PowerCMS for MT Ver.2

Power CMS for MT2.05のパフォーマンスと
大規模サイト構築・運用のHow To

アルファサード有限会社 野田純生

Ver2.05での高速化施策

- プラグインファイル軽量化(1.3MB→170KB)
- モジュール化とプラグイン統合
- SQL最適化
- メモリキャッシュ、ファイルキャッシュ、DBキャッシュ
- FastCGI対応、Memcached対応
- JavaScript/CSS圧縮

Ver2.05での高速化施策

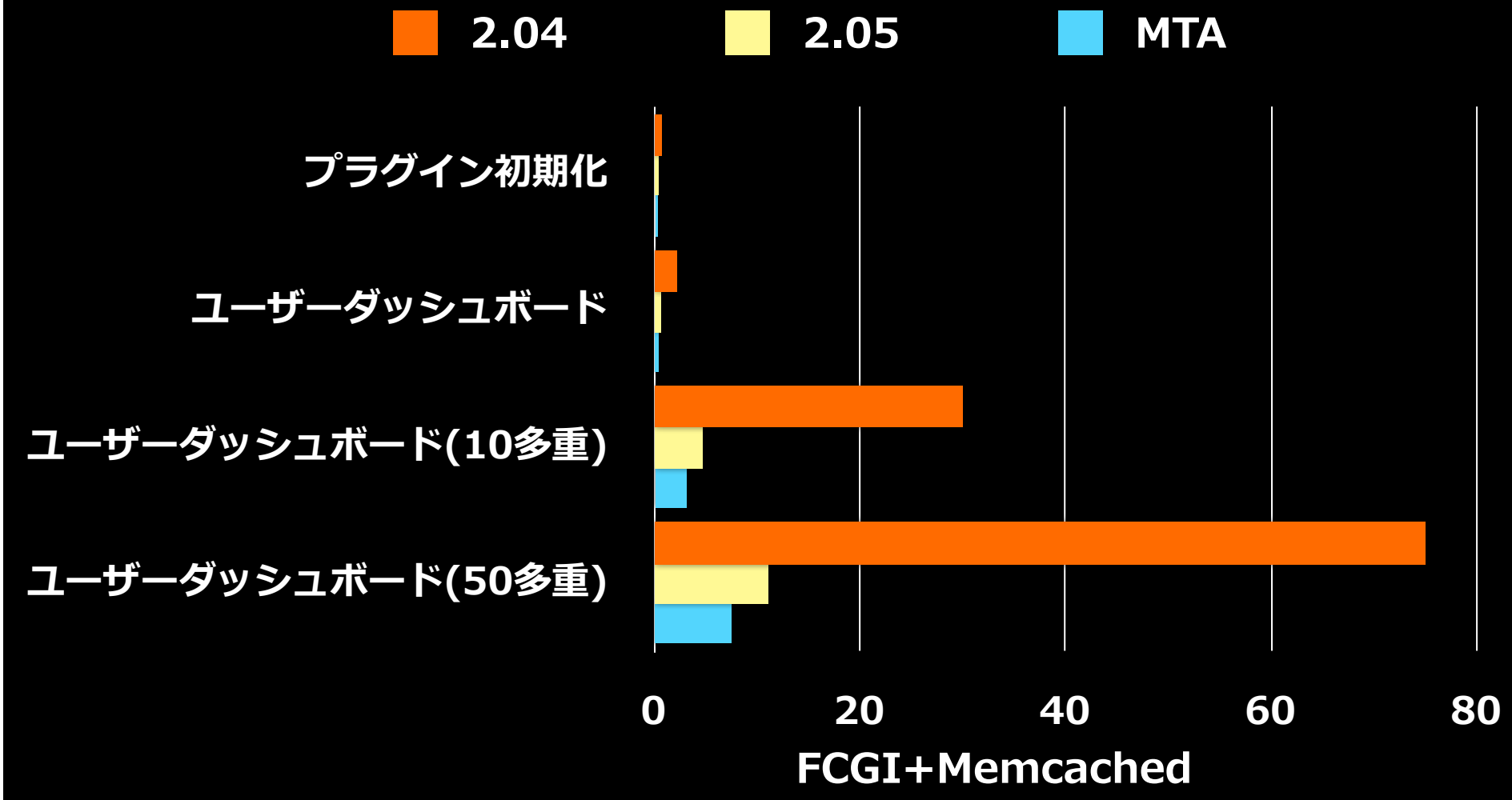
<$MT:Foo$>	[SQL]SELECT FROM ‘mt_foo’
→Foo	‘Foo’
…(別の処理)…	…
<$MT:Foo$>	メモリ内の値‘Foo’を返す
----------	----------
…次回…	…Memcached→ファイルキャッシュ
<$MT:Foo$>	→DBキャッシュの順で検索‘Foo’を返す* 見つからなければSQL発行

