エンドユーザレスポンスタイムを高精度に予測するための Webサーバ処理性能モデル化方法

峰野 博史　　川原 亮一
NTT サービスインテグレーション基盤研究所
E-mail:{mineno.hiroshi, kawahara.ryoichi}@lab.ntt.co.jp

ネットワークシミュレーションは，ネットワーク部分も含むシステム全体の性能評価をプロアクティブに行う手段として有効であるが，トラヒック条件，ネットワーク条件，ノード性能をどのようにモデル化するかが非常に重要である．本稿では，Web サーバが返送するホームページを構成する各オブジェクトの生成方法に応じてサーバ処理時間（サービスタイム）の計算式を変化させ，HTTP トランザクション内に適切にマッピングさせることによりエンドユーザレスポンスタイムを高精度に予測するための Web サーバ処理性能モデル化方法を検討した．また，実機への負荷実験結果から各パラメータを設定する方法も示し，シミュレーション結果を検証することにより本モデル化方法の有効性を示した．

Web Server Performance Modeling Method for Accurate Simulation of End User Response Time

Hiroshi Mineno　　Ryoichi Kawahara
NTT Service Integration Laboratories
E-mail:{mineno.hiroshi, kawahara.ryoichi}@lab.ntt.co.jp

Network simulation is one of the effective ways for a proactive performance evaluation of enterprise systems, and it is especially important to determine how to model the traffic, network, and node performances. In this study, we propose the modeling method of the web server performance with a feature of changing the service-time calculations according to the types of responsing inline-objects. We also describe how to set the parameters of this model from the stress test results, and show the effectiveness of this modeling method through some examples.

1　はじめに

WWW とその技術を利用した Web システムは急激な勢いで普及しており，WWW 上で Web サイトとして判明しているものだけでも約 3,800 万の Web サイトが存在する [1]．Web システムの構成は，プレゼンテーション，ビジネスロジック，データソースという 3 つの論理構造を物理的にどのように実装するかによって多種多様であり，XML や SOAP といった技術を核として Web 上で提供されるサービスをダイナミックに発見し連携させる Web サービスの出現など，その構成はますます複雑化している．

このような Web システムの性能評価法には様々な方法があるが，ネットワーク部分まで含めたシステム全体の性能評価をシステム設計時や開発初期に行う手段としてネットワークシミュレーションが有効である．ネットワークシミュレーションでは，トラヒック条件，トポロジ，プロトコル動作，ノード性能をどのようにモデル化するかが重要であるが，評価目的，評価期間，要求精度に応じてモデルの詳細度を変え，それらをバランス良くモデル化することが重要である．

[2] のように Web サーバや Web システム全体を待ち行列モデルの組み合わせでモデル化し，性能評価に役立てようとする研究も数多く行われているが，TCP の輻輳制御や HTTP の動作など実際は複雑な処理が発生しているネットワーク部分や，様々なハードウェア，OS，アプリケーションソフトウェアの組合せによって構成される Web サーバの性能などまで考慮して評価する場合には，待ち行列モデルの組合せで表現するのは困難である．また，[3] のように CPN：Coloured Petri Nets モデルを用い，Web システムをシステム構造，アプリケーション動作，リソース性能というような階層構造で表現し，HTTP の基本的な動作や Web サーバの性能をモデル化して性能評価に利用しようという研究もある．しかし，実測値とシミュレーション結果が近くなるようリソース性能のパラメータ調整を何度も行って検証したと述べており，パラメータ設定方法に関する定量的な方法が確立されていない．[4] では，Web サーバ性能のモデル化に注目しているが，比較的高精度なノード性能の模擬，ハードウェア構成変更が性能に与える影響

