# guisu，程序人生。 逆水行舟，不进则退。

能干的人解决问题。智慧的人绕开问题(A clever person solves a problem. A wise person avoids it)



【CSDN会员专属福利】OpenStack Days China 大会门票，先到先得 【收藏】Html5 精品资源汇集 我们为什么选择Java

个人资料

真实的归宿

+ 加关注 发私信

访问： 3882507次
积分： 25634
等级： BLOG 7
排名： 第150名

原创： 212篇 转载： 2篇
译文： 0篇 评论： 1200条



# 原 深入理解php底层：php生命周期

标签： php apache 脚本 扩展 web服务 function

2012-03-21 13:21 13516人阅读 评论(4) 收藏 举报

分类： PHP（40） PHP内核（10）

版权声明：本文为博主原创文章，未经博主允许不得转载。



## 1、PHP的运行模式：

PHP两种运行模式是WEB模式、CLI模式。无论哪种模式，PHP工作原理都是一样的，作为一种SAPI运行。

1、当我们在终端敲入php这个命令的时候，它使用的是CLI。

它就像一个web服务器一样来支持php完成这个请求，请求完成后再重新把控制权交给终端。

2、当使用Apache或者别web服务器作为宿主时，当一个请求到来时，PHP会来支持完成这个请求。一般有：

多进程(通常编译为apache的模块来处理PHP请求)

多线程模式

## 2、一切的开始: SAPI接口

通常我们编写php Web程序都是通过Apache或者Nginx这类Web服务器来测试脚本. 或者在命令行下通过php程序来执行PHP脚本. 执行完成脚本后，服务器应答，浏览器显示应答信息,或者在命令结束后在标准输出显示内容. 我们很少关心PHP解释器在哪里. 虽然通过Web服务器和命令行程序执行脚本看起来很不一样. 实际上她们的工作是一样的. 命令行程序和Web程序类似, 命令行参数传递给要执行的脚本,相当于通过url 请求一个PHP页面. 脚本戳里完成后返回响应结果,只不过命令行响应的结果是显示在终端上. 脚本执行的开始都是通过SAPI接口进行的.

**1)、启动apache**：当给定的SAPI启动时，例如在对/usr/local/apache/bin/apachectl start的响应中，PHP由初始化其内核子系统开始。在接近启动例程的末尾，它加载每个扩展的代码并调用其模块初始化例程（MINIT）。这使得每个扩展可以初始化内部变量、分配资源、注册资源处理器，以及向ZE注册自己的函数，以便于脚本调用这其中的函数时候ZE知道执行哪些代码。

**2)、请求处理初始化**：接下来，PHP等待SAPI层请求要处理的页面。对于CGI或CLI等SAPI，这将立刻发生且只发生一次。对于Apache、IIS或其他成熟的web服务器SAPI，每次远程用户请求页面时都将发生，因此重复很多次，也可能并发。不管请求如何产生，PHP开始于要求ZE建立脚本的运行环境，然后调用每个扩展的请求初始化（RINIT）函数。RINIT使得扩展有机会设定特定的环境变量，根据请求分配资源，或者执行其他任务，如审核。session扩展中有个RINIT作用的典型示例，如果启用了session.auto_start选项，RINIT将自动触发用户空间的session_start()函数以及预组装$_SESSION变量。

**3)、执行php代码**： 一旦请求被初始化了，ZE开始接管控制权，将PHP脚本翻译成符号，最终形成操作码并逐步运行之。如任一操作码需要调用扩展的函数，ZE将会把参数绑定到该函数，并且临时交出控制权直到函数运行结

NOSQL Mongo (0)
分布式 (1)
数据结构与算法 xi (0)
协议 (1)
信息论的熵 (0)
关于php的libevent扩展的应用 (0)
libevent简单介绍 (0)
SOA (0)