Ver2.05での高速化施策

テンプレートでキャッシュを制御できます。

```
<MTCMSCacheBlock
    key="Foo"                    # キャッシュキー
    blog_id="$blog_id"           # ブログID
    object_ds="category"         # 対象オブジェクト
    object_id="$cat_id"          # オブジェクトID
    by_user="1"                  # ユーザー毎かどうか
    children="1">                # 子オブジェクトに変化が
                                 # あったらキャッシュクリア
...
</MTCMSCacheBlock>
```

Ver2.05での高速化施策



Ver2.05での高速化施策

http://code.google.com/p/minify/

minify

Combines, minifies, and caches JavaScript and CSS files on demand to speed up page loads.

Project Home | Downloads | Wiki | Issues | Source

Summary | Updates | People

Minify is a PHP5 app that helps you follow several of Yahoo!'s Rules for High Performance Web Sites.

It combines multiple CSS or Javascript files, removes unnecessary whitespace and comments, and serves them with gzip encoding and optimal client-side cache headers.

Before

URL	Body
/wp-content/chili/recipes.css	2,447
/wp-content/themes/orangesky/style.css	10,923
/wp-content/chili/jquery-1.2.3.min.js	53,997
/wp-content/chili/chili-1.8b.js	11,793
/wp-content/chili/recipes.js	15,637
/js/email.js	771
/js/waitFor.jquery.js	241

```
Request Count:  7
Bytes Sent:     2,681
Bytes Received: 98,453

ACTUAL PERFORMANCE
--------------
Requests started at:
Responses completed at:
Total Sequence time:

RESPONSE CODES
--------------
```

After

URL	Body	Caching
/min/b=wp-cor...	2,959	max-age=1800, public, must-re
/min/g=js	24,971	max-age=1800, public, must-re

```
Request Count:  2
Bytes Sent:     783
Bytes Received: 29,049
```

The stats above are from a brief walkthrough which shows how easy it is to set up Minify on an existing site. It eliminated 5 HTTP requests and reduced JS/CSS bandwidth by 70%.

The design is somewhat similar to Yahoo's Combo Handler Service, except that Minify can combine *any* local JS/CSS files you need for your page.

Ver2.05での高速化施策

http://search.cpan.org/~pmichaux/JavaScript-Minifier/lib/JavaScript/Minifier.pm

CPAN

Home · Authors · Recent · News · Mirrors · FAQ · Feedback

in All CPAN Search

Peter Michaux > JavaScript-Minifier > JavaScript::Minifier

Module Version: 1.05 Source

NAME
SYNOPSIS
DESCRIPTION
EXPORT
SEE ALSO
AUTHORS
COPYRIGHT AND LICENSE

NAME

JavaScript::Minifier - Perl extension for minifying JavaScript code

SYNOPSIS

To minify a JavaScript file and have the output written directly to another file

```
use JavaScript::Minifier qw(minify);
open(INFILE, 'myScript.js') or die;
open(OUTFILE, '>myScript-min.js') or die;
minify(input => *INFILE, outfile => *OUTFILE);
close(INFILE);
```

