部分評価を応用した動的 Web ページのキャッシュ機構

竹辺靖昭[†] 湯淺太一[††]

現在の多くの Web サイトでは，Web サーバ上で動作するプログラムで，データベースへのクエリを行い，動的に Web ページを生成する処理が行われている．我々は，部分評価の手法を応用し，こうした動的 Web ページの生成を高速化するシステムを開発している．このシステムは，Web サイトの開発において広く使用されている PHP という言語を対象とした部分評価器と，部分評価によって生成されたプログラムを Web サーバに配置するシステムからなる．この部分評価器は，更新の頻度が低いデータを静的と見なすことにより，これらのデータへのクエリを部分評価時に行うことができる．部分評価および生成されたプログラムの配置は，これらのデータが更新されるタイミングなどで行われる．Web ページを生成する時点では，変換結果に残された動的な部分のみが実行される．これにより，リアルタイムに更新される情報を含むページやパーソナライズ機能を持つページなど，さまざまなタイプの動的 Web ページを生成する負荷を低減することができる．本論文では，このシステムで使用している部分評価の手法および実装方法を紹介するとともに，パーソナライズ機能を持つ Web ページなどに対して実際にこの手法を適用した結果を報告する．

A Caching System for Dynamic Web Pages Using Partial Evaluation

Yasuaki Takebe[†] and Taiichi Yuasa[††]

On many Web sites today, Web pages are generated dynamically by programs, which are deployed on Web servers and execute queries to databases. We are developing a system to reduce the cost of dynamic Web page generation on those Web sites. This system consists of a partial evaluator for PHP, a widely used programming language for Web site development, and of a deployment system which installs residual programs to Web servers. This partial evaluator regards those data that are not updated frequently as static and executes queries on them during partial evaluation. This system performs partial evaluation and program deployment when such data are updated. On Web servers, only residual programs are executed to generate Web pages. By this method, we can reduce the cost to generate many sorts of Web pages, including personalized Web pages and Web pages that contain real-time information. In this paper, we describe the partial evaluation technique used in this system and implementation of this system. We also report the result of experiment in which we apply this system to a Web site with personalized pages.

1. はじめに

現在の多くの Web サイトでは，利用者に対してより適切な情報を提供するために，動的 Web ページを用いる機会が増えている．これらの動的 Web ページの実現手法として，Web サーバ上で動作するスクリプトで，データベース（以下 DB）からのデータの取得および Web ページの生成を行うというものが一般的になっている．こうしたシステムは Web-DB システムと呼ばれている．

Web-DB システムでは，ブラウザからの処理要求があるたびに，スクリプトが実行され，DB へのクエリなどの処理が行われるため，システムへの負荷が高くなってしまう．この問題を解決するため，動的なコンテンツを Web サーバ側でキャッシュするためのシステムが開発されているが，ページ単位のキャッシュを基本としているものが多く，リアルタイムに変更される情報を含むページやパーソナライズ機能を持つページになどには適用が難しい．また，スクリプト実行系の高速化手法も，DB サーバの負荷の低減を主眼とはしていない．

今回我々が開発中のシステム PHP-Mix [1)]は，スクリプト言語である PHP [2)]のサブセットを対象とした部分評価器と，部分評価によって生成されたスクリプ

† コグニティブリサーチラボ株式会社
Cognitive Research Laboratories
†† 京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University

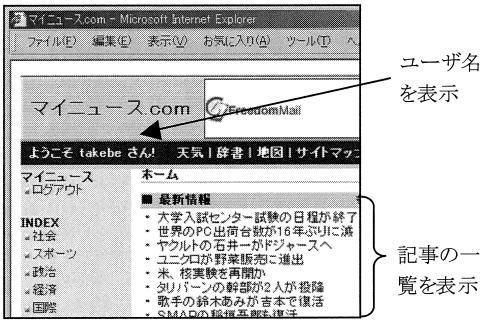

図 1 認証機能を持つ Web ページの例
Fig. 1 Example Web page with authentication.

トを Web サーバに配置するシステムからなる．部分評価により，更新の頻度が低いデータへのクエリは部分評価時に行われ，リアルタイムに変更されるデータへのクエリなど，ブラウザからの要求を受けてからでないと実行できない処理のみを行うスクリプトが生成される．

例として，**図 1** のように，認証機能を持つ動的 Web ページを部分評価することを考える．このページを生成するための仮想的なスクリプトは以下のようになる．

```
<html>
<? 認証情報からユーザ名を得る ?>
...
ようこそ<? ユーザ名表示 ?>さん!
...
■最新情報
<?
  データベースから記事を検索
  for ($title in 記事) {
    記事のタイトル表示;
  }
?>
</html>
```

これを部分評価することによって，記事の検索のようにブラウザからの要求と関係なく実行できる処理が行われ，以下のようなスクリプトが生成される．

```
<html>
<? 認証情報からユーザ名を得る ?>
...
ようこそ<? ユーザ名表示 ?>さん!
...
■最新情報
・大学入試センター試験の日程が終了
・世界の PC 出荷台数が 16 年ぶりに減
・ヤクルトの石井一がドジャースへ
・ユニクロが野菜販売に進出
...
</html>
```

認証情報の取得やユーザ名の表示のように，ブラウザからの要求に依存した処理はそのまま残される．

PHP-Mix は，DB の更新に合わせてこの例のようにスクリプトを部分評価し，結果を Web サーバに配置する．Web ページを生成する際には，部分評価されたスクリプトが実行されるため，更新の頻度が低いデータへのクエリは行われなくなる．このため，Web サーバ側でのキャッシュと同様に DB サーバの負荷を低減することができ，リアルタイムに変更される情報を含むページやパーソナライズ機能を持つページに対しても適用が可能である．

