2004－EVA－11　(2)
2004／11／4

汎用観測データを用いたユーザレスポンス予測のための性能モデル作成手法

大河原　英喜[†]　河場基行[†]　西川克彦[†]

近年，ITシステム全体の性能予測を行うためのシミュレーション技術が注目されている．高精度の性能予測を行うためには，適切なワークロードの定義が重要であり，運用環境やテスト環境における観測データから，サーバ間のトラフィックフローや必要サーバ資源情報を抽出する必要がある．本論文では，sar等のサンプリング情報に加えてtcpdump等のトラフィックデータも用いることで細かいユーザアクション毎の必要CPU資源情報を抽出して，詳細なワークロード定義を行う手法について検討を行った．定義したワークロードモデルの精度検証を行った結果，ユーザアクション毎の平均レスポンスタイムが実測に対して10%程度の誤差で得られた．

A Server System Modeling Method for Predicting Detailed User Action Response Time with General Measured Data

Hideki Okawara,[†] Motoyuki Kawaba[†]
and Katsuhiko Nishikawa [†]

Prediction of IT system performance is a common theme and simulation methods have been studied. For high-accurate evaluation it is significant to define appropriate workload, traffic flow between servers and required server resources. This paper describes our study to analyze generally measured data and extract required CPU resource for each detailed user actions. We analyze both CPU sampling data(sar) and traffic flow data(tcpdump) measured on real system or test system. We define WEB application workload model including 6 user actions and compare simulation results with measured each user action response time.

1. はじめに

近年，ITシステムの重要性が増すにつれて，機能実現のみならずユーザアクションのレスポンスタイム等の性能要件の実現への要求が高まっている．資源不足による性能問題や，過剰な資源投入によるコスト増を防ぐためには，十分な性能評価を行って適切な資源量のシステム構築を行う事が必要である．性能評価の際，精度の面だけを考えれば、実アプリケーションが稼動している運用システムやテスト環境上で性能を実測することが望ましいが、現実には稼動システムで任意の負荷をかけるのが困難であったり、時間的制約等により十分な実機テストを行うことができない事も多い．また，システム開発へのフィードバックという観点では，より初期段階で性能評価を行いたいが，実機テストが行えるのはアプリケーションの実装が完了した開発後期になってからである．そこで、運用システムないし同様な業務の別システム上で実測可能なデータを基に，ワークロードの理論式モデルないしシミュレーションモデルを作成し，仮想的にITシステムの性能見積りを行う技術が必要となる．

既存のITシステムシミュレーション製品[1),2)]では，図1の様なワークロードモデルを作成してシミュレーションを行う．ワークロードモデル作成には，汎用ツールで観測可能な以下のデータを用いる．

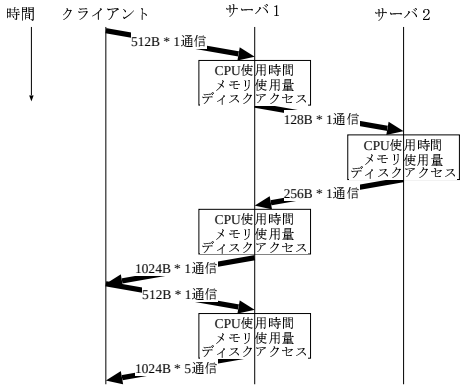

図1　ワークロードモデル概要

† 富士通研究所
FUJITSU LABORATORIES LTD.

ネットワークトレース　tcpdump や windump 等で測定可能な、サーバ間で発生する通信パケットをキャプチャしたトレースデータ．通信の回数・データサイズや処理タイミングを抽出する．解析容易性を考慮して，一多重でのデータ測定が望ましい．

サーバ資源使用情報　Windows の PerformanceMonitor や UNIX 系の sar,ps,acct 等で測定可能なサーバ資源情報のサンプリングデータ．BMC Patrol や Concord SysEDGE などの商用ツールを用いても良い．これらのツールでの測定では、システムへの影響を考慮して数十秒〜数分程度の周期でサンプリングが行われるのが一般的である．