Ver2.05での高速化施策


```
/*
# Movable Type (r) (C) 2001-2010 Six Apart, Ltd.
# This code cannot be redistributed without perm
# For more information, consult your Movable Typ
#
# $Id: mt.js 5417 2010-05-12 02:58:17Z takayama
*/

var pager;
var CMSScriptURI;
var ScriptURI;
var ScriptBaseURI;
var StaticURI;
var HelpBaseURI;
var Lexicon = {};
var itemset_options = {};

if ((!(navigator.appVersion.indexOf('MSIE') != -
      (parseInt(navigator.appVersion)==4))) {
    document.write("<style type=\"text/css\">");
    document.write("body { margin-top: -8px; mar
    document.write("</style>");
}

var origWidth, origHeight;
if ((navigator.appName == 'Netscape') &&
    (parseInt(navigator.appVersion) == 4)) {
```

```
var pager;var CMSScriptURI;var ScriptURI;var
ScriptBaseURI;var StaticURI;var HelpBaseURI;var
Lexicon={};var
itemset_options={};if((!(navigator.appVersion.indexOf('MSIE'
)!=-
1)&&(parseInt(navigator.appVersion)==4))){document.write("<s
tyle type=\"text/css\">");document.write("body { margin-top:
-8px; margin-left: -8px; }");document.write("</style>");}
var
origWidth,origHeight;if((navigator.appName=='Netscape')&&(pa
rseInt(navigator.appVersion)==4)){origWidth=innerWidth;origH
eight=innerHeight;window.onresize=restore;}
function
restore(){if(innerWidth!=origWidth||innerHeight!=origHeight)
location.reload();}
function
doRebuild(blogID,otherParams){window.open(CMSScriptURI+'?
__mode=rebuild_confirm&blog_id='+blogID+'&'+otherParams,'reb
uild_blog_'+blogID,'width=400,height=400,resizable=yes');}
function openManual(section,page){var url;if(page)
url=HelpBaseURI+'help/'+section+'/'+page+'/';else
if(section)
url=HelpBaseURI+'help/'+section+'/';else
url=HelpBaseURI+'help/';window.open(url,'mt_help','scrollbar
s=yes,status=yes,resizable=yes,toolbar=yes,location=yes,menu
bar=yes');return false;}
function countMarked(f,nameRestrict){var count=0;var
```

Ver2.05での高速化施策

<MTCSSCompressor>
　　...CSS::Minifierでコードを圧縮
</MTCSSCompressor>
<MTJSCompressor>
　　...JavaScript::Minifierでコード圧縮
</MTJSCompressor>

Ver2.05での高速化施策

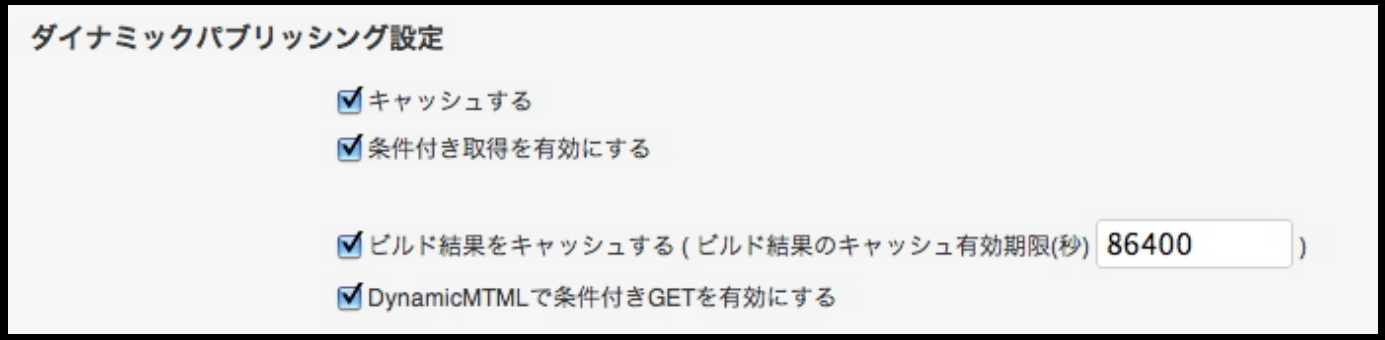
New Configration for mt-config.cgi

```
EnableCMSFileCache 1
# ファイルキャッシュを有効に
EnableCMSMemcached 1
# Memcachedを有効に
DisableSystemMenu 1
# システムメニューを非表示に
DisableCreateMenu 1
# システムスコープの新規作成メニュー
# を無効化(ブログ/ウェブサイトが多い場合)
CSSCompressor 1 #cssを圧縮
JsCompressor 1 #JavaScriptを圧縮
```

Ver2.05での高速化施策

DynamicMTMLのパフォーマンス対策

新たに条件付きGETに対応(304を返しブラウザのキャッシュを使わせることが可能に)