論文の構成は以下のとおりである．2 章では，論文の背景として，Web-DB システムとその高速化手法の概要および PHP の特徴について述べる．3 章では，PHP-Mix の全体構成と実装方法について説明する．4 章では，部分評価器が対象とする言語の仕様と，用いている部分評価の手法について述べる．5 章では，実験的に作成したパーソナライズ機能を持つニュース閲覧サイトに PHP-Mix を適用した結果を考察し，提案する方式を評価する．最後に，6 章でまとめと今後の課題について述べる．

2. 背　　景

この章では，Web-DB システムとその高速化手法および PHP の特徴について述べる．

2.1 Web-DB システムの概要

動的 Web ページの実現手法として一般的なものに，Web-DB システムがある．これは，Web サーバ上で動作するスクリプトで，DB からのデータの取得および Web ページの生成を行うというものである（**図 2**）．

Web サーバ上で動作するスクリプトは，以前は Perl などの汎用のスクリプト言語を用いて記述されていたが，近年では，Web ページのデザインの容易さなどから，JSP [3] などの HTML 埋め込み型のスクリプト言語を用いることが多くなっている．HTML 埋め込み型言語では，HTML 中に特殊なタグを用いてプログラムを記述することができる．また，Web-DB システムの開発に特化されているため，HTTP による要求の解析や DB へのクエリなど，Web-DB システムを開発するのに便利な機能をライブラリとして備えているものが多い．

本システムが対象とする PHP も HTML 埋め込み型言語の 1 つである．以下に PHP による動的 Web

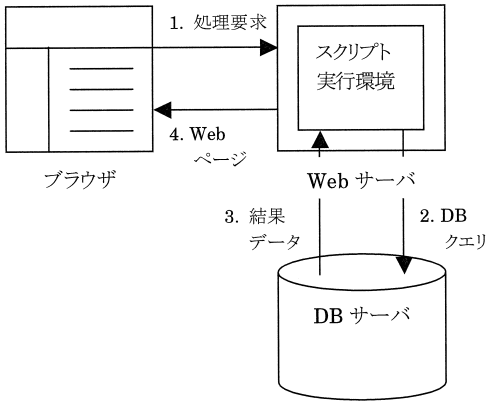

図 2 Web-DB システムの構成
Fig. 2 Structure of Web-DB system.

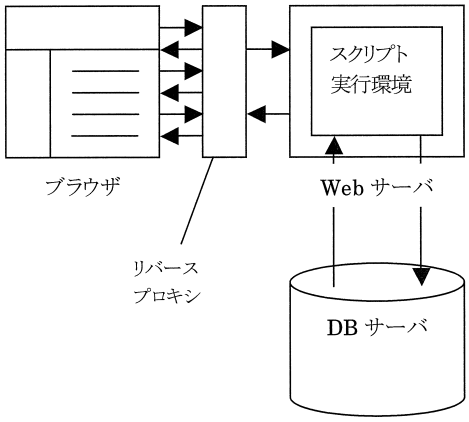

図 3 リバースプロキシによるキャッシュ
Fig. 3 Caching by reverse proxy.

ページのプログラム例をあげる．

```
<html>
<?
  $rs = mysql_query("SELECT * FROM tbl");
  while ($row = mysql_fetch_row($rs)) {
    echo $row[0] . "\n";
  }
?>
</html>
```

PHP では，<?...?> で囲まれた部分にプログラムが埋め込まれている．中で呼び出されている mysql_ で始まる関数は，MySQL[4]という DBMS へのアクセスを行う．ブラウザから要求を受けると，Web サーバでこのスクリプトが実行され，DB へのクエリが行われる．ブラウザには以下の HTML が返却される．

```
<html>
こんにちは
こんにちは
こんにちは
</html>
```

ここでは，DB 中のテーブル tbl には，3 行の「こんにちは」というデータが格納されているとしている．

2.2 従来の Web-DB システムの高速化手法

Web-DB システムでは，ブラウザからの処理要求があるたびに，スクリプトが実行され，DB へのクエリなどの処理が行われる．そのため，Web サーバだけでなく DB サーバの負荷も高くなってしまう．このため，システムの負荷を低減するためのさまざまな手法が開発されている．これらの手法は，大きく以下の 2 つに分けられる．

- スクリプト実行系の改良
- 生成した Web ページのキャッシュ

スクリプト実行系の改良としては，Web サーバと同一のアドレス空間で実行することにより，実行系を別プロセスとして起動するオーバヘッドを削減する手法があげられる．PHP の実行系は Web サーバと同一のプロセスで実行されるため，こうしたオーバヘッドは生じない．また，LibCGI[5]は，一般的な CGI プログラムを Web サーバと同一アドレス空間で実行することにより，性能を向上させている．また，実行するスクリプトをメモリにキャッシュするという手法もよく使われる．動的 Web ページの生成においては，同じスクリプトが何度も実行される場合が多いため，メモリへのキャッシュでも性能を向上することができる．PHP のスクリプトをメモリにキャッシュするシステムとしては APC[6]，Zend Accelerator[7]などがある．

スクリプト実行系を改良する手法の問題点としては，スクリプト内で実行される DB へのクエリは改良前と同様に行われるため，スクリプトを実行する Web サーバの負荷を低減することはできても，DB サーバの負荷は減らせないということがあげられる．

生成した Web ページのキャッシュは，Web サーバとブラウザの間に，リバースプロキシやキャッシュサーバを配置して行う（**図 3**）．リバースプロキシは，一度動的に生成された Web ページをキャッシュしておき，次に同じページのリクエストを受けたら，Web サーバに要求を流すことなく，キャッシュしてあるページを結果として返却する．DB の更新などのイベントによりキャッシュの内容は破棄される．この手法を採用したシステムとしては，DCache[8]や Oracle 社の Web Cache[9]があげられる．

この手法により，DB サーバの負荷も減らすことができるが，リバースプロキシで Web ページ全体をキャッ

シュした場合，内容がすべて固定の Web ページでしか効果が得られないため，リアルタイムに変更される情報を含むページやパーソナライズ機能を持つページに対して適用が困難になる．たとえば，DCache では，静的なデータの組合せだけから構成される動的 Web ページを準動的 Web ページと定義し，キャッシュの対象をそれらに限っている．

