SSL セッションマイグレーションの実装と評価

島崎 聡史† 中澤 昌史† 金田 健太郎† 齋藤 孝道††
†明治大学大学院 ††明治大学

1 はじめに

SSL (Secure Socket Layer) [1] では，Handshake の処理コストを抑えるため，一度確立したセッション情報を再利用するセッション再開という機能を用意しているが，通常，セッション情報はセッションを確立したサーバで管理されているため，別のサーバでセッション再開をすることはできない．ここで，例えばメンテナンスのために処理を別のサーバへ切り替える場合，切り替え直後には一斉に FullHandshake を行わなければならず，サーバの負荷が急激に上昇してしまう．

そこで，本論文では，SSL のセッション情報を複数のサーバ間で共有することにより，セッション情報を確立したサーバ以外のサーバでセッション再開を行うシステムの提案とその実装を示し，サーバの負荷の観点からその評価を行う．

2 提案システム

2.1 主体構成

提案システムは，Active サーバと Standby サーバからなり，Internet を介してクライアントと接続している．これとは別に，Active サーバと Standby サーバ間は，通報用のネットワークでも接続している．

Active サーバ

応答サーバの切り替え前，すなわち通常稼動時に応答する SSL-Web サーバである．クライアントとの間でセッション情報[1]を確立した際に，Standby サーバにマイグレーション通報 (後述) を送信し，セッション情報を共有する．また，実装は OpenSSL0.9.7a と Apache2.2.9 を改変して行った．

Standby サーバ

応答サーバの切り替え後に応答をする SSL-Web サーバである．応答サーバの切り替え前は，Active サーバからのマイグレーション通報を受信し，格納する．応答サーバの切り替えによって処理を引き継ぎ，その後の応答をする．実装は，同じく OpenSSL0.9.7a と Apache2.2.9 を改変して行った．

2.2 提案システムの詳細

2.2.1 応答サーバの切り替え

応答サーバの切り替えは，ARP リクエストにレスポンスを返さない設定 (以降，ARP 無視設定と呼ぶ)[2]と，GratuitousARP を用いて行う．ここで，GratuitousARP とは，自身の IP と MAC Address をブロードキャストし，同一セグメント内のネットワーク機器の ARP テーブルを変更させるパケットである．以下に，その手順を示す：

(0) 予め，Active サーバ，Standby サーバの双方のネットワークインターフェースに同じ IP を設定して起動し，そのうち Standby サーバは ARP 無視設定をしておく
(1) Active サーバのネットワークインターフェースを停止する
(2) Standby サーバの ARP 無視設定を解除する
(3) Standby サーバから，GratuitousARP をブロードキャストする

以降，この操作を応答サーバの切り替えと呼ぶ．

[1] 本論文では，セッション情報とは，マスターシークレット，セッション ID といったセッション再開に必要な情報を言う．
[2] Linux では，/proc/sys/ipv4/conf/インターフェース名/arp_ignore に 8 を記入して設定する．

2.2.2 Apache のセッションキャッシュモード

Apache mod_SSL のセッションキャッシュモードのうち，今回は，データベースファイルに書き出すモードを用いた．

このモードでは，データベースファイル (以降，セッションキャッシュデータベースと呼ぶ) に，セッション ID を検索キーとして，ASN.1 フォーマットの SSL_SESSION 構造体[3]と当該セッションの有効期限を格納する．

2.2.3 通報

提案システムでは，Active サーバと Standby サーバ間でセッション情報を共有するため，Active サーバから Standby サーバへ，通報用ネットワークを介し，以下のフォーマットの通報を送信する．以降，この通報を，マイグレーション通報と呼ぶ．

```
Struct{
    Session_id
    SSL_SESSION ASN.1 converted
}
```

ただし，*SSL_SESSION ASN.1 converted* とは，ASN.1 フォーマット の SSL_SESSION 構造体である．

2.3 動作の流れ

提案システムの応答サーバ切り替え前後の動作の流れを図 1 に示す．図中の番号は以下の説明のそれに対応している：

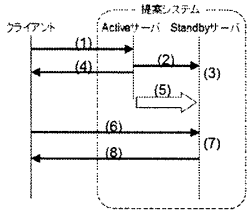

図 1: 提案システムの動作の流れ

(1) クライアントが HTTPS リクエストを送信する
(2) Active サーバが FullHandshake 後，マイグレーション通報を Standby サーバに送信する
(3) Standby サーバがマイグレーション通報を受信し，セッション情報をセッションキャッシュデータベースに格納する
(4) Active サーバがクライアントへレスポンスを送信する
(5) 応答サーバの切り替えを行い，Standby サーバが応答サーバとなる
(6) Web ブラウザが再び HTTPS リクエストを送信する
(7) Standby サーバが HTTPS リクエストを受信し，セッションキャッシュデータベース内の情報を利用し，セッション再開を行う
(8) Standby サーバがクライアントへレスポンスを送信する

[3] OpenSSL で，マスターシークレット，セッション ID といったセッション再開に必要な情報を格納する構造体

3 評価

3.1 評価の方法

提案システムの評価のために，システムに対し，負荷生成器を用いて一定の負荷をかけている最中に応答サーバの切り替えを行い，その切り替え後のサーバのCPU負荷を計測した．同様に，比較対象として，セッション情報の共有を行わない形で応答サーバの切り替えを行うオリジナルのシステム(以降，既存システムと呼ぶ)についても計測した．

3.2 計測

3.2.1 計測環境

計測に用いたサーバのシステム構成を以下に示す．

Active サーバ・Standby サーバ

富士通社のPRIMEQUEST540[4]を用いた．本計測では，次のようなパーティション[4]を2つ用意し，それぞれの上でActiveサーバ，Standbyサーバを構成した．

