Webシステムの性能評価に関する一手法

金木佑介[†] 飯塚剛[†] 浅見可津志[††]

近年，Web システムのアーキテクチャは生産性や保守性を向上させるため，フレームワークやミドルウェアによって多層化された構成となっている．そのため，システムは複雑化し，多様な要素が，Web システムの性能に影響を与えるようになってきた．本稿では，このように複雑化した Web システムの性能設計を支援するため，多層化したアーキテクチャを評価モデルに適用することによって，各層のコンポーネントが性能に与える影響を比較可能な性能評価手法を提案する．また，本提案手法を実装したベンチマークツールの構成を示し，それを使用してデータアクセス層に利用するフレームワークの性能比較を行った例も示す．

An Evaluation Method for Web System Performance

Yusuke Kaneki,[†] Tsuyoshi Iizuka[†] and Katsushi Asami[††]

Recently, Web systems have been constructed of framework and middleware based on multilayered architecture in order to improve productivity and maintainability. They consist of various components which affect system performance. In this paper, we propose an evaluation method of Web system performance to compare performances of components on each layer by applying the multilayered architecture to the evaluation model. This paper also shows the details of a benchmark tool using the proposed method and an example of performance evaluation for components of data access layer with the benchmark.

1. はじめに

近年，インターネットの活用はサービスを提供する手段として標準的な手法となり，企業活動においても，企業サイトやポータルサイト，e コマース等の様々なサービスが Web システムを介して提供されている．そのような中，企業活動における Web システムの役割がますます大きくなっており，その開発現場では生産性と性能の両立が非常に重要な課題となっている．これに対して，Web システムを構築する技術も急激な勢いで進歩しており，生産性や保守性，再利用性を向上させるため，Web アプリケーションのアーキテクチャは**図 1** のように，ミドルウェアやフレームワークを多用することによって，多層化されたアーキテクチャが採用されている．さらに，各層で実用可能なオープンソース・ソフトウェア（OSS）が多く存在し，コスト削減を目的として，それらの活用も重要となっている．

このように，アーキテクチャの多層化と各層で選択可能なミドルウェアやフレームワークの増加から，Web システムを構築するスタックは幾通りもの組合せが存在し，複雑化している．そのため，システム開発の現場では，それらを適切に選択し，生産性やコスト，性能のバランスをとっていかなくてはならない．しかし，性能に影響を及ぼす要因が多岐にわたるため，Web システムの性能設計は非常に困難となっている．

そこで，本稿では，複雑化した Web システムの性能設計を支援するため，各層のコンポーネントを比較可能な性能評価手法を提案する．以下に本稿の構成を示す．2 章では，Web システムの性能に関する既存研究や評価手法について述べ，その課題を明らかにする．3 章では本提案手法について述べる．そして，4 章では本提案手法を実装したベンチマークツールの構成について述べ，5 章には，実装したベンチマークツールを使用して，データアクセス層で利用するフレームワークの性能比較を行った例を示す．最後に 6 章で，本稿をまとめる．

2. 従来の性能評価手法

Web システムの性能評価手法には大きく分け，2 つの手法がある．1 つは，Web システムをモデル化し，それを入力とするシミュレータ[1),2)] を用いて，擬似的

† 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
†† 三菱電機インフォメーションシステムズ株式会社
Mitsubishi Electric Information Systems Corporation

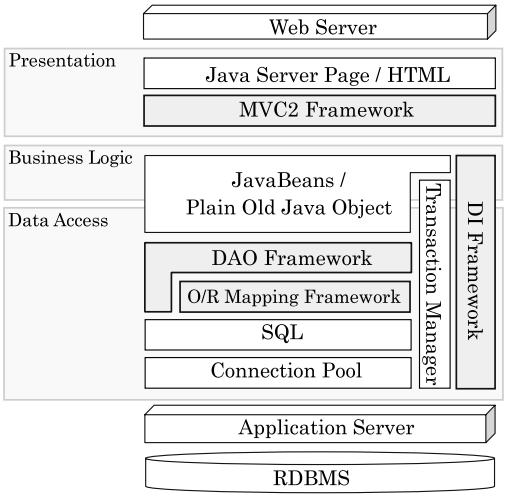

図 1 Web アプリケーションアーキテクチャ
Fig. 1 Web application architecture.