文章存档

2015年12月 (2)
2015年10月 (4)
2015年05月 (2)
2015年04月 (1)
2015年01月 (2)
展开

阅读排行

八大排序算法 (343034)
Mysql 多表联合查询效率 (155080)
深入理解java异常处理机 (132473)
socket阻塞与非阻塞，同 (115049)
Linux的SOCKET编程详解 (103864)
设计模式 ( 十八 ) 策略模 (102393)
hbase安装配置（整合到 (96003)
Hadoop Hive sql语法详解 (93761)
高性能Mysql主从架构的 (86443)
Nginx工作原理和优化、 (83399)

评论排行

八大排序算法 (116)
深入理解java异常处理机 (100)
socket阻塞与非阻塞，同 (55)
硬盘的读写原理 (43)
设计模式 ( 十八 ) 策略模 (41)
设计模式（一）工厂模式 (34)
Redis应用场景 (28)
海量数据处理算法—Bit-M (26)
设计模式 ( 十七) 状态模式 (24)
PHP SOCKET编程 (23)

推荐文章

*Android RocooFix 热修复框架
*笑谈Android图表------MPAndroidChart
*Nginx正反向代理、负载均衡等功能实现配置
* 浅析ZeroMQ工作原理及其特点
*Android开源框架Universal-Image-Loader基本介绍及使用
*Spring Boot 实践折腾记（三）：三板斧，Spring Boot下使用Mybatis

束。

**4)、脚本结束**：脚本运行结束后，PHP调用每个扩展的请求关闭（RSHUTDOWN）函数以执行最后的清理工作（如将session变量存入磁盘）。接下来，ZE执行清理过程（垃圾收集）– 有效地对之前的请求期间用到的每个变量执行unset()。

**5)、sapi关闭**：一旦完成，PHP继续等待SAPI的其他文档请求或者是关闭信号。**对于CGI和CLI等SAPI，没有“下一个请求”，所以SAPI立刻开始关闭**。关闭期间，PHP再次遍历每个扩展，调用其模块关闭（MSHUTDOWN）函数，并最终关闭自己的内核子系统。

简要的过程如下：

1. PHP是随着Apache的启动而运行的；
2. PHP通过mod_php5.so模块和Apache相连（具体说来是SAPI，即服务器应用程序编程接口）；
3. PHP总共有三个模块：内核、Zend引擎、以及扩展层；
4. PHP内核用来处理请求、文件流、错误处理等相关操作；
5. Zend引擎（ZE）用以将源文件转换成机器语言，然后在虚拟机上运行它；
6. 扩展层是一组函数、类库和流，PHP使用它们来执行一些特定的操作。比如，我们需要mysql扩展来连接MySQL数据库；
7. 当ZE执行程序时可能会需要连接若干扩展，这时ZE将控制权交给扩展，等处理完特定任务后再返还；
8. 最后，ZE将程序运行结果返回给PHP内核，它再将结果传送给SAPI层，最终输出到浏览器上。

## 3、PHP的开始和结束阶段

**开始阶段有两个过程：**

**第一个过程：apache启动的过程，即在任何请求到达之前就发生**。是在整个SAPI生命周期内(例如Apache启动以后的整个生命周期内或者命令行程序整个执行过程中)的开始阶段(MINIT),该阶段只进行一次.。启动Apache后，PHP解释程序也随之启动； PHP调用各个扩展（模块）的MINIT方法，从而使这些扩展切换到可用状态。看看php.ini文件里打开了哪些扩展吧； MINIT的意思是“模块初始化”。各个模块都定义了一组函数、类库等用以处理其他请求。 模块在这个阶段可以进行一些初始化工作,例如注册常量,定义模块使用的类等等.典型的的模块回调函数MINIT方法如下：

[cpp] view plain copy print ?

```
PHP_MINIT_FUNCTION(myphpextension) { /* Initialize functions, classes etc */ }
{
    // 注册常量或者类等初始化操作
    return SUCCESS;
}
```

[cpp] view plain copy print ?

```
PHP_MINIT_FUNCTION(myphpextension) { /* Initialize functions, classes etc */ }
{
    // 注册常量或者类等初始化操作
    return SUCCESS;
}
```