ワークロードモデル作成は図 2の手順で行われ，ユーザアクション等のシミュレーション対象の処理単位毎に切り分けてモデルを定義する．

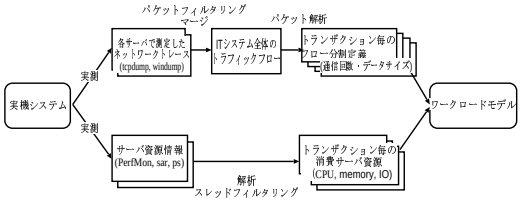

図 2　ワークロードモデル作成手順

本論文では、一般的な WEB アプリケーションシステムを対象に、上記の汎用観測データを基に，ユーザアクション毎のワークロードモデルを作成することを考える．ネットワークフローモデルに関しては、一多重でネットワークトレース測定を行い、ポート番号や実行時間などによるパケットフィルタリングを行う事で、ユーザアクションの切り分けは比較的容易にできる．しかし，サーバ資源情報に関しては、高負荷でも数秒以内のレスポンスタイムが求められるユーザアクション毎の情報抽出は，数十秒〜数分周期のサンプリングデータからでは困難である．そこで本論文では、サーバ資源のうち特に支配的性能要因である CPU 使用時間に着目し、測定したサーバ資源情報とネットワークトレースの両方とを併せて、ユーザアクション毎の CPU 使用時間を抽出する方法について検討する．

2. 既存製品及び関連研究

2.1 IT システムシミュレータ製品

既存の IT システムシミュレーション製品として、OPNET[1] や HyPerformix[2] が知られている．これらでは、CPU やディスク、ネットワーク機器などをキューとした待ち行列シミュレーションを行う．個々の構成要素の内部アーキテクチャが把握できれば，より詳細なハードウェアモデルを作成して高精度シミュレーションも可能ではあるが，近年の IT システムのオープン化・多様化の中では全てのハードウェアの内部詳細を知る事は困難であり，またモデル作成コストやシミュレーション時間とのトレードオフからもある程度は抽象度の高いモデルにならざるを得ない[3] のが現実である☆．

また，いずれの製品も，ワークロードモデル定義を支援するツールは存在するものの，トランザクション単位毎の CPU 使用時間などは人手で決定する必要がある．

2.2 IT システムシミュレータ応用に関する関連研究

2.1節で延べた IT システムシミュレータを用いた研究もいくつか発表されている．米国に目を向ければ OPNET 社，HyPerformix 社ともにユーザカンファレンス[4][5] を開き様々な発表がされている．日本国内でも，HyPerformix を用いたシミュレーション支援 GUI ツール[6] や実適用例[7] に関する論文が発表されている．WEB システムのワークロードモデル作成に関する研究としては，静的コンテンツと動的コンテンツに分けて定義した簡単な数式モデルを，OPNET に実装してシミュレーションを行った研究[8] が発表されているが，本論文の狙いとしている，実測データから複数トランザクションを抽出したモデル化は行っていない．

2.3 性能情報取得手法に関する関連研究

イベントトレース手法に基づく詳細なサーバ性能データ収集ツールの研究として，堀川らの mevalet[9] がある．この研究では，Linux カーネルに特別な hook を組み込むことで，各プロセスの割り込み処理イベントとタイムスタンプを記録する．イベントトレース方式では，高精度な詳細性能データ収集ができる半面，OS に手を加える必要があるため一般的に適用する事は難しい．

また，西岡らの論文[10] では，モデル作成のためのトランザクション単位毎の処理時間測定に要していた，環境構築・測定の工数を削減するために，独自開発の PC/AT システムハードウェアエミュレータ上で，仮想的にアプリケーション動作時のイベントトレースを採取した事例について紹介している．

これらの論文では，詳細なサーバ資源情報を測定するために独自の Linux カーネルやハードウェアエミュレータを用いており，その上でアプリケーションを動作させて測定可能な場合にのみ適用できる．本論文では，より一般的に適用可能なモデル作成手法として，sar と tcudump で得られる汎用観測データのみを用いてワークロードモデルを定義する方法を検討した．

