TPC-App ベンチマークを用いた Web アプリケーション フレームワークの性能評価

上田　誠治† 仲前　晋太郎†成　凱‡

†‡九州産業大学大学院 情報科学研究科 〒813-8503 福岡市東区松香台 2-3-1

E-mail: †k09gjk08@ip.kyusan-u.ac.jp, ‡chengk@is.kyusan-u.ac.jp

あらまし Web アプリケーションフレームワーク（以下はフレームワークと略称）は複数の再利用技術を取りまとめた Web アプリケーション開発技術であり、フレームワークを利用することにより、Web アプリケーション開発の生産性・保守性を大幅に向上させることができると期待されている。一方、フレームワークを利用する場合はアプリケーションの性能が著しく劣ると指摘されており、フレームワークを導入する際に問題視されている。本稿は実際に TPC-APP に準じた評価対象システムを symfony で開発し､利用パターンをパラメータとして指定可能なワークロードジェネレータ PW-GEN を実現した。さらに、Web アプリケーションのボトルネックとなりうる箇所にキャッシュを配置したり、アクセスパターンを変化させたりして、性能の変化をリクエスト応答数/秒で計測を行った。その結果、言語処理とアプリケーションにおけるキャッシュは最も効果的であることが分った。

キーワード ライブラリ、フレームワーク、Web アプリケーション、ボトルネック、キャッシュ、性能評価

Performance Evaluation of Web Application Framework Using the TPC-App Benchmark

Seiji UEDA† Shitaro NANAMAE† Kai CHENG‡

†‡Graduate School of Information Science, Kyushu Sangyo University

3-1, Matukadai 2-chome, Higashi-ku, Fukuoka, 813-8503

E-mail: †k09gjk08@ip.kyusan-u.ac.jp, ‡chengk@is.kyusan-u.ac.jp

Abstract Web application frameworks (hereafter “framework” for short) are useful technology to achieve high development productivity and maintainability of web-based software. However, applications developed under frameworks often suffer from low performance due to bottlenecks existing in several spots in the layered architecture. To deal with this problem, it is necessary to identify the bottlenecks that really cause the performance deterioration. In this paper, we use TPC-APP benchmark to evaluate the performance of symfony, a well known PHP framework for web application development. To do this, we first developed the TPC-APP SUT (System Under Test) using symfony and then implemented a workload generator PW-GEN. The results showed that caches at language processing layer and application layer are most effective but caches at web server level are not apparent.

Keyword Library, Framework, Web Application, Bottleneck, Cache, Performance Evaluation

1. はじめに

近年､インターネットの急速な普及にともない､Web 利用の多様化が進み、ショッピング、オークション、金融取引､映像・音楽の視聴､SNS の閲覧・書き込み、オンラインゲーム等､幅広い分野に Web が広がれている。従来、Web は HTML 言語で記述された静的な Web 文書の配信を中心に行ってきたが、現在ではユーザ入力に応じてビジネスロジックに従った一連の処理を行い､提供する情報を変化する Web アプリケーションが一般的である。新しい情報サービスは We アプリケーションとして続々登場し、高い人気が得られている。

一方、利用形態の多様化につれ、システムの複雑化・大規模化が進んでいる。かつて企業の一部業務を効率化するために構築された業務システムは業務全体を担うようになったり、競合他社と差別化を図るため

により高度の機能を提供したり、システムは大規模化・複雑化の道をたどらざるを得なくなっている。また、企業を取り巻く環境は年々変化のスピードが加速化され、新しいサービスを一日も早く提供し競争の優位性を確保するニーズも高まり、開発期間の短期化が要求される。

大規模化・複雑化・開発期間の短期化が進むなかで、Web アプリケーションを如何に効率的かつ安全に開発し、開発者の経験やスキルによるバラツキを抑えて、開発品質を保つかは、ソフトウェア開発の重要な課題である。これに対して、ソフトウェア工学では再利用技術により生産性の向上を図ってきた[6]。再利用技術は、様々なレベルで行うことができる。例えば、ライブラリによるコードの再利用、オブジェクト指向プログラミングによる部品単位の再利用、パターンによる設計思想の再利用がある。

Web アプリケーションフレームワーク（本稿ではこれからフレームワークと略称）は複数の再利用を取りまとめた Web アプリケーション開発技術であり、ソフトウェアの階層化アーキテクチャに基づき、オブジェクト指向プログラミング、O/R マッピングなどの技術を駆使して開発を行うソフトウェア開発環境である。アプリケーションの独自の部品がフレームワーク内の既存のクラスを継承することができるため、新たに開発する必要なコードの量を削減できるだけではなく、システムの保守性も高められる。

