# PHP-FPM不完全指南

hello　2017 年 01 月 09 日

## fpm工作流程

fpm全名是FastCGI进程管理器（FastCGI是啥？了解下cgi和fastcgi）。

fpm启动后会先读php.ini，然后再读相应的conf配置文件，conf配置可以覆盖php.ini的配置。
启动fpm之后，会创建一个master进程，监听9000端口（可配置），master进程又会根据fpm.conf/www.conf去创建若干子进程，子进程用于处理实际的业务。
当有客户端（比如nginx）来连接9000端口时，空闲子进程会自己去accept，如果子进程全部处于忙碌状态，新进的待accept的连接会被master放进队列里，等待fpm子进程空闲；这个存放待accept的半连接的队列有多长，由`listen.backlog`配置。

## fpm配置说明

配置里面的所有相对路径，都是相对于php的安装路径。

## 进程池

除了有php-fpm.conf配置文件外，通常还有其他的*.conf配置文件（也可以不要，直接在php-fpm.conf配置）用于配置进程池，不同的进程池可以用不同的用户执行，监听不同的端口，处理不同的任务；多个进程池共用一个全局配置。

include=/opt/remi/php56/root/etc/php-fpm.d/*.conf 载入其他的配置文件。

## 全局配置：

1.0 `pid = /opt/remi/php56/root/var/run/php-fpm/php-fpm.pid` pid进程文件，默认为none。

2.0 `error_log = /opt/remi/php56/root/var/log/php-fpm/error.log` 错误日志位置，默认：安装路径 #INSTALL_PREFIX#/log/php-fpm.log。如果设置为`syslog`，log就会发送给syslogd服务而不会写进文件里。

3.0 `syslog.facility = daemon` 把日志写进系统log，linux还不够熟悉，暂时不用理会。

3.1 `syslog.ident = php-fpm` 系统日志标示，如果跑了多个fpm进程，需要用这个来区分日志是谁的。

4.0 `log_level = notice` 日志等级，默认notice，可选：alert, error, warning, notice, debug

5.0 `emergency_restart_threshold = 60` 配合5.1用

5.1 `emergency_restart_interval = 60s` 如果在此参数设置的时间内，出现SIGSEGV或SIGBUS的子进程数超过5.0号参数设置的值，那么fpm就会优雅的重启，值是0表示off这个功能，可用的单位有：s秒,m分,h时,d天。

6.0 `process_control_timeout = 0` 设置子进程接受主进程复用信号的超时时间。这个每天明白，是过了这个时间就不能复用了?

7.0 `process.max = 128` 当动态管理子进程时，fpm最多能fork多少个进程，0表示无限制，这是所有进程池能启动子进程的总和，谨慎使用。

8.0 `process.priority = -19` 设置子进程的优先级，在master进程以root用户启动时有效；如果没有设置，子进程会继承master进程的优先级，值范围-19(最高)到20(最低)，默认不设置。

9.0 `daemonize = yes` 设置成no用于调试bug，默认为yes。

10.0 `rlimit_files = 1024` 设置master进程最多能打开的文件，默认为系统的值。

11.0 `rlimit_core = 0` master进程核心rlimit限制值；可选unlimited或>=0的整数，默认为系统的值。

12.0 `events.mechanism = epoll` 事件处理机制，默认自动检测，可选值：select，poll，epoll(linux>=2.5.44)，kqueue，/dev/poll，port

13.0 `systemd_interval = 10s` 当fpm被设置为系统服务时，多久向服务器报告一次状态，单位有s,m,h。

## 进程池配置pool Definitions：

在不同的监听端口和不同的管理选项下可以跑任意数量的池，并没有个数限

制；池的名字用于logs和stats。
下图定义了一个叫www的池：

1.0 `user = apache，group = apache` 以什么用户什么组的权限来运行池fpm。

> 用apache可以像httpd服务一样去访问某些目录

2.0 `listen = 127.0.0.1:9000`监听的ip和端口，可以`/path/to/unix/socket`来监听unix socket，性能更好。
3.0 `listen.backlog = 65535` 未accept处理的socket队列大小，-1 on FreeBSD and OpenBSD，其他平台默认65535，高并发时重要，合理设置会及时处理排队的请求；太大会积压太多，处理完后nginx在前面都等超时断开这个和fpm的socket连接了，就杯具了。不要用-1，建议1024以上，最好是2的幂值。

