インテル® Xeon® プロセッサー E5 v3 における AES-GCM 暗号化のパフォーマンス

この記事は、インテル® デベロッパー・ゾーンに公開されている「AES-GCM Encryption Performance on Intel® Xeon® E5 v3 Processors」の日本語参考訳です。

このケーススタディーでは、AES ブロック暗号の GCM (Galois/Counter Mode、ガロア/カウンターモード) のパフォーマンスを向上するため、インテル® Xeon® プロセッサー E5 v3 ファミリーに対して行われたアーキテクチャーの改良を検証します。具体的には、OpenSSL* SSL/TLS ライブラリーを利用して、Nginx* Web サーバーにおけるこれらの改良の影響を検証します。この新しい世代のインテル® Xeon® プロセッサーで、CBC モードの AES と HMAC+SHA1 ダイジェストから AES-GCM へ切り替えることにより、Web サーバーの最大スループットは大幅に向上します。

はじめに

このケーススタディーの目標は、SSL Web サーバーのパフォーマンスについて、インテル® Xeon® プロセッサー v3 ファミリーで行われたマイクロアーキテクチャーの改良の影響を検証することです。暗号化パフォーマンスに関連する 2 つの重要な拡張は、インテル® AES New Instructions (インテル® AES-NI) 命令と PCLMULQDQ 命令のレイテンシーの軽減でした。これらの変更は、一般に AES-GCM と呼ばれる、AES のガロア/カウンターモードのパフォーマンスを向上することを目的に行われたものです。

AES-GCM の重要な機能の 1 つは、メッセージ認証に使用されるガロア域における乗法をブロック暗号化と並列に計算できることです。CBC (Cipher Block Chaining、暗号ブロック連鎖) モードのような、AES の連鎖モードで可能なレベルよりも、はるかに高いレベルの並列化が可能です。AES-CBC と HMAC+SHA1 ダイジェストに対する AES-GCM のパフォーマンス・ゲインは、インテル® Xeon® プロセッサー v2 ファミリーのような古い世代の CPU でも大幅なものでしたが、インテル® Xeon® プロセッサー v3 ファミリーに対するアーキテクチャーの改良により、パフォーマンス・ギャップはさらに広がりました。

図 1 は、インテル® Xeon® プロセッサー E5 v2 システムおよびインテル® Xeon® プロセッサー E5 v3 システムの両方で、128-gcm EVP を選択して OpenSSL* スピードテストを行った場合の aes-128-cbc-hmac-sha1 に対するスループット・ゲインを示しています。データのテストに使用したハードウェアおよびソフトウェア構成を表 1 に示します。表 1 から、インテル® Xeon® プロセッサー E5 v2 で AES-GCM は AES-CBC と HMAC+SHA1 の約 2.5 倍、インテル® Xeon® プロセッサー E5 v3 で約 4.5 倍になっていることが分かります。GCM と CBC 間のパフォーマンス・ギャップは、インテル® Xeon® プロセッサー E5 v2 から v3 で約 2 倍です。

	E5 v2 System	*E5 v3 System*
Processor	Xeon E5-2697 v2	Xeon E5-2697 v3
Turbo	Off	Off
C States	Off	Off
Clock Speed	2.7 GHz	2.6 GHz
Memory	32 GB	64 GB
OS	Ubuntu 13.10	Ubuntu 13.10

表 1. OpenSSL* スピードテストのハードウェアおよびソフトウェア構成

この OpenSSL* 本来の性能が SSL Web サーバーのスループットにどのように影響するか評価するため、このケーススタディーでは、これらの 2 つの暗号を使用したときに Nginx* Web サーバーで達成可能な最大スループットを調べました。

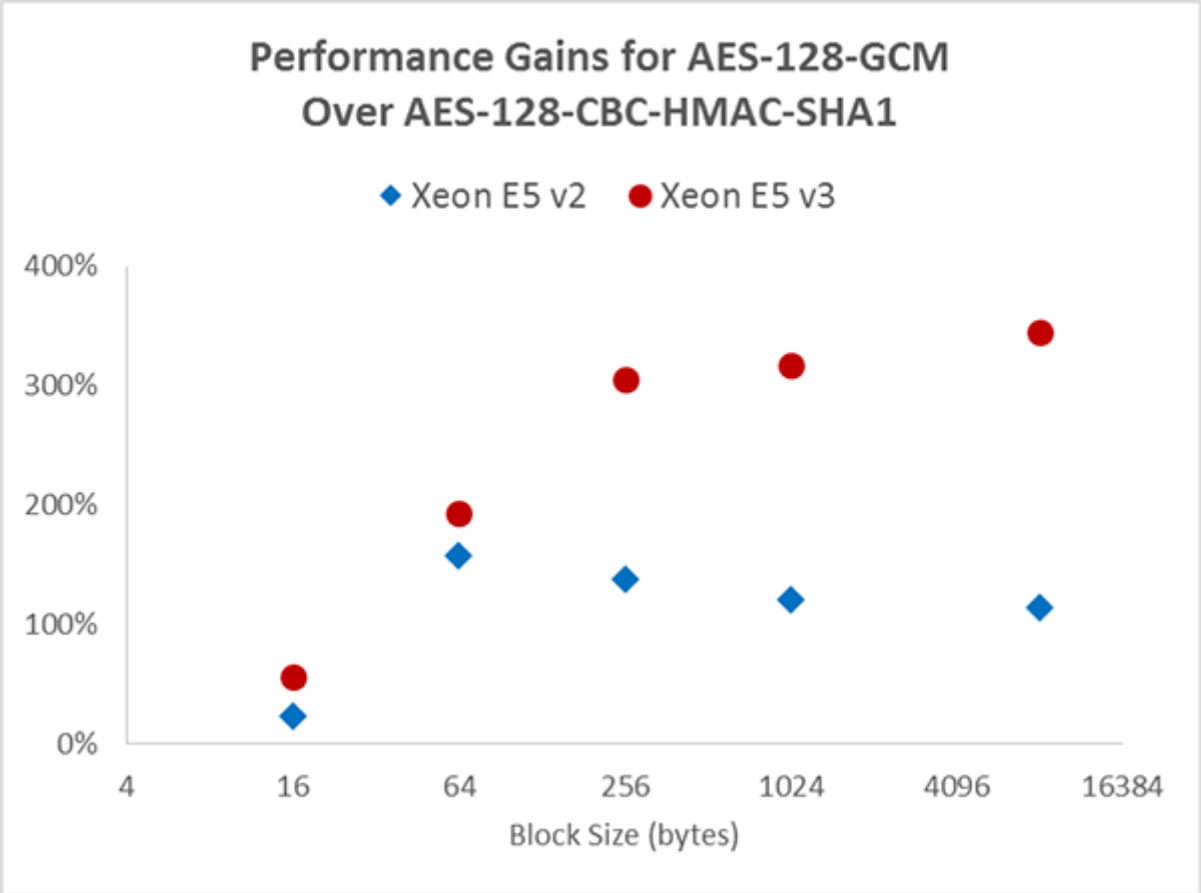

図 1. インテル® Xeon® プロセッサー E5 v2/v3 プロセッサーにおける aes-128-gcm と aes-128-cbc-hamc-sha1 EVP の OpenSSL* 1.0.2a スピードテストの相対的な結果

テスト環境

多くの並列接続リクエストを生成して、合計 2 分間でそれらの接続をできるだけ速く繰り返すことにより、2 つの暗号について Nginx* の性能限界をテストしました。この 2 分間の最後に、すべてのリクエスト中の最大レイテンシーがスループットとともに検証されました。2 秒を超える接続レイテンシーがない期間について Nginx* が達成可能な最大スループットを見つけるため、同時接続の数を実行間で調整しました。このレイテンシー限界は、小さな Web ページをロードする際に許容可能な遅延は 2 秒であると結論を下した、研究論文「A Study on tolerable waiting time: how long are Web users willing to wait?」(英語) の数値に基づいています。

Nginx* は、プリプロダクションの、2 ソケット インテル® Xeon® プロセッサー・サーバー・システム (2 つのプロダクション インテル® Xeon® プロセッサー E5-2697 v3 @ 2.60GH を搭載、ターボオン、ハイパースレッディング・オフ) にインストールしました。使用したオペレーティング・システムは Ubuntu* Server 13.10 です。インテル® Xeon® プロセッサー E5 のコア数は 14 コアで、ハードウェア・スレッドの数は合計 28 でした。システム RAM は合計 64GB でした。

Nginx* の SSL 機能は OpenSSL* ライブラリーにより提供されました。OpenSSL* は、汎用暗号関数に加えて SSL および TLS プロトコルを実装するオープンソース・ライブラリーです。1.0.2 ブランチはインテル® Xeon® プロセッサー v3 向けに最適化されています。OpenSSL* の詳細は、http://www.openssl.org/ (英語) を参照してください。このケーススタディーのテストを実行する時点ではプロダクション・リリースが登場していなかったため、これらのテストは 1.0.2-beta3 を使用して行われました。

サーバーロードは、必要に応じて、インテル® Xeon® プロセッサー E5 およびインテル® Xeon® プロセッサー E5 v2 クラスのハードウェアで構成された、最大 6 つのクライアント・システムにより生成されました。各システムと Nginx* サーバーは、複数の 10G ビット・ダイレクト・コネクト・リンクにより接続されました。サーバーには、2 つの 4x10G ビット・ネットワーク・カードと 2 つの 2x10G ビット・ネットワーク・カードが搭載されていました。2 つのクライアントには 4x10G ビット・カード、残りの 4 つにはシングル 10G ビット NIC が搭載されていました。

テスト環境のネットワーク・ダイアグラムを図 2 に示します。

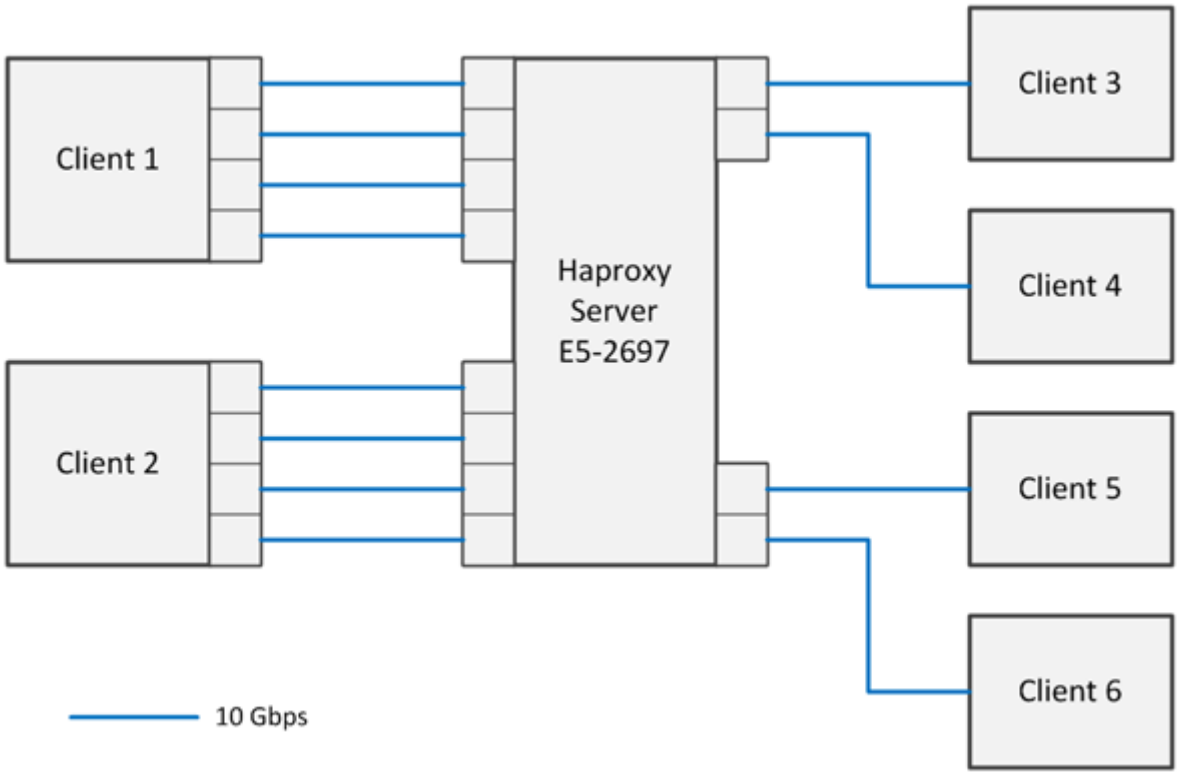

図 2. テストのネットワーク・ダイアグラム

実際のサーバーロードは、Apache* Benchmark ツール *ab* (Apache* サーバー・ディストリビューションに含まれているオープンソース・ユーティリティー) の複数のインスタンスを使用して生成されました。Apache* Benchmark の 1 つのインスタンスではサーバーの制限に達する十分なロードを作成できなかったため、複数のプロセッサー、およびクライアント CPU の要求により、複数のホストに分割する必要がありました。

しかし、各 Apache* Benchmark インスタンスは完全に自己完結型であるため、分散実行用のビルトイン・メカニズムはありませんでした。ロード・クライアント、CPU、ネットワーク・インターフェイスにわたる *ab* の複数のインスタンスの起動を調整して、結果を照合するために、同期サーバーおよびクライアント・ラッパーを記述しました。

テストプラン

テストの目標は、ターゲットファイルに対して 2 分間繰り返される着信接続リクエストを Nginx* が保持できる最大スループットを調べて、インテル® Xeon® プロセッサー E5-2697 v3 プラットフォームで AES128-SHA 暗号と AES128-GCM-SHA256 暗号の結果を比較することです。GCM 暗号スイートでは、_SHA サフィックスは暗号の擬似乱数関数アルゴリズムとして使用される SHA ハッシュ関数 (この場合は SHA-256) を指すことに注意してください。

OpenSSL Cipher	*RFC Cipher Suite*	*Cipher*	*Auth*
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA	AES_128_CBC	SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	AES_128_GCM	

表 2. 選択した TLS 暗号

固定のターゲット・ファイル・サイズについて、各テストが繰り返されました。サイズは 1MB で開始し、4GB まで 4 の累乗で増やしました (1GB = 1024MB、1MB = 1024KB、1KB = 1024 バイト)。1MB 以上のファイルを使用することで、セッション・スループットにおける鍵交換の影響は最小化されました。各接続で単一のファイルをフェッチするように、キープアライブ機能は無効にされました。

次のハードウェア構成で、各暗号のテストを実行しました。

- 2 コア有効 (ソケットあたり 1 コア)
- 4 コア有効 (ソケットあたり 2 コア)
- 8 コア有効 (ソケットあたり 4 コア)
- 16 コア有効 (ソケットあたり 8 コア)

ハイパースレッディングはすべての設定で無効にしました。ソケットあたり 1 つのアクティブ・コア (テストシステムの最小構成) にシステムを減らすと、低コア数のシステムが効率的にシミュレーションされ、Nginx* のパフォーマンスがほかのシステムリソースではなく CPU によって制限されるようになります。これらの測定は、コアあたりの全体的なパフォーマンスの予測に加えて、多くのコアを備えたシステムの推定パフォーマンスの予測にも使用できます。

メニーコアでの実行は、システムのスケーラビリティーをテストします。また、CPU 使用率を超えるシステムリソース制限の可能性についても調べます。

システム構成およびチューニング

Nginx* は、システムの各物理スレッドにつき 1 つのワーカーで、マルチプロセッサー・モードで動作するように構成されました。

設定ファイル nginx.conf からの抜粋を図 3 に示します。

```
worker_processes 16; # Adjust this to match the core count

events {
    worker_connections 8192;
    multi_accept on;
}
```

図 3. nginx.conf から抜粋

より小さなターゲット・ファイル・サイズで発生する可能性のある大量の同時接続をサポートするには、システムおよびカーネルのチューニングが必要でした。最初に、/etc/security/limits.conf のファイル・ディスクリプターの数を増やしました。

```
*               soft    nofile          150000
*               hard    nofile          180000
```

図 4. /etc/security/limits.conf から抜粋

次に、いくつかのカーネル・パラメーターを調整しました (これらの設定のいくつかは、大量の暗号化により適しています)。

```
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_fin_timeout = 30

# Increase system IP port limits to allow for more connections

net.ipv4.ip_local_port_range = 2000 65535
net.ipv4.tcp_window_scaling = 1

# number of packets to keep in backlog before the kernel starts
# dropping them
net.ipv4.tcp_max_syn_backlog = 3240000

# increase socket listen backlog
net.ipv4.tcp_max_tw_buckets = 1440000

# Increase TCP buffer sizes
net.core.rmem_default = 8388608
net.core.wmem_default = 8388608
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_mem = 16777216 16777216 16777216
net.ipv4.tcp_rmem = 16777216 16777216 16777216
net.ipv4.tcp_wmem = 16777216 16777216 16777216
```

図 5. /etc/sysctl.conf から抜粋

これらのパラメーターのいくつかは非常にアグレッシブですが、ここでの仮定は、このシステムは専用 SSL/TLS Web サーバーであることです。

Ubuntu* Server 13.10 に対するほかの調整は行われませんでした。

結果

ファイルサイズ別に、各暗号で達成した最大スループット (Gbps) を図 6 に示します。最小のファイルサイズ (1MB) では、SSL ハンドシェークがトランザクションの多くを占めているため GCM と CBC 暗号間の違いはわずかですが、より大きなファイルサイズでは、GCM 暗号の最大スループットは CBC 暗号の 2 倍から 2.4 倍になっています。本来の GCM の性能は約 8 Gbps/コアです。最大スループットが CPU による制限ではなくなる 8 コアまでは、これに該当しています。この 8 コアのケースは、ほかのシステム制限が Web サーバーでより高い転送レートを達成する妨げになるポイントであり、16 コアのケースでは、その傾向がさらに顕著になっています。ここで、CBC 暗号には大きな恩恵があることが分かりますが、どちらの暗号もスループットはあまり増加していません。

各ケースで 2 分間の実行中に Nginx* の最大 CPU 使用率をプロットした図 7 では、この傾向がより明確に示されています。2 コアおよび 4 コアのケースでは、どちらの暗号の CPU 使用率も 90% 台後半ですが、8 コアのケースでは CPU 使用率が 80% (大きなファイル) から 98% (小さなファイル) まで変動しています。

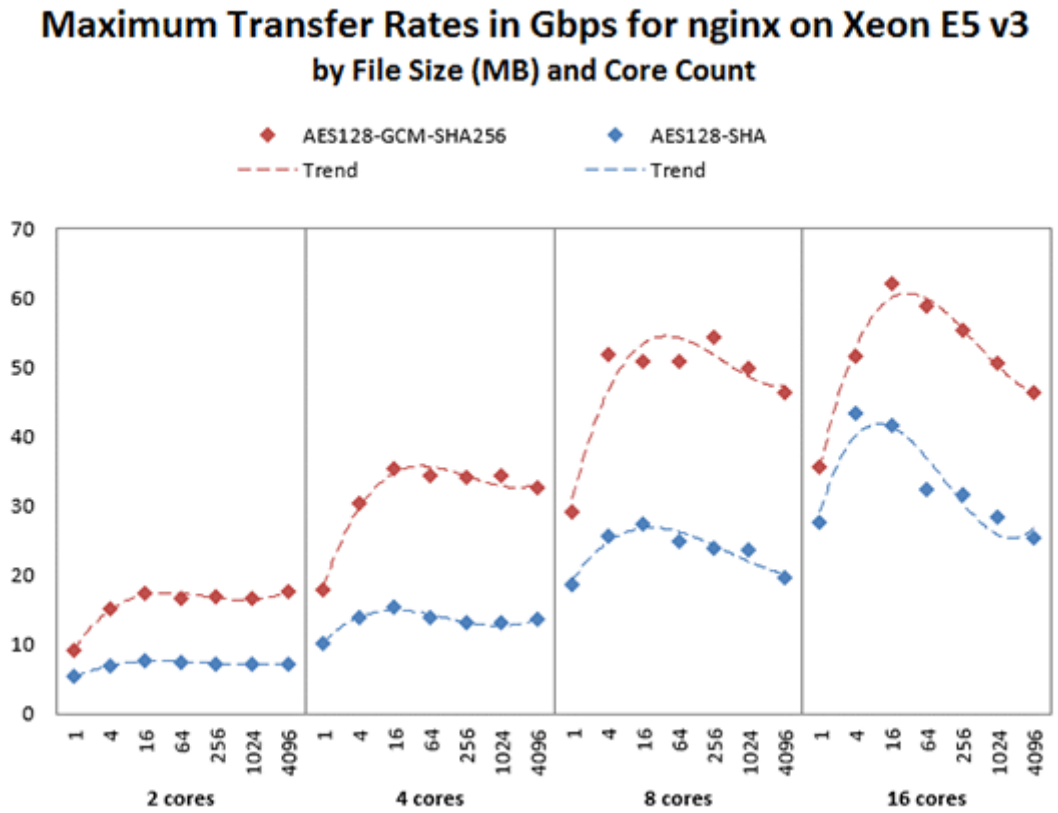

図 6. 指定されたコア数におけるファイルサイズごとの最大の Nginx* スループット

システムリソースの制限が顕著になる 16 コアのケースでは、暗号のパフォーマンスが大きく異なっています。ここで、合計のスループットは 8 コアのケースからはインクリメンタルに増加しています。つまり、8 コアのケース以降は追加のコアを活用できていないことは明白です。GCM 暗号は利用可能な CPU の 50% から 70% しか使用していません。GCM 暗号が、大幅に少ない計算能力で (CBC 暗号よりも多くのスループットを提供して) より多くのことを行っていることも分かります。

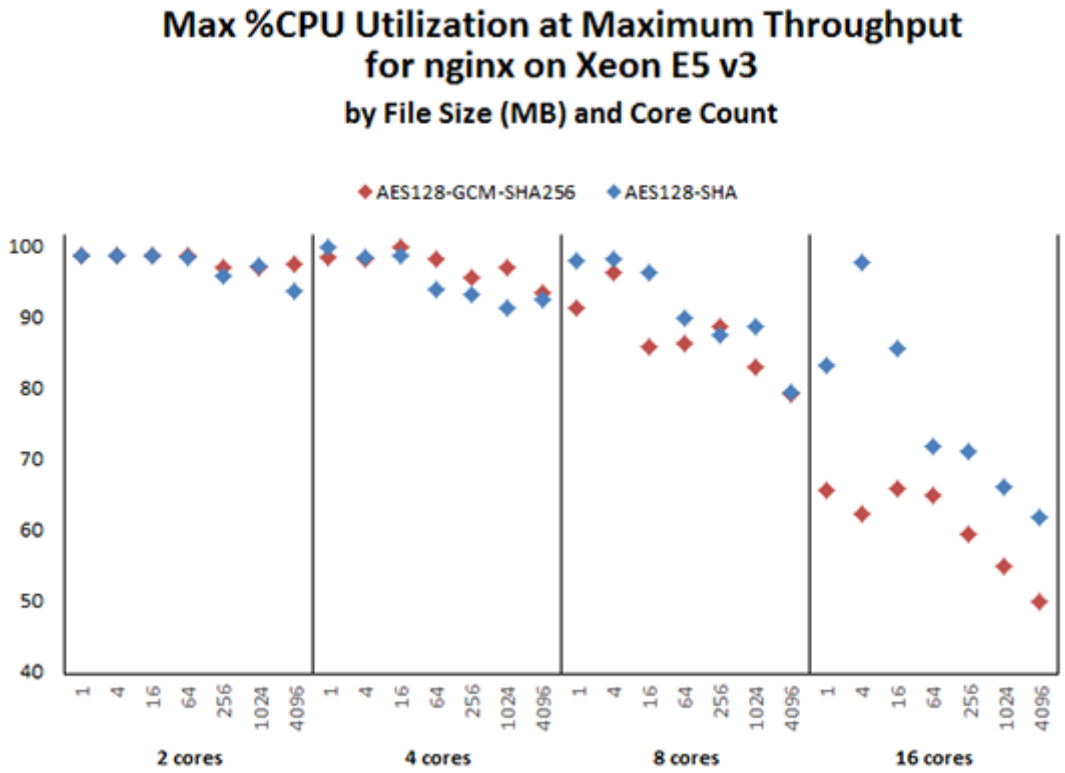

図 7. 最大の Nginx* スループットにおける最大の CPU 使用率

まとめ

インテル® Xeon® プロセッサー v3 ファミリーのアーキテクチャーの変更は、AES-GCM 暗号のパフォーマンスに大きな影響を与え、SSL/TLS Web サーバーで AES-CBC と HMAC+SHA1 ダイジェストよりも AES-GCM を選択する理由を提示しました。

ロー OpenSSL* スピードテストでは、GCM と CBC 間のパフォーマンス・ギャップはインテル® Xeon® プロセッサー E5 v2 ファミリーのほぼ 2 倍になりました。Web サーバーテストでは、AES-GCM 暗号を使用すると AES-CBC 暗号のスループットの約 2 倍から 2.4 倍になり、絶対データレートは約 8 Gbps/コアになりました。メニーコア構成は、システムの制限に達する前に 50Gbps を超える合計データ転送レートを達成することができます。このレベルのスループットは、最小限のチューニングを行った既製の Linux* システムで達成されました。

AES-GCM を活用できない大量のクライアントのために AES-CBC と HMAC+SHA1 ダイジェストのサポートは続ける必要がありますが、インテル® Xeon® プロセッサー v3 ファミリーで実行する Web サーバーでは、この暗号が提供する最良のパフォーマンスを引き出すために、AES-GCM を有効にすべきです。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。