include the U.S. Departments of Commerce, Labor, and Interior as well as telecommunication giants Sprint, T-Mobile, Telefonica and CenturyLink. For more information, visit http://www.secure64.com.



For More Information:

(303) 242-5890
www.secure64.com

Secure64 Software Corporation
5600 South Quebec Street, Suite 320D
Greenwood Village, CO 80111