な性能値を取得し，評価する手法，もう 1 つは，実際の Web サーバ等のミドルウェアや簡略化された Web アプリケーションを用いて，実際のシステムに近い評価用システムを構築し，性能評価を行う手法である．

モデル化を行う手法には次のような検討がなされている．文献 3) では，CPU やメモリアクセス等のハードウェアとソフトウェアの動作を把握し，分析することのできるシステム・ハードウェアエミュレータを利用し，対象となるアプリケーションをその上で動作させ，そのデータを基にシミュレーションモデルを作成，Web システムの性能予測を行っている．文献 4) では，Web システムの処理性能を基本処理性能と競合時の処理性能劣化度という 2 つの観点からモデル化し，ネットワークシミュレータ OPNET [1] を用いて HTTP のトランザクションを模擬，Web システムの性能を予測している．ほかにも文献 5)〜7) のような研究がなされている．これらは，Web システムの性能予測に対しては有用だが，あくまでシミュレーションであるため，実際のフレームワークやミドルウェアによる性能への影響を知ることは難しい．

実システム相当の評価を行う手法には，次のような実アプリケーションを利用したベンチマークがある．SPECjAppServer [8] は，自動車製造業者とその関連ディーラのやりとりをシミュレートする J2EE アプリケーションである．EJB を使用するため Tomcat 等のアプリケーションサーバでは動作しない等，ミドルウェアへの依存性が高く，評価対象が限定されるという問題点がある．JBento [9] は，HttpSession レプリケーション性能，HTTP リクエスト性能，JNDI 参照性能といったアプリケーションサーバの個々の機能に着目した J2EE マイクロベンチマークである．特定の J2EE アプリケーションサーバに依存しない作りになっている．本稿が目的とするマクロなベンチマークを行うためのアプリケーション（JBentoStore）も存在するが，アプリケーションサーバの性能評価を目的としており，データベースアクセス部分が省略されているため，データアクセス層のコンポーネントを性能比較することはできない．Web3 階層を評価モデルとしたベンチマーク手法には OSDL-DBT-1 [10] がある．OSDL-DBT-1 は，TPC-W [11] 相当のトランザクションを模擬しデータベースサーバの性能評価を行うベンチマークツールである．Web3 階層システムを評価モデルとした性能評価が可能であるが，アプリケーションサーバに擬似的なものを使用しており，フレームワーク等，アプリケーション層のコンポーネントを性能比較することはできない．

表 1 従来の性能評価手法との比較
Table 1 Comparison with conventional methods of a performance evaluation.

性能評価対象		性能評価手法が対象とする範囲		
ソフトウェア	フレームワーク			○
	ミドルウェア		○	○
	アプリケーション	○	○	○
ハードウェア		○	○	○
性能評価手法		参考文献 3, 4, 5, 6, 7	参考文献 8, 9, 10, 11	提案手法

これら従来の性能評価手法を**表 1** にまとめた．

3. 提案手法

表 1 に示すように，多くの性能評価手法が検討されているが，多層化された Web システムを構成するフレームワーク等の各コンポーネントを性能比較可能な評価手法はない．

そこで本稿では，次のような特徴を持つ Web システムの性能評価手法を提案する．(1) Web3 階層システム全体の性能を評価可能である．(2) 評価モデルにミドルウェアやフレームワークを実際に使用し，実システム相当の環境で性能を評価できる．(3) 評価モデルに使用するミドルウェアやフレームワーク等のコンポーネントを低い実装コストで交換可能であり，評価モデルの拡張性が高い．

本稿で提案する手法は，**図 2** に示すようにクライアント層，アプリケーション層，データベース層の Web3 階層モデルで構成されるベンチマークツールである．アプリケーション層は，Web サーバやアプリケーションサーバ，その上で動作する Web アプリケーションで構成される．データベース層は，データベースサーバ

と Web アプリケーションがアクセスするデータベースで構成され，アプリケーション層とデータベース層でベンチマークモデルを構成している．クライアント層では，ベンチマークモデルに対して負荷を与えるシミュレータが動作する．また，各層のサーバにはシステム負荷を測定する sar や iostat コマンドが動作する．

ベンチマークモデルは，実装言語に Java を採用し，図 1 に示すミドルウェアやフレームワークを多用した多層構造の Web アプリケーションとする．ミドルウェアやフレームワークを比較可能とするため，できる限り，特定のミドルウェアやフレームワークに依存しない作りとする．

ベンチマークの設定には，ベンチマークモデルに関する設定（コンポーネントの導入やデータベースパラメータ）とシミュレーションに関する設定（シミュレーションパラメータ）がある．

ベンチマークの動作を次に説明する．まず，クライアント層のシミュレータがスレッドを複数生成，それぞれがベンチマークモデルに対して HTTP リクエストを発行する．発行された HTTP リスクエストをアプリケーション層が受けると，リクエストに応じてデータベース層への JDBC によるアクセスが行われ，その結果がアプリケーション層を経由し，シミュレータに HTTP レスポンスとして返るという動作をする．