**第二个过程发生在请求阶段**,当一个页面请求发生时.则在每次请求之前都会进行初始化过程(RINIT请求开始).

请求到达之后，SAPI层将控制权交给PHP层，PHP初始化本次请求执行脚本所需的环境变量,例如创建一个执行环境,包括保存php运行过程中变量名称和变量值内容的符号表. 以及当前所有的函数以及类等信息的符号表.例如是Session模块的RINIT，如果在php.ini中启用了Session 模块，那在调用该模块的RINIT时就会初始化$_SESSION变量，并将相关内容读入； 然后PHP会调用所有模块RINIT函数,即“请求初始化”。 在这个阶段各个模块也可以执行一些相关的操作, 模块的RINIT函数和MINIT函数类似 ， RINIT方法可以看作是一个准备过程，在程序执行之间就会自动启动。

[cpp] view plain copy print ?

```
PHP_RINIT_FUNCTION(myphpextension)
{
    // 例如记录请求开始时间
    // 随后在请求结束的时候记录结束时间.这样我们就能够记录下处理请求所花费的时间了
    return SUCCESS;
}
```

[cpp] view plain copy print ?

```
PHP_RINIT_FUNCTION(myphpextension)
{
    // 例如记录请求开始时间
    // 随后在请求结束的时候记录结束时间.这样我们就能够记录下处理请求所花费的时间了
    return SUCCESS;
```

最新评论

程序的装入和链接
风雨同舟-: 谢谢博主！

架构师知识体系(5)--建立自己的
black_OX: 哥，这文章我转载了啊！这篇文章太棒了！我关注博主已经很长时间了，我的博客也加了你的链接.

深入理解java异常处理机制
u010535582: 你开通的例子在finally里return有点骚啊，兄弟。容易误导人的

数据结构-线性表
magic__hu: @hguisu:博主，好像C定义函数的时候不带用&...

高性能Mysql主从架构的复制原理
奋进中的巨人: echo "very good!";

八大排序算法
qq_26014757: @Jack_Stan:我想了半天，想到一个更好的方法，不用分那么多情况讨论。代码是这样的：packa...

八大排序算法
qq_26014757: @Jack_Stan:我想了半天，想到一个更好的方法，不用分那么多情况讨论。代码是这样的：packa...

mysql or条件可以使用索引而避免
从别后: or两边的条件都有索引的话，是可以用的上索引的，type是index_merge

PHP7新特性 What will be in PHP
Frank_Monkey_Lee: 大大，不好意思，我把您的这篇博客转载到自己新开的博客里了，已注明出处，还望海涵。

八大排序算法
qq_26014757: @Jack_Stan:厉害，知道错，就是找不到原因，看了你的终于知道了。不过二元插入这么写感觉代码比...

友情链接

图灵机器人：聊天api的最佳选择

```
}
```

**结束阶段分为两个环节：**

请求处理完后就进入了结束阶段, 一般脚本执行到末尾或者通过调用exit()或者die()函数,PHP都将进入结束阶段. 和开始阶段对应,结束阶段也分为两个环节,一个在请求结束后(RSHUWDOWN),一个在SAPI生命周期结束时(MSHUTDOWN).

**第一个环节：请求处理完后结束阶段**：请求处理完后就进入了结束阶段，PHP就会启动清理程序。它会按顺序调用各个模块的RSHUTDOWN方法。 RSHUTDOWN用以清除程序运行时产生的符号表，也就是对每个变量调用unset函数。典型的RSHUTDOWN方法如下：

```
[cpp] view plain  copy  print  ?
PHP_RSHUTDOWN_FUNCTION(myphpextension)
{
    // 例如记录请求结束时间，并把相应的信息写入到日至文件中.
    return SUCCESS;
}
```

```
[cpp] view plain  copy  print  ?
PHP_RSHUTDOWN_FUNCTION(myphpextension)
{
    // 例如记录请求结束时间，并把相应的信息写入到日至文件中.
    return SUCCESS;
}
```