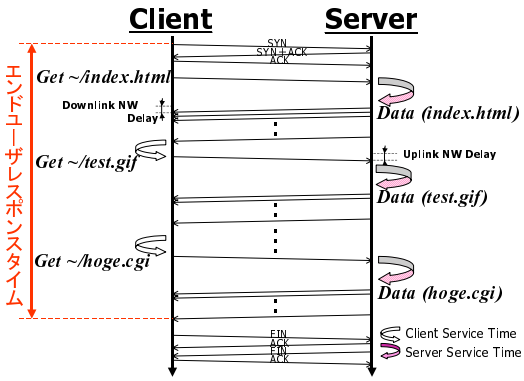

図 1: HTTP トランザクション分析

を簡易評価できる反面，シミュレーション時間が極端に長くなったり，モデル設定のためのパラメータ測定が大変といった課題が挙げられる．このように，ネットワークシミュレーションを用い比較的高精度な性能評価を短期間に実施するためには，プロトコル動作とノード性能のバランスを考慮したモデル化方法を検討する必要がある．

本稿では，性能評価項目として重要な項目の一つであるエンドユーザの感じるレスポンスタイム（エンドユーザレスポンスタイム）を簡単な設定で高精度に予測することを目的に，Web サーバ処理性能を簡易な測定によって得られるパラメータによってモデル化する方法を検討した．なお，プロトコル動作のモデル化については，パケットフローベースにイベントをスケジューリングし，HTTP/1.1[5] のトランザクションをほぼ模擬することのできるディスクリートイベントネットワークシミュレーションツール OPNET[1] を採用した．

本検討では，Web サーバの処理性能を基本処理性能のモデル化，競合時の処理性能劣化度のモデル化という以下の 2 つの観点からモデル化している．なお，その基本的な概念は OPNET の実装方法を参考とした．

- 1 ユーザが Web サーバへアクセスする時の処理性能を基本処理性能と考え，その時の Web サーバにおける総サービスタイムを分析し，シミュレーションで模擬される HTTP/1.1 トランザクションへ適切にマッピングさせてホームページを構成する各オブジェクトの処理で生じるサービスタイムを模擬する．この時，オブジェクトの生成方法に応じてサービスタイムの計算方法を変化させ，Web サーバの基本処理性能を表現
- 複数ユーザが Web サーバへ次から次へとアクセスする負荷実験を行い，競合するユーザ数に応じて平均レスポンスタイムがどれ程増加するかを処理性能劣化度と考え，その度合いに従って競合処理時の性能を表現

また，処理性能劣化度測定のための負荷実験結果より，同時アクセスユーザ数とレスポンスタイムが線形近似できる場合は，処理性能劣化度を単純な式に近似できることを示した．そして，実測値と本モデルを用いたシミュレーション結果を比較し，本論で提案するモデル化方法と処理性能劣化度の近似法の有効性を検証した．

以下に本稿の構成を示す．第 2 章で HTTP トランザクションの分析と提案する Web サーバ処理性能モデル化方法を記し，第 3 章でモデルに設定するパラメータ測定方法について述べる．そして，第 4 章で本モデル化方法を用いたシミュレーション結果と実測結果を検証し，第 5 章でまとめと今後の進め方について述べる．

2 Web サーバ処理性能のモデル化

2.1 HTTP トランザクションの分析

これまで多くの Web ブラウザや Web サーバソフトウェアが開発されてきたが，その多くが標準で実装している HTTP/1.1 を用いた基本的なデータのやり取りを分析し，エンドユーザレスポンスタイムを構成する重要な要素について検討した（図 1）．この図からも，エンドユーザレスポンスタイムはクライアント-サーバ間の複数のデータエクスチェンジとそれらを形成する両方向のネットワーク遅延，クライアント処理時間，サーバ処理時間の総和であるといえる．つまり，ネットワークシミュレーションによってエンドユーザレスポンスタイムを高精度に模擬するためには，ネットワーク遅延などのネットワーク部分のモデル化だけでなく図 1 に示されるような HTTP トランザクションやクライアント，サーバといったノードにおける処理性能のモデル化が重要となる．

プロトコル動作のモデル化に関して TCP 輻輳制御のモデル化 [6, 7] やトランスポートプロトコルを考慮した HTTP 性能のモデル化 [8] などの検討もあるが，本稿では，トランスポートプロトコルや HTTP/1.1 トランザクションなどのプロトコル動作をほぼ模擬できるディスクリートイベントネットワークシミュレーションツールを採用することとし，サービスタイム（最終的にはレスポンスタイム）に大きな影響を与えるノード性能のモデル化方法について検討している．