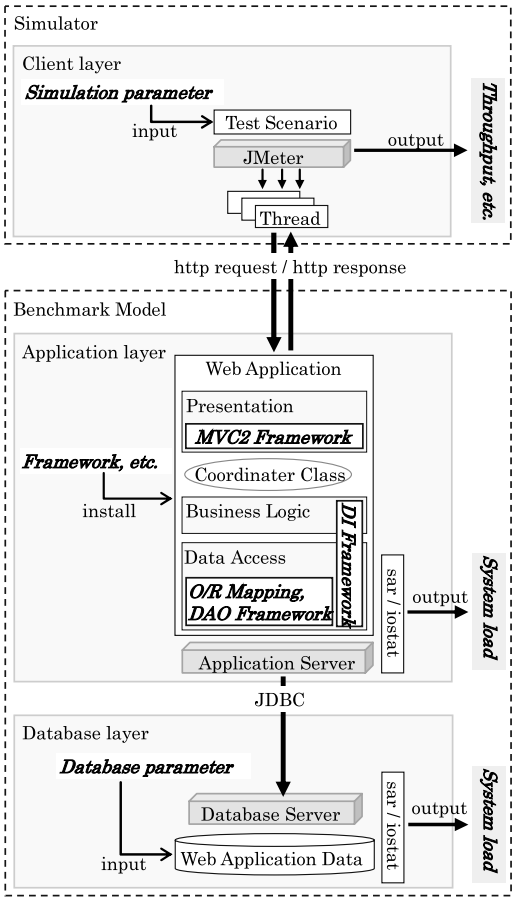

図 2 提案手法の全体像
Fig. 2 Overview of proposed method.

本提案手法が定義する Web システムの性能は，シミュレータから見たときのスループットや平均応答時間と，ベンチマーク時の各サーバに発生したシステム負荷である．ベンチマークテストは，シミュレータが生成するスレッド数を増やしていき，アプリケーション層，データベース層のいずれかのサーバが処理限界となり，スループットが横這いとなるまで行い，各スレッド数ごとのスループットや平均応答時間，システム負荷を測定する．

4. ベンチマークツールの構成

4.1 シミュレータ

クライアント層のシミュレータは，シミュレーション機能と性能計測機能から構成される．シミュレーション機能では，与えられたシミュレーションパラメータに応じて，ベンチマークモデルである Web アプリケーションを操作するユーザを模擬し，システムに実運用相当の負荷を与える．性能計測機能では，擬似ユーザが各 Web ページにアクセスする際の平均応答時間やスループット等を計測する．

シミュレータは，オープンソースの Web アプリケーション用テストツール JMeter [12] を利用して実装した．JMeter は，プログラム可能なテストシナリオに従って HTTP リクエストを送信，その際のレスポンス性能やスループットを計測および表示することのできるツールである．シミュレーションロジックを JMeter のテストシナリオとして作成し，測定結果は JMeter の表示機能を利用して表示する．シミュレータに与えるパラメータを**表 2** に示す．

browsers は，ベンチマークモデルに対して何人のユーザがアクセスしたときの性能を測定するかを示し，システムに与える負荷に直接影響するパラメータであ

表 2 シミュレーションパラメータ
Table 2 Simulation parameters.

項目	説明
browsers	シミュレータが生成する擬似クライアント数
customers	生成するクライアント ID の最大値
items	生成する商品 ID の最大値
thinktime	トランザクションの間隔
duration	シミュレーションを行う時間
transition rate	Web ページ間での遷移比率

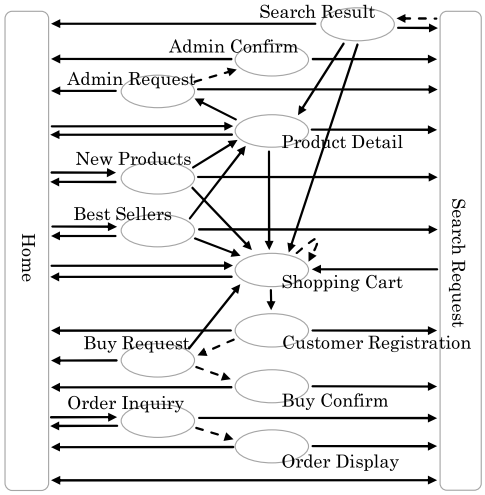

図 3 ベンチマークモデルの画面遷移
Fig. 3 Page transition of benchmark model.

