NGINX SSL Performance

NGINX is commonly used to terminate encrypted SSL and TLS connections on behalf of upstream web and application servers. SSL termination at the edge of an application reduces the load on internal servers, simplifies certificate management and reduces certificate costs. However, because it is extremely CPU-intensive, it can create a scalability bottleneck that may limit growth.

This paper investigates the performance of NGINX's SSL termination under a range of traffic types and ciphers. It seeks to establish a correlation between OpenSSL benchmarks and NGINX performance, to enable users to rapidly estimate the capacity of selected hardware or virtual machines.

Summary of Results

A single virtualized Intel core can typically perform up to 350 full 2048-bit SSL handshake operations per second, using modern cryptographic ciphers. This equates to several hundred new users of your service per second per core.

NGINX's SSL performance scales with the number of cores available on the host server, until other limits (typically bandwidth) are met, so an 8-core virtual machine could accept traffic from over 1,000 new users per second and still have resources to spare.

Older, less compute-intensive ciphers can give significantly better performance in some benchmarks, but the benefits are outweighed by the security improvements of modern SSL and TLS ciphers.

Interpreting the Results – some background on SSL

SSL connections are complex to analyze because there are many variables that affect performance – the server key size, the key exchange protocol, the bulk cipher and the quantity of data transferred down that SSL connection. Furthermore, the industry standards for SSL and TLS operations have changed in the last 3 years in response to developments in encryption-breaking technology and concerns about government snooping on traffic.

Server Key size – 1024 or 2048-bit?

An SSL connection will begin with an authentication step, where the server presents an identifying 'public certificate' and the client verifies the server owns the corresponding RSA private key.

In this step, the client encrypts some random data using the server's public key (in the public certificate) and the server then decrypts it using the private key. Both parties must agree on the value of the client's random data for the SSL handshake to proceed successfully.

What has changed? Until recently, 1024-bit RSA private keys were common, but industry standards have now fully migrated to 2048-bit keys. In general, operations using 2048-bit keys are 5 times slower than 1024-bit keys.

Key Agreement - RSA or Perfect Forward Secrecy

The client and server then need to negotiate a shared master secret that is used to derive the encryption key for the SSL connection. The results of the RSA operation used in the authentication step can be used to generate the shared secret, or an additional key-exchange step can be used.

What has changed? Many published benchmarks select an SSL cipher that uses the RSA operation in the authentication step to generate the shared secret. However, RSA-encrypted session keys can be decrypted if the RSA key is compromised in the future.

Modern SSL implementations favor 'Perfect Forward Secrecy' methods[1]. This involves an additional step in the SSL handshake where the server generates an ephemeral (temporary) key pair for the connection. A shared secret is negotiated securely using this ephemeral key, and the ephemeral key is then destroyed. However, improved security comes at the cost of greater computation on behalf of the server, and a corresponding performance impact.

[1] http://vincent.bernat.im/en/blog/2011-ssl-perfect-forward-secrecy.html

Bulk Ciphers – RC4 or AES?

Once the shared secret is determined, both parties must negotiate a stream or block cipher (to encrypt the data) and signing method (to generate message authentication codes) for data transmission. The shared secret is used to derive the encryption and signing keys.

What has changed? The RC4 bulk cipher is now regarded as weak and now the slower AES128 cipher is preferred. MD5 and SHA-1 signing methods have now been replaced with SHA-256 or other more secure but more expensive signing methods.

SSL Session Reuse

The expensive authentication and key negotiation operations in an SSL handshake do not need be performed for every HTTP request. Clients are extremely efficient at using single HTTPS connections for multiple GETs and reusing SSL session credentials across multiple SSL connections.

The number of RSA operations per second that a server can perform provides an upper limit on the number of new clients per second, not a limit on the number of individual requests per second the server can perform.

Summary

It is very difficult to compare SSL performance benchmarks from different sources or vendors without knowing the precise details of the SSL ciphers used. In this study, we will consider two cipher combinations:

- **TLS_RSA_WITH_RC4_128_SHA** illustrates ‘legacy performance’;
- **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256** gives a more representative measure of contemporary performance.

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (or a related cipher) is necessary to score an ‘A’ on Qualys’ SSL Server Test.[2]

It’s even more difficult to infer real-world performance from benchmark results. The aim of this study is to help you make some good judgements, but it’s no substitute for evaluating performance on your own hardware, with your own content, applications and users.

[2] https://www.ssllabs.com/ssltest/

System Under Test

NGINX 1.7.2 was tested on Ubuntu 14.04 on a range of virtual machine sizes from DigitalOcean. No operating system or NGINX tuning was applied, other than to increase NGINX's `worker_processes` to match the number of cores in each virtual machine.

	CPU cores	Network Throughput (MBs)	2048 RSA ops per second	RC4 throughput (MBs)	AES throughput (MBs)
Server-1	1	112	619.5	516	175
Server-2	2	111	1120	1101	406
Server-3	4	111	2380.4	2352	861
Server-4	8	110	4404.8	4300	1575

Network throughput was measured using `ab` (apachebench) to transfer large files from client to server using HTTP. 111 MBs application-layer throughput is consistent with a 1 GbE network, and this should be regarded as the peak possible application network performance.

Cryptographic speed tests were taken using `openssl speed`. This is an easy-to-run process, and this study investigates the correlations between `openssl speed` results and the benchmark results:

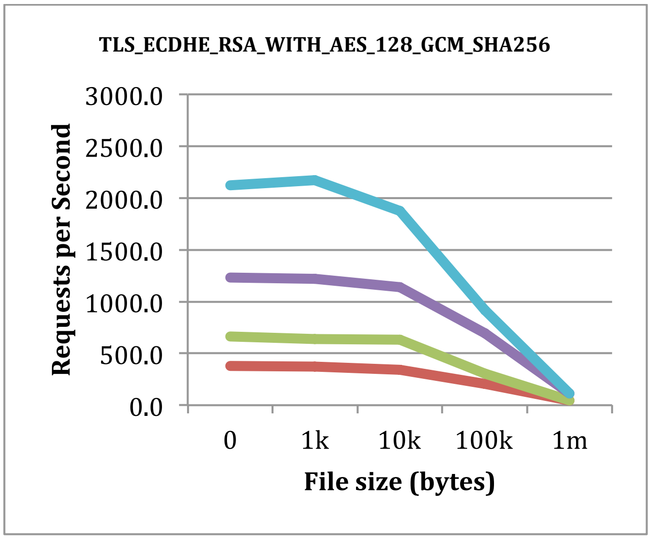
- **RSA operations per second** measured using `openssl speed rsa` (signatures per second with 2048-bit keys).

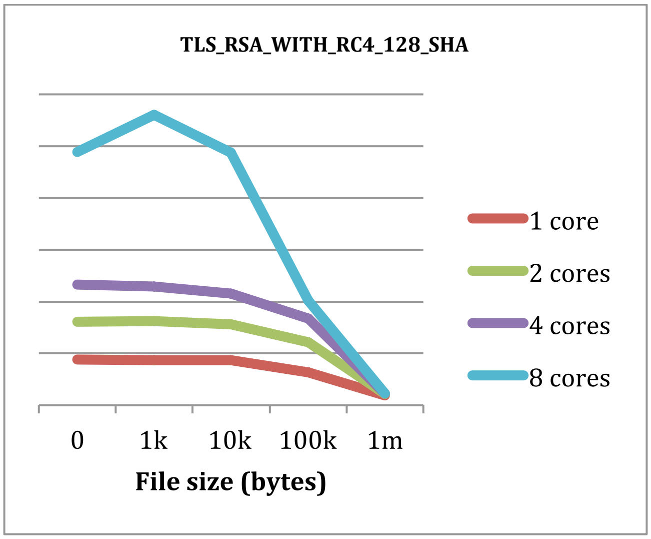
- **RC4 throughput measured** using `openssl speed rc4` (bulk encryption with 1024-byte block size).

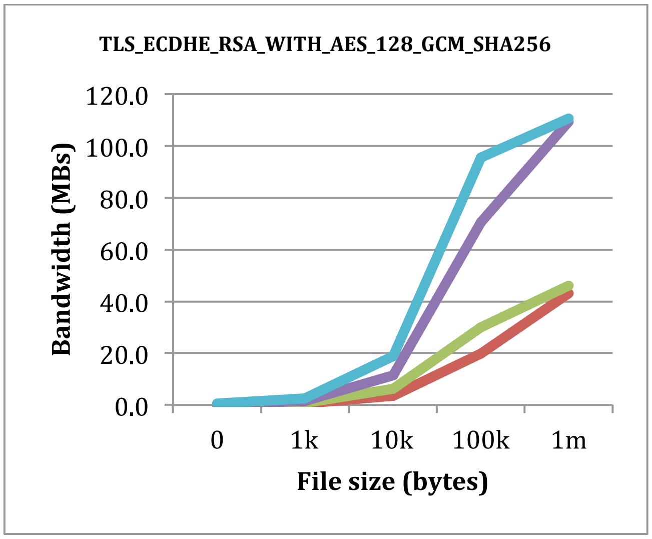
- **AES throughput measured** using `openssl speed aes`, (128-bit key, 1024-byte block sizes).

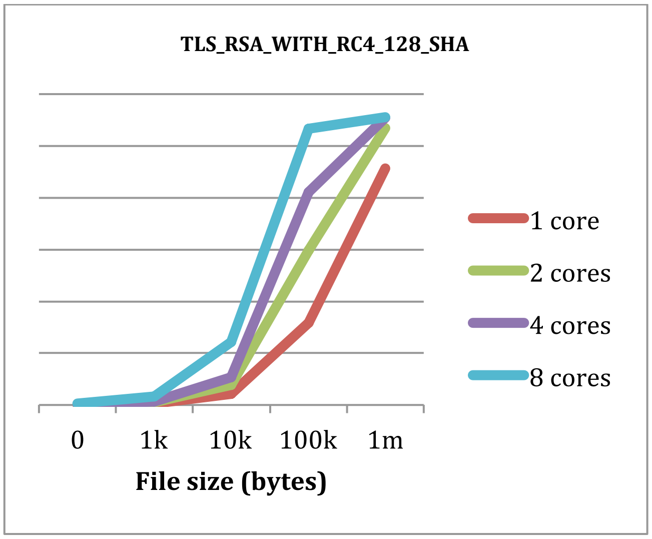
OpenSSL speed results are multiplied by the number of cores in the server to get total server capacity.

Results – Requests per second and Bandwidth









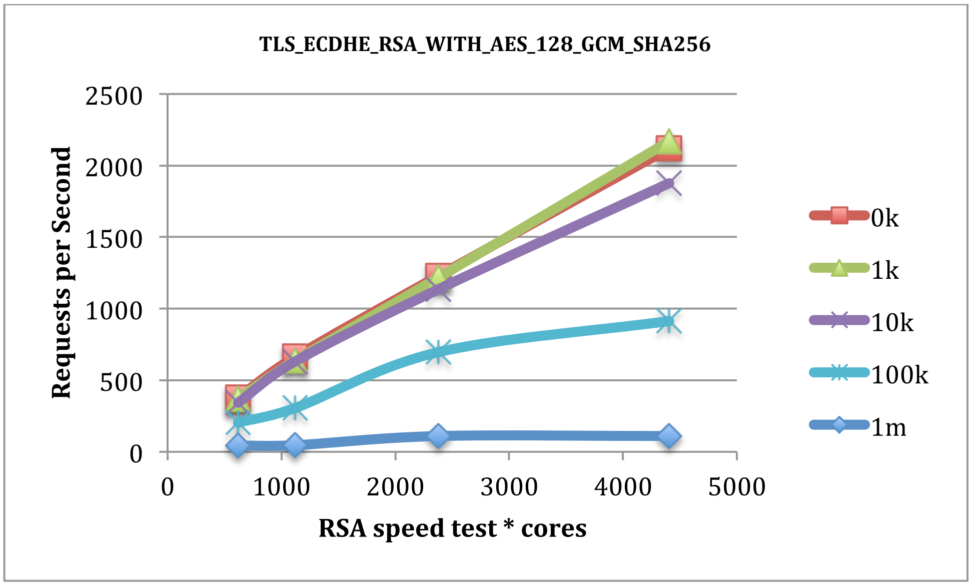
For small requests, where the performance is dominated by the handshake, the additional ECDHA key exchange step reduces the performance of the ECDHA-RSA cipher to approximately 85% the performance of the simple RSA cipher.

For larger requests, where the performance is dominated by the bulk cipher, the AES-based ciphers are approximately 35% the performance of the RC4 ciphers. Where performance is dominated by available bandwidth, both ciphers are then limited by network bandwidth.

Correlating Performance with other measurements

Performing full benchmark tests is complex and error prone. It's useful to determine if there is a strong correlation between easy-to-measure performance metrics (e.g. openssl speed) and SSL performance.

Estimating Requests per Second

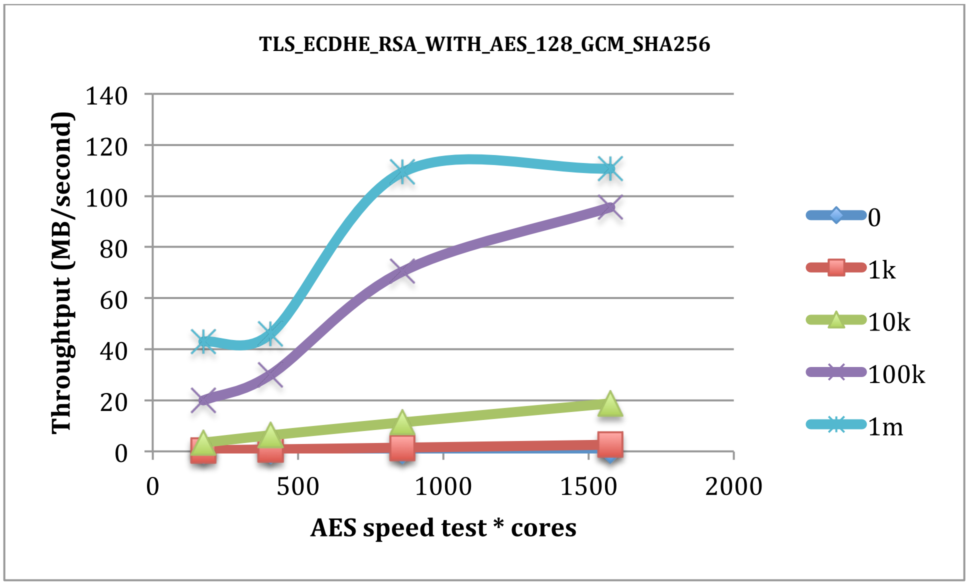


The strongest correlation for Requests per Second for small files is against the RSA speed test:

Number of cores	RSA speed	0 bytes RPS	Ratio	1k RPS	Ratio
1 core	619.5	379.7	0.61	371.4	0.60
2 cores	1120	663.2	0.59	636.7	0.57
4 cores	2380.4	1230.8	0.52	1218.5	0.51
8 cores	4404.8	2120.5	0.48	2169.4	0.49

… giving a rough approximation that requests per second for small files is between 50% and 60% of the RSA speed, tailing off for more powerful machines where other factors (e.g. interrupt handing) begin to affect performance.

Estimating Bandwidth - AES

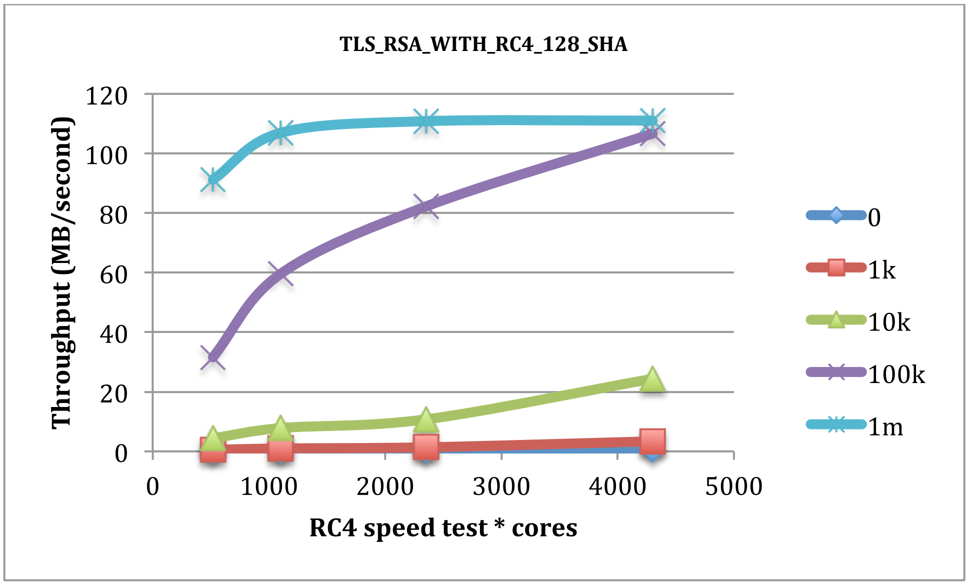


Determining a correlation between bandwidth and AES speed is more challenging because the test network was only 1GbE-capable (approx. 111MB/s effective throughput), and because bulk ciphers such as AES are relatively efficient; the performance of the cipher does not have a significant bearing on the performance of the system.

Number of cores	AES speed (MB/s)	Throughput (100K files)	Ratio	Throughput (1M files)	Ratio
1 core	175	19.9	0.11	43.0	0.25
2 cores	406	30.0	0.07	46.1	0.11
4 cores	861	70.5	0.08	109.3	0.13
8 cores	1575	95.6	0.06	110.7	0.07

The 'sweet spot' is clearly on low-powered machines (where bandwidth limits cannot be met), with very large files (where the very-expensive RSA operation has less effect). We saw a peak throughput of 25% the theoretical AES speed, but overall throughput limits quickly dominated. Because the cipher is relatively lightweight, there is not a strong correlation between theoretical speed and actual speed.

Estimating Bandwidth – RC4



Note that even the low-powered machines in the test quickly saturated the 1 GbE network when using the RC4 bulk cipher.

Number of cores	RC4 speed (MB/s)	Throughput (100K files)	Ratio	Throughput (1M files)	Ratio
1 core	516	31.6	0.06	91.2	0.18
2 cores	1101	59.6	0.05	106.9	0.10
4 cores	2352	82.2	0.03	110.8	0.05
8 cores	4300	106.7	0.02	111.0	0.03

AES Acceleration with AES-NI

AES-NI is a set of instructions on modern Intel processors that accelerates the encryption speed of the AES algorithm.

The virtual machines used in these tests were not AES-NI-capable. Informal reports indicate that AES-NI is between 4-8x the performance of AES, so one would expect that the performance difference against RC4 would be significantly reduced or eliminated.

Conclusions

NGINX software can handle large volumes of SSL traffic on a modern 8-core or beyond server. Scalability is very cost-effective; NGINX leverages general-purpose physical or virtual hardware and SSL connections-per-second scales linearly with the number of cores, up to other limitations in the hardware or operating system.

A single virtualized Intel core can typically perform up to 350 full 2048-bit SSL handshake operations per second; this equates to several hundred new users of your service per second per core. NGINX's SSL performance scales with the number of cores available on the host server, until other limits (typically bandwidth) are met, so an 8-core virtual machine could accept traffic from over 1,000 new users per second and still have resources to spare.

For other hardware, you can use openssl tests to determine approximately what the SSL capacity could be.

Specialized hardware devices exist that can perform many thousands of RSA operations per second. You should consider whether the cost of these devices (acquisition, support, upgrades) is merited given the traffic levels you need to terminate, and the corresponding cost of an NGINX-based solution.