第二个环节：最后，所有的请求都已处理完毕，SAPI也准备关闭了， PHP调用每个扩展的MSHUTDOWN方法，这是各个模块最后一次释放内存的机会。（**这个是对于CGI和CLI等SAPI，没有“下一个请求”，所以SAPI立刻开始关闭。**）

典型的RSHUTDOWN方法如下：

```
[plain] view plain  copy  print  ?
PHP_MSHUTDOWN_FUNCTION(extension_name) {
    /* Free handlers and persistent memory etc */
    return SUCCESS;
}
```

```
[plain] view plain  copy  print  ?
PHP_MSHUTDOWN_FUNCTION(extension_name) {
    /* Free handlers and persistent memory etc */
    return SUCCESS;
}
```

这样，整个PHP生命周期就结束了。要注意的是，只有在服务器没有请求的情况下才会执行“启动第一步”和“关闭第二步”。

SAPI运行PHP都经过下面几个阶段:

1、**模块初始化阶段(Module init)** ：

即调用每个拓展源码中的的PHP_MINIT_FUNCTION中的方法初始化模块,进行一些模块所需变量的申请,内存分配等。

2、**请求初始化阶段(Request init)** ：

即接受到客户端的请求后调用每个拓展的PHP_RINIT_FUNCTION中的方法,初始化PHP脚本的执行环境。

3、**执行PHP脚本**

4、**请求结束(Request Shutdown)**：

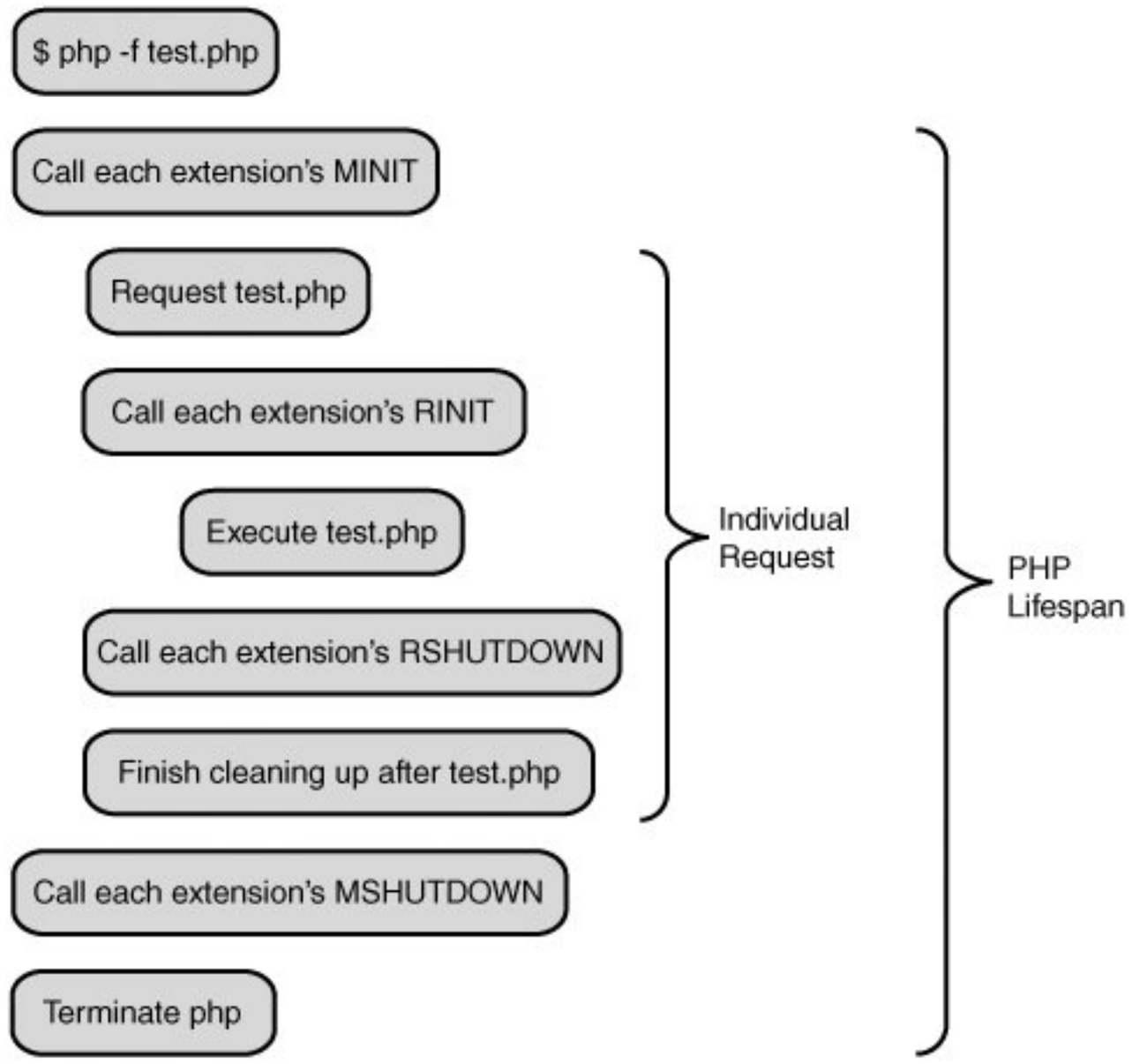
这时候调用每个拓展的PHP_RSHUTDOWN_FUNCTION方法清理请求现场,并且ZE开始回收变量和内存。

