[轉]優化 PHP代碼的 42條建議

作者是 Administrator
週六 , 27 六月 2009 18:09 - 最近更新 週二 , 11 八月 2009 12:17

英文版權歸 Reinhold Weber所有，中譯文作者 yangyang（aka davidkoree）。
雙語版可用於非商業傳播，但須註明英文版作者 、 版權信息，以及中譯文作者 。 翻譯水平有限，請廣大 PHPer指正。如果一個方法可靜態化，就對它做靜態聲明 。 速率可提升至 4倍 。 **2.ec**
ho比 print快。
錯誤
:
根據 **php**所在服務器環境的不同，某些情況下 **echo**會比 **print()**慢。
這裡有一篇 **echo**和 **print()**對比的帖子，對 **echo**和 **print()**的速度進行了測試： **http://www.learnphponline.com/php-basics/php-echo-vs-print**
。
3.
使用 **echo**的多重參數（譯註：指用逗號而不是句點）代替字符串連接
。
錯誤
:
基準測試證明這又是一條與事實完全相反的建議。
php引擎處理以點號連接（或者使用雙引號）傳遞給 **echo**的變量實際上是更快的。
4.在執行 for循環之前確定最大循環數，不要每循環一次都計算最大值 。
5.註銷那些不用的變量尤其是大數組，以便釋放內存 。
6.盡量避免使用 __get，__set，__autoload。
7.require_once()代價昂貴 。
8.在包含文件時使用完整路徑，解析操作系統路徑所需的時間會更少 。
9.如果你想知道腳本開始執行（譯註：即服務器端收到客戶端請求）的時刻，使用 $_SERVER['REQUEST_TIME']要好於 time()。
10.函數代替正則表達式完成相同功能 。
11.str_replace函數比 preg_replace函數快，但 strtr函數的效率是 str_replace函數的四倍 。
12.如果一個字符串替換函數，可接受數組或字符作為參數，並且參數長度不太長，那麼可以考慮額外寫一段替換代碼，使得每次傳遞參數是一個字符，而不是只寫一行代碼接受數組作為查詢和替換的參數 。

13.使用選擇分支語句（譯註：即 **switch case**）好於使用多個 **if**，**else if**語句
。
錯誤
:
這條建議毫無意義。
關於是要使用 **switch/case**還是 **if/else**的問題，其實完全是個代碼風格問題，而它們的效率相差無幾。
14.用 @屏蔽錯誤消息的做法非常低效 。
15.打開 apache的 mod_deflate模塊 。
16.數據庫連接當使用完畢時應關掉 。
17.$row['id']的效率是 $row[id]的 7倍 。
18.錯誤消息代價昂貴 。
19.盡量不要在 for循環中使用函數，比如 for ($x=0; $x < count($array); $x)每循環一次都會調用 count()函數 。
20.在方法中遞增局部變量，速度是最快的 。 幾乎與在函數中調用局部變量的速度相當 。
21.遞增一個全局變量要比遞增一個局部變量慢 2倍 。
22.遞增一個對象屬性（如： $this->prop++）要比遞增一個局部變量慢 3倍 。
23.遞增一個未預定義的局部變量要比遞增一個預定義的局部變量慢 9至 10倍 。
24.僅定義一個局部變量而沒在函數中調用它，同樣會減慢速度（其程度相當於遞增一個局部變量） 。 PHP大概會檢查看是否存在全局變量 。
25.方法調用看來與類中定義的方法的數量無關，因為我（在測試方法之前和之後都）添加了 10個方法，但性能上沒有變化 。

26.派生類中的方法運行起來要快於在基類中定義的同樣的方法。

27.調用帶有一個參數的空函數，其花費的時間相當於執行 7至 8次的局部變量遞增操作。 類似的方法調用所花費的時間接近於 15次的局部變量遞增操作。

28.