この手法の改良として，情報の更新の頻度に応じてあらかじめ Web ページをフラグメントに分割し，それらをキャッシュするというものがある．Challenger らは，Web ページをフラグメントに分割してキャッシュし，フラグメントと Web ページの依存関係に従ってデータの更新を伝播する手法により，キャッシュのヒット率を大幅に向上させている[10]．ただし，Challenger らの手法は，最終的にはフラグメントから構成される Web ページ全体をキャッシュすることを目的としており，パーソナライズ機能を持つ Web ページへの適用は難しい．ESI[11]では，フラグメントのみをキャッシュの対象とし，リバースプロキシでそれらを動的に組み合わせることにより，こうした用途にも適用が可能である．

これらの手法の問題点としては，あらかじめ Web ページをフラグメントに分けておかなければならないということがあげられる．このため，既存の Web サイトを変更して適用することは難しい．

2.3 PHP の概要

ここでは，PHP のプログラミング言語としての特徴について述べる．特徴としては以下のものがあげられる．

- C 言語に似た制御構文（goto 文はない）
- オブジェクト指向
- 文字列，連想配列などのデータ構造
- スクリプト言語的な柔軟性

はじめの制御構文については，既存のプログラミング言語と似ているため説明を割愛する．詳細については，PHP のマニュアル[2]を参照してほしい．

2 つめと 3 つめの特徴としてあげたオブジェクトや文字列，連想配列などのデータ構造も，Perl などの既存のスクリプト言語のものに類似しているが，代入や関数の引数として渡すときにコピーが行われるというところに PHP の特徴がある．たとえば，PHP で以下のスクリプトを実行すると，`$a[0]` は 4 になるが，`$b[0]` は 1 のままである．

```
$a = array(1, 2, 3);
$b = $a;  // 配列全体をコピー
$a[0] = 4;
```

一般に，手続き型言語の最適化を正しく行うためには，別名解析が欠かせないが，PHP の場合，上記の特徴のため別名が生じにくくなっている．部分評価器を実装するうえで好都合な特徴といえる．

また，上記の例にあるように，連想配列をリテラルとしてスクリプト中に直接書くことができる．部分評価を行ったときに，静的な値として得られたものが連想配列であっても，これを変換結果に埋め込むことができる．

4 つめのスクリプト言語的な柔軟性にはいろいろなものがあるが，部分評価を行う際に特に問題になるものとして，C などの大多数の言語では静的とされている機能が，実行時の動的な機能となっていることがある．ここでは，以下のものを例としてあげる．

- include 文，define 文
- eval
- 関数，クラスの定義
- 可変変数，可変関数

include 文，define 文は，C ではプリプロセッサによって前処理されるが，PHP では実行時に処理される．引数には PHP の式を指定することができる．include 文の例を以下にあげる．

```
$a = "foo.php";
include($a);
```

define 文は定数を定義する文であるが，PHP の場合，定数も実行時に定義されるため，プログラムを実行してみないと定数の値は決まらない．

eval は，渡された文字列をプログラムとして実行するものであり，これもどのようなプログラムを実行するのかは実行時に決まることになる．

また，PHP では，関数やクラスの定義も実行時に行われる．このため，以下のように条件分岐の中で関数を定義することもできる．

```
if ($a == 0) {
  function func() { 定義その 1 }
} else {
  function func() { 定義その 2 }
}
```

この場合，実行時に `$a` がとる値によって関数 `func` の定義が変わってくることになる．

可変変数は，PHP の式の評価結果を変数名として使う機能である．PHP では変数名の前に ‘$’ をつけるが，$ に続けて変数名の代わりに式を書くことにより，その式の評価結果を名前に持つ変数を参照することができる．たとえば，以下のプログラムを実行すると，変数 `$a` の値は 5 になる．

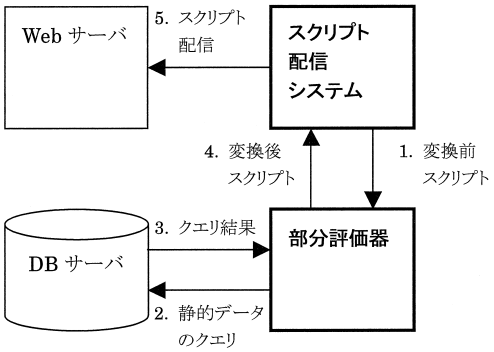

図 4 PHP-Mix の構成
Fig. 4 Structure of PHP-Mix.

```
$b = "a";
$$b = 5;
```

同様に，可変関数は，関数名として PHP の式の評価結果を使う機能である．

4 章で説明するように，これらのスクリプト言語的な特徴を使用したプログラムを部分評価するのは困難なため，部分評価器が対象とする PHP のサブセットでは使用が制限されている．

3. PHP-Mix の全体構成と実装

PHP-Mix は，スクリプト配信システムと部分評価器からなる（**図 4**）．太い線の部分が PHP-Mix のモジュールである．

スクリプト配信システムは，データの更新を検出すると，部分評価器を呼び出し，そのデータに依存しているスクリプトを変換する．データとスクリプトの依存関係は，設定ファイルによりユーザが指定する．また，スクリプト配信システムは，変換されたスクリプトを Web サーバに配信する処理も行う．

現在の実装では，データが更新されていることの検出は，一定の間隔で DB のデータの最終更新日時を示すファイルのチェックすることによって行う．ファイルに示されているデータの最終更新日時の方が部分評価結果よりも新しい場合，データが更新されていると判断される．DB のトリガ機構などは利用していない．また，実際の DB のデータと，DB のデータの最終更新日時を示すファイルとの同期をとる処理は，Web サイトの開発者が行う必要がある．

部分評価器は，静的なデータのクエリなど，ブラウザからの要求を受けなくても実行できる処理を実行し，入力されたスクリプトを変換する．この変換に用いている手法は 4 章で説明する．

