SYMETRIC

GENECODE

ジーンコード ver 3

ユーザマニュアル

パフォーマンスチューニングガイド

第 1 版

最終更新日 2018/12/20

目次

改訂履歴

版数	発行日	改訂内容
第 1 版	2018 年 12 月 20 日	・初版発行

第1章　　はじめに

ジーンコードのパフォーマンスチューニングを行うには、以下の 2 つの方法があります。

- アプリケーションのチューニング：テンプレートやコントローラのパフォーマンスチューニング
- サーバーのチューニング：Apache のパフォーマンスチューニング

これらのチューニングを行うことで、ネットワーク構成やハードウェア構成を変更することなく、パフォーマンスを改善することができます。

第2章　　テンプレートのパフォーマンスチューニング

テンプレートのパフォーマンスチューニングには、以下のポイントがあります。

- jQuery セレクタの見直し
- <gc-script>タグ内における jQuery オブジェクトの変数化
- 負荷の高い jQuery 関数の使用を避ける

一般的にjQuery セレクタの見直しと、jQuery オブジェクトの変数化による効果は大きく、これだけでパフォーマンスが 2 倍以上に改善することもあります。

2-1　　jQuery セレクタの見直し

jQuery セレクタは、パーツのセレクタや<gc-script>タグ内での jQuery()関数などで使用されています。指定されている jQuery セレクタを以下の 4 つのルールに基づいて変更してください。

なお、以下の例では<gc-script>タグ内で jQuery セレクタを指定している場合のチューニング例を記載していますが、パーツのセレクタも同様のルールでパフォーマンス改善が可能です。また、<script>タグ内でのセレクタ見直しはサーバーパフォーマンスに影響しません。

(1)　　先頭が ID セレクタになるようにする

セレクタ書式の先頭に、親となる要素を ID セレクタで追加指定します。

例:

【変更前】

```
var $e = $(".header");
```

【変更後】

```
var $e = $("#mainContents .header");
```

なお、既に先頭の要素が ID セレクタであれば、不要です。

(2) クラスセレクタや属性セレクタは要素セレクタと同時に使用する

クラスセレクタを使用する際には、その要素名も合わせて指定します。

例:

【変更前】

```
var $e = $(".news");
```

【変更後】

```
var $e = $("ul.news");
```

以下のような属性セレクタの場合にも、要素名を指定します。

例:

【変更前】

```
var $e = $("[type=radio]");
```

【変更後】

```
var $e = $("input[type=radio]");
```

(3) 可能なら子孫セレクタを子セレクタに変更する

DOM の構造上、子孫セレクタを使用する必要がない場合には、子セレクタで代用します。たとえば ul 要素直下の li 要素を指定する場合は、子セレクタでの代用ができます。

例:

【変更前】

```
var $e = $("ul.news li");
```

【変更後】

```
var $e = $("ul.news > li");
```

(4) :contains()や:has()疑似セレクタの使用を避ける

:contains()セレクタや:has()セレクタは非常に負荷の高い処理のため、使用を避けるようにします。回避不能な場合は、繰り返し処理の中では使用しないように変更するなど、使用回数を削減します。

2-2　jQuery オブジェクトの変数化

jQuery オブジェクトは生成の度に要素の検索負荷がかかるため、jQuery オブジェクトの生成回数を削減することが重要です。生成した jQuery オブジェクトを変数化することで、jQuery オブジェクトの生成回数を削減することができます。

以下の手順に従い、<gc-script>タグ内のスクリプトコードを変更してください。

(1)　生成した jQuery オブジェクトを変数化する

同一の jQuery オブジェクトを 2 回以上生成している場合は、一度生成した jQuery オブジェクトを変数化します。

例:

【変更前】

```
$("ul.news").removeAttr("style");
$("ul.news").addClass("top");
```

【変更後】

```
var $news = $("ul.news");
$news.removeAttr("style");
$news.addClass("top");
```

(2)　$(this)も変数化する

each()関数などで、$(this)を複数回使用している場合も変数化します。

例:

【変更前】

```
$("#mainContainer").find("div").each(function() {
    if (!$(this).hasClass("main")) {
        $(this).addClass("sub");
    }
});
```

【変更後】

```
$("#mainContainer").find("div").each(function() {
    var $this = $(this);
    if (!$this.hasClass("main")) {
```

```
            $this.addClass("sub");
        }
});
```

(3) 他の jQuery オブジェクトを起点に jQuery オブジェクトを生成する

セレクタの途中までが一致する、複数のセレクタを使用する場合は、共通となる起点に対する jQuery オブジェクトを事前に生成します。既存のセレクタは、起点からのセレクタ指定に変更します。

例:

【変更前】

```
$("#container div.thumbnail ul li a").addClass("compact");
$("#container div.thumbnail ul li img").addClass("small-image");
```

