

**MỘT SỐ THỦ THUẬT**

**TĂNG TỐC**

**ỨNG DỤNG PHP**

**V1.0**

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Change Description** | **Author** | **Approved By** |
| 1.0 | 24/09/2013 | Created | Dungvd3 | HoanHK |

**Introduction:**

Đây là tài liệu mô tả một số thủ thuật nhằm tăng tốc độ xử lý trên PHP. Mọi người có thể tham khảo để tối ưu hóa hiệu suất ứng dụng của mình.

Do đây là tài liệu sưu tầm nên không thể tránh khỏi thiếu sót. Nếu có điểm nào chưa đúng hay cần bổ sung, mọi người vui lòng phản hồi để giúp cho tài liệu ngày càng hoàn thiện hơn.

- Nếu có thể khai báo hàm tĩnh thì có thể tăng tốc độ lên 4 lần.

- Echo thì nhanh hơn Print.

- Sử dụng nhiều câu lệnh Echo thay vì Echo 1 chuỗi nối vào nhau.

- Đặt giá trị lớn nhất cho vòng lặp for trước, và ko đặt trong vòng lặp for.

- Hủy các biến bằng hàm unset() để giải phóng bộ nhớ sau khi sử dụng, đặc biệt là những mảng lớn.

- Tránh dùng các định nghĩa kiểu: \_\_get, \_\_set, \_\_autoload.

- Tránh dùng hàm require\_once().

- Sử dụng đường dẫn đầy đủ trong includes và requires.

- Nếu cần tính thời gian lúc code được thực thi, thì nên dùng $\_SERVER['REQUEST\_TIME'] thay vì time().

- Chú ý sử dụng các hàm strncasecmp, strpbrk and stripos thay vì regex.

- Str\_replace thì nhanh hơn preg\_replace nhưng mà strtr thì nhanh hơn str\_replace 4 lần.

- Tốt hơn nên dùng câu lệnh Select hoặc Switch thay vì nhiều câu lệnh if, else.

- Các biến lỗi sử dụng @ thì rất chậm.

- Bật Apache mod\_deflate.

- Đóng kết nối DB sau khi dùng xong.

- $row[‘id'] thì nhanh gấp 7 lần $row[id].

- Hạn chế gửi Error messeage.

- Không khai báo biến khi dùng các vòng lặp. Ví dụ: for($x=0; $x

- Sử dụng biến cục bộ trong hàm thì nhanh hơn.

- Sử dụng biến toàn cục thì chậm hơn 2 lần so với dùng biến cục bộ.

- Sử dụng biến cục bộ ko được định nghĩa thì chậm hơn 9-10 lần biến được định nghĩa trước.

- Định nghĩa biến toàn cục mà không sử dụng trong hàm thì cũng chậm hơn.

- Hàm được triệu gọi thì độc lập với các hàm ở trong lớp định nghĩa nó. Ví dụ sau khi sử dụng gọi hàm đó, ta định nghĩa thêm vào vào lớp chứa hàm đó, thì tốc độ performance không thay đổi.

- Hàm được định nghĩa từ lớp thì chạy nhanh hơn hàm định nghĩa từ lớp cơ sở.

- Hàm có 1 tham số hoặc hàm có body rỗng thì tương đương với thực thi 7-8 lần phép toán $localvar++. Một hàm bình thường thì tương đương 15 lần phép toán $localvar++.

- Nếu khai báo chuỗi mà không có biến ở trong thì sử dụng ‘’ sẽ nhanh hơn dùng “”.

- Khi dùng Echo thì dùng phân cách bằng dấu , sẽ nhanh hơn bằng dấu .

- Đoạn mã PHP thực thi thì sẽ chậm hơn 2-10 lần so với trang HTML tĩnh khi chạy Apache.

- Mã PHP được biên dịch lại mỗi lần, trừ khi các đoạn mã được cache. Cài đặt các sản phẩm cache PHP cơ bản để tăng hiệu suất từ 25-100% qua việc giảm số lần biên dịch lại.

- Cache với tất cả khả năng có thể. Memcache là hệ thống cache đối tượng với hiệu suất cao để tăng tốc các ứng dụng web động bằng việc giảm số lần truy suất DB. Opcode cache thì rất hữu ích, giúp các đoạn mã không phải biên dịch lại sau mỗi lần request.

- Dùng trick để tính độ dài của chuỗi bằng hàm isset thay vì sử dụng hàm strlen().

Ví dụ:

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

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

Dùng hàm isset() sẽ nhanh hơn dùng strlen() vì isset() sẽ không tính toán trường hợp lowercase.

- Viết ++$i thay cho $i++ sẽ nhanh hơn (3 opcodes so với 4).

- Không phải mọi thứ đều dùng hướng đối tượng, sẽ tốn bộ nhớ.

- Không phải cài đặt mọi dữ liệu bằng lớp, có thể dùng mảng cũng hữu ích.

- Không nên chia ra quá nhiều hàm, tận dụng sử dụng lại code.

- Sử dụng các hàm được định nghĩa trước.

- Profile code sẽ giúp ta có cái nhìn tổng quan nếu có hiện tượng thắt cổ chai.

- mod\_gzip có sẵn trong module Apache sẽ nén dữ liệu, và giảm dữ liệu truyền đi tới 80%.

- Sử dụng foreach để duyệt các mảng, các tập. Các phần tử PHP4 là theo giá trị, hay hơn nhiều so với theo tham chiếu ở PHP5.

- Sử dụng mẫu thiết kế Singleton với các lớp PHP phức tạp.

- Sử dụng POST thay cho GET với tất cả các giá trị vì lý do performance của TCP/IP.

- Sử dụng các hàm kiểm tra kiểu dữ liệu thay vì dùng Biểu thức chính quy (Regular Expression).

- Dùng include/require thay cho include once/require once để đảm bảo opcode cache.

- Dùng proxy để truy cập các Web Services (XML hay JSON) từ các tên miền nước ngoài dùng XMLHTTP để tránh lỗi cross-domain.

- Dùng thông báo lỗi (E\_ALL) trong quá trình debug.

- Đặt allowoverride của Apache là “none ” để cải thiện performance khi truy cập file/ thư mục.

- Dùng một fileserver nhanh để chạy nội dung tĩnh (thttpd). Ví dụ: static.domain.com.

- Hạn chế dùng các câu SQL: SELECT \*.

- Dùng các phép logic database: queries, joins, views, procedures.

- Khởi tạo các biến trước khi sử dụng.

- Dùng đường dẫn đầy đủ thay vì đường dẫn tương đối.

- Tăng 1 thuộc tính của đối tượng thì chậm hơn việc dùng biến cục bộ 3 lần. ($this->prop++).

- Khi xử lý dữ liệu XML, sử dụng Regular Expression thì nhanh hơn dùng DOM hoặc SAX.

- Dùng tham chiếu làm tham số cho hàm thì nhanh hơn dùng tham số bình thường.

- Dùng INSERT DELAYED nếu như ko cần biết thời gian đưa dữ liệu vào.

- Dùng multiple-row INSERT để INSERT nhiều dữ liệu bằng 1 câu Query.