Ver2.05での高速化施策

DynamicMTMLのパフォーマンス対策

コールバックプラグインでキャリア毎にキャッシュする

```
<?php
function myplugin_pre_run( $mt, &$ctx, &$args,  ) {
    $cache = $args[ 'cache' ];
    if ( $cache )  {
        $agent = get_agent(); // ex: iPhone
        $args[ 'cache' ] = $agent . '_' . $cache;
        // iPhone用のキャッシュ
    }
}
?>
```

ここからは一般的なお話です

- システムの設計
  - 原則：下から順に設計する
    下位レイヤから検討する
- ネットワークなら

→回線速度

→ネットワークカード

→CPU性能

→サーバーの調整

→ブラウザ

- サーバーマシンなら
  - →メモリ
    - →CPU性能
      - →HDD
        - →OS
          - →専用か共用か
            - →サーバー設定
              - →CMS
                - →ブラウザ

- サーバーマシンのスペック
  - CPU

    プロセッサ数、コア数よりはクロック数を優先

    Perlはマルチコアを有効に利用できない
  - メモリ

    できるだけ多く積む。4GB程度推奨

    FAQ：4GB以上のメモリを利用するなら64bit OS、4GB以下は32bit OSの方がパフォーマンスがいい

● 共用サーバーと専用サーバー

MTは再構築時、CPUをほぼ90%程度利用する

共用で再構築が重なるとパフォーマンス低下

→ 共用サーバー、VPSはメモリも少ない

→ 専用サーバーをお勧めします

● ブラウザについて

Power CMS2では、管理画面にJavaScriptを多用してユーザービリティを向上

→ IE9のJavaScriptは、IE6の70倍高速

→ 最新のブラウザをご利用下さい

- DBのチューニング(1)
  - 基本

    DBサーバーとWebサーバーを分ける
    DBサーバーは十分なスペックのマシンを割り当てる(メモリは4GB以上)
    memcachedなどのメモリキャッシュを利用する(mt-config.cgiに設定を追記するだけ)
    例:

    ```
    MemcachedServers 127.0.0.1:11211
    MemcachedDriver Cache::Memcached::Fast
    ```

- DBのチューニング(2)
  - MyISAMではなくInnoDBを利用する
  - mysqltunerの利用
    (http://blog.mysqltuner.com/)
    実行すると、my.cnf の設定地をアドバイスしてくれる
    ある程度長期に利用したDBに対してのみ有効
    再起動後24時間以内ではチェックしない

- FastCGIの利用
  - CGIの代わりにFastCGIで動かす
  - mot_fcgidの利用(http://www.movabletype.jp/documentation/developer/server/fastcgi.html/)

    メモリに常駐するためインスタンスの起動が高速化される

    メモリを多く積むこと、定期的にtouch(又はApacheを再起動すること)

続いてMTにフォーカスした話を

パブリッシュオプションを見直す

パブリッシングオプションを見直す

- スタティック(既定)
- ダイナミック
- 公開キュー経由
- 手動
- 公開しない
- ＋部分的に動的(Power CMS for MT)

パブリッシングオプションを見直す

スタティック	閲覧者に高速(安全)、管理側(再構築)に負荷	共通部分をいかにキャッシュ、モジュール化するか
ダイナミック	更新頻度＞閲覧数 動的処理	キャッシュ、条件付きGETの利用、部分的にスタティック
公開キュー経由	リアルタイム性を問わない(ある程度犠牲にできる)	別サーバー(DBサーバー等)でのタスク実行を検討
手動	.cssや.js等	設定等の変更時のみ再構築するもの
部分的動的	動的処理を部分的に行う	キャッシュ設定、DBダウン時の対策