フレームワークは、大企業による開発実績が蓄積され、近年、Web アプリケーションのためのフレームワークにも大きな注目を集めている[5]。　現在、Java、Ruby、PHP など Web アプリケーション開発の代表的な言語においては、Struts、Ruby on Rails、symfony、CakePHP など、アプリケーションフレームワークが数多く出回っており、フレームワークは導入しやくなりつつある。

しかし、フレームワークを利用する場合は、できあがったシステムの性能が著しく劣ると指摘されており、フレームワークの導入を検討する際に問題視されている。Web アプリケーションはネットワーク環境で実行されるのが一般的であり、性能に影響しえるボトルネックは複数考えられる。しかし、ボトルネックを特定する研究は、少なくとも我々知る範囲内には存在せず、フレームワークを採用するなら、アプリケーションの性能がどこまで期待できるか、性能改善のためにどんな方策を講じるべきか、効果はどうなるか、より客観的に評価する必要がある。

本稿では、Web アプリケーションを開発するにあたり、アプリケーションフレームワークを導入する必要性及び導入に伴う問題点を分析した上、TPC-APP ベンチマークを用いて、代表的な Web アプリケーションフレームワーク symfony の性能評価を行い。ボトルネックの特性やキャッシング等の効率化技術の効果を計測することにより、Web アプリケーションの性能向上の手がかりを提供する。

1.1. 関連研究

文献[1]では、複雑度と機能量に基づくアプリケーションフレームワークの実験的評価が述べられている。ソフトウェアの品質と工数削減の観点から、フレームワークの有効性を実験的に評価することを目的とし、フレームワークを用いた再利用とモジュール単位の再利用を 2 つの開発ケースで使用し、それらの差異を機能量と複雑度のメトリクスを用いて評価を行っている。

文献[2]は、フレームワークの要求仕様に対する適合性の評価手法が報告されている。要求仕様に対して適切なフレームワークを選択の支援をする手法の提案を目的としている。フレームワークは複数存在するが、それぞれ利用方が異なっている。またフレームワークが提供する機能が複雑になるほどフレームワークの可変部分間の依存関係が複雑となり、利用が困難となる。それらを解決ためにフレームワークをその振る舞いでモデル化し、それらをラベル付き遷移システムで表現し、プロセス代数における観測同値性に基づけて対応することによりフレームワークの要求仕様に対する適合性を評価している。

2. Web アプリケーションフレームワーク

プリケーションフレームワークはソフトウェアの再利用技術のひとつとして知られている。再利用技術がソフトウェア開発の生産性を高める鍵となると言われている。ソフトウェア再利用の歴史は、ソフトウェアを部品として扱い、それを組み立てることで開発を行う考え方が提唱されたことが始まりである[8]。再利用の対象は、プログラム・コードから、要求仕様や設計、さらにソフトウェアの開発や保守のプロセスそのものも重要な再利用対象とされてきた。オブジェクト指向技術の発展にともない、大規模な再利用は現実的に可能となりつつある[4]。

オブジェクトを部品とすることにより、従来は部品の大きさやレベルをどうするかという問題は解消される。部品を使う場合の加工度については、そのまま使うこともできるし，クラス継承という手段によって既に定義されているデータ（属性）や処理（メソッド）を利用しながらさらに特殊化したり変更したりすることも可能である。また、再利用されるクラス部品では仕様のみを定義して、具体化は個別のクラスやインスタンスを作るときに行うような，部品の用意の仕方も有効である。またクラスからインスタンスを生成する

際、パラメータを指定して用途に応じたものを作り出せるようにすることも、自然にできる。部品同士の結合はオブジェクト間のメッセージ交換によって行われるので、必要な部品（オブジェクト）さえそろえれば、組み立てはある意味では自然に可能であり、機能の追加のために新たにオブジェクトを導入するにも、メッセージ交換のインタフェースに従えばよい.

しかし、クラス単位の再利用では単位が小さすぎて、再利用の効果が限定的であるという問題点が指摘されている。この問題は、より大きな単位としてのフレームワークや設計パターンの再利用が促進されてきた。

これらの再利用技術は相互に影響している。現在の多く商用フレームワークにはライブラリによるコードの再利用、オブジェクト指向における小さな単位の再利用、MVC やいくつかのパターンが含まれている。