3. 提 案 手 法

3.1 概　　要

測定区間におけるトランザクション i の発生回数 n_i，トランザクション当りの CPU 使用時間 τ_i，定常的なバックグラウンド CPU 使用時間 C とすると，単位時間のトータル CPU 使用時間 T は以下の式で現される．

$$T = \sum_{i=1}^{N} (\tau_i \times n_i) + C \qquad (1)$$

☆ 既存製品の特徴を比較すると，OPNET はネットワーク，HyPerformix はサーバ・ミドルウェアのモデルが詳細である

もし分割したいトランザクション毎の発生回数 n_i が独立で，十分な負荷バリエーションの測定点が存在すれば．式 1を解くことができる．しかし， n_i が同様な傾向を持つ場合や，十分な測定バリエーションが存在しない場合も存在する．

そこで，本論文ではネットワークトレースデータを解析して各トランザクションの見掛けのサーバ処理時間 τ_i' を算出し，それを CPU 処理時間としてワークロードモデル定義する事を考える．実際には，見掛けのサーバ処理時間 τ_i' にはディスク I/O などの影響も含まれるが，一般的に多くの場合は CPU がボトルネックである事から，ここでは見掛けのサーバ処理時間の大半が CPU 処理時間に現れると仮定し，極端に CPU 以外にボトルネックがあるトランザクションは含まないと考えている．

ワークロードモデルの精度は，いかに適切なサーバ処理時間をネットワークトレースデータから抽出できるかが鍵となる．算出値が妥当か否かの一つの指標として式 2が考えられる．この値が 1 に近ければ，見掛けのサーバ処理時間と実際の CPU 使用時間が近く，このワークロードモデル定義が妥当であると考えられるが，式 2の指標値が大きい場合には，本質的に見掛けのサーバ処理時間の誤差要因を追求する必要がある．

$$\frac{\sum_{i=1}^{N}(\tau_i' \times n_i)}{\sum_{i=1}^{N}(\tau \times n_i)} = \frac{\sum_{i=1}^{N}(\tau_i' \times n_i)}{(T-C)} \qquad (2)$$

また，見掛けのサーバ処理時間を如何に適切に見積もっても， CPU 以外のサーバやネットワークの影響で多少の誤差が含まれてしまう可能性もある．そこで，sar で測定されるトータル CPU 使用時間を守りつつ，トランザクション毎の CPU 使用時間を定義するための粗い近似として式 3が考えられる．式 3では，ネットワークトレースから算出したサーバ処理時間で, sar で測定されるトータル CPU 使用時間を比例配分することで，トランザクション毎の CPU 使用時間を定義する．

$$\tau_i = (T-C) \times \frac{\tau_i'}{\sum_{i=1}^{N}(\tau_i' \times n_i)} \qquad (3)$$

3.2 サンプリングツールの選択

サンプリング期間のトータル CPU 時間 T とバックグランド CPU 使用時間 C を求める方法として，大きく以下の 2 通りの求め方がある。いずれの方法でも，高精度の定義をするためにはある程度トランザクション数のある中〜高負荷時のデータを用いるのが望ましい．

スレッド毎の CPU 使用時間から求める方法 psやacctなどを用いてスレッド毎の CPU 使用を測定し、注目するトランザクション毎にフィルタリングを行って合計し、トランザクション発生回数で割る。バックグラウンド CPU 使用時間は、いずれのトランザクションにも属さない CPU 使用時間とする。

トータル CPU 使用時間から求める方法 sarなどを用いて、トータル CPU 使用時間を測定する。バックグランド CPU 使用時間を得るために負荷のかかっていない状況でも測定を行う．

前者のスレッド毎の CPU 使用時間の測定を行う場合には、図 3の 4 つのケースが考えられる。サンプリング時に稼動しているスレッドの情報を採取する ps などでは、図中のスレッド B、スレッド D の CPU 使用時間を測定することはできない。また、 acct ではプロセス終了時に情報が記録されるため、スレッド A やスレッド C の様な実行継続中のスレッドの情報は測定できない。この様に，スレッド毎の CPU 使用時間を累積する方法は、スレッドフィルタリングの手間がある上、測定誤差が考えられる。そこで、本論文では sar によるトータル CPU 使用時間から求める方法を採用する．