【変更後】

```
var $li = $("#container div.thumbnail ul li");
$li.find("a").addClass("compact");
$li.find("img").addClass("small-image");
```

2-3　負荷の高いjQuery 関数の使用を避ける

HTML 文字列を引数に指定する jQuery 関数は、サーバー処理の負荷が高い関数です。したがって、HTML 文字列を指定しない別の関数で代替したり、呼び出し回数を制限したりするようにしてください。

例えば、以下のような関数はサーバー負荷が高い処理になります。

- .append("HTML 文字列")
- .html("HTML 文字列")
- .prepend("HTML 文字列")
- .replaceAll("HTML 文字列")
- .replaceWith("HTML 文字列")
- jQuery("HTML 文字列")

以下はサーバー負荷の高い html()関数を避けた例になります。

■変更前

```
$(".container").html($("ul.news").html());
```

■変更後

```
$(".container").empty().append($("ul.news"));
```

第3章　　サーバーのチューニング

ジーンコードサーバー（Apache）のチューニングを行う場合は、主に 4 つのポイントがあります。

- Apache プロセス数の上限を設定する
- 短いタイムアウトを設定する
- デバッグログ出力を抑制する
- 変換対象外 URL を無変換設定する
- キャッシュを利用する

3-1　　Apache プロセス数の上限設定

使用可能メモリと Apache プロセスの使用メモリから、Apache プロセスの上限数を設定します。

Apacheを停止した状態でfreeコマンドを実行すると、使用可能メモリの確認ができます。以下の例では、空きメモリが 7,510[MB]です。

```
[root@genecode ~]# free -m
             total       used       free     shared    buffers     cached
Mem:          8010       1950       6060          0        220       1230
-/+ buffers/cache:        499       7510
Swap:         8015          0       8015
```

Apache プロセスの使用メモリは、ジーンコードの設定ファイル httpd-genecode.conf により指定が可能です。出荷時設定では 150[MB]です。

```
LoadModule gc_html_module modules/mod_gc_html.so
<IfModule gc_html_module>
   :
   :
  GCHtmlMaxRss 150
</IfModule>
```

プロセス数の上限値は、(Apache に割り当て可能なメモリ)÷(Apache プロセスの使用メモリ)で算出が可能です。算出した値は、httpd.conf で ServerLimit ディレクティブ及び MaxClients ディレクティブの設定値に指定します。

以下の例では、7500[MB]÷150[MB]で求まった 50 を最大プロセス数として設定しています。

例:/etc/httpd/conf/httpd.conf

```
<IfModule prefork.c>
StartServers       8
MinSpareServers    5
MaxSpareServers   20
ServerLimit       50
MaxClients        50
MaxRequestsPerChild 4000
</IfModule>
```

3-2　タイムアウトの設定

オリジンサーバーからのレスポンスが非常に遅い場合、オリジンサーバーからレスポンスが完了するまでの間、Apache プロセスが占有されます。そのため、新規リクエストを処理可能な Apache プロセスが減少し、ジーンコードサーバーのレスポンスタイム低下が発生することがあります。

タイムアウトを現実的に許容可能な範囲内での最小値に設定することで、オリジンサーバーのレスポンス低下に伴ってジーンコードサーバーのレスポンスタイムが急激に悪化する現象を抑えることができます。

以下の例では、タイムアウトを 30 秒に設定しています。

例:/etc/httpd/conf/httpd.conf

```
Timeout 30
ProxyTimeout 30
```

3-3　デバッグログを抑制する

アプリケーションログ（app_log）のデバッグログ出力を抑制することで、ログ出力コストを抑えることができます。デバッグログ出力量が多い場合に効果的です。

具体的には、httpd-genecode-vhost.conf でログレベル GCHtmlLogLevel をデバッグを表す 2 から 1 へと変更します。

例: httpd-genecode-vhost.conf

```
GCHtmlLogLevel 1
```

3-4　変換対象外 URL を無変換設定する

ジーンコードは Content-Type ヘッダが text/html となっているレスポンスを HTML 変換します。変換対象外 URL のうち、Content-Type ヘッダが text/html になる URL があれば、ジーンコードが動作しないように無変換設定します。Ajax レスポンスなど、変換を行わない HTML コンテンツが多い場合に効果的です。

以下の例では、.ashx 拡張子の URL をすべて変換対象外に設定します。

例: httpd-genecode-vhost.conf

```
<Location ~ "¥.ashx$">
  GCHtmlEngine Off
</Location>
```

3-5　キャッシュを利用する

ジーンコードには HTML キャッシュ機能があります。HTML キャッシュ機能は、キャンペーンページのような静的ページを変換するような場合に効果を発揮します。

なお、ログインしたユーザーによって表示内容が変化するといった動的ページではキャッシュを利用できません。