O-015

レガシーラッピングで提供される Web サービスの応答高速化手法
Response Improvement of Web Services Provided with Legacy Wrapping Technique

塚本 良太† 吉村 礼子† 山足 光義†
Ryota Tsukamoto Ayako Yoshimura Mitsuyoshi Yamatari

1. 研究背景と目的

既存のシステムを再利用してサービス化する技術は、サービス指向のシステムアーキテクチャを実現するために欠かせない技術の一つである。

既存システムを再利用するアプローチは大きく White-Box 型と Black-Box 型の二種類が知られている[1]。特に Black-Box 型はレガシーラッピングと呼ばれ、複雑化した昨今の情報システムの現状から、その開発コスト面で有利であり注目されている。

近年の情報システムにおいては多くの場合、Web インタフェースを持っていることもあり、Web システムを対象としたラッピング手法がいくつか提案されている[2][3][4][5]。Web システムをラッピングするシステムの多くは基本的な機能構成として図 1に示すような、リクエスト処理機能、シナリオ処理機能、エミュレータの組み合わせで提案されている。これらはシナリオ処理機能のロジックに大きな違いがあり、如何にラッピングの定義を容易にするか、という視点が主であった。

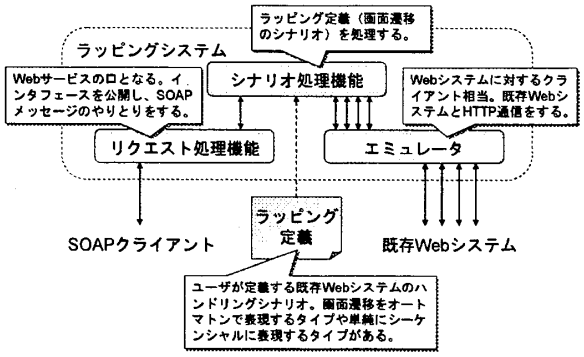

図 1：Web システムラッピングの基本構成

しかしながら、レガシーラッピングはサービス化対象を Web システムなどの画面遷移を含めたものにすると、エミュレータによる Web サーバとの通信や処理時間のオーバーヘッドが大きくなる。これにより、レガシーラッピングによって提供する Web サービスの応答時間が長くなる問題がある。

そこで本稿では、レガシーラッピングで提供される Web サービスの高速化手法を提案、評価する。

2. レガシーラッピングのオーバーヘッド

レガシーラッピングでは通常、エミュレータを利用して、結果的にユーザ操作をシミュレーションする。そのため、サービスの入出力とは直接関係しないエミュレーションが含まれる場合がある。例えば、既存 Web システムにおいて、ユーザがブラウザで入力画面へ画面遷移するまでの処理がこれにあたる（図 2における 1 回目の入力）。

よって、サービスの入出力とは直接関係しない処理を省くことで、サービスとしての応答時間を高速化することが期待できる。

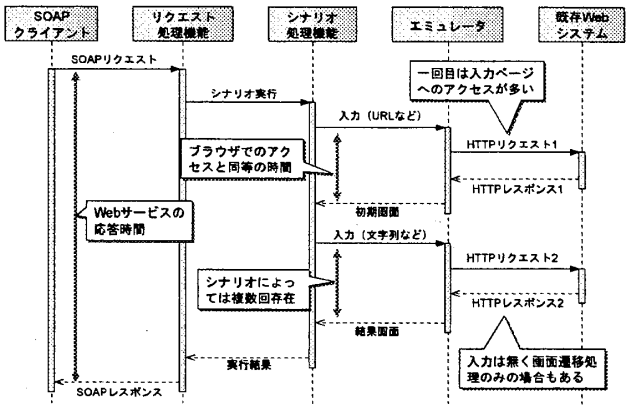

図 2：レガシーラッピングの主な処理シーケンス

3. 状態のプールによる高速化

Web システムを対象としたラッピングの場合、通常はエミュレータによる入力画面取得が最初の処理となる。よって、あらかじめ特定の SOAP リクエストに対して入力待ち状態となっているエミュレータを準備しておくことで高速化が図れると考えられる。

本稿ではその実現方法としてエミュレータの状態をプールする手法を提案する（図 3）。適当な状態でプールし、共有化されたエミュレータをシナリオ処理機能が取り出すことにより高速化を図る。

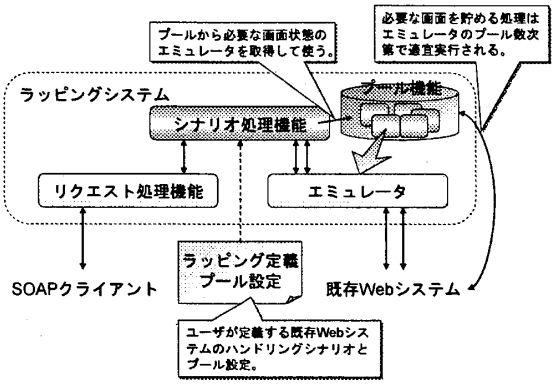

図 3：高速 Web システムラッピングの構成

図 3に示した構成の動作シーケンスを図 4に示す。本手法では、シナリオ処理機能からのプール初期化があらかじめ実行されているため、従来 SOAP リクエストを受信してから処理していた内容がプール機能からのエミュレータ取得に置き換わる。

†三菱電機株式会社 情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation

よって、Web サーバとの通信などによるオーバーヘッドだけでなく、ラッピングしたシナリオの一部をスキップする効果も得られ、サービス全体としての応答時間が大きく短縮されると考えられる。また、プール機能では複数のエミュレータを保持しておけるため、多重リクエストなどで高いスループットを期待できる。

