[PHP-FPM详解](https://www.cnblogs.com/followyou/p/9460058.html)

PHP-FPM(PHP FastCGI Process Manager)意：PHP FastCGI 进程管理器，用于管理PHP 进程池的软件，用于接受web服务器的请求。  
PHP-FPM提供了更好的PHP进程管理方式，可以有效控制内存和进程、可以平滑重载PHP配置。

(1). 为什么会出现php-fpm

fpm的出现全部因为php-fastcgi出现。为了很好的管理php-fastcgi而实现的一个程序

(2). 什么是php-fastcgi

php-fastcgi 只是一个cgi程序,只会解析php请求，并且返回结果，不会管理(因此才出现的php-fpm)。

(3)为什么不叫php-cgi

其实在php-fastcgi出现之前是有一个php-cgi存在的,只是它的执行效率低下，因此被php-fastcgi取代。

(4)那fastcgi和cgi有什么区别呢？

亲们，这区别就大了，当一个服务web-server(nginx)分发过来请求的时候，通过匹配后缀知道该请求是个动态的php请求，会把这个请求转给php。

在cgi的年代，思想比较保守，总是一个请求过来后,去读取php.ini里的基础配置信息，初始化执行环境，每次都要不停的去创建一个进程,读取配置，初始化环境，返回数据，退出进程，久而久之，启动进程的工作变的乏味无趣特别累。

当php来到了5的时代，大家对这种工作方式特别反感，想偷懒的人就拼命的想，我可不可以让cgi一次启动一个主进程(master),让他只读取一次配置，然后在启动多个工作进程(worker),当一个请求来的时候，通过master传递给worker这样就可以避免重复劳动了。于是就产生了fastcgi。

(5)fastcgi这么好，启动的worker用完怎么办？  
当worker不够的时候，master会通过配置里的信息，动态启动worker，等空闲的时候可以收回worker

(6)到现在还是没明白php-fpm 是个什么东西?  
就是来管理启动一个master进程和多个worker进程的程序.

PHP-FPM 会创建一个主进程，控制何时以及如何将HTTP请求转发给一个或多个子进程处理。PHP-FPM主进程还控制着什  
么时候创建(处理Web应用更多的流量)和销毁(子进程运行时间太久或不再需要了)  
PHP子进程。PHP-FPM进程池中的每个进程存在的时间都比单个HTTP请求长,可以处  
理10、50、100、500或更多的HTTP请求。

**安装**

PHP在 5.3.3 之后已经把php-fpm并入到php的核心代码中了。 所以php-fpm不需要单独的下载安装。  
要想php支持php-fpm，只需要在编译php源码的时候带上 --enable-fpm 就可以了。

**全局配置**

在Centos中，PHP-FPM 的主配置文件是 /etc/php7/php-fpm.conf。

指定一段时间内有指定个子进程失效了，PHP-FPM重启：

#在指定的一段时间内，如果失效的PHP-FPM子进程数超过这个值，PHP-FPM主进程将优雅重启。

emergency\_restart\_threshold = 10

#设定emergency\_restart\_interval 设置采用的时间跨度。

emergency\_restart\_interval = 1m

**配置进程池**

PHP-FPM配置文件其余的内容是一个名为Pool Defintions的区域。这个区域里的配置用户设置每个PHP-FPM进程池。PHP-FPM进程池中是一系列相关的PHP子进程。通常一个PHP应用有自己一个进程池。

Centos在PHP-FPM主配置文件的顶部引入进程池定义文件：

include=/etc/php7/php-fpm.d/\*.conf

www.conf 是PHP-FPM进程池的默认配置文件。

user= nobody

#拥有这个 PHP-FPM进程池中子进程的系统用户。要把这个设置的值设用的非根用户的用户名。

group = nobody

#拥有这个 PHP-FPM进程池中子进程的系统用户组。要把这个设置的值设应用的非根用户所属的用户组名。

listen=[::]]:9000

#PHP-FPM进程池监听的IP地址和端口号,让 PHP-FPM只接受 nginx从这里传入的请求。

listen. allowed clients =127.0.0.1

#可以向这个 PHP-FPM进程池发送请求的IP地址(一个或多个)。

pm.max children =51

#这个设置设定任何时间点 PHP-FPM进程池中最多能有多少个进程。这个设置没有绝对正确的值,你应该测试你的PHP应用,确定每个PHP进程需要使用多少内存，然后把这个设置设为设备可用内存能容纳的PHP进程总数。对大多数中小型PHP应用来说,每个PHP进程要使用5~15MB内存(具体用量可能有差异【罗总的是20M，使用阿里云内存4G除以20M 大概就是800，所以服务器01开启的进程池就是800】）。 假设我们使用设备为这个PHP-FPM进程池分配了512MB可用内存，那么可以把这个设置设为（512MB总内存）/(每个进程使用10MB) = 51个进程。

编辑保存，重启PHP-FPM主进程：

sudo systemctl restart php-fpm.service

PHP-FPM进程池的配置详情参见 <http://php.net/manual/install.fpm.configuration.php>

**参考Company开发环境**

测试环境的配置如下：

[www]

user = nobody #进程的发起用户和用户组，用户user是必须设置，group不是 nobody 任意用户

group = nobody

listen = [::]:9000 #监听ip和端口，[::] 代表任意ip

chdir = /app #在程序启动时将会改变到指定的位置(这个是相对路径，相对当前路径或chroot后的“/”目录)

pm = dynamic #选择进程池管理器如何控制子进程的数量 static：　　对于子进程的开启数路给定一个锁定的值(pm.max\_children) dynamic:　 子进程的数目为动态的，它的数目基于下面的指令的值(以下为dynamic适用参数)

pm.max\_children = 16 #同一时刻能够存货的最大子进程的数量

pm.start\_servers = 4 #在启动时启动的子进程数量

pm.min\_spare\_servers = 2 #处于空闲"idle"状态的最小子进程，如果空闲进程数量小于这个值，那么相应的子进程会被创建

pm.max\_spare\_servers = 16 #最大空闲子进程数量，空闲子进程数量超过这个值，那么相应的子进程会被杀掉。

catch\_workers\_output = Yes #将worker的标准输出和错误输出重定向到主要的错误日志记录中，如果没有设置，根据FastCGI的指定，将会被重定向到/dev/null上

生产环境配置：

**转发请求给PHP-FPM**

nginx为例：

server {

listen 83;

server\_name mobile.com;

root /app/mobile/web/;

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

location / {

index index.html index.htm index.php;

# Redirect everything that isn't a real file to index.php

try\_files $uri $uri/ /index.php$is\_args$args;

}

#把HTTP请求转发给PHP-FPM进程池处理

location ~ .\*\.php include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

fastcgi\_pass 192.168.33.30:9000; #监听9000端口

fastcgi\_index index.php;

try\_files $uri =404;

#include fastcgi.conf;

}

location ~ /\.(ht|svn|git) {

deny all;

}

access\_log /app/wwwlogs/access.log;

error\_log /app/wwwlogs/error.log;

}

**思考**

* [x] 思考问题：  
  服务器的并发量取决哪些因素？

PHP-FPM最大进程数、nginx并发数（nginx只是充当代理服务器的角色）、内存的占用、cpu的开销

服务器的QPS（每秒处理请求数） = 平均请求连接数\*（1/响应时间）；

并发量 = 有效时间\*QPS；

* 解决：

1. 加机器分流
   * 根据并发量计算需加机器
2. 增加单机新能
   * 开启php的opcache，增加内存空间
   * 接缓存系统
   * sql优化
   * php优化  
     ...