2.1. Web アプリケーションの階層化アーキテクチャ

階層化は抽象化と分割で複雑な問題を対処する効果的な方法と知られている。Web アプリケーションが複雑になるにつれ、階層に分けて理解する必要性が高まる。階層化の度合いはアプリケーションの複雑さによって異なり、複雑になればなるほどより階層化される。階層化により複数の技術者や開発チームが互いに影響を与えず、それぞれ異なる層に対して同時に作業が可能となる。また各開発チームはあらかじめインタフェースを定義しておけば、自分の担当部位以外の層についての内容を知る必要性がなくなる。

表1　Web アプリケーションの階層アーキテクチャ

Web アプリケーションの階層	役　割
プレゼンテーション層	ページの見栄えを定義する
マークアップ層	データを表示するためにタグを付ける
ページロジック層	データをページ別に纏め上げる
ビジネスロジック層	データに対するアクセス方法と操作方法を定義する
永続化層	データの格納と検索の方法を提供する

大規模複雑な Web アプリケーションを理解・開発しやすいため、一般的に表 1に示すように、プレゼンテーション層、マークアップ層、ページロジック層、ビジネスロジック層、永続化層の 5 つの階層に分けられる。ユーザインタフェースからデータストレージまでの間の処理をいくつかの役割に分離したものである[3]。下位になるに従ってデータストレージに近づき、階層が上位になるに従って Web アプリケーションを操作するユーザインタフェースに近づく。Web アプリケーションだけではなく、マークアップ層とプレゼンテーション層を適切に変えると、ユーザインタフェースをもつ全ての会話的アプリケーションには、このアーキテクチャが適用できる。

最下層は永続化層と呼ばれ、データに該当する。データにアクセスや操作、画面表示するためにはデータが格納されていなければならない。Web アプリケーションにおいて、データはディスク上のファイルまたはデータベースのレコードの形で保存される。

永続化層の上には、ビジネスロジック層が存在する。ビジネスロジック層ではデータに対するアクセス方法と操作方法が定義されており、データにアクセスするには必ずビジネスロジック層を通される。

ビジネスロジック層の上には、ページロジック層が存在する。ページロジック層はビジネスロジック層で得られたデータを纏め上げる役割を担う。ビジネスロジック層によってデータへのアクセスと操作は可能であるが、データの更新手順やどの部分のデータを更新するかといったことは定められていない。

ページロジック層の上にはマークアップ層が存在する。マークアップ層はユーザへ下位層にアクセスする手段を提供する。Web においてはマークアップ、デスクトップにおいては GUI ツールキット、API においては XML が該当する。

最上位層にはプレゼンテーション層が存在する。プレゼンテーション層はアプリケーションを装飾する。Web において装飾は CSS (Cascading Style Sheets) に該当する。装飾は軽視されがちではあるが、一般のユーザは装飾を必要以上に重要視する。Web アプリケーションにおいて装飾は集客に影響する重要な要素である。

2.2. MVC モデル

上記の階層化アーキテクチャの具体化として、MVC (Model View Controller) モデルが多くのフレームワークに採用されている。MVC モデルはもともと Smalltalk-80 のユーザインタフェース構築のためのアーキテクチャとして提唱された[5][6]。MVC モデルはモデル (Model) 、ビュー (View) 、コントローラ (Controller) の 3 つの部分で構成され、それぞれが役割分担を行っている。頻繁に発生するユーザインタフェースの変更・追加要求に対して、アプリケーションロジックを修正することなく対応することができる。現在ではこのモデルは Web アプリケーションにおいても有効であることが認知され、Web アプリケーションフレームワークのシステムデザインのベースとして広く使われている。

モデル部分はそのアプリケーションが扱う領域のデータと手続き、すなわちビジネスロジックを表現する。ビュー部分はモデルからデータを受け取り、ユーザが見るのに適した形で表示する。すなわちユーザインタフェースへの出力を担当する。コントローラ部分はユーザからの入力を解釈し、モデルやビューの制御を行う。３つの部分は連携しており、制御のフローは一般に以下のとおりになる。

1. ユーザがユーザインタフェースを通してコントローラに入力する。
2. コントローラは入力イベントを処理し、ユーザのアクションに応じてモデルのメソッドを呼ぶ。この際モデルのデータが書き換わることがある。
3. ビューがモデルから関連するデータを取得し、出力を更新する。
4. コントローラはユーザの次の操作を待つ。

MVC モデルもユーザインタフェースをもつアプリケーションを実装するための構造として、Web アプリケーションの階層化アーキテクチャと役割分担という点で共通である。MVC モデルではビューとモデルが連携している。それに対して Web アプリケーションの階層化アーキテクチャでは原則としてユーザへの表示部分とデータの操作部分が直接通信することはなく、必ず中間層を通さなければならないという点で異なる。

2.3. Web アプリケーションフレームワーク symfony

symfony は Web アプリケーションフレームワークの中で人気の高いものの 1 つであ[7]。PHP 言語に対応しており、MVC モデルに基づいている。以下では symfony の MVC 構造および symfony による Web アプリケーションの開発について詳述する。

symfony でも MVC モデルが採用されている。まず、symfony では、コントローラがフロントコントローラとアクションで構成されている。フロントコントローラはクライアントからリクエストを受け取って、処理に必要な情報を取り出す。取り出された情報により、適切なアクションを呼び出し、処理を始める。クライアントからの全てのリクエストをフロントコントローラで受け取り、そのリクエストの URL から実行するアクションを決定する。これは、**ルーティング処理**と呼ばれる。

アクションの処理では、まず HTTP リクエストを確認し、使用するメソッド（GET、POST、PUT、DELETE、HEAD）を特定する。リクエストメソッドが確認されたら、次に入力データをバリデートする。その後 3 つのアクションが実行される。プレアクションはアクション本体の前処理を行い、ポストアクションはアクション本体の後処理を行う。これらは全てのアクション本体に共通した処理を記述する。

アクションが実行されるときに、データベースアクセスが必要とされる場合、モデルによってデータが取得される。モデルはデータベース抽象化レイヤとデータベースアクセスレイヤで構成されている。データベース抽象化レイヤはデータロジックの記述を行う。たとえば、データベースから取得した値を整形する処理などである。データベース抽象化レイヤは複数のデータベースに対し、統一的な命令で操作するための仕組みである。

symfony ではデータベースを行うために O/R マッパを利用する。O/R マッパを利用することにより、モデルクラスが自動的に生成され、プログラムのデータオブジェクトをデータベースと対応させる。それゆえ、プログラマはデータベースを意識せずに外部データを操作することができる。symfony では Propel という O/R マッパをデフォルトに利用している。

ビューはテンプレートと各ページ固有のビューロジック、全体で共通したレイアウトで構成している。レイアウトはフッタやヘッダを含んだ HTML など、アプリケーション全体の共通した画面デザインを定義する。テンプレートはレイアウトの中に組まれる、ページごとに異なるビューを記述する。レイアウトとテンプレートはいずれも雛形をベースに動的に生成されるコンテンツであるが、ページ全体を定義したレイアウトの中に、ページごとのテンプレートが埋め込まれて表示される。

symfony で Web アプリケーションを開発する際に、図 1n にするように、プロジェクトからアプリケーション、モジュール、そして、アクション、テンプレートの順に構築していく。ほとんどの作業はコマンドで自動的に行える。

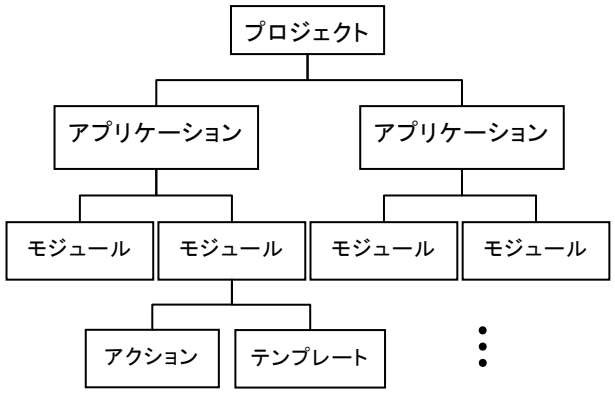

図 1 symfony の Web アプリケーション構造

2.4. Web アプリケーションのボルトネック

Web アプリケーションはクライアント/サーバ環境で実行される。アプリケーションの性能に影響する劣るネックは複数個所に存在する可能性がある。図 2では Web アプリケーション、とりわけ、フレームワークによって開発された Web アプリケーションのボトルネックが枠で示されている。