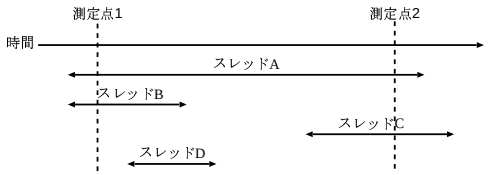

図 3 サンプリング期間におけるスレッド実行パターン

3.3 見掛けのサーバ処理時間の算出方法

ここでは、一多重で測定したネットワークトレースデータから，適切な見かけのサーバ処理時間を算出する方法を検討する．

トラフィックフロー作成方法 IT システム全体のワークロードモデルを作成するためには，複数箇所で測定したネットワークトレース群をマージして，一つのトラフィックフローモデルを作成する必要がある．通信を行っている両端で測定したネットワークトレースでは，図 4左に示す様に同一のパケットが異なる時間タイミングで観測される．そのため，複数のネットワークトレースをマージする際に，この各サーバ間で重複して観測されるパケットをどう扱うかによって，処理発生タイミングや見掛けのサーバ処理時間の算出値に違いが現れる．

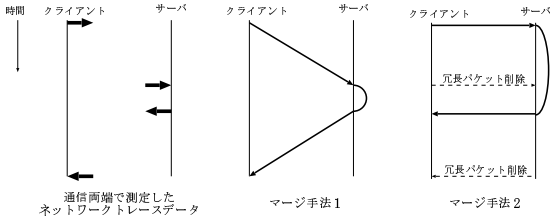

図 4 複数箇所のネットワークトレースのマージ方法

図 4のマージ手法 1 の様に，ネットワークの両端の時間情報を考慮したモデルを解析した方が，サーバ処理時間の算出は正確である．それに対し，手法 2 の様に，片側のパケットのみを残す形でマージしてから解析してしまうと，サーバ処理時間の見積りと

しては誤差が生じてしまう．この違いは，測定環境のネットワーク帯域・レーテンシの性能が劣る場合に，より顕著に現れてしまう．本論文では，ネットワーク遅延の影響まで考慮するために，マージ手法１を採用することにした．

サーバ処理時間の考え方　ネットワークトレース中の，依存関係のある一連のパケット群のやりとりの中で，どの区間をサーバ処理時間とするかによっても算出結果に違いが現れる．図 5では，HTTP 要求も HTML データ送信も複数のパケットにまたがる例を示している．

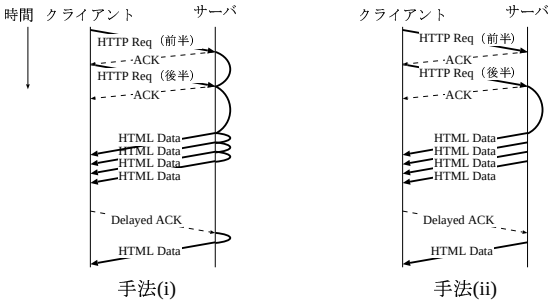

図 5　サーバ処理時間の考え方の違い

図 5の算出方法 (i) では，最初の HTTP 要求パケットを受信してから最後の HTML データ送信までの区間に関して，パケットの依存関係を詳細に解析しながらサーバ処理時間を算出している．この方法では，ネットワーク通信性能の制約で連続パケット送信に時間がかかっている場合に，サーバ (CPU) 処理時間としては過剰に見積もってしまう可能性がある．そこで，測定環境のネットワーク性能を基に，各パケットの送信処理にかかる時間を見積もり，見掛けのサーバ処理時間を補正する必要がある．

一方，算出方法 (ii) では，一連処理のデータ送信の方向が変化する前後のパケットのみに注目している．すなわち，最後の HTTP 要求パケット受信から最初の HTML データ送信までの区間をサーバ処理時間としている．この方法では，同方向のパケット連続送信の間に CPU を消費していた場合には，実際より小さく見積もられる可能性がある．

この両手法はどちらにも誤差要因が含まれるが，ネットワークトレース解析が適切に行えれば算出方法 (i) の方が精度が高いと判断し，本論文では算出方法 (i) を用いた．ネットワーク処理時間の補正については，パケットデータサイズをネットワークスループット (bps) で割った値を見掛けのサーバ処理時間から引くという単純な補正のみを行った．

同時実行処理の取り扱い　今回の様に一多重でネットワークトレースを測定しても，HTTP 処理における parallel GET の様に複数の要求が同時発生される事がある．例えば，図 6の様に４つの HTTP 要求がオーバーラップして発行された場合，ネットワークトレース情報のみから見積もろうとすると，処理の重なり具合いを考慮せずに何らかの処理がサーバ側で処理されている時間を求める算出方法 a か，各コネクションの要求〜応答間の時間を累積する算出方法 b が考えられる．

