[php-fpm 启动参数及重要配置详解](http://www.cnblogs.com/argb/p/3604340.html)

## 约定几个目录

* /usr/local/php/sbin/php-fpm
* /usr/local/php/etc/php-fpm.conf
* /usr/local/php/etc/php.ini

## 一，php-fpm的启动参数

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #测试php-fpm配置  /usr/local/php/sbin/php-fpm -t  /usr/local/php/sbin/php-fpm -c /usr/local/php/etc/php.ini -y /usr/local/php/etc/php-fpm.conf -t    #启动php-fpm  /usr/local/php/sbin/php-fpm  /usr/local/php/sbin/php-fpm -c /usr/local/php/etc/php.ini -y /usr/local/php/etc/php-fpm.conf    #关闭php-fpm  kill -INT `cat /usr/local/php/var/run/php-fpm.pid`    #重启php-fpm  kill -USR2 `cat /usr/local/php/var/run/php-fpm.pid` |

## 二，php-fpm.conf重要参数详解

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79 | pid = run/php-fpm.pid  #pid设置，默认在安装目录中的var/run/php-fpm.pid，建议开启    error\_log = log/php-fpm.log  #错误日志，默认在安装目录中的var/log/php-fpm.log    log\_level = notice  #错误级别. 可用级别为: alert（必须立即处理）, error（错误情况）, warning（警告情况）, notice（一般重要信息）, debug（调试信息）. 默认: notice.    emergency\_restart\_threshold = 60  emergency\_restart\_interval = 60s  #表示在emergency\_restart\_interval所设值内出现SIGSEGV或者SIGBUS错误的php-cgi进程数如果超过 emergency\_restart\_threshold个，php-fpm就会优雅重启。这两个选项一般保持默认值。    process\_control\_timeout = 0  #设置子进程接受主进程复用信号的超时时间. 可用单位: s(秒), m(分), h(小时), 或者 d(天) 默认单位: s(秒). 默认值: 0.    daemonize = yes  #后台执行fpm,默认值为yes，如果为了调试可以改为no。在FPM中，可以使用不同的设置来运行多个进程池。 这些设置可以针对每个进程池单独设置。    listen = 127.0.0.1:9000  #fpm监听端口，即nginx中php处理的地址，一般默认值即可。可用格式为: 'ip:port', 'port', '/path/to/unix/socket'. 每个进程池都需要设置.    listen.backlog = -1  #backlog数，-1表示无限制，由操作系统决定，此行注释掉就行。backlog含义参考：http://www.3gyou.cc/?p=41    listen.allowed\_clients = 127.0.0.1  #允许访问FastCGI进程的IP，设置any为不限制IP，如果要设置其他主机的nginx也能访问这台FPM进程，listen处要设置成本地可被访问的IP。默认值是any。每个地址是用逗号分隔. 如果没有设置或者为空，则允许任何服务器请求连接    listen.owner = www  listen.group = www  listen.mode = 0666  #unix socket设置选项，如果使用tcp方式访问，这里注释即可。    user = www  group = www  #启动进程的帐户和组    pm = dynamic #对于专用服务器，pm可以设置为static。  #如何控制子进程，选项有static和dynamic。如果选择static，则由pm.max\_children指定固定的子进程数。如果选择dynamic，则由下开参数决定：  pm.max\_children #，子进程最大数  pm.start\_servers #，启动时的进程数  pm.min\_spare\_servers #，保证空闲进程数最小值，如果空闲进程小于此值，则创建新的子进程  pm.max\_spare\_servers #，保证空闲进程数最大值，如果空闲进程大于此值，此进行清理    pm.max\_requests = 1000  #设置每个子进程重生之前服务的请求数. 对于可能存在内存泄漏的第三方模块来说是非常有用的. 如果设置为 '0' 则一直接受请求. 等同于 PHP\_FCGI\_MAX\_REQUESTS 环境变量. 默认值: 0.    pm.status\_path = /status  #FPM状态页面的网址. 如果没有设置, 则无法访问状态页面. 默认值: none. munin监控会使用到    ping.path = /ping  #FPM监控页面的ping网址. 如果没有设置, 则无法访问ping页面. 该页面用于外部检测FPM是否存活并且可以响应请求. 请注意必须以斜线开头 (/)。    ping.response = pong  #用于定义ping请求的返回相应. 返回为 HTTP 200 的 text/plain 格式文本. 默认值: pong.    request\_terminate\_timeout = 0  #设置单个请求的超时中止时间. 该选项可能会对php.ini设置中的'max\_execution\_time'因为某些特殊原因没有中止运行的脚本有用. 设置为 '0' 表示 'Off'.当经常出现502错误时可以尝试更改此选项。    request\_slowlog\_timeout = 10s  #当一个请求该设置的超时时间后，就会将对应的PHP调用堆栈信息完整写入到慢日志中. 设置为 '0' 表示 'Off'    slowlog = log/$pool.log.slow  #慢请求的记录日志,配合request\_slowlog\_timeout使用    rlimit\_files = 1024  #设置文件打开描述符的rlimit限制. 默认值: 系统定义值默认可打开句柄是1024，可使用 ulimit -n查看，ulimit -n 2048修改。    rlimit\_core = 0  #设置核心rlimit最大限制值. 可用值: 'unlimited' 、0或者正整数. 默认值: 系统定义值.    chroot =  #启动时的Chroot目录. 所定义的目录需要是绝对路径. 如果没有设置, 则chroot不被使用.    chdir =  #设置启动目录，启动时会自动Chdir到该目录. 所定义的目录需要是绝对路径. 默认值: 当前目录，或者/目录（chroot时）    catch\_workers\_output = yes  #重定向运行过程中的stdout和stderr到主要的错误日志文件中. 如果没有设置, stdout 和 stderr 将会根据FastCGI的规则被重定向到 /dev/null . 默认值: 空. |