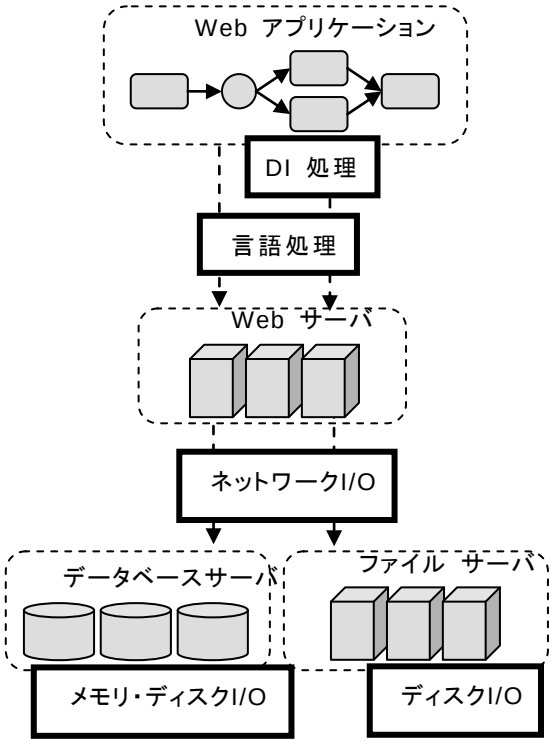

図 2　Web アプリケーション性能に影響するボトルネック

依存性注入（DI：Dependency Injection）処理とは、ソフトウェアコンポーネント間の依存関係をプログラムのソースコードから排除し、外部の設定ファイルなどで注入できるようにするソフトウェアパターンである。PHP のようなスクリプト言語では、DI 処理は実行時に行うので、システムの性能に影響する。
言語処理とは主にスクリプト言語に関わる。プログラムが実行する前に、スクリプトファイルをメモリにロードし、ソースコードをコンパイルする処理である。PHP言語の場合は、ソースコードをコンパイルしてopcodeツリーを生成してから、opcodeツリーを実行することが一般的である。

コンパイルされたソースコードであっても毎回行われる。長いプログラムや、利用頻度の高いプログラムの場合はmコンパイルするために多くの時間がかかり、全体の処理時間に大きく影響する。

Web サーバ、データベースサーバ、ファイルサーバにおける I/O 関連の部分があり、遅いデバイスとデータをやり取りしないといけない。そういう部分はボトルネックになりやすい。

2.5. キャッシュの適用

ボトルネックによる影響を改善するために、キャッシュを適用することは一般的である。キャッシュとはデータや処理結果を一時的に保管して再利用する仕組みである。使用頻度の高いデータを高速な記憶装置に蓄えておくことで、いちいち低速な記憶装置から読み出す手間を省いて高速化する。例としてキャッシュメモリが挙げられる。キャッシュメモリは CPU が頻繁に使用するデータをあらかじめメインメモリから読み込んでおくことで、全てをメインメモリから読み込む時よりも高速化される。

2.5.1. サーバにおけるメモリ・ディスクキャッシュ

Web サーバおよび PHP 言語処理層ではキャッシュを適用できる。例えば、代表的な Web サーバ Apache ではローカルのコンテンツやプロキシされたコンテンツのキャッシングを実現し、CGI を使って生成された動的なコンテンツのレンダリング結果の再利用を可能にする mod_cache が存在する。コンテンツのキャッシュは URI に基づいたキーが使用され、ファイル送信要求があった時、ファイルがキャッシュされかつ期限が切れていなければ、mod_cache の利用するストレージからキャッシュされているファイルを取り出し送信する。

ストレージによって、ディスクを利用したものである mod_disk_cach および、メモリを利用した mod_mem_cache を使える。mod_disk_cache を選択した場合、やや低速ではあるが容量は大きい。mod_mem_cache を選択した場合、高速ではあるが容量に制限がある。mod_cache では以下の条件に該当するものをキャッシュする。

1. HTTPのGETリクエストでレスポンスのHTTPステータスコードが200、203、300、301または410。
2. ”Authorization:”ヘッダが含まれないリクエスト。
3. URLにクエリ文字列が含まれない。但し”Expires:”ヘッダがある場合は除外する。
4. レスポンスのステータスが 200 なら、”Etag”、”Last-Modified”、”Expires”のいずれかを含む。
5. レスポンスの”Cache-Control:”ヘッダにて”private”が含まれていない。

6. 同様に”Cache-Control:”ヘッダにて”no-store”が含まれていない。

2.5.2. 言語処理層のキャッシュ

PHP 言語処理層でコンパイルされたコードが opcode と呼ばれ、opcode キャッシュを使って、ソースコードの更新時刻が変わってなければ、キャッシュから取り出し実行する。opcode キャッシュが搭載された製品としては、APC (Alternative PHP Cache) 、eAccelerator、XCache などが知られている。

APC (Alternative PHP Cache) は、アクセスがあった際に中間形式にコンパイルした中間コード (opcode) を捨ててしまわず、ファイル(もしくは共有メモリに)として保管しておいて、以後同じファイルに対してアクセスがあった際には、保管してあるキャッシュを利用するという方法を用いて高速化をはかる。