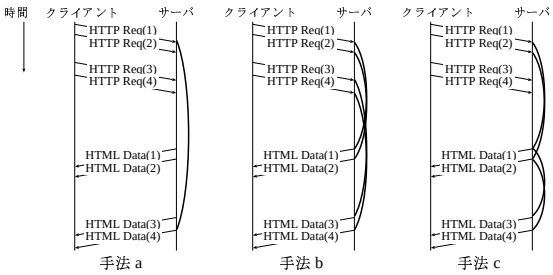

図 6　HTTP parallel GET の取り扱い

もしサーバの CPU が一つであれば手法 a は正しいが，サーバに複数の CPU が存在して並列実行された場合には実際より小さい値になってしまう．一方，CPU 数の範囲で同時要求が発生していれば手法 b は正しいが，CPU 数以上の同時要求が発生して CPU 処理待ちが発生している場合には，実際の CPU 使用時間より大きな値となってしまう．そこで本論文では，測定環境のサーバ CPU 数も考慮した算出手法 c を用いた．この手法では，各サーバの CPU 数までの重なりは累積するが，それ以上の重なりについては無視をする．図 6では 2CPU での算出例を示している．

4. 評　　価

4.1 評 価 条 件

本節では，以下の特徴を持つアプリケーションの実測データについてワークロードモデル定義を行い，本論文の算出方法の妥当性を検証する．

- 図 7に示すテスト環境での測定データ
- １ユーザの処理は処理 A 〜処理 F の 6 つの処理を順番に実行する．
- 各処理は主に CPU を消費し，特に WEB サーバの CPU に負荷がかかる．
- サーバ間のネットワーク遅延は十分に小さい．

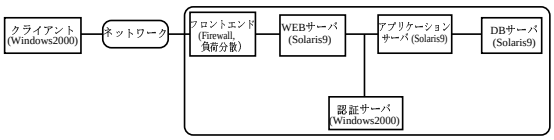

図 7　測定システム概要

本論文の見掛けのサーバ処理時間の算出方法との比較として，表1に示す比較手法 A，比較手法 B を用いた評価も行う．比較手法 A はパケット解析は詳細に行うが同時実行要求の重なりを二重に考慮しない，よりネットワークからの観点から見た解析手法となっている．一方，算出手法 B は詳細なネットワーク解析は行わず，同時実行要求の重なりは全て累積する手法となっている．既存製品である OPNET や HyPerformix においても，ワークロードモデル作成支援のための解析は行っているが，見掛けのサーバ処理時間☆は比較手法 A ないし比較手法 B の算出結果と同様の値を出している．

表 1　評価するサーバ処理時間算出手法

	提案手法	比較手法 A	比較手法 B
トラフィックフロー作成	手法 1	手法 1	手法 2
サーバ処理時間の考え方	手法 (i)	手法 (i)	手法 (ii)
同時実行処理の取り扱い	手法 c	手法 a	手法 b

4.2　見掛けのサーバ処理時間の算出結果

提案方法および比較手法 A，比較手法 B を用いて，見掛けの WEB サーバ処理時間の算出を行った結果を表2に示す．また実測の 1 ユーザ当り（処理 A ～処理 F）の合計 CPU 処理時間を示す．

表 2　各手法で算出したサーバ処理時間と実測 CPU 時間（秒）

	提案手法	比較手法 A	比較手法 B	実測
処理 A	0.26498	0.16031	0.47216	
処理 B	0.05152	0.03327	0.05585	
処理 C	0.03192	0.02345	0.03869	
処理 D	0.01948	0.01615	0.01416	
処理 E	0.01834	0.01834	0.01361	
処理 F	0.01080	0.01080	0.00769	
合計	0.39704	0.26232	0.60216	0.35556

1 ユーザ処理の合計時間を見ると，本提案手法での算出値は実測 CPU 使用時間の約 110% と若干大きいものの，式 2の指標値が 1 に近い値となっている．それに対して，算出手法 A では約 73%，算出手法 B では約 170% と大きくずれている．また，既存製品での算出値でも同様のずれが認められた．

個別の処理に着目すると，処理 A ～処理 C では比較手法A ＜ 提案手法 ＜ 比較手法B となっている．これは図 6の同時実行処理 (parallel GET) の取り扱いによる影響である．また，処理 E ～処理 F では比較手法B ＜ 提案手法 ＝ 比較手法A となっているが，これは図 5のサーバ処理時間の考え方の違いによるものである．処理 D では両方の影響を受けている．

4.3　シミュレーション結果と実測値の比較

本節では，表 2の各手法で算出したサーバ処理時間および，提案手法での算出値を数式 3の様に比例配分補正した値を，それぞれ CPU 使用時間としてワークロードモデルを定義し， OPNET でシミュレーションを行った結果を示す．ここでは，ワークロードモデル作成手法の妥当性を確認するために，実測環境と同じシステム構成のシミュレーションを行い，実測値とシミュレーション結果の比較を行う．図 8に，多重度（ユーザ数）が変化した場合の処理 A と処理 B の平均レスポンスタイムを示す．

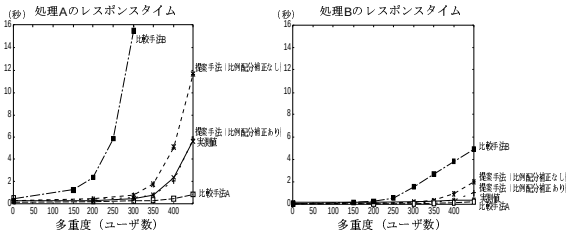

図 8　シミュレーション結果（処理 A, 処理 B のレスポンスタイム）

レスポンスタイムの悪化が最も著しい処理 A に関しては，比較手法 A, 比較手法 B では性能曲線が実測と大きく異なるが，提案手法では 10% ほど横軸方向にずれる程度の誤差に納まっている．さらに，提案手法に対して比例配分補正を行った場合には，シミュレーションと実機がほぼ同じ性能曲線を示しており，高精度でレスポンスタイムの延びが予測されている．

処理 B に関しては，実機では多重度が上がってもレスポンスタイムが悪化する傾向が小さいが，提案手法モデルのシミュレーションではレスポンスタイムが悪化する様子が現れ始めている．この違いは，ハードウェアモデルやジョブスケジューリングモデルの精度の問題や，待ち行列で表現しきれない実アプリケーションの特性の影響などが考えられる．レスポンスタイムの短い処理 C ～ F に関しても同様の傾向が得られた．

5.　考　　察

5.1　マージ手法による影響について

4節の測定システムはネットワーク遅延が小さかったため， 3.3節で述べたマージ手法による誤差が現れなかった．そこで，マージ手法による誤差が現れた例として，クライアントと WEB サーバ間が実効帯域 600kbps, レーテンシ 10ms のシステムで測定したデータを解析した結果を表 3に示す．このシステムでは， 60 種類程の非常に多くのトランザクションが混在していたため，ここでは特徴的な一処理のみを取り上げている．

表 3　ネットワーク遅延が大きい場合のサーバ処理時間算出値 (秒)

	提案手法	比較手法 A	比較手法 B
処理 1	0.016211	0.016211	0.694583

☆ OPNET では processing delay, HyPerformix では loacl time と読んでいる

マージ手法 1 を用いている論文の提案手法および比較手法 A では見掛けのサーバ処理時間が 16ms 程度なのに対して，マージ手法 2 を用いている算出手法 B では 700ms と非常に大きな値が算出されている．また，全トランザクションに関して見掛けのサーバ処理時間を算出した結果，式 2に示される誤差指標値は約 130% 程度となった．4.2節の結果よりは，実測 CPU 使用時間との差が大きいが，これはネットワーク処理時間の補正の影響がより大きいという点などが関係していると考えられる．

5.2 ネットワーク遅延の詳細見積りについて

各サーバでネットワークトレースが測定でき，かつ通信の両側で同一パケットの対応ができる場合には，図 4で示した様に両側の時間情報を保持したまま解析することで，ネットワーク遅延情報が得られる．しかし，以下の場合には両端でのパケットの対応がとれず，何らかの方法でネットワーク遅延を見積もってパケット送受信タイミングを調整する必要がある．