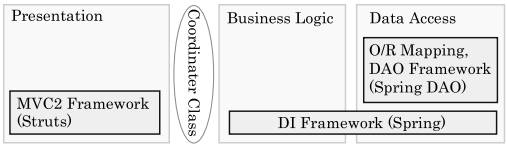

図 4 ベンチマークモデルの構造
Fig. 4 Structure of benchmark model.

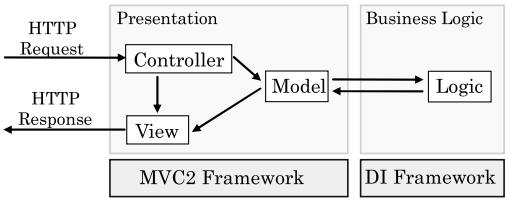

図 5 プレゼンテーション層
Fig. 5 Presentation layer.

る．thinktime は，模擬される各ユーザがベンチマークモデル中の Web ページを遷移する平均の間隔を指定する．duration は，シミュレーションを行う時間を指定する．transition rate では，各 Web ページから次の Web ページへ遷移する確率を指定する．これを変更することによって，たとえば検索処理（商品検索ページへのアクセス）を増加させる，更新処理（商品注文ページへのアクセス）を増加させる等ができ，シミュレーションモデルが変更可能となっている．

4.2 ベンチマークモデル

ベンチマークモデルには，実際の業務システムを意識した Web アプリケーションとして，TPC-W のモデルを採用した．TPC-W のモデルは，インターネット上で実現される典型的な電子商取引を想定して設計され，「インターネット上の Web サイトで書籍を販売する」というシナリオを採用した電子書店である．この電子書店は，ブラウザから商品の検索，閲覧，注文等ができるほか，ユーザ登録，商品情報の変更等，現実のシステムに即した各種の機能が含まれている．各機能ごとに 14 の画面を持ち，**図 3** に示すような画面構成となっている．

このような Web アプリケーションを，**図 4** に示すようなフレームワークを使用した多層構造で構築した．大きく分けると，プレゼンテーション層，ビジネスロジック層，データアクセス層の 3 つの階層から構成されている．

プレゼンテーション層は，Model-View-Controller（MVC）フレームワークの Struts [13] と Java Server Pages（JSP）によって構築した．ビジネスロジック層は，ミドルウェアやフレームワークに依存しないように JavaBeans 等の POJO で作成，データアクセス層は，Data Access Object（DAO）パターンで作成し，それらを Spring Framework [14] が提供する DI コンテナによって管理する構成とした．

a) プレゼンテーション層

プレゼンテーション層は，**図 5** に示すように MVC パターンで構成している．MVC パターンでは，すべての HTTP リクエストを受け，ページフローを制御する 1 つのコントローラサーブレット，それから呼び出されるモデル，その結果を表示するビューで構成される．このような MVC パターンに準拠したフレームワーク Struts を使用して実装している．

Struts では，プレゼンテーション層の各コンポーネントが XML 形式の定義ファイルに設定されており，ベンチマークモデルを拡張して，画面を追加する場合にも，拡張が容易である．

プレゼンテーション層のモデルからビジネスロジック層のロジックを呼び出す際に，MVC フレームワークと DI コンテナを連携させる必要がある，このような方式はいくつかあるが，フレームワークが提供する手法は利用せず，連携用のクラスを作成した．これは，フレームワーク間の依存性を減らし，フレームワークを交換して性能比較をする場合の影響を考慮している．

b) ビジネスロジック/データアクセス層

ビジネスロジック層とデータアクセス層は，**図 6** に示すように，DI コンテナ Spring を利用して構成する．データベースへの接続情報，トランザクションマネージャ，データアクセスオブジェクト等の関連が DI コンテナによって管理されており，XML 形式の定義

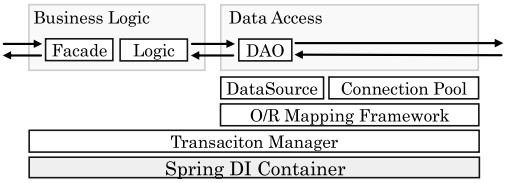

図 6 ビジネスロジック層/データアクセス層
Fig. 6 Business logic layer/Data access layer.

表 3 データベースのパラメータ
Table 3 Database parameters.

項目	説明
item	データベースに登録される商品数
user	データベースに登録される顧客数

表 4 測定項目
Table 4 Measurement items.

大項目	小項目	使用するツール
処理性能	スループット	JMeter
	平均応答時間	JMeter
システム負荷	CPU 使用率	sar コマンド
	メモリ使用量	sar コマンド
	ネットワーク負荷	sar コマンド
	ディスク負荷	iostat/sar コマンド

ファイルに設定されている．

DI コンテナを利用することで，管理されるコンポーネント間の依存性をなくし，DI コンテナが対応する範囲で各コンポーネントが交換可能である．

c) データベース

データベースには，TPC-W の仕様に基づいたテーブルが配置される．データベースは，**表 3** に示すパラメータによって，サイズ（各テーブルの件数）を変更して作成することができ，それによりベンチマークモデルの変更が可能である．

4.3 性能測定結果

本ベンチマークツールの性能測定方法について説明する．本ベンチマークツールの性能測定では，ベンチマークモデルとなる Web アプリケーションの処理性能と，シミュレーション時の各サーバに対するシステム負荷を測定する．**表 4** にその測定項目を示す．

Web アプリケーションの処理性能としてはスループットと平均応答時間を取得する．

スループットは，シミュレーションにおいて，1 秒間にどれくらいのトランザクションを処理可能であったかを示すものである．表 2 で指定する thinktime（平均思考時間）と browsers（擬似クライアント数）から理論値が導き出される．実測値はこの理論値以上にはならないことに注意が必要である．

平均応答時間は，各 HTTP リクエストの応答時間の平均である．平均応答時間を見ることで，web 操作にどのくらいの時間がかかっているかが分かる．

システム負荷は，OS（Linux）の sar コマンドと iostat コマンドを使用し，CPU 使用率，メモリ使用量，ネットワーク負荷，ディスク負荷を計測する．

CPU 負荷では，さらにカーネルレベルの使用率，ユーザレベルの使用率，IO 待ち，アイドルの 4 種類に分けて取得する．カーネルレベルの使用率では，システムコール等がどの程度使用されているかが分かる．ユーザレベルの使用率では，システムコール以外のユーザアプリケーション部分の処理がどのくらいを占めているかが分かる．IO 待ちでは，ディスクアクセスがどれくらいシステムに負荷を与えているかが分かる．この値が増加している場合，ディスクアクセスがシステムのボトルネックになっているという判断が可能である．このように，CPU 負荷を計測することで，スループットが横這いとなり，システムが処理限界となった場合に，どの部分がボトルネックになっているかの大まかな見当をつけることができる．

メモリ使用量では，未使用，ユーザ使用，システム使用，バッファ，キャッシュ，スワップといった各領域の容量を取得する．メモリ使用量を見ることで，たとえば，システムのメモリが不足しており，スワップが大量に発生しているため，メモリ容量がシステムのボトルネックになっているといった判断が可能である．

ネットワーク負荷では，各ネットワークデバイスごとの通信量を取得する．CPU 使用率を分析することで，システム負荷のほとんどを把握できるが，通信量による負荷は CPU 使用率には現れないため，シミュレーション中の送受信量を把握する必要がある．また，事前にネットワーク性能を Netperf [15] 等のベンチマークツールを用いてネットワークの限界性能を測っておくことで，ネットワークの帯域を使いきってしまっているかどうかを判断することができる．

ディスク負荷では，データベースへのアクセスで使用するディスクデバイスへのアクセス量を取得する．CPU 使用率の IO 待ちを見ることで，CPU から見たディスクアクセスの負荷は分かるが，実際にどのデバイスにどの程度のアクセスが行われているかまでは分からない．そこで iostat コマンドにより，より詳細な情報を取得する．この場合も，事前にディスクの限界性能を IOzone [16] 等で測定しておき，ディスク性能の限界まで使用していないかどうかを判断することができる．

これら測定項目で，Web システムの性能のほとんどを把握することができる．

5. 提案手法による評価

本提案手法を用いて，データアクセス層で使用するコンポーネント，O/R マッピングフレームワークの性能評価を行った．O/R マッピングフレームワークには iBATIS[17] と Hibernate[18] の 2 種類を用意した．

ベンチマークモデルのデータアクセス層に，iBATIS と Hibernate を組み込み，それぞれの場合でベンチマークを実施，スループットや平均応答時間，システム負荷の観点から 2 種類のフレームワークの性能比較を行った．

5.1 評価環境

評価環境は**図 7** のとおり．クライアント層，アプリケーション層，データベース層に各 1 台のサーバとデータベース層にストレージを配置，サーバとストレージはファイバチャネルで接続，サーバ間は 1 Gbps のネットワークで接続する構成とした．サーバ上の OS はすべて Linux としたが，本提案手法および実装したベンチマークツールは，プラットフォームを限定するものではない．