PRIMEQUEST540
Red Hat EnterprisedLinux4 AS (Update4)
Intel Itanium64 1.66GHz
8GB RAM
1000Mbps NIC

また，いずれのパーティションも2GbpsのFC (Fibre Channel)で接続されたストレージをSAN (Storage Area Network)として利用している．

負荷生成器

Spirent社のAvalanche2700モデルB[5] (以降，Avalancheと呼ぶ)を用いた．Avalancheは，Webサーバやネットワーク機器に対し負荷テストを行う機器である．

計測で利用した機器のシステム構成を以下に示す．

Avalanche2700
Intel Xeon 3.6GHz
8GB RAM
1000Mbps NIC

本計測は，PRIMEQUESTの各パーティションと筐体内部で結線されている1GbpsのスイッチングハブであるGSWBとAvalancheを直接接続して行った．

3.2.2 生成した負荷パターン

Avalancheは，HTTPSリクエストを送信するユーザを複数エミュレートし，負荷を生成する．そのエミュレートするユーザ1人あたりの挙動を以下に示す：

1. HTTPSリクエストをシステムへ送信する
   (FullHandshakeを行う)
2. システムからレスポンスを受け取り，一定時間
   (以降，リクエスト待機時間と呼ぶ)待機する
3. HTTPSリクエストをシステムへ送信する
   (セッション再開を試みる)
4. 2. 3. をもう一度だけ繰り返す

リクエスト待機時間は5秒，10秒，30秒，60秒の4種類を同じ割合で混成してある．また，取得するファイルのサイズは50KByteであり，公開鍵暗号スイートはRSA，共通鍵暗号スイートはRC4 (128bit)，ハッシュアルゴリズムはMD5，サーバ認証モードである．

負荷生成開始から90秒間は，1秒あたり800個のHTTPSリクエストを送信し，その後は待機時間を経過したユーザのリクエストを送信する．

ここで，1秒当たりのリクエスト数は，システムが処理できる上限が1秒あたり1200リクエスト程度であることを考慮し，設定した．

3.2.3 計測手順

計測は，以下の手順で行った：

(1) Avalancheから，Activeサーバに負荷を生成する
(2) Standbyサーバで，sarコマンドを用いて，CPU負荷を記録し始める
(3) Activeサーバが応答しているのを確認し，負荷を生成し始めて30秒経過後に応答サーバ切り替えを行う
(4) Standbyサーバが応答しているのを確認する

[4] PRIMEQUESTでは，論理的に構成したサーバ1台ずつをパーティションと呼ぶ

3.3 計測結果

計測結果を図2に示す．このグラフは提案システムと既存システムの計測結果を応答サーバ切り替え完了時点を基準に重ね合わせ，さらに両方の6次多項式近似曲線を付加してある．

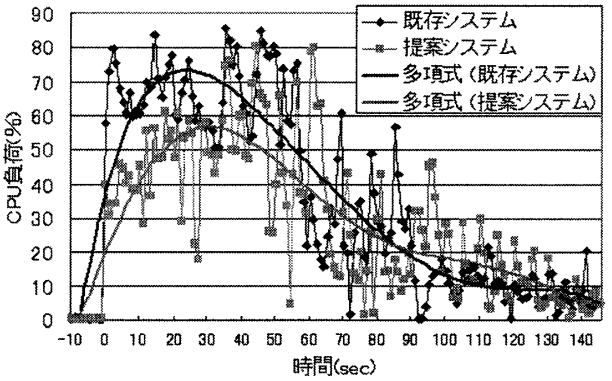

図2: 計測結果

図2で，折れ線グラフに着目すると，およそ0秒から25秒の範囲で，既存システムでピーク時にCPU負荷が80 %前後まで上昇してしまっているのに対し，提案システムでは60 %前後までしか上昇していない．

また，図2で，近似曲線に着目すると，およそ85秒から130秒の範囲では，既存のシステムより提案システムの方がCPU負荷が高くなっている．

3.4 考察

セッションマイグレーションによって，Activeサーバで確立したセッションをStandbyサーバでセッション再開させることで，0秒から約25秒の範囲で，提案システムが既存システムよりCPU負荷を20 %ほど抑えられる事がわかった．

また，約85秒から約135秒の範囲で提案システムが既存システムよりCPU負荷が高くなっているのは，提案システムがマイグレーションしたセッション情報を保持しているため，セッションキャッシュデータベースの検索コストが増えたからだと考えられる．これは，セッション情報を保持する時間[5]を経過すれば既存システムと同程度に落ち着くと考えられる．

4 まとめ

本論文では，SSLセッションのマイグレーションを用いて，セッションを確立したサーバ以外のサーバでセッション再開を行う方式を提案し，サーバへの負荷の観点からその評価を行った．提案システムでは，既存のシステムに比べ切り替え後のピーク時のCPU負荷を抑えることができた．

参考文献

[1] Alan O.Freier, Philip Kaltorn, and Paul C.Kocher, "The SSL Protocol Version 3.0 draft", March 1996
[2] The OpenSSL Project, http://www.openssl.org/
[3] John Viega,Matt Messier,Pravir Chandra 共著 齋藤孝道 監訳 "OpenSSL 暗号・PKI・SSL/TLS ライブラリの詳細" オーム社
[4] PRIMEQUEST, http://primeserver.fujitsu.com/primequest/
[5] Avalanche, http://www.toyo.co.jp/spirent/avalanche/2700series.html

[5] Apacheの標準は300秒