スクリプト配信システムは Perl，部分評価器は C により実装した．

PHP の処理系はオープンソースとして公開されているため，部分評価器にその一部を流用することができた．ただし，オリジナルの PHP のうち，言語処理の実装は，構文木を作成せずにスクリプトを直接実行する構造だったため，4 章で説明する部分評価器の実装に再利用するのは困難であった．このため，一度構文木を作成してからスクリプトを実行する構造に変更するために，構文解析器や，構文木の実行処理を新たに実装した．

オリジナルの PHP のうち，文字列，配列といった PHP の基本的なデータ構造やそれらの演算の実装，API の実装を行っている部分は再利用が可能だった．

新たに開発した言語処理部の規模は，部分評価機能も含めて約 12,000 行になった．また，再利用できたもののうち，データ構造の処理などは約 6,000 行，API は約 21,000 行になった．

4. 部分評価器の概要

部分評価[12)]とは，プログラム変換の一手法であり，あるプログラムへの入力の一部だけが与えられているときに，与えられた入力のみで行える計算をあらかじめ行い，残りの不明な入力を処理する専用のプログラムを生成するというものである．部分評価では，プログラム中の変数や式などが，部分評価時にあらかじめ計算できることを静的，計算できないことを動的という．また，部分評価の過程で，プログラム中に出現する変数や式が動的か静的かの判定を行う解析を束縛時解析という．

PHP-Mix の部分評価器は，まず束縛時解析を行い，続いて部分評価を行うという手順で部分評価を行っている．ここでは，まず部分評価器が対象とする言語の仕様について述べた後，束縛時解析の手法，部分評価の手法についてそれぞれ説明する．

4.1 部分評価器が対象とする言語の仕様

PHP-Mix の部分評価器は，PHP のサブセットを対象としている．ここでは，元の PHP と比べて制限されている部分および追加されている部分について述べる．また，制限されている部分には，将来的には対応が可能なものと，原理的に対応が困難と思われるために制限しているものとの 2 種類がある．それらについては分けて説明する．

将来的には対応が可能な制限には以下のものがある．

- オブジェクト指向機能
- リファレンス
- 多くの API

PHP においては，リファレンスとは，変数の別名を作成する機能である．以下に例をあげる．

```
$a = array(1, 2, 3);
$b = &$a;  // リファレンスの代入
$a[0] = 4;
```

このプログラムを実行すると，単純に代入した場合とは異なり，`$b[0]` が 4 になる．

オブジェクト指向機能およびリファレンスは，開発期間の制約から現時点ではサポートされていない．また，オリジナルの PHP には 1200 以上の API が用意されているが，現状では，PHP-Mix はそのうちの 291 個のみをサポートしている．

原理的に対応が困難と思われる制限には以下のものがある．

- include 文，define 文
- 動的な関数およびクラスの定義
- eval，可変変数，可変関数
- eval や可変関数と同等の処理を行う API

include 文，define 文は，定数のみを引数にとることができるように制限されており，define 文は制御構文の中では使用できないように制限されている．この制限は，構文解析の時点で実行するコードを確定するために必要になる．

また，関数やクラスの定義も制御構文の中では行えないように制限されている（現時点の実装ではクラスの定義は制御構文の中でなくても行えない）．この制限は，構文解析の時点で関数やクラスの定義を確定するために必要になる．

eval，可変変数，可変関数はまったく使用できないように制限されている．また，PHP の API には，eval や可変関数と同様の処理を行うものがある．たとえば，`call_user_func` は，ユーザが定義した関数の名前を引数に渡すことにより，その関数を呼び出すことができる．こうした API もまったく使用できない．

追加した構文には以下のものがある．

- dynamic 宣言
- bind 文

dynamic 宣言は，変数を動的と宣言するためのものである．この構文は以下のようになっている．

`// dynamic` $\$var_1$`, ...,` $\$var_n$

この宣言により，束縛時解析の際に変数 $\$var_1, \ldots, \$var_n$ が動的とされるようになる．普通の PHP の処理系ではコメントとして処理されるように，‘//’ ではじめている．

bind 文は，変数がいくつかの決まった値をとることが分かっているということを部分評価器に指示するためのものである．この構文は以下のようになっている．

`// bind (`$\$var$ `:` val_1`, ...,` val_n`)`

stmt

`// endbind`

この文により，部分評価器は，文 *stmt* が開始される時点では $\$var$ が，$val_1, \ldots, val_n$ のいずれかの値をとると判断する．この構文も，普通の PHP の処理系ではコメントとして処理されるように，‘//’ ではじめている．

また，このほかに，部分評価を行うのに都合のよいように，ライブラリとして提供している関数の束縛時解析の結果をあらかじめ指定するための構文も追加している．この構文は以下のようになっている．

`bogusapi` $func$`(`$param\_list$`)`
`{`
  `bogusbt` bt_list`;`
  stmt
`}`

この宣言により，関数 $func$ は，束縛時 bt_list を持つということが指示される．bt_list には，4.2 節で説明する関数の束縛時と 1 対 1 に対応した定数の列が指定できる．指定できる定数は以下のとおりである．

- BT_STATIC
  引数がすべて静的な場合，関数の返り値が静的かつリテラルで表現可能であることを示す．
- BT_UNLIFTABLE
  引数がすべて静的な場合，関数の返り値が，静的だがリテラルでは表現できないことを示す．
- BT_DYNAMIC
  引数の束縛時に関係なく関数の返り値が動的であることを示す．
- BT_DEFINE
  関数の使用方法が define 文と同様の制限を受けることを示す．
- BT_UPDATEn
  関数の呼び出しにより，$n+1$ 番目の引数が更新されることを示す．n は 0〜9 が指定可能である．