ベンチマークモデルは，表 2，表 3 のパラメータを調節し，顧客数を 288 万人，商品数を 10 万点，約 7 GB のデータベースを持つベンチマークモデルを設定，シミュレーションに関しては，トランザクション間隔の平均を 7.5 秒，シミュレーション時間を 10 分とした．

評価では，シミュレーションパラメータの擬似クライアント数を 100 から 1,000 まで変化させて，ベンチマークテストを行った．

5.2 結果の比較

ベンチマークモデルのデータアクセス層を iBATIS と Hibernate の 2 つのフレームワークを利用して，それぞれ実装した．その際に記述または修正したコード量を**表 5** に示す．

iBATIS，Hibernate それぞれを使用した場合の，測定結果を次に示す．**図 8** は，両者の性能をスループットで比較したものである．横軸に擬似クライアント数，縦軸にスループット，グラフ内には理論値も示している．**図 9** は，平均応答時間を比較したものである．横軸に擬似クライアント数，縦軸に平均応答時間を示している．**図 10** は，システム負荷（CPU 使用率）で比較したものである．横軸には擬似クライアント数，縦軸には平均 CPU 使用率を示し，アプリケーション層，データベース層ごとに示している．**図 11** は，ネットワーク負荷を比較したものである．横軸には擬似クライアント数，縦軸にはネットワークデバイスを通過したデータ量を示している．**図 12** は，ディスク負荷で比較したものである．横軸には擬似クライアント数，縦軸にはデータベースサーバからストレージへのアクセス量を読み込み，書き込み別に示している．

なお，メモリ使用量も測定したが，今回の評価では Java 仮想マシンのガベージコレクションや OS の影響が大きく，有意な結果を得られなかったため，ここ

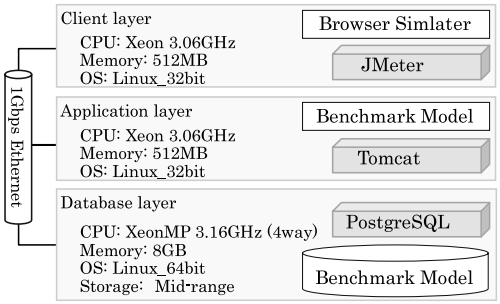

図 7 評価環境
Fig. 7 Evaluation environment.

表 5 記述または修正したコード量（単位：ライン）
Table 5 Lines of code (lines).

	iBATIS 利用時	Hibernate 利用時
Spring 定義	20	20
DAO クラス	450	510
マッピング定義	220	380
合計	690	910
（モデル全体）	9,000	9,000

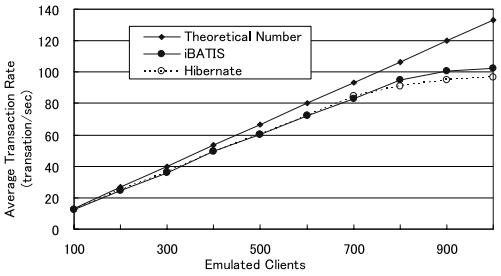

図 8 スループット
Fig. 8 Throughput.

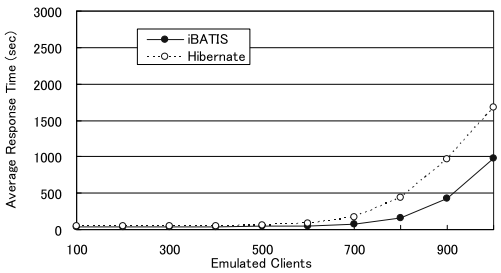

図 9 平均応答時間
Fig. 9 Average of response time.

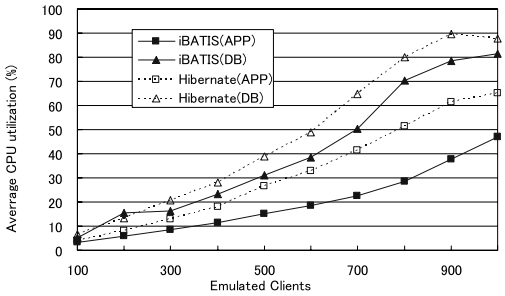

図 10 平均 CPU 使用率
Fig. 10 Average of CPU utilization.

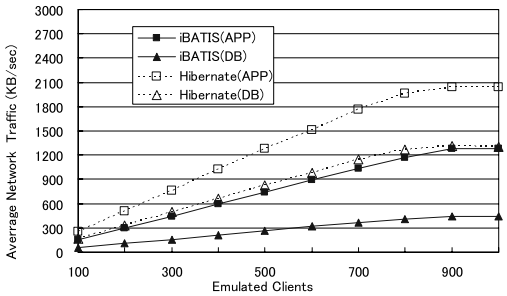

図 11 ネットワークトラフィック
Fig. 11 Network traffic.

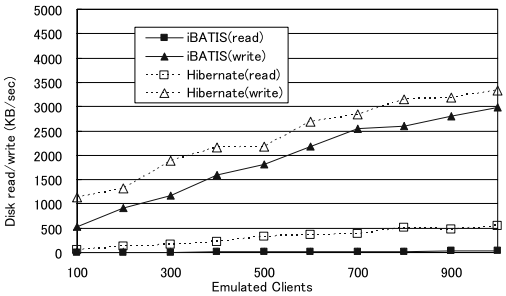

図 12 ディスクアクセス
Fig. 12 Disk access.

では触れていない．

5.3 考　　察

a) ベンチマークモデルの拡張性

評価を行うため，ベンチマークモデルのデータアクセス層に iBATIS と Hibernate の 2 種類のフレームワークをそれぞれ組み込み，拡張した．本提案手法ではコンポーネント間の依存関係を極小化するアーキテクチャを採用しているため，データアクセス層のみを実装すればよく，他の部分には手を加える必要はなかった．実装に必要だったコード量を表 5 に示したが，iBATIS では約 690 ライン，Hibernate では 910 ラインとなり，ベンチマークモデル全体のコード量，約 9,000 ラインに対して 1 割程度である．さらに，これらのコードは，業務ロジックを含まず，機械的に生成可能なコードのため，実装は容易であった．また，iBATIS と Hibernate の切替えは，表 5 の Spring 定義ファイルを 20 行程度修正するだけでよく，容易にコンポーネントの切替えが可能であった．

b) ベンチマークの測定結果

図 8 より，iBATIS，Hibernate 両者ともに，擬似クライアント数 800 程度からスループットが横這いとなっている．擬似クライアント数 800 以前では，スループットはほぼ同じ値となり，それ以降のスループットは，iBATIS の方が若干上回っていることが分かる．図 9 より，両者の平均応答時間を見ると，擬似クライアント数 800 程度から Hibernate の処理速度が遅くなっていることが分かる．図 10 より，両者のデータベース層の CPU 使用率を見ると，擬似クライアント数 800 程度から CPU 使用率が 70% 以上となっている，一方でアプリケーション層の CPU 使用率は，データベース層ほど高くなく，全体的にデータベース層が高負荷となっている．これより，図 8 で，スループットが横這いとなっている原因がデータベースサーバの CPU が高負荷となり，処理が滞っているためであることが分かる．また，iBATIS と Hibernate を比較すると，アプリケーション層，データベース層ともに Hibernate 使用時の CPU 負荷が iBATIS 使用時に比べて高いことが分かる．図 11 より，通信量に関しても同様で，Hibernate 使用時に iBATIS 使用時を比べて 2 倍近く通信量が多くなっている．また，通信量の最大値は，約 2 MB/sec（16 Mbps）であり，使用しているネットワーク（1 Gbps）の帯域を使いきるような負荷はかかっていない．図 12 より，ディスク負荷では，読み込み，書き込みともに Hibernate の方がアクセス量が多い．また，アクセス量の最大値は，約 3 MB/sec であり，ストレージが処理限界となるような負荷はかかっていない．

以上より，iBATIS と Hibernate では，Web システム全体のスループットで見ると，処理限界近くのデータを除けばほぼ性能が等しいが，CPU 使用率やネットワーク負荷，ディスク負荷の観点から見ると Hibernate の方が全体的にシステムに与える負荷が高い．そのため，Hibernate を利用した Web アプリケーションを構築する場合には，これらの性能特性を考慮した設計を行う必要があることが分かった．

また，これら測定結果は，図 7 に示したハードウェア環境で Web システムを構築した場合の Web システム全体の性能も示している．測定結果から，この Web システムは約 800 クライアントからの同時アクセス

を処理可能で，それ以上の負荷が発生すると，データベースサーバの CPU が高負荷となりシステム全体が処理限界となるような性能傾向を示すことが分かる．

このように，本提案手法を用いることで Web システムで利用される各コンポーネントを容易に切り替え，性能を比較可能である．そして，あるハードウェア上に Web システムを構築した場合の性能を取得可能である．このような性能データは，Web システムを構築する際の性能設計やフレームワークの選定に活用することができ，複雑化した Web システムの性能設計に役立てることができる．ただし，本提案手法による測定結果は，実装方式により変化するもので，絶対的な性能の差異を示すものではない．