APC のインストールが終わってから使えるまでまたいくつかの設定をしないといけない。まず、apc.mode は、mmap、 shm、 off のいずれかを選択する。mmap は、中間形式にコンパイルしたものをファイルに保存し、shm は共有メモリに保存する。 off は APC を使用しない。次に、apc.cachedir は mmap の時に有効で、どのディレクトリにファイル生成するかを設定する。実際にはここで指定したディレクトリの下に、階層的にディレクトリが作成され、ファイ ル名は元のスクリプトのファイル名の最後に _apc をつけたものになる。cachedir で指定したディレクトリには、書き込みを行なうユーザ(Apache 経由の場合は Apache の実行ユーザ、php コマンドからの場合はそのコマンドの実行ユーザ)が書き込みできる権限がなければならない。

最後に、apc.check_mtime は、キャッシュされているファイルに更新があった場合に、自動的に再コンパイルさせるためのオプションである。ドキュメントでは shm のときに有効となっている。

Web サーバ上と PHP 言語処理層にはそれぞれに複数のキャッシュが存在し、組み合わせることが可能である。しかしながら性能を最も向上させる組み合わせは明確ではない。そこで Web サーバレベルのキャッシュと PHP 言語処理層を組み合わせた場合、性能がどの程度変化するかを測定した。

2.5.3. Web アプリケーションのキャッシュ

Web アプリケーションレベルのキャッシュはフレームワークに搭載されている。フレームワークでは MVC モデルにおけるビューのキャッシュ、関数呼び出し結果のキャッシュ、オブジェクト・スタティックメソッドのキャッシュなどが利用できる。

symfony においては、以下の項目でキャッシュを活用できる。下に行けばよりキャッシュされる範囲が狭められる。

1. レイアウトを含むページ全体
2. レイアウトを含まないアクションの結果
3. パーシャル、コンポーネント、コンポーネントスロットと呼ばれるテンプレートで共通に使用することができる部品
4. パーシャル、コンポーネント、コンポーネントスロットを除くテンプレートの中の一部分

3. TPC-App ベンチマークを用いた性能評価

TPC-App は、TPC (Transaction Processing Performance Council :「トランザクション処理性能評議会」)によって制定された Web コマース向けベンチマークである[9][10]。トランザクション処理とデータベース 性能のベンチマークを定義し、それらのベンチマークに基づく客観的な性能 データを広く知らせることを目的に設立された非営利団体である。TPC によって策定された指標(ベンチマーク)には TPC-A、TPC-B、TPC-C、TPC-D、TPC-W、TPC-App 等があるが、よく使われているのは一部しかない。

TPC-App はインターネット・ブックストア（オンライン書店）を模倣するものであり、トップページ、書籍の検索、注文購入など画面が合計 14 個ある。性能結果は、WIPS（Web Interaction Per Second : 1 秒間の Web インタラクション数）を測定して判定する。

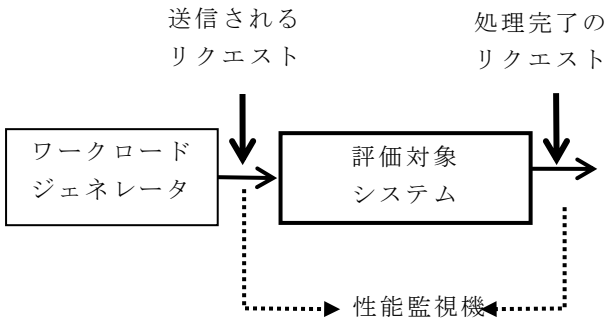

図 3 TPC-APP による性能評価の仕組み

ワークロードジェネレータは評価対象システム (System Under Test）に対しリクエストを送信し、ワークロードを発生する。SUT はそのリクエストを受け取り、処理を実行しレスポンスを返す。性能監視機構は SUT をモニタリングすることによって測定基準に従って性能評価する。SUT はネットワーク接続、Web サーバ、アプリケーションサーバ、データベースサーバなどを含むハードウェアおよびソフトウェア全体を指し

ている。その中の Web サーバはワークロードジェネレータからのリクエストに応じて Web アプリケーションを実行し、レスポンスを返す。

3.1. TPC-APP の評価対象システム

TPC-APP の評価対象システムは 14 の Web インタラクションが含まれているが、読み込みを中心とするインタラクションは HOME、Search Request、Search Result を評価する。ユーザが HOME から Search Request へリンクによって移動し、商品（書籍）の検索を行うと、Search Result によってその検索結果が表示される。

HOME は新商品およびベストセラー商品リストのリンクを含む Web ページであり、全てのユーザはこの Web ページからセッションを開始する。

Search Request は検索条件と検索文字列を指定して商品の検索を要求する Web ページである。検索条件は著者、書籍タイトル、書籍のテーマから選択できる。例えば著者が「John Doe」の書籍を探す場合、検索条件を「著者」とし、検索文字列を「John Doe」とする。

Search Result は Search Request の要求に対して、検索条件にマッチした商品のリストを表示するページである。タイトルの昇順にソートし、上位 50 件の著者名および書籍タイトルを表示する。

3.2. ワークロードジェネレータ PW-GEN

TPC-APP においては、EB (Emulated Browser) によってワークロードの生成を行うことになっている。EB は SUT に対してブラウザを通して HTML コンテンツを送受信して、通信するユーザをエミュレートするものである。ネットワーク接続を通して、HTML コンテンツを送受信するユーザをエミュレートする。

ワークロードを自動的に発生するために、我々は EB の仕様に準じて遷移確率をパラメータとして設定可能なワークロードジェネレータ PW-GEN（Parameterized Workload GENerator）を開発した。

PW-GEN は Web サーバ Apache に標準で搭載されたベンチマークツール ab (Apache Bench)に基づいて実現されている。Ab がコマンドラインで実行される場合は、以下の構文形式がある。

ab [options] URL

option にはテストで発行するリクエストの回数、同時接続数、クッキーの指定などが可能である。例えば、オプション「-n 数値」は テストで発行するリクエストの回数を数値で指定する。オプション「-c 数値」はテストで同時に発行するリクエストの数を数値で指定する。オプション「-t 数値」はサーバからのレスポンスの待ち時間（秒）を数値で指定する。

本稿では、リクエストの回数および同時接続数を 1 とし、スクリプトによって下記の遷移確率で Web ページ間を 1000 回遷移することにした。

集計結果は平均リクエスト応答数/秒を求めた。各アクション利用の割合をパラメータとして以下の三つのパターンを使ってある。

表 2 利用パターンの設定

	HOME (%)	Search Request(%)	Search Result(%)
パターン 1	50	25	25
パターン 2	25	50	25
パターン 3	25	25	50

PW-GEN では Search Request における検索文字列の生成を行わなければならない。検索文字列はヒット数を一定にするため、形式が決まっている。

4. 評価結果

前述の評価対象システムに対して、評価実験を行った。以下はその結果を述べる、

サーバにおけるメモリ・ディスクキャッシュ及び言語処理層のキャッシュ効果を評価した。図 4はその結果を示している。これによりキャッシュを利用することで、性能を数倍から十数倍まで改善できることが確認できた。PHPソース実行前のコンパイル時間を節約できるOpcodeキャッシング（APC）、アプリケーションレベルのキャッシングは最も効果的である。サーバレベルのキャッシュにおいて、毎回更新されるWebページに対し、性能の改善の効果が見られなかった。

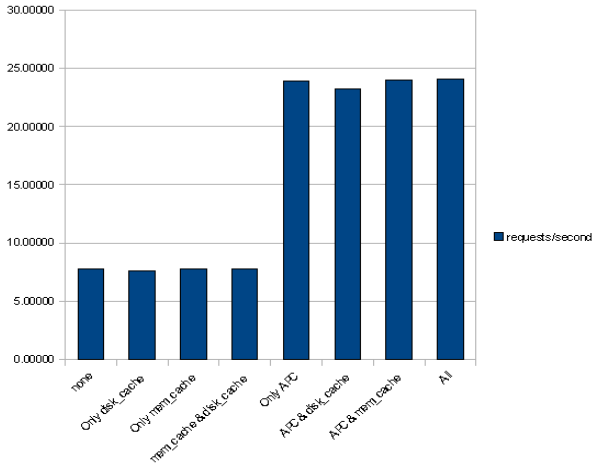

図 4 サーバおよび言語処理キャッシュの効果

次にアプリケーションレベルのキャッシュを以下のケースを計測した。

- **ケースa**：レイアウトを含まないキャッシュ
- **ケースb**：レイアウトを含むキャッシュ
- **ケースc**：キャッシュ未使用

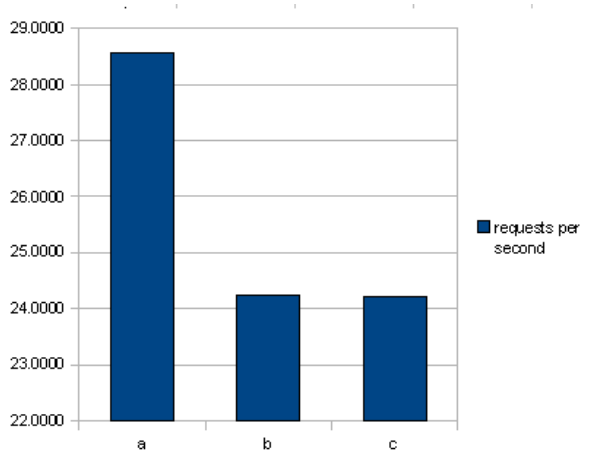

図 5 symfony アプリケーションのキャッシュ

結果は、図 5に示すようにレイアウトを含まないキャッシュ（ケース a）が一番効果的であった。これはレイアウトと組み込まない場合は、同じ結果とみなしているのではないかと推測している。