## 三，常见错误及解决办法整理

### 1,request\_terminate\_timeout引起的资源问题

request\_terminate\_timeout的值如果设置为0或者过长的时间，可能会引起file\_get\_contents的资源问题。

如果file\_get\_contents请求的远程资源如果反应过慢，file\_get\_contents就会一直卡在那里不会超时。我们知道php.ini 里面max\_execution\_time 可以设置 PHP 脚本的最大执行时间，但是，在 php-cgi(php-fpm) 中，该参数不会起效。真正能够控制 PHP 脚本最大执行时间的是 php-fpm.conf 配置文件中的request\_terminate\_timeout参数。

request\_terminate\_timeout默认值为 0 秒，也就是说，PHP 脚本会一直执行下去。这样，当所有的 php-cgi 进程都卡在 file\_get\_contents() 函数时，这台 Nginx+PHP 的 WebServer 已经无法再处理新的 PHP 请求了，Nginx 将给用户返回“502 Bad Gateway”。修改该参数，设置一个 PHP 脚本最大执行时间是必要的，但是，治标不治本。例如改成 30s，如果发生 file\_get\_contents() 获取网页内容较慢的情况，这就意味着 150 个 php-cgi 进程，每秒钟只能处理 5 个请求，WebServer 同样很难避免”502 Bad Gateway”。**解决办法是request\_terminate\_timeout设置为10s或者一个合理的值，或者给file\_get\_contents加一个超时参数。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | $ctx = stream\_context\_create(array(      'http' => array(          'timeout' => 10    //设置一个超时时间，单位为秒      )  ));    file\_get\_contents($str, 0, $ctx); |

### 2,max\_requests参数配置不当，可能会引起间歇性502错误：

|  |  |
| --- | --- |
| 1 | pm.max\_requests = 1000 |

设置每个子进程重生之前服务的请求数. 对于可能存在内存泄漏的第三方模块来说是非常有用的. 如果设置为 ’0′ 则一直接受请求. 等同于 PHP\_FCGI\_MAX\_REQUESTS 环境变量. 默认值: 0.  
这段配置的意思是，当一个 PHP-CGI 进程处理的请求数累积到 500 个后，自动重启该进程。

**但是为什么要重启进程呢？**

一般在项目中，我们多多少少都会用到一些 PHP 的第三方库，这些第三方库经常存在内存泄漏问题，如果不定期重启 PHP-CGI 进程，势必造成内存使用量不断增长。因此 PHP-FPM 作为 PHP-CGI 的管理器，提供了这么一项监控功能，对请求达到指定次数的 PHP-CGI 进程进行重启，保证内存使用量不增长。

正是因为这个机制，在高并发的站点中，经常导致 502 错误，我猜测原因是 PHP-FPM 对从 NGINX 过来的请求队列没处理好。不过我目前用的还是 PHP 5.3.2，不知道在 PHP 5.3.3 中是否还存在这个问题。

目前我们的解决方法是，把这个值尽量设置大些，尽可能减少 PHP-CGI 重新 SPAWN 的次数，同时也能提高总体性能。在我们自己实际的生产环境中发现，内存泄漏并不明显，因此我们将这个值设置得非常大（204800）。大家要根据自己的实际情况设置这个值，不能盲目地加大。

话说回来，这套机制目的只为保证 PHP-CGI 不过分地占用内存，为何不通过检测内存的方式来处理呢？我非常认同高春辉所说的，通过设置进程的峰值内在占用量来重启 PHP-CGI 进程，会是更好的一个解决方案。

### 3,php-fpm的慢日志，debug及异常排查神器：

request\_slowlog\_timeout设置一个超时的参数，slowlog设置慢日志的存放位置

|  |  |
| --- | --- |
| 1 | tail -f /var/log/www.slow.log |

上面的命令即可看到执行过慢的php过程。  
大家可以看到经常出现的网络读取超过、Mysql查询过慢的问题，根据提示信息再排查问题就有很明确的方向了。

**原文链接：**[http://levi.cg.am/?p=3127](http://levi.cg.am/?p=3127" \t "http://www.cnblogs.com/argb/p/_blank)