- 一个池共用一个backlog队列，所有的池进程都去这个队列里accept连接。
- 最大数量受限于系统配置`cat /proc/sys/net/core/somaxconn`，系统配置修改：`vim /etc/sysctl.conf`，增加`net.core.somaxconn = 2000`则最大为2000，然后php最大的backlog可以到2000。

4.0 用socket连接方式时，指定拥有unix socket权限的用户，默认和运行的用户一样；用tcp连接可以注释掉

```
listen.owner =apache
listen.group =apache
listen.mode = 0660
```

5.0 `listen.allowed_clients = 127.0.0.1` 设置允许连接fpm的地址，比如nginx就要来连，多个地址用逗号隔开，如果不配置，则默认任意地址都能来连。
6.0 `process.priority = -19` 池进程的权限，同样要master进程是root用户才有效，和全局那个一样，不设置的话会继承master进程的优先级。
7.0

pm = dynamic 启动时子进程管理方式，可选值：static(启动时创建指定个数)， dynamic(启动
pm.max_children = 5 该池同时最多存在5个进程， 三种管理方式都要配置
pm.start_servers = 2 fpm启动时创建2个子进程，只适用动态dynamic管理方式
pm.min_spare_servers = 2 服务器闲置时最少保持2个子进程，不够这个数就会创建，只适用动
pm.max_spare_servers = 3 服务器闲置时最多要有几个，多了会kill，只适用动态dynamic管
pm.process_idle_timeout = 10s; 子进程闲置10s后就会被杀掉。

8.0 pm.max_requests = 500 每个子进程最大处理500请求就被回收，可防止内存泄露。
9.0 有几个设置，可以看fpm的status，和nginx的差不多。
10.0 access.log = var/log/$pool.access.log 访问文件日志，没啥用处，比如yii2每次都记录访问index.php，只是记录真实的PHP文件。
11.0 slowlog = var/log/$pool.log.slow PHP文件执行过慢的日志，会准确的记录具体哪一行代码太慢，这个非常有用，在设置了时间时生效。
11.1 request_slowlog_timeout = 2s 超过这个运行时间就会写慢日志
12.0 request_terminate_timeout = 3s 单个请求的超时时间，有时候php.ini设置的最大执行时间未生效，这个就会来干掉那个执行太久的请求。
13.0 rlimit_files = 1024 最大打开句柄数，默认为系统值。
14.0 rlimit_core = 0 最多的核心使用数，默认为系统分配。
15.0 chroot = /path 路径必须是绝对路径，改变子进程的跟目录，可以把进程对文件系统的读写与实际的操作系统文件系统隔离，对安全有好处。
16.0 chdir = /var/www 改变当前工作目录，可以用相对路径，默认是当前目录或者chroot。
17.0 catch_workers_output = yes 重定向标准输出stdout和标准错误stderr到主错误日志，如果不设置，这两个日志就会定向到/dev/null，在高负载情况下，这个配置会引起页面延迟几毫秒，默认不开启。
18.0 clear_env = no 创建work进程时是否清除环境变量，如果是yes，那么该子进程getenv()就访问不到$_ENV和$_SERVER了。
19.0 security.limit_extensions = .php .php3 .php4 .php5 为了安全，限制能执行的脚本后缀
20.0 为当前池指定另外的php.ini配置，比如指定当前池的错误日志写在哪个地方

php_value/php_flag 可以设置php.ini的内容，可以被ini_set覆盖
php_admin_value/php_admin_flag 这个同上，但是不会被ini_set覆盖。

其中flag设置的，值只能是on, off, 1, 0, true, false, yes or no，其他类型的值需要用

```
php_flag[display_errors] = off
php_admin_value[error_log] = /var/log/fpm-php.www.log
php_admin_flag[log_errors] = on
php_admin_value[memory_limit] = 32M
```