6. おわりに

本稿では，複雑化した Web システムの性能設計を支援するため，次のような特徴を持つ手法を提案した．(1) Web3 階層システム全体の性能を評価可能である．(2) 評価モデルにミドルウェアやフレームワークを実際に使用し，実システム相当の環境で性能を評価できる．(3) 評価モデルに使用するミドルウェアやフレームワーク等のコンポーネントを低い実装コストで交換可能であり，評価モデルの拡張性が高い．

(1) に対しては，図 2 で示したように評価手法の構成をクライアント層にシミュレータ，アプリケーション層とデータベース層に性能評価モデルを持つ 3 階層とすることで解決し，(2) では，性能評価モデルに TPC-W を採用し，実際の Web アプリケーションと同様の実装とすることとした．(3) に関しては，性能評価モデルのアーキテクチャを多層化することにより，コンポーネント間の依存性をなくし，各層のコンポーネントを容易に比較可能とした．

これら特徴を活かし，データアクセス層で利用する 2 つの O/R マッピングフレームワークについて，本提案手法を用いた性能比較を行い，本提案手法を用いることで Web システムで利用される各コンポーネントの性能を容易に比較可能であることを示した．

本稿では，評価例として，データアクセス層のフレームワーク iBATIS と Hibernate の比較を紹介したが，たとえばコネクションプーリングやトランザクションマネージャ等，Web アプリケーションを構成する他のコンポーネントに関しても，評価例と同様にベンチマークモデルに組み込み，ベンチマークを実施することで性能比較が可能である．また，本提案手法は，Web アプリケーションを構成するコンポーネントに着目しているが，アプリケーションサーバやデータベースサーバ等のミドルウェアの性能比較やチューニング検証にも活用することができる．

今回の実装では，ベンチマークモデルに TPC-W を採用し，画面構成やデータベース，SQL 等をその仕様に忠実に実装したが，今後はそれらをパラメータ化し，ベンチマークモデルを柔軟に変更可能とすることで，より汎用的な Web システム性能評価手法としていく予定である．

参考文献

1) OPNET Technologies, Inc: OPNET. http://www.opnet.com/
2) HyPerformix, Inc: HyPerformix. http://www.hyperformix.co.jp/
3) 西岡大祐，亀山　伸，垂井俊明，庄内　亨：情報システム性能見積りのためのシミュレーションモデル作成方法の開発，情報処理学会研究報告 2003-EVA-10，pp.7–12 (2004).
4) 峰野博史，川原亮一：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，情報処理学会研究報告 2002-EVA-3, pp.37–42 (2002).
5) Menasce, D. and V.A.F., A.: *Scaling for E-Business*, Prentice Hall (2000).
6) Wells, L., Christensen, S., Kristensen, L. and Mortensen, K.: Simulation based performance analysis of web servers, *Petri Nets and Performance Models*, pp.59–68 (2001).
7) 岡田昭宏：サーバ処理性能の評価方法に関する一検討，情報処理学会研究報告 2004-EVA-9, pp.47–52 (2004).
8) Standard Performance Evaluation Corporation: SPECjAppServer. http://www.spec.org/
9) The JBento Project: JBento. http://jbento.oscj.net/
10) Open Source Development Labs: Database Test 1 (DBT-1). http://old.linux-foundation.org/lab_activities/kernel_testing/osdl_database_test_suite/
11) Transaction Processing Performance Council. TPC-W. http://www.tpc.org/
12) Apache Software Foundation: JMeter. http://jakarta.apache.org/jmeter/
13) Apache Software Foundation: Struts. http://struts.apache.org/
14) springframework.org: Spring Framework. http://www.springframework.org/
15) netperf.org: Netperf. http://www.netperf.org/netperf/
16) iozone.org: IOzone. http://www.iozone.org/
17) Apache Software Foundation: iBATIS.

http://ibatis.apache.org/
18) hibernate.org: Hibernate.
http://www.hibernate.org/

(平成 19 年 5 月 7 日受付)
(平成 19 年 9 月 25 日採録)

金木 佑介（正会員）
平成 16 年三菱電機（株）情報技術総合研究所入社．コンピュータプラットフォーム技術，Web アプリケーションフレームワークに関する研究に従事．

飯塚 剛
昭和 62 年三菱電機（株）入社．現在は同社情報技術総合研究所に勤務．産業用計算機ハードウェア，通信装置の監視制御，コンピュータプラットフォーム技術に関する研究に従事．

浅見可津志（正会員）
昭和 52 年三菱電機（株）入社．現在は三菱電機インフォメーションシステムズ（株）に勤務．システム構築のための共通技術基盤の開発に従事．アーキテクチャ設計に関心を持つ．