5、**关闭模块(Module shutdown)** ：

Web服务器退出或者命令行脚本执行完毕退出会调用拓展源码中的PHP_MSHUTDOWN_FUNCTION 方法

## 4、单进程SAPI生命周期

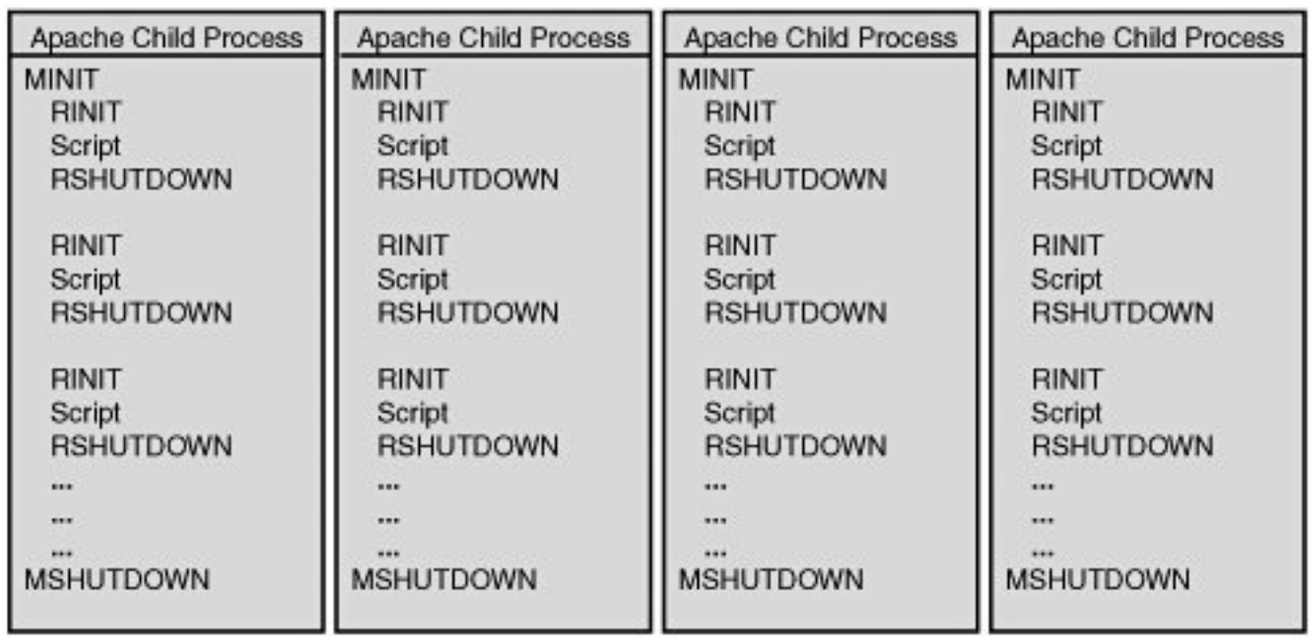
CLI/CGI模式的PHP属于单进程的SAPI模式。这类的请求在处理一次请求后就关闭。也就是只会经过如下几个环节：开始 - 请求开始 - 请求关闭 - 结束 SAPI接口实现就完成了其生命周期。如图所示：