用單引號代替雙引號來包含字符串，這樣做會更快一些。因為 PHP會在雙引號包圍的字符串中搜尋變量，單引號則不會。

當然，只有當你不需要在字符串中包含變量時才可以這麼做。

錯誤

:

無數的基準測試已經證明，使用雙引號表示純字符串並不比使用單引號時慢。

29.輸出多個字符串時，用逗號代替句點來分隔字符串，速度更快。

注意：只有 echo能這麼做，它是一種可以把多個字符串當作參數的“函數”（譯註： PHP手冊中說 echo是語言結構，不是真正的函數，故把函數加上了雙引號）。

30.Apache解析一個 PHP腳本的時間要比解析一個靜態 HTML頁面慢 2至 10倍。 盡量多用靜態 HTML頁面，少用腳本。

31.除非腳本可以緩存，否則每次調用時都會重新編譯一次。 引入一套 PHP緩存機制通常可以提升 25%至 100%的性能，以免除編譯開銷。

32.盡量做緩存，可使用 memcached。 memcached是一款高性能的內存對象緩存系統，可用來加速動態 Web應用程序，減輕數據庫負載。

對運算碼 (OP code)的緩存很有用，使得腳本不必為每個請求做重新編譯。

33.當操作字符串並需要檢驗其長度是否滿足某種要求時，你想當然地會使用 strlen()函數。

此函數執行起來相當快，因為它不做任何計算，只返回在 zval結構（ C的內置數據結構，用於存儲 PHP變量）中存儲的已知字符串長度。

但是，由於 strlen()是函數，多多少少會有些慢，因為函數調用會經過諸多步驟，如字母小寫化（譯註：指函數名小寫化， PHP不區分函數名大小寫）、 哈希查找，會跟隨被調用的函數一起執行。 在某些情況下，你可以使用 isset()技巧加速執行你的代碼。

（舉例如下）

if (strlen($foo) < 5) { echo “Foo is too short”$$ }

（與下面的技巧做比較）

if (!isset($foo{5})) { echo “Foo is too short”$$ }

調用 isset()恰巧比 strlen()快，因為與後者不同的是， isset()作為一種語言結構，意味著它的執行不需要函數查找和字母小寫化。

也就是說，實際上在檢驗字符串長度的頂層代碼中你沒有花太多開銷。

34.當執行變量 $i的遞增或遞減時， $i++會比 ++$i慢一些。

這種差異是 PHP特有的，並不適用於其他語言，所以請不要修改你的 C或 Java代碼並指望它們能立即變快，沒用的。

++$i更快是因為它只需要 3條指令 (opcodes)，$i++則需要 4條指令。 後置遞增實際上會產生一個臨時變量，這個臨時變量隨後被遞增。

而前置遞增直接在原值上遞增。 這是最優化處理的一種，正如 Zend的 PHP優化器所作的那樣。

牢記這個優化處理不失為一個好主意，因為並不是所有的指令優化器都會做同樣的優化處理，並且存在大量沒有裝配指令優化器的互聯網服務提供商（ ISPs）和服務器。

35.並不是事必面向對象 (OOP)，面向對象往往開銷很大，每個方法和對象調用都會消耗很多內存。

36.並非要用類實現所有的數據結構，數組也很有用。

37.不要把方法細分得過多，仔細想想你真正打算重用的是哪些代碼?

38.當你需要時，你總能把代碼分解成方法。

39.盡量採用大量的 PHP內置函數。

40.如果在代碼中存在大量耗時的函數，你可以考慮用 C擴展的方式實現它們。

41.評估檢驗 (profile)你的代碼。 檢驗器會告訴你，代碼的哪些部分消耗了多少時間。

Xdebug調試器包含了檢驗程序，評估檢驗總體上可以顯示出代碼的瓶頸。

42.mod_zip可作為 Apache模塊，用來即時壓縮你的數據，並可讓數據傳輸量降低 80%。

43.

在可以用 file_get_contents替代 file、 fopen、 feof、 fgets等系列方法的情況下，盡量用 file_get_contents，因為他的效率高得多!

但是要注意 file_get_contents在打開一個 URL文件時候的 PHP版本問題;

44.對 global變量，應該用完就 unset()掉;