这种方法设置`disable_functions`和`disable_classes`时，不会覆盖php.ini的设置，只会

注意：PHP配置值通过 php_value 或者 php_flag 设置，并且会覆盖以前的值。请注意 disable_functions 或者 disable_classes 在 php.ini 之中定义的值不会被覆盖掉，但是会将新的设置附加在原有值的后面。
使用 php_admin_value 或者 php_admin_flag 定义的值，不能被 PHP 代码中的 ini_set() 覆盖。

自 5.3.3 起，也可以通过 web 服务器设置 PHP 的设定。

# nginx通过unixsock与php-fpm通信：

适用场景：nginx和php-fpm在同一台服务器上，这时可以直接用unixsocket进程间通信，不走tcp端口通信，可以节约创建连接的时间，从而提高性能。

1，设置php-fpm的listen为/opt/remi/php56/root/var/run/php-fpm/php567-fpm.sock(可以用相对路径)，然后重启fpm就会自动创建该php567-fpm.sock文件
2，nginx的fastcgi_pass参数修改为 unix:/opt/remi/php56/root/var/run/php-fpm/php567-fpm.sock;
通过php567-fpm.sock文件去和fpm通信
需要保证该php567-fpm.sock文件nginx有权限访问。

总结：sock文件随便创建到哪里都可以，只要fpm有权限在那个目录里写文件，nginx有权限去读就可以。
tcp连接会更稳定，因为有tcp协议保证数据的正确性，但是sock有更少的数据拷贝和上下文切换，更少的资源占用。
不过只能在nginx和fpm在同一台机器上才能用sock。

# fpm进程状态监控

1，nginx配置：遇到status的请求，直接转发给php

```
location ~^/status$ {
      fastcgi_param SCRIPT_FILENAME $fastcgi_script_name;
      include fastcgi_params;
      fastcgi_pass 127.0.0.1:9000;
  }
```

2，fpm配置：pm.status_path = /status

3， 然后重新fpm和nginx，在浏览器里访问就能看到了：

默认以text/plain展示结果，可以传参数?json/html/xml分别得到json等格式的结果；参数full可以查看每个子进程的明细

```
pool  进程池名称
process manager 进程管理方式
start time 进程什么时候启动的
start since 进程已经运行了多少秒
accepted conn  该池总共accept了多少连接
listen queue   等待accept的连接的数量
max listen queue   fpm启动后，历史最高等待accept的连接的数量
listen queue len 配置的监听队列最大长度 受限于`listen.backlog`和系统`cat /proc/s
idle processes 闲置的进程数
active process  正在工作的进程数（加上限制的，就是总的子进程数）
total processes 总的子进程数量
max active processes  fpm启动后，历史最多同时工作的进程数
max children reached  进程管理模式为 'dynamic'和 'ondemand'时，此数值是当子进程不
slow requests  慢请求个数

full参数下
pid                    子进程ID;
state                  子进程状态(Idle, Running, ...);
start time             子进程启动的时间;
start since            子进程启动后运行了多少秒;
requests               当前子进程一共处理了多少个请求;
request duration     请求耗费的纳秒数;
request method       请求方法 (GET, POST, ...);
request URI            请求参数;
content length         POST请求时，请求的内容长度;
user                   - the user (PHP_AUTH_USER) (or '-' if not set);
script                 请求的哪个php文件;
last request cpu  上次请求耗费的cpu资源
last request memory 上次请求耗费的内存峰值
如果进程是闲置状态，那这些信息记录的就是上次请求的相关数据，否则就是当前本次请求的相关数
```

# backlog配置问题

一个fpm子进程在同一时间只能处理一个请求，如果，
backlog设置得过大，nginx之类的客户端发起的请求一直没有fpm子进程进行accept，nginx就会直接断掉这个连接，等fpm忙过来了再去accept的时候，就会发现断开了，于是报错。
backlog设置得过小，访问量大时fpm子进程全部处于忙碌状态，backlog也塞满了，就会拒绝新的连接，此时nginx再请求，就会直接被拒。
所以需要合理的设置backlog参数。