## 5、多进程SAPI生命周期

通常PHP是编译为apache的一个模块来处理PHP请求。Apache一般会采用多进程模式， Apache启动后会fork出多个子进程，每个进程的内存空间独立，每个子进程都会经过开始和结束环节， 不过每个进程的开始阶段只在进程fork出来以来后进行，在整个进程的生命周期内可能会处理多个请求。 只有在Apache关闭或者进程被结束之后才会进行关闭阶段，在这两个阶段之间会随着每个请求重复请求开始-请求关闭的环节。

如图所示：

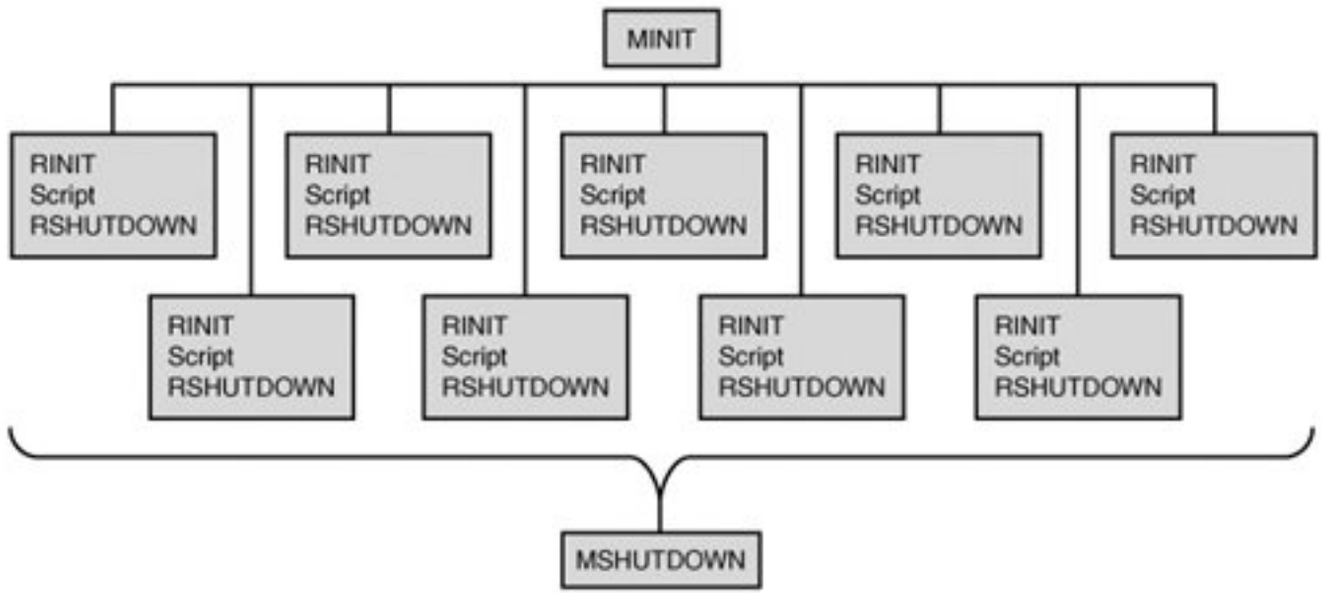



## 6、多线程的SAPI生命周期

多线程模式和多进程中的某个进程类似，不同的是在整个进程的生命周期内会并行的重复着 请求开始-请求关闭的环节.

