Aceleración de SSL con GPU

Morganella Julian A., Pastor H. Leonardo., Ezequiel Silvero, Ezequiel Valdez.

1Universidad Nacional de La Matanza,

Departamento de Ingeniería e Investigaciones Tecnológicas,

Florencio Varela 1903 - San Justo, Argentina

[julianmorganella@hotmail.com](mailto:julianmorganella@hotmail.com), [leonardopastor1984@gmail.com](mailto:leonardopastor1984@gmail.com), [hezequiel.silvero@gmail.com](mailto:hezequiel.silvero@gmail.com), Zequiiel@gmail.com

**Resumen.**

SSL / TLS es un protocolo estándar para la comunicación segura de Internet. A pesar de su gran éxito, la implementación SSL de hoy es en gran medida limitada a los dominios críticos para la seguridad. La baja tasa de adopción de SSL es principalmente debido a la alta sobrecarga de cómputo en el lado del servidor. En este documento, proponemos unidades de procesamiento de gráficos (GPU) como una nueva fuente de poder de cómputo para reducir la sobrecarga del lado del servidor.

**Palabras claves:** SSL, GPU

1. Introducción

Secure Sockets Layer (SSL) y Transport Layer Security (TLS) han servido como un canal de comunicación segura en Internet para los últimos 15 años. SSL proporciona una comunicación segura contra escuchar a escondidas y permite la autenticación de los hosts finales.

A pesar de su gran éxito, el número de sitios web habilitados para SSL es ligeramente superior a uno millón, La razón principal detrás de la baja adopción de SSL se encuentra en alta gastos generales de computación en el lado del servidor.

En este trabajo se busca implementar de un acelerador SSL con GPU trabajando como proxy para servidores web, y muestra que las GPU aceleran los procesos SSL con una pequeña sobrecarga de latencia a la vez que aumenta significativamente rendimiento al mismo tiempo.

En la actualidad existe un proyecto similar llamado SSLShader [1] que plantea y busca solucionar la misma problemática.

1. Desarrollo

SSL usa tres algoritmos diferentes para una comunicación segura. Al principio, un cliente y un servidor intercambian aleatoriamente secretos usando un cifrado asimétrico. Los cifrados asimétricos (por ejemplo, RSA, DSA o DHE) garantizan el intercambio seguro de la clave secreta incluso cuando un atacante está escuchando a escondidas. Después del intercambio de la clave secreta, el servidor y el cliente generan claves de sesión simétricas cifradas (por ejemplo, AES, DES o RC4) y algoritmos de autenticación (por ejemplo, SHA1 o MD5). Los cifrados simétricos se utilizan para encriptar los datos y los algoritmos de autenticación se utilizan para evitar la manipulación. Para pequeñas transacciones SSL, el cifrado asimétrico ocupa la mayor parte del cálculo, y para transacciones grandes, el cifrado simétrico y el algoritmo de autenticación consume la mayor cantidad de cálculos. Se implementa operaciones criptográficas en GPU y se evalúa la efectividad para la aceleración de SSL. Se puede elegir entre RSA, AES128-CBC (Cipher-Block Chaining) y HMAC-SHA1 (Mensaje basado en hash Authentication Code). Pero el descifrado de clave pública se convierte rápidamente el cuello de botella cuando se debe establecer una gran cantidad de conexiones en el lado del servidor [2, 3]. El núcleo de la CPU solo puede manejar alrededor de 1,000 transacciones HTTPS por segundo (TPS) con RSA de 1024 bits, mientras que el mismo núcleo puede servir más de 10,000 HTTP TPS sin formato.

Las GPU alcanzan una gran potencia informática con cientos de procesamientos núcleos y gran ancho de banda de memoria, pudiendo ejecutar miles de hilos de hardware en paralelo. En el nivel de grano grueso cada transacción SSL se puede ejecutar en paralelo. En el nivel de grano fino, el algoritmo de cifrado en sí mismo puede procesar los datos de entrada en paralelo. Las operaciones de RSA dependen en gran medida de la aritmética de enteros grandes tales como una multiplicación de enteros de 1024 bits. Esto puede implementarse a la conexión utilizada en la sesión web rtc bajo https

1. Explicación del algoritmo.

La aritmética entera grande está dividida en series de operaciones enteras nativas y ejecutadas en serie en la CPU. En el GPU, cada operación independiente puede ejecutarse en paralelo. Se implementa un algoritmo paralelo con la complejidad del tiempo O (k) para la multiplicación de dos enteros de k bits. En el cifrado AES128-CBC, no hay paralelismo en el algoritmo en sí mismo como encriptación de cada bloque AES de 128 bits o 16B ya que depende de los resultados encriptados del bloque anterior. Por otra parte, el descifrado AES-128CBC se puede paralelizar en el nivel de bloque. El algoritmo HMAC-SHA1 no se puede paralelizar en el nivel de bloque debido a la dependencia de datos entre bloques pero se puede paralelizar en el nivel de registro Ssl.

1. Pruebas que pueden realizarse

La prueba más simple sería elegir un proxy SSL como lighttpd en lugar de una biblioteca de cifrado basada en GPU o un motor OpenSSL por la facilidad de integración con servidores existentes y el soporte para el bloqueo de aplicaciones controladas por eventos en operaciones criptográficas. Se puede utilizar un algoritmo de descarga oportunista para lograr un alto rendimiento mientras se mantiene la latencia baja. La idea clave es enviar todas las solicitudes a la CPU cuando el número de operaciones criptográficas son lo suficientemente pequeñas como para ser manejadas por la CPU. Si las solicitudes comienzan a acumularse en la cola, el algoritmo descarga operaciones criptográficas y vuelca al GPU y se aprovecharían los beneficios de la ejecución paralela para alto rendimiento.

Pseudocodigo:

-Llega una petición https (en la cual puede aprovecharse el paralelismo) a un proxy reverso como lighttpd que es usado para traducir las sesiones ssl a tcp

-La parte de desencriptacion de manda al gpu mientras el resto lo atiende el cpu

1. Conclusiones

Con HPC con GPU se puede optimizar en gran parte los procesos criptigraficos de SSL sobre un servidor y esto a su vez permite al CPU tomar mas peticiones SSL.

1. Referencias

1. https://shader.kaist.edu/sslshader/

2. Performance Analysis of TLS WebServers, NDSS 2002.

3. Transport Layer Security: How much does it really cost?, IEEE Infocom 1999