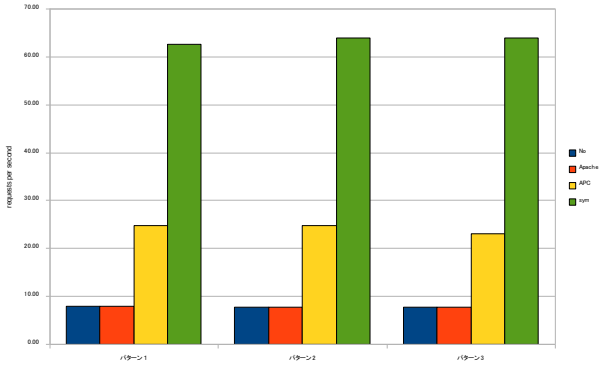

図 6 各種ワークロードにおけるキャッシュ

最後に図 6は利用パターンによってキャッシュ効果の違いの評価である。違いは確認できなかった。

5. 終わりに

本論文は、TPC-App ベンチマークを用いて、代表的な Web アプリケーションフレームワーク symfony を対象として性能評価を行い、ボトルネックを特定するとともに、性能改善の手がかりを提供する内容をまとめたものである。Web アプリケーションフレームワーク（以下はフレームワークと略称）は複数の再利用技術を取りまとめた Web アプリケーション開発技術であり、フレームワークを利用することにより、Web アプリケーション開発の生産性・保守性を大幅に向上させることができると期待されている。一方、フレームワークを利用する場合はアプリケーションの性能が著しく劣ると指摘されており、フレームワークを導入する際に問題視されている。本論文は実際に TPC-APP に準じて評価対象システムを symfony で開発し、利用パターンをパラメータとして指定可能なワークロードジェネレータ PW-GEN を実現した。さらに、Web アプリケーションのボトルネックとなりうる箇所にキャッシュを配置したり、アクセスパターンを変化させたりして、性能の変化をリクエスト応答数/秒で計測を行った。その結果、言語処理とアプリケーションにおけるキャッシュは最も効果的であることが分った。

今度の課題として、TPC-APP の全 14 通りの Web インタラクションの実装や、O/R マッピングにおけるデータベースアクセス最適化を行うことが挙げられる。

参考文献

[1] 藤原晃、楠本真二，井上克郎，大坪稔房，湯浦克彦，複雑度と機能量に基づくアプリケーションフレームワークの実験的評価，情報処理学会オブジェクト指向 2001 シンポジウム論文集,pp.85-94,2001.

[2] 善明晃由、小林隆志、佐伯元司，フレームワークの要求仕様に対する適合性の評価手法，電子情報通信学会技術研究報告.SS,ソフトウェアサイエンス，Vol.103,No.481 (20031120) pp.7-12 (2003) .

[3] Cal Henderson 著，スケーラブル Web サイト，オライリー・ジャパン，2006

[4] 玉井哲雄，ソフトウェア工学の基礎，岩波書店，2004

[5] 黒住幸光，芦沢嘉則著，最強フレームワーク Apache Struts1.2 パーフェクトガイド，技術評論社，2005

[6] マーチン・ファウラー著，長瀬嘉秀監訳，エンタープライズアプリケーションアーキテクチャパターン，株式会社翔泳社，2005

[7] 人気フレームワークは CakePHP，symfony，Zend---PHP の開発と労働環境の調査結果，http://itpro.nikkeibp.co.jp/article/NEWS/20081024/317714/

[8] 八木達矢，変化対応を考慮したシステム構築の考察，UNISYS TECHNOLOGY REVIEW，第 89 号，2006, http://www.unisys.co.jp/tec_info/tr89/8905.pdf

[9] TPC-APP，TPC Tranasction Processing Performance Council http://www.tpc.org/tpcw/default.asp

[10] Daniel Menasce，TPC-W: A Benchmark for E-Commerce, IEEE INTERNET COMPUTING，2002

[11] Ralph E. Johnson，中村宏明，中山裕子，吉田和樹，ソフトウェアテクノロジーシリーズ 1 パターンとフレームワーク，共立出版株式会社，1999

[12] 徒然なるままに Blog http://www.tsujita.jp/blojsom/blog/default/PHP/2007