6R-4

クラスタリング SSL WEB システムの パフォーマンス計測に関する考察

初谷 良輔† 齋藤 孝道†

† 明治大学

1 はじめに

インターネット上の通信の安全性を確保するため, セキュリティプロトコルの一つである SSL (Secure Socket Layer) [1] が広く利用されている. SSL 通信は, 処理コストの高い多数の暗号処理を含み, さらに, 今日の Web システムが C/S (Client/Server) モデルであることから, SSL を用いた Web (以降, SSL-Web と呼ぶ) サーバに SSL 通信が集中することで, サーバの負荷が急激に上昇することがある [2][3]. つまり, SSL の利用は, Web 通信の「安全性」を確保する一方で, SSL-Web サーバのパフォーマンスや稼働率を低下させる要因となることがある.

このような SSL-Web サーバの問題に対する解決策の一例として, 負荷分散クラスタ (Load-Balanced Cluster) がある. 本論文では, SSL-Web システムにおいて動的なシステムリソースの再配分による負荷分散を実現する新たなクラスタリング方式 (以降, C-SSL と呼ぶ) の提案と実装, および, そのパフォーマンス評価について示す.

2 提案方式と実装

2.1 概要

本来の SSL 通信は, SSL Handshake とバルク暗号通信を同じ SSL サーバ上で一貫して処理する. それに対して, C-SSL では, SSL Handshake とバルク暗号通信を切り離し, 自由に役割を交換できるそれぞれの処理に特化したサーバによって一連の SSL 通信を実行する.

2.2 システム構成と実装

2.2.1 通報とデータベースのデータ構造

提案方式にて独自に実装した主要なデータ構造, および, 通報とそのパラメータについて示す.

データ構造について

(1) $STRUCT\_OBJ$ $\{Version, Cipher, Master\_Secrets\}$:
これは, プロトコルバージョン, 暗号スイート, セッション鍵など SSL 通信に必要なパラメータ群を格納するための構造体である.

(2) $STRUCT\_G$ $\{SockID,\ Session\_id\}$:
これは, クライアントと GAgent (後述) 間の TCP コネクションを特定するための識別子 $SockID$ と $Session\_id$ の対を格納するための構造体である.

(3) $STRUCT\_D$ $\{Session\_id,\ SSL\_OBJ\}$:
これは, $Session\_id$ と SSL_OBJ の対を格納するための構造体である.

通報とパラメータについて

(1) MSG_SSL_ID $(Session\_id)$:
これは, SSL Handshake の *Server Hello* に含まれるセッション識別子 $Session\_id$ を通知する.

(2) MSG_SSL_OBJ $(Session\_id,\ SSL\_OBJ)$:
これは, $Session\_id$ と SSL_OBJ を通知する.

2.2.2 サーバプログラムの役割

C-SSL では, GAgent, HAgent, DAgent の 3 つの主体を用いてクラスタリングシステムを構成する. 各主体の役割を以下に示し, その構成例を図 1 に示す. ここで, 図中の実線は, SSL Handshake や MSG_SSL_ID を交換するため TCP 通信を, 細かな破線は暗号化されたデータを含む通信 (バルク暗号通信) を, 粗い破線は HAgent が DAgent に MSG_SSL_OBJ を送信するための TCP 通信を示している. また, 図 1 は, 提案するクラスタリング方式を用いたシステム構成の一例であり, 図中の HAgent および DAgent は, 適宜増やすことができる. さらに, システム構成の前提として, サーバサイド中のネットワークは, 通信の盗聴などを試みる攻撃者のいない安全なものであるとする:

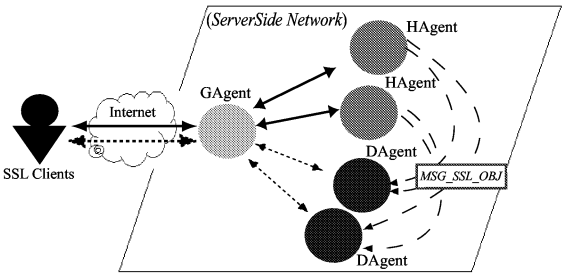

図 1: 提案方式を用いたシステム構成例

表 1: SSLv3 レコードタイプと転送先

Protocol Type	Hexadecimal	Destination
ChangeCipherSpec	0x14	HAgent
Alert	0x15	HAgent,DAgent
Handshake	0x16	HAgent
Application Data	0x17	DAgent

GAgent (Guiding Agent)

GAgent は, 「SSL クライアント」と「HAgent および DAgent」との間に存在するディスパッチャであり, SSL レコードヘッダを解析し, 表 1 にもとづいて TCP ポートフォワーディングを行う. GAgent の後方に配置された HAgent と DAgent への通信の分散には, Round Robin 方式*, ヘルスチェック機能には ICMP (Internet Control Message Protocol) Echo Request/Reply, また, パーシステンス管理にはクライアントのソース IP アドレスを今回の実装では利用した. また, GAgent は, SSL Handshake にて確立した SSL セッションとバルク暗号通信に利用する SSL セッションとの対応付けを行うために, $STRUCT\_G$ を利用する.

GAgent は, RedHat Linux-9 (kernel 2.4.20) 上で gcc-3.2.2-5 (C 言語) を用いて実装した.

HAgent (Handshake Agent)

HAgent は, GAgent の後方に配置された SSL Handshake

† Ryosuke HATSUGAI (hatsugai@cs.meiji.ac.jp)
† Takamichi SAITO (saito@cs.meiji.ac.jp)
Meiji University, 1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa, 214-8571, Japan. (†)

* いわゆる DNS Round Robin ではなく, Web サーバに対して順番に処理を割り当てていく負荷分散アルゴリズムである.

の処理に特化したサーバプログラムである. また, SSL Handshake にて生成した SSL セッションから *MSG_SSL_OBJ* を作成し, これをサーバサイドに配置されたすべての DAgent に転送する.

HAgent は, RedHat Linux-9 (kernel 2.4.20) 上で gcc-3.2.2-5 (C 言語) と OpenSSL-0.9.7g [4] を用いて実装した.

DAgent (Data Transfer Agent)

DAgent は, GAgent の後方に配置されたバルク暗号通信に特化したサーバプログラムであり, また, バルク暗号通信に必要となるセッションパラメータの管理には, *STRUCT_D* を用いる.

DAgent は, RedHat Linux-9 (kernel 2.4.20) 上で gcc-3.2.2-5 (C 言語) と OpenSSL-0.9.7g を用いて実装した. また, Web サーバプログラムとして今回の提案方式では, Apache-2.0.54 [5] を利用した.

2.3 通信の詳細

ここでは, 提案するクラスタリング方式での SSL 通信の処理手順について示す. ただし, 以下に示す説明文に割り当てた番号は, 図 2 における通報に割り当てたそれと対応している. また, 図 2 の線種は, 図 1 のそれと対応している.

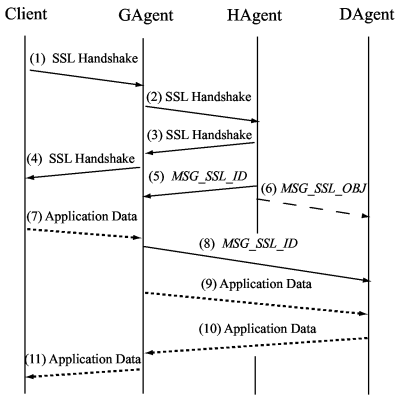

図 2: 提案方式における SSL 通信

(1): クライアントは, GAgent と TCP コネクションを確立し, その後, SSL Handshake を開始する.

(2): GAgent は, クライアントからの通報を表 1 の Protocol Type にもとづき転送するために TCP コネクションを HAgent と確立する. その後, クライアントから受信した通報を HAgent に転送する.

(3)(4): クライアントと HAgent は, SSL Handshake を GAgent 経由で実行し, SSL セッションを確立する.

(5): SSL セッションを確立した HAgent は, 当該セッションにもとづき *MSG_SSL_ID* を生成し, これを GAgent に送信する. ここで GAgent は, 当該 *Session_id* と *SockID* を *STRUCT_G* に格納する.

(6): HAgent は, DAgent と TCP コネクションを確立した後, *MSG_SSL_OBJ* を直接送信する. これを受信した DAgent は, *Session_id* と *STRUCT_OBJ* を *STRUCT_D* に格納する.