2.2 ノード性能のモデル化

ノード性能の模擬で重要なのは，アクセス数が 1 である時の基本処理性能のモデル化とアクセス数が増加し同時実行される処理が競合した時の処理性能劣化の様子のモデル化である．アクセス数 1 の基本処理時のサービスタイムを ST_1，n ユーザ競合時の処理性能劣化度を $Load_n$ とすれば，n ユーザ競合時

[1] OPNET Technologies, Inc. OPNET Modeler V.8.0.C PL16. http://www.opnet.com/

のサービスタイム ST_n は以下の式で表せるといえ，OPNET の標準サーバモデルもこのように実装されている．

$$ST_n = ST_1 \times Load_n \quad (1)$$

ノード性能を模擬すべき対象はクライアントとサーバがあるが，クライアント処理性能は，ハードウェア構成，使用 OS，ブラウザ等の組合せに大きく依存し，モデル化するとしてもサーバと同様なモデルが流用可能であると考え，サーバ処理性能のモデル化に焦点を絞った．ただし，モデルの設定やパラメータ測定が困難でないよう，できるだけ少なくかつ容易に測定可能なパラメータによってモデル化することに努め，OPNET の実装方法を参考とした．

2.3 基本処理性能のモデル化

まず基本処理性能のモデル化方法について検討する．Web サーバの役目は，要求されたホームページやそのページを構成するインラインオブジェクトを返送することである．一般にホームページやインラインオブジェクトは，登録されたコンテンツをそのまま返送する“ 静的 ”なものと，リクエストに応じて Web サーバ上のプログラムが起動し自動的にコンテンツを生成する“ 動的 ”なものの 2 種類に分類できる．このように Web サーバの返送するオブジェクトが“ 静的 ”なのか“ 動的 ”なのかに応じてアクセス数 1 の時の ST_1 算出方法を変化させることを考えた．

ここで，各々の ST_1 を ST_{static}，$ST_{dynamic}$ と定義する．そして，各 ST_1 を返送するオブジェクトサイズ（FS[bytes]）に依存する部分と CGI やサーバ API，FastCGI 等の処理で生じるオブジェクトサイズに依存しないサービスオーバヘッド（OH[sec]）という単純なパラメータを用い ST_{static} と $ST_{dynamic}$ を以下のように表した．

$$ST_{static} = \frac{FS_{static}}{PS} \quad (2)$$

$$ST_{dynamic} = OH + \frac{FS_{dynamic}}{PS} \quad (3)$$

ここで，アプリケーションがハードディスクもしくはキャッシュからデータを読み込むのに要する時間は，そのサイズに依存すると考えられるため，アプリケーション処理速度（PS[bytes/sec]）というパラメータを用い，また，OH は CGI やサーバ API，FastCGI によって生じる FS に依存しない処理時間だけでなく，OS がスレッド切り替える時に発生するコンテキストスイッチに要する時間なども含めるものとする．

このように，ST_1 を比較的少ないパラメータで表し，ホームページを構成するオブジェクトの生成方法（静的か動的）に応じて計算式を変化させるところが大きな特徴である．

2.4 処理性能劣化の様子のモデル化

次に複数ユーザが同時にアクセスした場合のように処理が競合した時の処理性能劣化の様子のモデル化方法について検討する．典型的な CPU の時分割ラウンドロビンスケジューラに基づく競合時のサービスタイム増加の様子を考えれば，競合処理数の増加に伴い競合した各処理に要するサービスタイムが増加すると考えるのが一般的である．ここで，エンドユーザレスポンスタイムを形成するネットワーク遅延やクライアント処理時間が無視できれば，競合処理数が増加した時の各サービスタイムの平均増加率とエンドユーザレスポンスタイムの増加率は等しいと見なせる．つまり，エンドユーザレスポンスタイムの増加率を測定することにより，競合処理数増加に対する処理性能劣化度 $Load$ を求めることができる．