モジュールとキャッシュの活用

モジュールとキャッシュの活用

尚、製品版については下記の違いがあります。

- OPMLに加えCSVからのインポート、CSVエクスポートに対応
- グループ機能に対応(リンクグループ)
- 本文テキスト欄のリッチテキストエディタ対応

以下、「リンクをオブジェクトで管理するLinkプラグイン」のページと一部重複する部分がありますが、概要についてご紹介します。

Linkプラグインについて

- 外部リンクの「名前」「URL」「RSS」等の情報をオブジェクトとして管理できます。
- リンク「タグ」をつけることができます。
- 管理画面での検索に対応しています。
- エディタにリンクを貼り付けることができます。
- リンクをMTタグで出力することができます。
- スタティック/ダイナミック・パブリッシングの両方に対応しています。
- OPML/CSV形式のファイルからリンクをインポートすることができます。
- CSVファイルへのエクスポートが可能です。
- 一覧画面、または定期実行タスクによってリンクチェックを行えます。
- カスタム・パーミッション/ロールを作成します。
- 個別リンクのページを表示する場合は、オプションプラグインのViewerをインストールします。Viewerのインストールと利用方法についてはカスタムオブジェクトの記事を参照してください。

スクリーンショット

Recent Entries

- リンクをオブジェクトで管理するLinkプラグイン(製品版)
- 携帯サイトで絵文字を利用する
- バナー/広告配信・管理を行う(Campaign)
- 管理画面でブログ/ブログ記事の呼称を変更する
- 新しくなったグループ&ソート機能

Categories

- Power CMS 2
- プラグイン
- テンプレート作成Tips
- 技術情報
- イベント・セミナー情報
- リリース&Update情報
- サポート
- その他

モジュールとキャッシュの活用

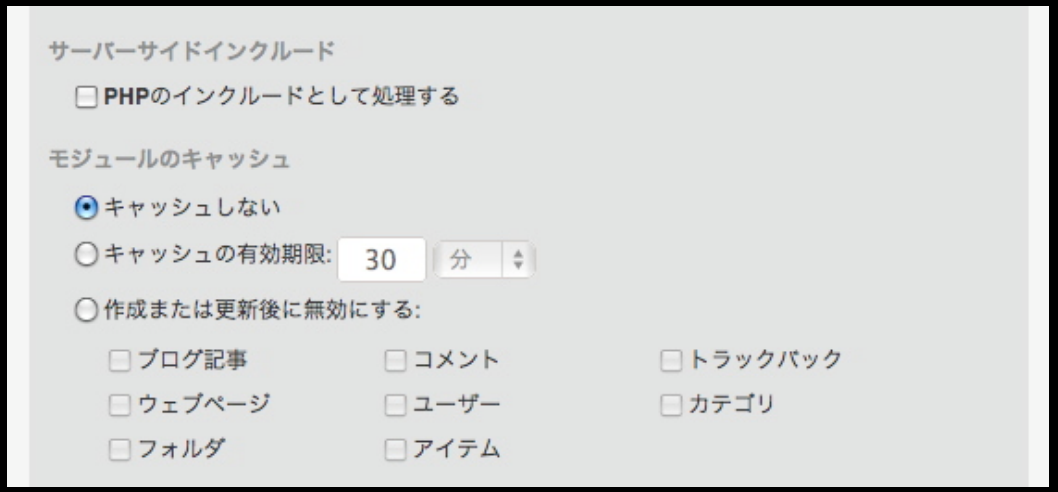
- MTのモジュールキャッシュを使う
- サーバーサイドインクルードを使う
- インデックス・アーカイブにしてインクルードする
- Power CMS for MTのキャッシュを使う

モジュールとキャッシュの活用

- サーバーサイドインクルードを使えば更新部分の反映に再構築は不要

  SSIを使うとHTTPヘッダでのブラウザキャッシュが使われなくなる点に注意

モジュールとキャッシュの活用



● モジュールのキャッシュは時間単位

<$MTInclude module="モジュール名" cache="1" ttl="600"$>

更新トリガを指定可能

Power CMS for MTのキャッシュ

●メモリキャッシュ(同一リクエストでのみ有効)

```
<MTRequestCacheBlock key=”foo”
blog_id="$blog_id">.....
</MTRequestCacheBlock>
```

●メモリ>Memcache>ファイル>DBキャッシュ順

```
<MTCMSCacheBlock key="Foo"
blog_id="$blog_id" object_ds="category"
object_id="$cat_id"  children="1">
...</MTCMSCacheBlock>
```

MultiBlogの再構築トリガ

MultiBlogの再構築トリガ

- アーカイブタイプ単位の設定のため負荷が高くなりやすい。単一アーカイブであればPower CMSのタグを利用可能

<$MTRebuilIndexByID template_id="1"$>

PowerCMS for MT Ver.2