なお，これらの定数は関数の束縛時と 1 対 1 に対応しているため，種類は上記のものに固定されている．

このようにして定義した関数を，以下では擬似 API と呼ぶ．

4.2 束縛時解析の手法

この節では，束縛時解析の手法について述べる．

4.2.1 基本的な構文の解析

PHP-Mix での束縛時解析は，変数の束縛時を基本として行われている．変数はプログラム全体で同じ束

縛時を持つとしている．

変数の束縛時を求める解析は以下のように行う．まず，初期状態では dynamic 宣言された変数のみが動的であり，また，動的な変数を含む式も動的になる．それ以外の変数，式はすべて静的である．

次に，プログラム全体を探索し，静的な変数に対して動的な式を代入する処理が行われていたら，その変数は動的とする．これを繰り返し行い，変数と束縛時の対応がそれ以上更新されなくなったら，解析は終了する．

解析中に静的な変数が動的となるケースには，大きく分けて以下の 3 つのものがある．

- 動的な式が代入されている場合
- 動的な制御構文の中で代入が行われている場合
- リテラル表現ができない値をスクリプト中に埋め込むのを避ける必要がある場合

まず，動的な式が代入されている場合，変数のとる値を部分評価時には計算できないため，動的にする必要がある．

動的な制御構文の中で代入が行われている場合の例としては以下のプログラムがある．

```
// dynamic $d
if ($d) {
  $x = "value1";
} else {
  $x = "value2";
}
echo $x;
```

変数 `$x` には静的な値しか代入されていないが，`$x` の値は `$d` に依存しているため，if 文を抜けた後の `$x` の値は部分評価時には計算できない．このため，制御構文の条件式が動的な場合，その内部で値を更新する処理が行われている変数は動的と解析している．while 文，do〜while 文，switch 文についても同様である．

リテラル表現ができない値をスクリプト中に埋め込むのを避ける必要がある場合の例としては以下のプログラムがある．

```
// dynamic $d
$rs = sql_query("...");
$row = sql_fetch_row($rs, $d);
```

`sql_query` は，DB へのクエリを行い，結果を返却する関数．`sql_fetch_row` は，クエリの結果から指定した行を 1 行だけ取り出す関数とする．2 番目の引数には行数を指定するとする．なお，PHP ではクエリの結果はリテラルで表現できない値である．

このプログラムでは，`sql_fetch_row` の 2 番目の引数は動的であるため，この関数呼び出しも動的となる．`$rs` には静的な値しか代入が行われていないが，`sql_fetch_row` の呼び出しの引数に使われているため，`$rs` を静的と解析すると，その値を `sql_fetch_row` の 1 番目の引数にすることになる．ところが，PHP では，クエリの結果のリテラル表現がないため，`$rs` がとっている値を部分評価の結果に埋め込むことができない．この問題を避けるため，`$rs` のようにリテラルで表現できない値をとる変数が，動的な関数呼び出しの引数などに使用されている場合には，その変数を動的と解析するようにしている．

4.2.2 API 関数の解析

PHP-Mix の部分評価器は，API 関数の束縛時をあらかじめ内部のテーブルに保持している．API 関数は，束縛時によって以下のように 3 つに分類されている．

- 動的な関数
- 静的な関数
- 定義を行う関数

動的な関数は，引数が静的であっても部分評価時には呼び出しが行われない．動的な関数の呼び出しは部分評価結果に残される．`printf` のように出力を行う関数，`date`，`rand` のように結果がリアルタイムに変更すべき関数などは動的としている．

静的な関数は，引数が静的であれば部分評価時に呼び出しが行われる．`sprintf` のように出力を行わない関数は静的としている．

定義を行う関数には，環境に影響を与えるものなどがある．以下のプログラムを考える．

```
// dynamic $str
setlocale("C");
$a = strtoupper($str);
$b = strtoupper("str");
```

`setlocale` はロケールの設定を行う関数である．`strtoupper` は，文字列を大文字にする関数であるが，このとき大文字にする文字はロケールによって違ってくる．たとえば，ドイツ語のロケールが設定されていたらウムラウトも大文字になる．

`$str` が動的のため，1 番目の `strtoupper` の呼び出しは動的になるが，2 番目の呼び出しは，引数が静的なため，部分評価時に行いたい．この場合，`setlocale` の呼び出しを静的にしても動的にしても変換結果に問題が生じる．動的としたときは部分評価時にロケールが設定されず，静的としたときには変換結果に `setlocale` の呼び出しが含まれなくなる．

PHP-Mix では，この問題を解決するため，こうした関数の呼び出しは定義と解析し，部分評価時に実行

されるとともに変換結果にも残すようにしている．定義を行う関数には，2 回呼び出しても問題が生じないように，define による定数の定義と同様の使用制限を設けている．

4.2.3 DB へのクエリ

PHP では，以下の機能を持つ API を組み合わせて DB へのアクセスを行う．

- DB への接続
- クエリの実行
- 行の取得

通常，DB への接続は一度だけ行い，クエリの実行は複数回行われる．クエリは動的にしか行われない場合もあるし，静的に行える場合もある．このように考えると，DB への接続を，前述の `setlocale` のように定義と見なすと都合が良い．

しかし，PHP の DB 接続 API は，接続ハンドルという値を返却する関数であり，クエリを実行する関数もこの値を使用するため，定義と見なすには適していない．このため，接続ハンドルを返却しない形に擬似 API を定義し，この中で DB 接続 API を呼び出すようにする．ユーザは，通常の DB 接続 API の代わりにこの擬似 API を使用する．

DB への接続を行う擬似 API の定義は以下のようになる．

```
bogusapi pm_sql_connect($host)
{
  bogusbt BT_DEFINE;
  global $_conn;
  $_conn = mysql_connect($host);
  if (!$_conn) {
    pm_exit("can't connect to db.");
  }
  ...
}
```

この関数の中では，MySQL という DBMS への接続 API が呼び出されている．束縛時は bogusbt 文により定義を行う関数と定義されている．DB への接続ハンドルは `$_conn` というグローバル変数に格納される．接続に失敗すると，`pm_exit` という関数が呼び出される．この関数は PHP-Mix の拡張であり，部分評価を失敗させるという特殊な関数である．

なお，擬似 API 中で使用するために PHP-Mix が拡張した関数には，`pm_exit` を含めて以下の 2 つがある．

- `pm_exit`
  引数を標準エラー出力に出力し，部分評価を失敗させる．
- `pm_print`
  引数を標準エラー出力に出力する．擬似 API のデバッグに使用する．

`pm_sql_connect` で得られた接続に対してクエリを行うための擬似 API の定義は以下のようになる．

```
bogusapi pm_sql_query_static($query)
{
  bogusbt BT_UNLIFTABLE;
  global $_conn;
  $rs = mysql_query($query, $_conn);
  return $rs;
}
```

この関数の中で呼び出している `mysql_query` 関数には，`$_conn` が渡される．これらの擬似 API を組み合わせることにより，呼び出す側のスクリプトでは接続ハンドルを使用することはなくなる．

DB へのクエリの実現でもう 1 つ考慮しなければいけないのが，クエリの結果中の行の位置である．クエリの結果は複数の行からなるが，DBMS によっては，最後に取得した行の位置が記憶されており，行の指定を行わなくても次々とクエリの結果中から行を取り出すことができる．MySQL の場合を例に見てみる．

```
$rs = mysql_query("SELECT ...");
while ($row = mysql_fetch_array($rs)) {
  echo $row["title"];
}
```

`mysql_fetch_array` はクエリの結果から行の取り出しを行う関数である．1 回目の呼び出しでは `$rs` に格納されたクエリの結果の 1 行目が，2 回目では 2 行目が，という具合に順番に結果を取り出すことができる．このようなプログラムを正しく解析するためには，`mysql_fetch_array` の呼び出しによって，`$rs` の値が更新されていると解析する必要がある．

このことを考慮し，クエリの結果から行を取り出す擬似 API を以下のように定義する．

```
bogusapi pm_sql_fetch_array($rs)
{
  bogusbt BT_STATIC, BT_UPDATE0;
  $row = mysql_fetch_array($rs);
  return $row;
}
```

bogusbt 文の指定により，この関数を呼び出すと 1 番目の引数が更新されると判断される．このため，たとえば，動的な制御構文の中で`pm_sql_fetch_array` 関数を呼び出すと，引数の式は動的と解析される．

表 1 コード生成の規則
Table 1 Code generation rule.

評価する文	実行される処理	生成されるコード
$\$\overline{x}$ = $\overline{exp}$	$\$x$ への exp の値の代入	
$\$\underline{x}$ = $\underline{exp}$		$\$x$ = exp
if ($\overline{\overline{test}}$) $stmt_1$ else $stmt_2$	$test$ が真なら $stmt_1$ を，偽なら $stmt_2$ を評価	$test$ が真なら $stmt'_1$，偽なら $stmt'_2$
if ($\underline{test}$) $stmt_1$ else $stmt_2$	$stmt_1$ と $stmt_2$ を評価	if ($test$) $stmt'_1$ else $stmt'_2$

4.2.4 bind 文の解析

PHP-Mix の部分評価器は，bind 文が現れると，一度 bind 文を if 文と同様の制御構文として全体の束縛時解析を行う．その後，bind 文が囲んでいる文の中だけで再度解析を行い，以下の条件を満たす変数について，可能ならば静的と解析する．

- bind 文の先頭で死んでおり，かつリテラルで表現可能な値をとる変数
- bind 文の先頭および bind 文から脱出するブロックで死んでいる変数

変数と束縛時の関係は，全体のものと，この解析の結果との 2 通りが保持される．この解析結果を用いて bind 文をどのように部分評価するかは 4.3.2 項で説明する．

4.2.5 ユーザ定義関数の解析

PHP-Mix の部分評価器は，ユーザ定義関数は，引数がすべて動的なものと，すべて静的なものの 2 つのバージョンがあるとして解析される．これにより，ある関数の呼び出しが複数回行われているときに，そのうちのどれかの引数が動的であるために，その関数呼び出しすべてが動的になるのを防ぐことができる．

また，PHP のユーザ定義関数では，グローバル変数やスタティック変数が使用できる．これらの変数の束縛時を正しく解析するためには，関数がどのような制御構文の中から呼び出されているかを考慮しなければならない．以下の例を考えてみる．

```
function foo()
{
  static $a = 0;
  $a++;
}
// dynamic $d
if ($d) {
  foo();
}
```

このプログラムでは，foo が呼び出されるかどうかは部分評価時には分からないため，$a は動的と解析するべきである．このように，関数の内部の解析は，その関数が動的な制御構文の中から呼び出されるかどうかを考慮して行っている．

4.3 部分評価の手法

この節では，部分評価の手法について述べる．

4.3.1 基本的な構文の部分評価

部分評価は，静的と評価された式，文を実行し，動的と評価された式，文をコードとして生成していくことで行っていく．束縛時解析の結果と，実行される処理，生成されるコードとの対応を**表 1** に示す．ここで，上線は静的な式，下線は動的な式を表す．また，$stmt'_1$，$stmt'_2$ はそれぞれ $stmt_1$，$stmt_2$ を部分評価して得られる結果である．

制御構文については if 文のみを載せているが，while 文，do〜while 文，switch 文についても同様である．

4.3.2 bind 文の部分評価

bind 文は以下のような構文をしている．

// bind ($var : $val_1, val_2, \ldots, val_n$)
$stmt$
// endbind

これは次のように変換される．

if ($var == val_1) {
  $stmt'[\$var/val_1]$
} else if ($var == val_2) {
  $stmt'[\$var/val_2]$
} else if (...) {
  ...
} else {
  $stmt'$
}

ここで，$stmt'[\$var/val_i]$ は，$var が val_i をとるとして $stmt$ を部分評価した結果であり，$stmt'$ は $stmt$ をそのまま部分評価した結果である．

リテラルで表現可能な変数で bind 文から脱出するブロックで死んでいない変数が bind 文内では静的と解析された場合，脱出時点でとっている値をその変数に代入する文を変換結果に挿入する．たとえば，以下

のスクリプトでは，$i は bind 文から脱出するブロックで生存している．

```
// bind ($i : 1)
$i = $i + 1;
echo $i;
// endbind
echo $i;
```

これは次のように変換される．

```
if ($i == 1) {
  echo 2;
  $i = 2;  // 脱出前に値を代入
} else {
  $i = $i + 1;
  echo $i;
}
echo $i;
```

bind 文の中では $i は静的と解析されるため，変換前のスクリプトにおける $i への代入は評価され，2 を値としてとるようになる．if 文を出る前にその値を $i に代入することにより，if 文の外の echo が正しく処理できる．

4.3.3 擬似 API の処理

擬似 API が定義されているファイルをインクルードすると，生成されたスクリプトでもそのファイルと同名のファイルをインクルードするように include 文が追加される．生成されたスクリプトから include されるファイルでは，擬似 API と同名の関数が定義されていることを前提としている．

このファイルの内容を，部分評価時と実行時とで別のものにしておくことにより，部分評価時と実行時で動作を変えることが可能である．たとえば，PHP-Mix で用意している DB 接続用の擬似 API で接続に失敗したときのエラー処理は，部分評価時には pm_exit の呼び出しによる部分評価の異常終了処理だが，実行時にはエラーメッセージの表示処理である．これにより，部分評価結果としてエラーメッセージが採用されるのを防ぐとともに，実行時にエラーが起きたときには適切なエラーメッセージを表示することが可能になる．

5. 評　　価

今回開発しているシステムを適用することにより，動的な Web ページを生成する負荷がどの程度低減されるかを実験するため，例題として**図 5** のようなデザインの Web ページを生成するスクリプトを作成し，有効な部分評価結果が得られるかどうかの検証および部分評価前と後での性能比較を行った．この Web ページは，ニュースを提供するサイトのトップページにパーソナライズ機能を持たせたものという想定である．

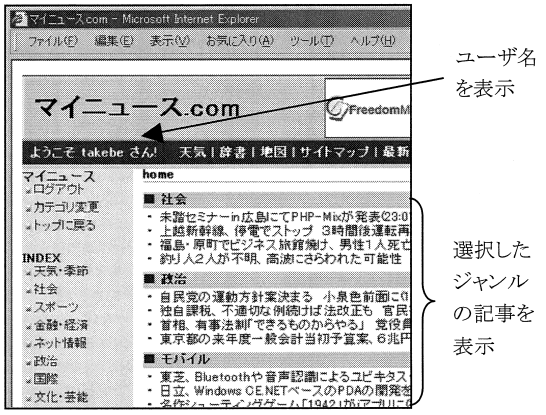

図 5 Web ページの例
Fig. 5 Example Web page.

この Web ページでは，パーソナライズ機能を実現するため，アクセスしている人を Cookie の情報をもとに識別し，DB に登録されたユーザ情報から興味あるジャンルを取得し，各ジャンルのニュースの一覧を DB から取得するという処理を行っている．図 1 のような単一のニュース一覧ではなく，訪問者が選択したジャンルのニュースの一覧を表示するため，はじめの例よりも複雑な処理が必要になる．

この Web ページの開発は PHP で行い，DB へのアクセスを PHP-Mix が用意している擬似 API を用いて行うようにした．スクリプトの行数は 218 行になった．

5.1 部分評価器の能力

Web ページ生成の処理のうち，各ジャンルのニュースの取得は部分評価時に行うことができるはずである．しかし，今回実装した部分評価器では，普通にスクリプトを書いた場合には，このようには部分評価されないことが多いと思われる．

うまく部分評価するために気をつけなければいけない点としては以下のようなものがある．

- 一時変数を同じ名前にしない．
- 動的な制御文の中ではできるだけ変数への代入を行わない．

2 点目の注意点のため，たとえば以下の仮想的なスクリプトのような処理はうまく部分評価できない．

```
if (認証が行われていたら) {
  個人情報を取得;
  $personal = パーソナライズされたニュース;
  $personal の情報を表示;
} else {
```

```
  $default = デフォルトのニュース;
  $default の情報を表示;
}
```

認証が行われているかどうかはブラウザからアクセスが行われてからでしか判断できないため，この if 文は動的になる．デフォルトのニュースは 1 種類だけなので，直感的には静的に評価が行われてもよさそうであるが，動的な if 文の中で代入が行われているため，`$default` のような変数は動的と判定されてしまう．

結果的には，上記の点に注意し，ユーザが選択しているジャンルを bind 文によって制限することによって，目標のとおりに部分評価を行うことができる．仮想的なスクリプトは以下のようになる．

```
if (認証が行われていたら) {
  個人情報を取得;
  パーソナライズされたジャンル一覧を取得;
} else {
  デフォルトのジャンル一覧を設定;
}
for ($i in ジャンル一覧) {
  $genre = ジャンル一覧[$i];
  // bind ($genre : 社会,  ...)
  そのジャンルのニュースを検索し表示;
  // endbind
}
```

ただし，今回の実験では，部分評価器の実装を行った本人がスクリプトを記述しているため，一般のユーザでも満足のいく結果を得られるかどうかについては今後評価していく必要がある．

5.2 負荷の低減

部分評価前と後のスクリプトで処理できるリクエスト数の比較，および部分評価に要する時間の測定を行った．実験に使用したシステムの構成は**表 2** のとおりである．

まず，処理できるリクエスト数は，クライアントから，マイクロソフトの Web Application Stress [13] というツールを使い，10 スレッドで間隔を空けずに 1 分間の連続アクセスを行うことにより測定した．部分評価前と後の処理リクエスト数および平均レスポンス時間を**表 3** に示す．ともに約 3 倍弱の性能を達成できていることが分かる．

また，クライアントより部分評価後のスクリプトに対して上記の連続アクセスを行っている状態で，スクリプトの部分評価をサーバ上で行い，要した時間を測定した．10 回の平均は 844 m 秒となった．

部分評価を行う時間は，部分評価前のスクリプトへの 1 アクセスに要する時間の 3 倍弱にとどまっている．部分評価は，新しい記事が入力されると行われるが，実際の運用においては，新しい記事が入力される間隔は 1 分以上は空いていると考えられるため，部分評価に要する時間が性能に及ぼす影響はほとんどないということができる．

表 2 実験に使用したシステムの構成
Table 2 Configuration of experimental system.

	クライアント	サーバ
CPU	Celeron/450 MHz	Celeron/600 MHz
メモリ	192 MB	384 MB
ソフトウェア	Windows 2000	RedHat Linux 6.2J Apache 1.3.20 MySQL 3.23.44 PHP 4.0.6 APC 1.1.0pl1

表 3 実験結果
Table 3 Result of experiment.

	部分評価前	部分評価後
リクエスト数	1,860	5,390
レスポンス時間	314 m 秒	110 m 秒

6. ま と め

部分評価の手法を応用し，動的な Web ページを生成する負荷を減らすシステム PHP-Mix を開発した．本システムは，PHP のサブセットを対象とした部分評価器とスクリプト配信システムからなる．実験により，パーソナライズ機能を持つ動的 Web ページに対しても，ページ内の静的な部分の部分評価を行うことにより，性能向上の効果を確認することができた．この Web ページは，簡単ながらもパーソナライズ機能を備えたものであり，認証にも Cookie を用いるなど，現実に運用されているものに近い機能を持っている．これを目標のとおり部分評価できたことで，本システムの実用性に関しては一応の評価ができる．

ただし，部分評価器に関しては改良すべき点が多い．まず，現時点ではサポートしていないオブジェクト指向などの PHP の言語仕様について考慮していく必要がある．また，実験の際にも明らかになったように，部分評価の手法が非常に素朴なものであるため，スクリプトの書き方によってはうまく部分評価が行われないという問題がある．現状でも，C などの手続き型言語に対しては，すでに優れた部分評価の手法が研究されており [14]，今後はこうした手法を適用していくことが必要と思われる．

今後の 1 つの応用として，ODB や XML-DB などの比較的低速で計算機資源を必要とする DB に対して

本手法を適用していくということが考えられる．現状では，Web-DB システムにおいては，高速な RDB を用いるのが一般的であるが，将来的には，開発効率の高さなどの利点から，ODB や XML-DB も普及していくことが予想される．DB が遅ければ遅いほど，DB へのアクセスを部分評価できたときに得られる効果が大きいため，本システムの利点をアピールすることができると考えられる．

謝辞 このシステムの開発は，IPA の平成 13 年度未踏ソフトウエア創造事業のプロジェクトとして行いました．未踏関係者の方々には，さまざまな機会において有益な助言をいただきました．また，富士通株式会社の上田健之氏には，言語処理系の開発一般について助言をいただきました．ここに感謝します．

参 考 文 献

1) PHP-Mix. http://www.cityfujisawa.ne.jp/~takebe/
2) Bakken, S.S., et al.: PHP Manual. http://jp.php.net/manual/en/
3) *JavaServer Pages(TM) Technology*. http://java.sun.com/products/jsp/
4) MySQL. http://www.mysql.com/
5) Venkitachalam, G. and Chiueh, T.: High-Performance Common Gateway Interface Invocation, *IEEE Workshop on Internet Applications* (*WIAPP '99*) (1999).
6) APC. http://apc.communityconnect.com/
7) Zend Accelerator. http://www.zend.com/store/products/zend-accelerator.php
8) Rajamani, K. and Cox, A.: A Simple and Effective Caching Scheme for Dynamic Content, Rice University Computer Science Technical Report, TR 00-371 (2000).
9) Oracle9iAS Web Cache. http://www.oracle.co.jp/9i/9ias/cache/
10) Challenger, J., Dantzig, P. and Iyengar, A.: A Scalable System for Consistently Caching Dynamic Web Data, *Proc. 18th Annual Joint Conference of the IEEE Computer and Communications Societies*, New York (1999).
11) Edge Side Includes. http://www.esi.org/
12) Jones, N., Gomard, C. and Sestoft, P.: *Partial Evaluation and Automatic Program Generation*, Prentice Hall (1993).
13) Microsoft Web Application Stress Tool. http://webtool.rte.microsoft.com/
14) Hornof, L. and Noyé, J.: Accurate Binding-Time Analysis for Imperative Languages: Flow, Context, and Return Sensitivity, *Partial Evaluation and Semantics-Based Program Manipulation* (*PEPM '97*), New York, pp.63–73, ACM (1997).

(平成 14 年 2 月 18 日受付)
(平成 14 年 6 月 10 日採録)

竹辺 靖昭（正会員）

1995 年京都大学理学部卒業．1997 年東京大学大学院理学系研究科情報科学専攻修士課程修了．同年富士通（株）入社．グループウェア，文書データベースの研究開発に従事．2001 年よりコグニティブリサーチラボ（株）勤務．Web サイトの開発およびスクリプト言語処理系の研究開発を行う．

湯淺 太一（正会員）

1977 年京都大学理学部卒業．1982 年同大学大学院理学研究科博士課程修了．同年京都大学数理解析研究所助手．1987 年豊橋技術科学大学講師．1988 年同大学助教授，1995 年同大学教授，1996 年京都大学大学院工学研究科情報工学専攻教授．1998 年同大学大学院情報学研究科通信情報システム専攻教授となり現在に至る．理学博士．記号処理，プログラミング言語処理系，超並列計算に興味を持っている．著書「Common Lisp 入門」（共著），「C 言語によるプログラミング入門」，「コンパイラ」ほか．日本ソフトウェア科学会，電子情報通信学会，IEEE，ACM 各会員．