例えば，レスポンス受信後すぐに次のリクエストを要求するよう設定されたクライアントを一定時間毎に 1 台づつ増加させる負荷実験を行い，その一定時間内にレスポンスを得られた全クライアントの平均レスポンスタイムを求める．そして，1 クライアント時の平均レスポンスタイム（RT_1）を基準に同時アクセスクライアント数 n の時の平均レスポンスタイム（RT_n）の増加率を求めれば，処理性能劣化度 $Load_n$ が分かる（4）．

$$Load_n = \frac{RT_n}{RT_1} \quad (4)$$

3 各パラメータ測定方法

第 2 章で検討した基本処理性能モデルと処理性能劣化モデルの各パラメータ測定方法について検討する．

基本処理性能や処理性能劣化度の測定はクライアント端末で行うが，Web サーバ端末とは 1m のクロスケーブルで直結し，ネットワーク遅延をできるだけ小さくしている．また，OS，Web サーバソフトの設定はほぼデフォルトのままとし，同時接続数の制限のみ無限にした．この測定は最大処理性能の計測が目的ではないため，レジストリ中の接続キューサイズ，最大スレッド数，CGI 用スレッド数，タイムスライス値などはデフォルトのままとしている．測定を Web サーバ上で行わないのは，OS 付属のネットワークモニタではタイムスライス値より短い周期でログを取得することができず，また，他の測定用アプリケーションをアドオンしたのでは，サービスタイムという非常に短い時間を測定するのに対しオーバヘッドの影響が少なくないからである．

また，処理性能劣化度を測定するための負荷実験では，実際のブラウザを用いて多数のクライアントがアクセスし続けるのは難しいため，ストレステストツールを採用した．様々なストレステストツールがあるが，TCP 同時接続数や仮想ユーザの各種設定変更，取得統計量の量と出力形式などから判断し，実

表 1: 基本処理性能パラメータの算出結果例

Homepage	p	q	$\sum_{i=0}^{p} FS_i$(bytes)	$\sum_{j=0}^{q} FS_j$(bytes)	PS(bytes/sec)	OH(sec)
Type-A:	14	0	42408	0	10046908	0
Type-B:	19	1	67492	307	10026115	0.019895

際のブラウザの動きをできるだけ忠実に模擬させられるツールを採用した[2]．

3.1 基本処理性能の測定

処理性能劣化度を測定する時と同じストレステストツールを用い，仮想ユーザ数 1 で Web サーバへ数回アクセスさせ基本処理性能の測定を行った．

オブジェクトサイズ（FS）の測定は，Web サーバ上の HTML ファイルサイズや各インラインオブジェクトサイズを調べたり，キャプチャ結果から各 TCP パケットのペイロードを足し合わせることによって求めることができる．

また，キャプチャ結果から，Web サーバが各オブジェクトへのリクエストを受信し，その要求に対するレスポンスを返送し始めるまでの時間を調べることにより，各オブジェクトの返送に要した実測サービスタイム（MST）を求められる．総オブジェクト数を p，その中に占める動的オブジェクト数を q とすると，(5)（6）の連立方程式を解くことによって対象ホームページに対するサービスオーバヘッド（OH）と処理速度（PS）を求めることができる．その結果例を表 1 に示す．

$$\sum_{i=0}^{p} MST_i = \sum_{j=0}^{q} OH_j + \frac{\sum_{i=0}^{p} FS_i}{PS} \qquad (5)$$

$$MST_k = OH_k + \frac{FS_k}{PS} \qquad (k = 0, .., q) \qquad (6)$$

3.2 処理性能劣化度の測定

処理性能劣化度の測定は，レスポンス受信後すぐに次のリクエスト送信を繰り返す仮想ユーザを一定時間毎に増加させるというシナリオのストレステストによって測定できる．このようなシナリオにより各一定時間内で同時アクセスクライアント数をほぼ一定に保つことができ，その時間内にレスポンスを得られた全クライアントの平均レスポンスタイムによって（4）を求めることができる．

表 1 のホームページに対し 15 秒毎に仮想ユーザ数を 1 ずつ 50 人まで増加させた時のレスポンスタイム（1 秒平均）の分布を図 2 に，x 軸を同時アクセス仮想ユーザ数とした時の平均レスポンスタイムの様子を図 3 に示す．また図 3 において，線形回帰，対数

[2] Mercury Interactive, Inc. LoadRunner7.02 を使用

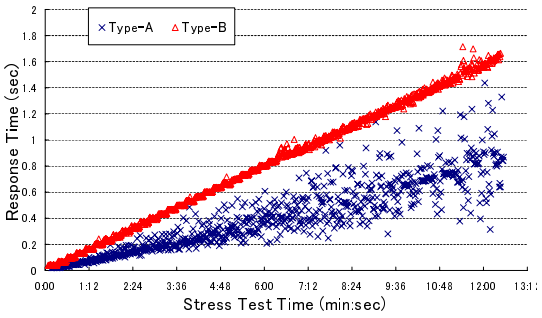

図 2: $Load_n$ 用ストレステスト結果

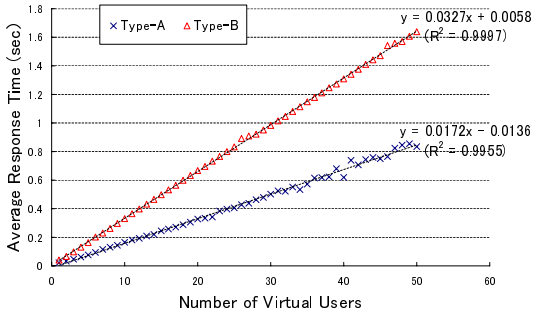

図 3: 仮想ユーザ数と平均レスポンスタイムの関係

回帰，累乗回帰，指数回帰，多項式回帰の中で最も決定係数の高かった線形回帰の式を横に記した．

ここで，図 3 のように平均レスポンスタイムが同時アクセスユーザ数に対し $Y = aX + b$ で近似できると（4）は以下のように表すことができる．

$$Load = \frac{a}{a+b}X + \frac{b}{a+b} \qquad (7)$$

仮に，傾き a が小さい場合は（低負荷なホームページに多いと思われる）$a = b$ に近似し，傾き a が大きい場合は（高負荷なホームページに多いと思われる）$b = 0$ に近似できたとすると，(4) はそれぞれ次のような a，b に依存しない単純な形となり，それほど精度を必用としない，もしくはストレステストを実施できない場合に利用できそうである．

$$Load_{low} = \frac{X+1}{2} \qquad (8)$$

$$Load_{high} = X \qquad (9)$$

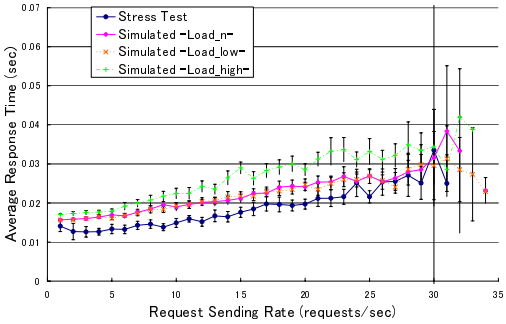

図 4: リクエスト送信レート毎の比較（A）

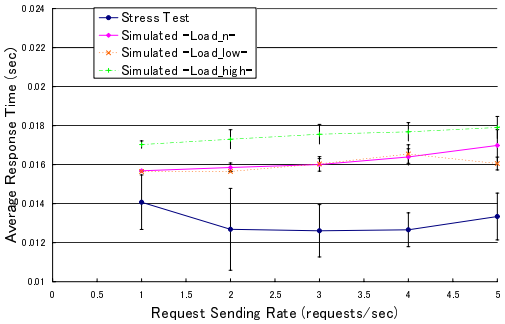

図 5: 拡大図（A）

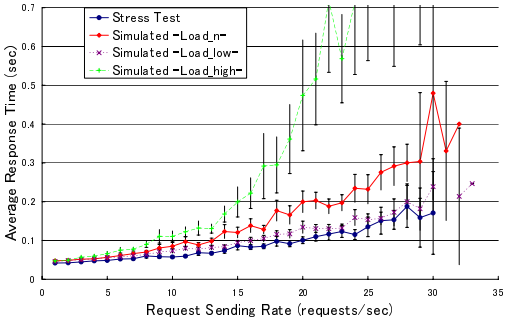

図 6: リクエスト送信レート毎の比較（B）

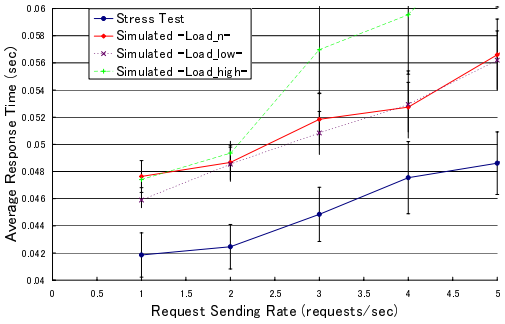

図 7: 拡大図（B）

4　モデル化方法の検証

4.1　モデルの設定

本稿で提案する Web サーバ処理性能モデル化方法の検証を行うために，OPNET におけるサーバノード内のサービスタイム計算部分を修正し，ホームページを構成するオブジェクト種別（静的か動的か）毎に ST_{static}，$ST_{dynamic}$ を使い分けるよう実装した．また，処理競合のモデル化は，競合処理数の増加・減少のタイミングで残っている競合処理全ての残り処理時間を再計算するようになっているため，その部分で測定された $Load$ に従って再計算を行うように設定した．ここで，ST_n 計算式中で確率的変動要素となっているのは，OH が分散 $OH^2/100$ の正規分布に従うという部分のみとし（デフォルト設定），競合処理数に依存する部分は確定的としている．

最後に，サーバ，クライアントノードの TCP 設定は標準設定項目の中から測定環境と同じものを選択し，OS の仕様に合わせた再送設定，バッファサイズ等を模擬させている．

4.2　検証実験の概要

実運用環境を想定したリクエスト到着間隔の確率変動を考慮したかったが，簡単のためレスポンスを受信してから 1〜3 秒の一様分布で次のリクエスト送信を繰り返す仮想ユーザ設定で，30 秒毎に 1 人から 50 人まで増加させるシナリオを設定し検証用負荷実験を実施した．また，OPNET にて上述の修正と設定を行ったサーバノードと 50 台のクライアントノードを用意し，各クライアントのリクエスト送信間隔を 1〜3 秒の一様分布に設定して 30 秒毎に 1 台から 50 台まで増加するシナリオ設定で検証用シミュレーションを実施した．ここで，両ツールの機能上の制限からリクエスト送信間隔を全く同設定にすることができなかったため，リクエスト送信レート（requests/sec）毎の平均レスポンスタイムを求めて結果の検証を行った．

4.3　結果の比較

第 3 章で例として挙げた 2 種類のホームページについて，検証用負荷実験結果とシミュレーション結果の平均レスポンスタイムの分布を図 4，6 に示す．また，両図における 0〜5[requests/sec] 部分を拡大したものを図 5，7 に示す．なお，各リクエスト送信レートにおける平均レスポンスタイムの 95%信頼区間も同時に記している．シミュレーション結果として，$Load$ 設定に（4）（8）（9）を用いた 3 例を記しており，OPNET 標準モデルの結果は省略している．これは，提案するモデル化方法での OH の扱い方が標準モデルの概念と異なっており，本稿におけるパラメータ設定方法がそのまま適用できないからである．

まず，図 4，5 の静的オブジェクトのみの A ページの結果を見てみる．下から負荷実験結果，$Load_{low}$，$Load_n$，$Load_{high}$ で設定したシミュレーション結果となる．グラフ全体の傾向を見ると，測定値ベースに設定した $Load_n$ の結果が検証用に実施した負荷実験結果よりも高く差が見られるが，リクエスト送信レートが増加した時の平均レスポンスタイム増加の傾向は類似しており，処理性能劣化の様子はうまく

模擬できているといえる．図5より，差の見られる原因は基本処理性能測定用実験と検証用負荷実験の違いによる誤差や測定誤差の影響と考えられ，レスポンスタイムが数十 msec のオーダと非常に小さいことを考えれば十分な結果であるといえる．また，両図から $Load_n$ と $Load_{low}$ で設定したシミュレーション結果がほぼ重なっていることから，A ページは $Load_{low}$ の設定で代用できるといえる．