在这种模式下，只有一个服务器进程在运行着，但会同时运行很多线程，这样可以减少一些资源开销，向Module init和Module shutdown就只需要运行一遍就行了，一些全局变量也只需要初始化一次，因为线程独具的特质，使得各个请求之间方便的共享一些数据成为可能。

多线程工作方式如下图




## 7、Apache一般使用多进程模式prefork

在linux下使用#http –l 命令可以查看当前使用的工作模式。也可以使用#apachectl -l命令。

看到的prefork.c，说明使用的prefork工作模式。

prefork 进程池模型，用在 UNIX 和类似的系统上比较多，主要是由于写起来方便，也容易移植，还不容易出问题。要知道，如果采用线程模型的话，用户线程、内核线程和混合型线程有不同的特性，移植起来就麻烦。prefork 模型，即预先 fork() 出来一些子进程缓冲一下，用一个锁来控制同步，连接到来了就放行一个子进程，让它去处理。

prefork MPM 使用多个子进程，每个子进程只有一个线程。每个进程在某个确定的时间只能维持一个连接。在大多数平台上，Prefork MPM在效率上要比Worker MPM要高，但是内存使用大得多。prefork的无线程设计在某些情况下将比worker更有优势：他能够使用那些没有处理好线程安全的第三方模块，并 且对于那些线程调试困难的平台而言，他也更容易调试一些。

顶 11
踩 0

上一篇 深入理解php内核 编写扩展_III- 资源
下一篇 使用ext_skel和phpize构建php5扩展

我的同类文章

PHP（40） PHP内核（10）

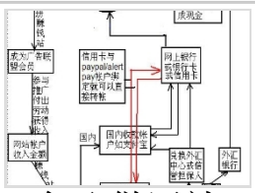
- PHP7新特性 What will be i… 2015-04-17 阅读 20409
- PHP编程注意事项 2013-02-22 阅读 3339
- PHP5.4最新特性 2012-09-05 阅读 4158
- gzip压缩输出 2012-07-27 阅读 3963
- PHP5.4的变化关注---What … 2012-04-24 阅读 4453
- PHP安全模式详解(PHP5.4… 2012-04-16 阅读 14233
- TIME_WAIT引起Cannot as… 2013-08-25 阅读 8878
- 正则表达式详解 2012-11-30 阅读 4969
- PHP字符串的编码问题 2012-08-07 阅读 3740
- php的serialize序列化和json… 2012-06-11 阅读 12468
- Windows PHP 中 VC6 X86 … 2012-04-24 阅读 7258

更多文章

猜你在找

- 最牛JavaScript课程

- Android开发精品课程【Java核心知识】
- 【中国ＰＨＰ教育大牛高洛峰】亲授ｐ ｈ ｐ教程
- Web前端从零基础到高手之路
- 韦东山嵌入式Linux第一期视频
- 最牛JavaScript课程
- Qt网络编程实战之HTTP服务器
- 【中国ＰＨＰ教育大牛高洛峰】亲授ｐ ｈ ｐ教程
- Web前端从零基础到高手之路
- HTML 5移动开发从入门到精通


云服务器免费

怎么做网站

视频直播技术

快速开发平台

app外包

电脑发短信平

游戏原画教程


查看评论

4楼 idong杨 2015-07-28 15:25发表


菜鸟虽然不是很懂 但也得踩一脚...

3楼 huashenghn 2014-05-30 15:43发表

很不错的文章，转载了

2楼 一把老刀 2014-02-20 06:47发表

现在想在买个php-fpm初始化的时候进行shared memory连接工作，一直找不到放在哪里？现在技术上处理是当此进程的session第一次运行的时候初始化。但是这样对于打用户量访问就浪费了些判断语句的时间，而且由于初始化shared memory在1g左右还需要3秒左右，因此第一个访问的人会有些迟钝。

1楼 binbinwan51a 2014-01-16 11:54发表

不错。讲得很精炼，通俗易懂。

发表评论

用 户 名: sinat_20684939

评论内容:

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC
coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
Angular Cloud Foundry Redis Scala Django Bootstrap




京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved