Architectural Impact of Secure Socket Layer on Internet Servers: A Retrospect

Krishna Kant and Ravishankar Iyer
Intel Corporation
Hillsboro, Oregon

Prasant Mohapatra
University of California
Davis, CA

***Abstract*—Secure socket layer (SSL) is the most popular protocol used in the Internet for facilitating secure communications. In this retrospective, we summarize our original paper which analyzed the performance and architectural impact of SSL on the servers and provided insights into the functioning and acceleration of SSL. In addition, we describe advancements in the area that have occurred on this topic and also discuss future research opportunities**

I. Introduction

The SSL protocol has two important functions: (a) authentication of the server and client at the beginning of the session, and (b) encrypted data exchange between the two parties during the session. The authentication is performed via the SSL handshake protocol, which involves 3 phases, including parameter negotiation, mutual authentication and secret key exchange. The second phase uses a 128-bit session key for encryption and appends each message with message authentication code (MAC) computed using the negotiated secure hash function (e.g., SHA256) over the entire message.

SSL cost is dominated by the first phase which uses public key encryption. RSA is by far the most commonly used public key encryption algorithm in practice and belongs to the class of exponentiation ciphers [3]. Let (e, d) denote the encryption/decryption key-pair, and $N = pq$ where p and q are large primes. Then, encrypted message C and plain-text message M are related as follows:

$$C = M^e \text{mod} N, \qquad M = C^d \text{mod} N \tag{1}$$

The algorithm to efficiently evaluate equation (1) involves unsigned arithmetic operations (multiplication, addition, shift and rotate) on large integers. Thus, efficient coding of exponentiation and availability of special hardware features to expedite exponentiation can have a tremendous influence on handshake performance.

Private (or symmetric) key algorithms such as RC4/6, AES, IDEA, etc. are typically used for bulk data encryption. By design, these algorithms are highly sequential in nature and use many rounds of computations. Consequently, bulk-data encryption is not only computationally intensive but also does not have much inherent parallelism that can be exploited by modern superscalar or vector-oriented (e.g., MMX-like) architectures.

In our paper [1], we did an experimental study of the SSL performance and its architectural impact using Intel Pentium III Xeon based servers. The configurations analyzed varied the following parameters: (a) number of processors in the SMP server (1P, 2P, 4P), (b) three different L2 cache sizes (512 KB, 1 MB and 2 MB), and (c) three different transfer sizes. The study exposes various architectural features that impact the performance of SSL secured communications.

II. SSL Resource Requirements

Because of large computational cost of SSL handshake, an important parameter is the amount of bulk data transfer per handshake. We considered 30 B, 1 MB and 36KB values in the experiments. The results, [1], show that SSL can increase path length 10-15 fold over the non-SSL case, however, the CPI (cycles per instruction) drops by more than a factor of 2, thereby resulting in a net increase in processing cost of only about 5-7 fold.

The small CPI for SSL is indicative of primarily computational type of workload. It is found that while L1 instruction miss ratios remain very low in all cases, the L1 data miss ratios are significant. Furthermore, the miss ratio increases slightly with number of processors because of larger data footprint and coherency misses.

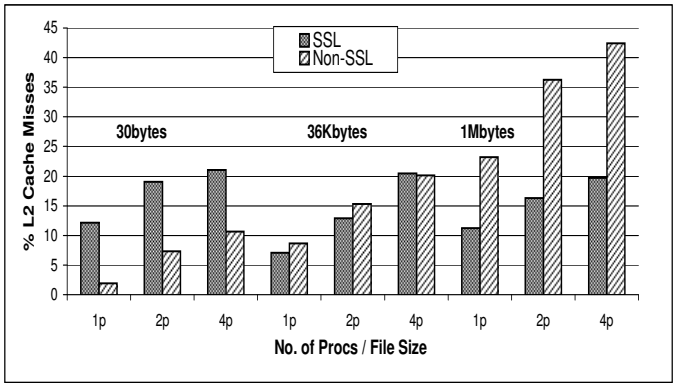

Fig. 1. L2 miss ratio vs. transfer size (2 MB L2)

Figure 1 shows the scaling of L2 miss ratios as a function of number of processors and transfer-size for the 2MB L2 cache size. It may be noted that L2 miss ratios are rather high without SSL because of high degree of locking/contention in TCP processing and increasing cache pollution due to large transfer sizes. With SSL, the bulk data encrption/decryption helps bring down the misses due to its sequentiality, but also causes cache pollution and substantial increase in hit-modified (HITM) cache lines. In view of this, a larger L2

cache should help so long as the performance is not I/O latency limited. More important, an architecture that reduces L2 cache pollution should result in a substantial performance improvement.

Some of the current processors already contain features that could be exploited for superior performance in secured communications. In the longer run, processors could provide special instructions that speeds up encryption/decryption. A factor of 2 reduction in CPI under SSL (along with a large increase in path-length) is a clear indication that SSL workload is primarily computational and could be speeded up via special prefetching and computational instructions.

We also examined the overall response time with and without SSL. The most significant result here is that the response time increase by a factor of about 10 under SSL! If we were to look at the time to last byte for a 1 MB file size, we find that a response time of 1-2 secs without SSL (which is quite tolerable) increases to 10-15 secs with SSL (which is large enough to result in significant abandonment and retries). It may be noted that the response time is significantly higher for 30B case because the handshake involves about 4 round-trip delays between the server and the client. With a large file transfer also involved, the competing connection setups go down drastically and the TCP data transfer becomes much more efficient (by virtue of slow-start mechanism). This results in a significant drop in queuing delays and hence a drop in first-byte response times. A consequence of these observations is that the client response time can be improved by reducing the frequency of SSL handshakes.

III. Advances and Future Opportunities

The original paper (as summarized above) presented an experimental analysis of the impact of SSL transactions on Internet server architecture. In particular, in addition to quantifying the overall drop in throughput due to SSL, the analysis also supports the following observations:

- SSL workload is processor bound and hence more powerful processors are very helpful in improving the performance.
- A processor with high pipeline depth can improve the performance of SSL transactions, whereas the increase in the issue width may not, especially in cases where the performance is dominated by bulk data encryption. This behavior results from the high sequentiality and low control dependencies in SSL code.
- Increasing the size of L1 cache should have a positive impact on the SSL performance.
- Increasing the size of L2 caches has only a small impact on SSL performance.
- SSL handshake and encryption/decryption of large webpages has very good scaling with respect to the number of processors in a SMP environment, which may promote the use of 4-way or 8-ways systems for these applications.

IV. Retrospect

Since the paper, there have been some notable advances in the area of SSL acceleration. For example, Intel processors now have ISA support for AES [5] that can be used to accelerate the performance of an AES implementation by 3 to 10X over a completely software implementation. With appropriate hardware acceleration, it is now possible to perform bulk data encryption at line rate, even at 100 Gb/sec. The SSL handshake operations can also be accelerated substantially using features such as long word arithmetic, special instructions and multicore architectures. For example, Intel has introduced a communications platform with technology to accelerate network processing including cryptography, compression and deep packet inspection [6].

With these developments, SSL processing is today far less of an issue than when the paper was written. Yet, as the technology improves, so do the key lengths, which keeps the acceleration issue relevant particularly because of ongoing movement to mobile devices with constrained computing power. Also, SSL is only one instance where all 3 basic cryptographic operations – namely public key encryption, symmetric key encryption, and secure hashing – are involved. With security concerns becoming central to the Internet economy, cryptography and cryptographic acceleration is becoming necessary in most data communications and storage instances. In particular, the rise of cloud computing and the resulting concerns about integrity and confidentiality of cloud-stored data, the need for authentication and encryption continues to expand. Furthermore, the need to operate on encrypted data has given rise to special forms of encryption such as homomorphic encryption, and hence the need for accelerating these operations as well.

References

[1] K. Kant, R. Iyer, and P. Mohapatra, "Architectural Impact of Secure Socket Layer on Internet Servers," Int. Conf. on Computer Design, pp. 7-14, 2000.

[2] D. Dunning, G. Regnier, et. al., "The virtual interface architecture: a protected, zero copy user-level interface to networks," IEEE Micro, March 1998, pp. 66-76.

[3] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Second Ed, Prentice Hall, 1999.

[4] T. Wilson, "E-Biz bucks lost under SSL strain", Internet Week online, May 20, 1999, at www.internetwk.com/lead/lead052099.

[5] White Paper on AES-NI - http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-aes-instructions-set

[6] Crystal-Forest: Intel's next generation communications platform, http://newsroom.intel.com/community/ intel_newsroom/blog/2012/02/14/