次に動的なオブジェクトの含まれる B ページの結果を見てみる．下から負荷実験結果，$Load_{low}$，$Load_n$，$Load_{high}$ で設定したシミュレーション結果となる．$Load_{high}$ の結果が途中で発散してしまっているが，これはシミュレーション途中で Web サーバへのリクエスト到着レートが処理レートを上回ってしまったからである．また，測定値ベースに設定した $Load_n$ の結果より $Load_{low}$ で設定した結果の方が負荷実験結果に近いという結果になった．この原因として考えられるのは，A ページの場合と同様に基本処理性能を設定するために実施した実験と検証用負荷実験の 1 ユーザ時の結果に若干のずれがあり，負荷が増加するに従ってそのずれが大きく現れた結果ではないかと考えられる．このような基本処理性能測定時の誤差を考えれば $Load_{low}$ の設定でも十分利用可能と考えられる．

以上の結果から，本稿で提案する Web サーバ処理性能モデル化方法とパラメータ測定法で Web サーバの処理性能を高精度にモデル化でき，ネットワークシミュレーションによってエンドユーザレスポンスタイムを予測することができるようになる．また，基本処理性能の設定が負荷増加時の結果に影響を与えることに注意が必要だが，今回検証を行った 2 種類のページでは $Load_{low}$ で近似しても十分な結果を得られ，処理性能劣化度の測定ができない場合などへの利用が期待できる．

5 おわりに

本稿では，Web システムを利用するエンドユーザの感じるレスポンスタイムを高精度に予測するための Web サーバ処理性能モデル化方法を検討した．そして，Web サーバの返送するオブジェクト生成方法に応じてサービスタイムの計算方法を変化させ，比較的単純な測定結果や近似値をモデルに設定することにより，基本処理性能と競合時の処理性能劣化の様子を模擬したネットワークシミュレーションを行えることを示した．

本モデル化方法を用いて Web サーバの処理性能を考慮した性能評価ができるだけでなく，ホームページを構成するファイル数やファイルサイズを変化させた場合の評価も可能である．また，ネットワーク更改の影響まで含めたシステム全体の性能評価や，新規システム導入の影響評価にも利用でき，システム設計時や開発初期の指針決定に役立てることができる．

今後の課題として，今回検討した以外のホームページ構成に対する本モデル化方法の適用と検証を始め，Web アプリケーションサーバや DB サーバ，キャッシュサーバ，ロードバランサなどを含む現実的なシステム構成に対する適用範囲の拡大，物品の購入を模擬した典型的な商用シナリオとして規定されている TPC-W[9] や代表的なワークロード [10] を考慮したストレステストを用い，様々な処理が同時に発生する場合への適用可能性などについて検討を進めていきたい．

参考文献

[1] Netcraft: *Web Server Survey* (2002). http://www.netcraft.com/Survey/Reports/.

[2] Menascé, D. A. and Almeida, V. A. F.: *Scaling for E-Business*, Prentice Hall (2000).

[3] Wells, L., Christensen, S., Kristensen, L. M. and Mortensen, K. H.: Simulation Based Performance Analysis of Web Servers, *Petri Nets and Performance Models*, pp. 59–68 (2001).

[4] Kant, K. and Won, Y.: Server Capacity Planning for Web Traffic Workload, *IEEE Transactions on Knowledge and Data Engineering*, No. 5, pp. 731–747 (1999).

[5] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1, *RFC2616* (1999).

[6] Padhye, J., Firoiu, V., Towsley, D. F. and Kurose, J. F.: Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation, *IEEE/ACM Trans. Networking*, Vol. 8, No. 2, pp. 133–145 (2000).

[7] Barakat, C.: TCP/IP Modeling and Validation, *IEEE Network*, Vol. 15, pp. 38–47 (2001).

[8] Heidemann, J., Obraczka, K. and Touch, J.: Modeling the Performance of HTTP Over Several Transport Protocols, *IEEE/ACM Trans. Networking*, Vol. 5, No. 5, pp. 616–630 (1997).

[9] TPC: *TPC-W*. http://www.tpc.org/tpcw/.

[10] Arlitt, M. F. and Williamson, C. L.: Internet Web Servers: Workload Characterization and Performance Implications, *IEEE/ACM Trans. Networking*, Vol. 5, No. 5, pp. 631–645 (1997).