(7): 通報 (1)(4) にて SSL セッションを確立したクライアントは, 当該 SSL セッションパラメータで暗号化したアプリケーションデータを GAgent に送信する.

(8): GAgent は, クライアントからの通報を表 1 の Protocol Type にもとづき転送するため DAgent と TCP コネクションを確立する. さらに, *SockID* を検索キーとして, *STRUCT_G* から *Session_id* を取り出し, *MSG_SSL_ID* として DAgent に送信する. これを受信した DAgent は, *Session_id* を検索キーとして *STRUCT_D* から *STRUCT_OBJ* を取り出す. さらに, *STRUCT_OBJ* からクライアントとのバルク暗号通信に用いるセッションパラメータを取り出す.

(9)(10)(11): 以降, クライアントと DAgent は, GAgent 経由でバルク暗号通信を行う.

3 パフォーマンス評価

C-SSL のパフォーマンス計測には, httperf [6] を利用し, その評価は 1 秒間あたりの平均 SSL Handshake 数とする. 以下に, C-SSL とのパフォーマンス比較を行うために同様のパフォーマンス計測を行った SSL-Web システムを示す:

(a): Apache-2.0.54/OpenSSL-0.9.7g :
Apache を用いたスタンドアロン型の SSL-Web サーバである (以降, Apache/SSL).

(b): Apache-2.0.54/OpenSSL-0.9.7g/mod_proxy:
Apache とその拡張モジュールである mod_proxy を用いて構築した SSL リバースプロキシサーバと Web サーバ (Apache-2.0.54) の 2 台から構成される SSL-Web システムである (以降, Apache/Proxy/SSL).

今回の実験では, SSL 通信は「サーバ認証モード」を利用し, その暗号スイートは, 「RSA(1024bits)-RC4-SHA1」とする. また, クライアントに提供する Web コンテンツには PHP プログラムによって生成された 1024 Byte のテキストデータを利用する. 各サーバプログラムは CPU:XEON (3.20 GHz), メモリ:1024 MB を搭載したマシン上で稼動しており, クライアントである httperf は CPU:Pentium4 (3.20GHz), メモリ:512 MB を搭載したマシン上で稼動しており, これを 4 台用いて SSL-Web システムに負荷を掛ける. また, クライアントとサーバサイド間のネットワーク帯域は 1000 Mbps, ラウンドトリップあたりの通信遅延は 30 msec である. 上記の環境下での各 SSL-Web システムのパフォーマンス計測結果を図 3 に示す.

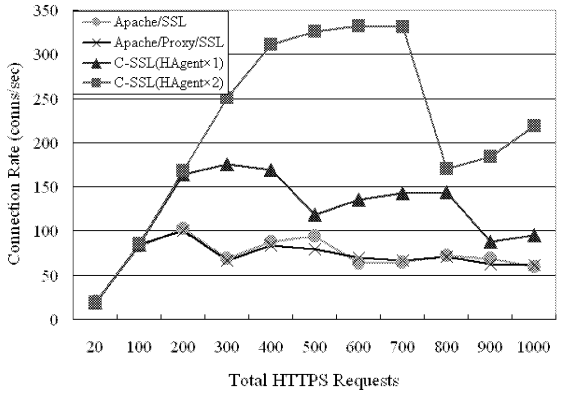

図 3: パフォーマンス評価

4 まとめ

本論文では, SSL プロトコルの分離によりシステムリソースの再配布を可能とする新たな SSL-Web システムの負荷分散クラスタリング方式の提案とその実装について示した. また, そのパフォーマンス評価を行った.

参考文献

[1] Alan O. Freier, Philip Kocher, and Paul C. Kaltorn, "The SSL Protocol Version 3.0 draft", March 1996, http://home.netscape.com/eng/ssl3/draft302.txt
[2] Cristian Coarfa, Peter Druschel, and Dan S. Walach, "Peformance Analysis of TLS Servers"
[3] George Apostolopoulos, Vinod Peris, and Debanjan Saha, "Transport Layer Security: How much does it really cost?"
[4] http://www.openssl.org/
[5] http://www.apache.org/
[6] http://www.hpl.hp.com/research/linux/httperf/