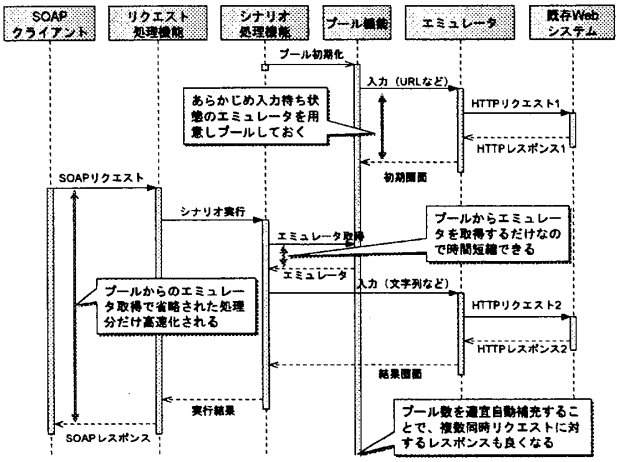

図 4：高速 Web システムラッピングの処理シーケンス

4. 比較評価

本手法の効果を定量的に評価するため、従来のラッピングシステムとして、文献[4]で既に我々が提案しているラッピングシステムと比較する。ラッピングするシナリオは Apache Tomcat 6.0 サーバに付属しているサンプル Web アプリケーションを利用し、文字列を逆順にするサービスとした。表 1に画面遷移などのシナリオを示す。

従来の手法ではこのシナリオを逐次実行するが、本手法では、テキストボックスに文字列を入力するまではサービスの入出力とは直接関係しないため、1～3 のブラウザ操作を完了した状態をプール機能によってあらかじめ用意しておき、4 からシナリオを実行することになる。

表 1：ラッピングするシナリオ

No.	ユーザのブラウザ操作
1*	サンプルアプリケーションページを開く。 http://hostname:8080/examples
2*	「JSP Examples」をクリックする。
3*	Functions の右の「Execute」をクリックする。
4	テキストボックスに文字列を入力する。
5	「クエリ送信」をクリックする。
6	表中の Result 列の 2 段目の値を取得する。

*：プール機能であらかじめ処理する部分。

以上のシナリオを従来の手法と本手法でラッピングし、提供された Web サービスの応答時間とスループットを計測した結果を図 5、図 6に示す。本手法ではプールを用いているため、その数を 20、40、80 とした場合も計測した。

結果から、プール機能を用いた本手法によって、Web サービスの応答時間が 70%～50%に短縮されていることがわかる。プール数による差は長期的に見たプール補充回数の差によるオーバーヘッドである。

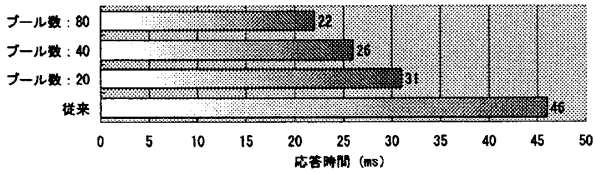

図 5：応答時間の比較結果

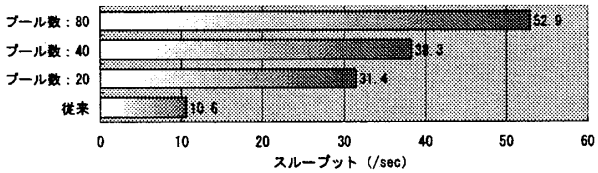

図 6：スループットの比較結果

また、スループットにおいては 3 倍～5 倍の性能を示しており、プール数を大きくすると性能が大きく向上している。これらのことから、本手法は従来手法に比べ提供するサービスの応答が高速であり、スループット性能も高いことを示せた。

5. まとめと今後の展望

本稿では、レガシーラッピングのオーバーヘッドを削減し、サービスを高速化する手法としてエミュレータをプールする機能を組み込むことを提案した。従来手法と比較することで、応答時間、スループットにおいて本手法は明らかに高性能であることを確認できた。

本手法は Web システムを対象とするラッピングだけでなく、文献[6]にあるような一般的なシステムのラッピングにおいても、同様に高速化ができると考えられ、多くの応用が期待できる。

しかしながら、今回の比較評価では単一サービスであったためプールを共通利用できる範囲やプール数などの最適化、プール中のエミュレータの異常系の処理等は考慮しなかったため、今後検討が必要である。

参考文献

[1] S. Comella-Dorda, K. Wallnau, R. C. Seacord and J. Robert, "A Survey of Legacy System Modernization Approaches", CMU/SEI-00-TN-003, 2000.

[2] G. D. Lorenzo, A. R. Fasolino, L. Melcarne, P. Tramontana and V. Vittorini, "Turning Web Applications into Web Services by Wrapping Techniques", WCRE 2007, pp.199-208.

[3] 高橋 健一, 立堀 道昭, 紫合 治, "Web アプリケーションの Web サービス変換", IPSJ SIG Notes, Vol.2006, No.35, pp.1-8.

[4] 塚本 良太, 吉村 礼子, 山足 光義, "スクリーンラッピングによる既存 Web システムのサービス化", IPSJ 第 70 回全国大会, 2008.

[5] 土屋 隆, 吉村 礼子, 塚本 良太, "スクリーンラッピングによる Web サービスを利用した SOA システム構築", IPSJ 第 70 回全国大会, 2008.

[6] D. Bovenzi, G. Canfora and A. R. Fasolino, "Enabling Legacy System Accessibility by Web Heterogeneous Clients", CSMR 2003, pp.73-81