- SSL 通信が行われていて，通信途中の SSL Accelerator で複合化されている場合
- 負荷分散装置などによる中継時にパケットが書き換えられる場合
- サーバ側の制約で片側しか測定できない場合

この場合，片側のパケットのみから帯域幅やレーテンシの実測値を基に遅延補正を行う事になるが，ネットワーク遅延が大きい場合や，補正してパケット順序関係が逆転した場合には，大きな誤差を生じる可能性があるので注意が必要である．例えば， ACK 受信と次の要求送信の順序関係が数 msec 単位で逆転してしまい，nagle アルゴリズムによる 200msec 程度の ACK 待ち時間が誤ってサーバ処理時間として算出されてしまったケースが存在した．

6. ま と め

本論文では， IT システム全体の性能予測・シミュレーションを行うために，運用システムないしテスト環境で実測可能な汎用観測データを基に，ワークロードモデルを作成する方法について検討を行った．複数のトランザクションが混在するアプリケーションを定義するために，低負荷時のネットワークトレースからトランザクション毎の見掛けのサーバ処理時間を見積もる方法を検討した結果，ネットワークトレースのマージ手法や，同時要求処理の扱いに注意する必要がある事がわかった．実際にモデル作成を行った結果，実測 CPU 使用時間と見掛けの処理時間算出値の誤差が 10% 程度に抑えられた．さらに，実測 CPU 使用時間を見掛けの処理時間で比例配分する補正を行う事によって，シミュレーション結果でもレスポンスタイムが高精度に得られた．

ワークロードモデル定義に関する今後の課題としては以下の事項が考えられる．

- 提案手法によってサーバ処理時間と CPU 使用時間の誤差を抑えられたが，依然として 10% 程度の誤差が含まれる場合がある．ネットワーク遅延をより詳細に補正する手法を検討する必要がある．
- 本論文では，見掛けのサーバ処理時間に占める CPU 時間の割合は非常に大きいと仮定して検討を行った．多くの場合はこの仮定が当てはまると思われるが，一部アプリケーションでは CPU 以外の影響が見えてくる可能性がある．例えば， I/O 待ちやスワップが頻発するアプリケーション等をモデル化・性能予測する場合には別途検討が必要である．
- 本論文では，一多重でのネットワークトレースデータを解析しているが，実際には複数トランザクションが重なりあっているトレースデータしか入手できない場合も考えられる．より複雑なトレースデータからの情報抽出を検討する必要がある．
- 今回は，ネットワークトレースデータから見掛けのサーバ処理時間を抽出したが，アプリの性質や入手可能データによっては，回帰分析などの数学的手法との併用や使い分けの検討も必要である．

また，今回はシミュレーション結果の精度そのものの議論は行わなかったが，キャパシティプランニング等に実適用していくためにはシミュレーションや理論式による性能予測手法の妥当性についても検証・改良をしていく必要がある．

参 考 文 献

1) OPNET. http://www.opnet.com/.
2) HyPerforix. http://www.hyperformix.com/.
3) 岡田昭宏. サーバ処理性能の評価方法に関する一検討. 情処研報,2004-EVA-9, pp. 47–52, 2004.
4) OPNETWORK2004. http://www.opnet.com/opnetwork2004/home.html.
5) HyPerformix Annual Conference. http://www.hyperformix.com/Default.asp?Page=113.
6) 西岡大祐, 今木常之, 長澤幹夫. Wise ツールによる情報システム構築支援. 情処研報,2001-EVA-1, pp. 1–6, 2001.
7) 渡辺聡, 鈴木芳生, 長澤幹夫. 統合運用ソフトウェア jp1 の性能評価. 情処研報,2001-EVA-1, pp. 7–12, 2001.
8) 峰野博史, 川原亮一. エンドユーザレスポンス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. 情処研報,2002-EVA-3, pp. 37–42, 2002.
9) 堀川隆. 性能基礎データ収集方法の比較検討. 情処研報,2004-EVA-10, pp. 1–6, 2004.
10) 西岡大祐, 亀山伸, 垂井俊明, 庄内亨. 情報システム性能見積もりのためのシミュレーションモデル作成方法の開発. 情処研報,2004-EVA-10, pp. 7–12, 2004.