高性能的服务器的架设

对于高性能网站 ,请求量大,如何支撑?
1 方面,要减少请求
对于开发人员----合并 css, 背景图片, 减少 mysql 查询等.

2: 对于运维 nginx 的 expires ,利用浏览器缓存等,减少查询.
3: 利用 cdn 来响应请求

4: 最终剩下的,不可避免的请求----服务器集群+负载均衡来支撑.

所以,来到第 4 步后,就不要再考虑减少请求这个方向了.
而是思考如何更好的响应高并发请求.

大的认识-------既然响应是不可避免的,我们要做的是把工作内容”平均”分给每台服务器.
最理想的状态 每台服务器的性能都被充分利用.

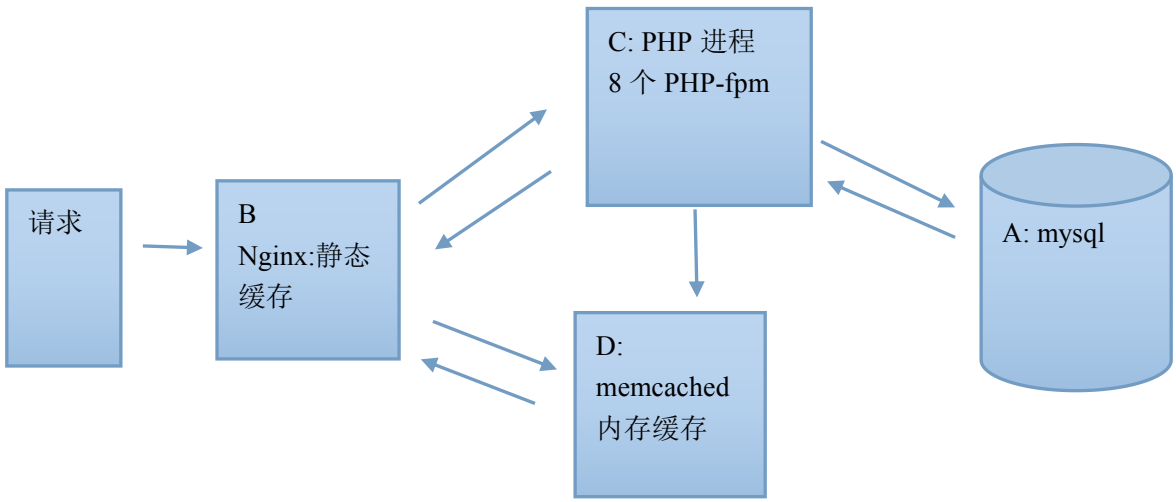
服务器介绍:

服务器 IP:
A　192.168.1.201
B　192.168.1.202
C 203
D204

Root: zixue.it

1 台 A
RAM: 2G
HD: 500G

3 台 B, C, D
RAM: 8G
Hd　: 200G



步骤:
1:A 号服务器
　　1.1 安装 mysql
　　1.2 并导入数据.
　　注意:先把表中的索引去掉,加快导入速度

2: C 号服务器:
　2.1: 编译 PHP
注意: enbale-fpm , with-mysql=mysqlnd　(编译成独立 fpm 进程,支持 mysql,)

2.2: 下载第 3 方的 memcached 扩展 编译进来

3: D 号服:
3.1 编译 memcached

4: B 号服:
编译 nginx ,并配置
Cd /app/pcre-8.12
./configure
Make && make install

Cd nginx-1.2.7
./configure --prefix=/usr/local/nginx --add-module=/app/ngx_http_consistent_hash-master
注:红线部分是 nginx 的第 3 方模块,需要自己下载.

安装统计模块,便于观察 nginx 的状态
./configure --prefix=/usr/local/nginx/ --add-module=/app/ngx_http_consistent_hash-master --with-http_stub_status_module

Php 安装配置
1 tar -xzvf /path/'
2 cd /path/
3 .configure --prefix=/usr/local/php --

服务器集群与负载均衡搭建完毕
1:问题 C-->A 的 mysql 连接很慢
解决: my.cnf 中的[mysqld]节点中,添加
skip-name-resolve // 这句话使 mysql 连接时忽略解析域名,在制定 Mysql 权限时,只能根据 IP 限制,不能根据域名限制.

2: 问题 当 memcache 中没有相应的数据,从后台回调数据时,
http 的状态码是 404,(虽然内容正常),这样不利于 SEO
解决: nginx/conf/nginx.conf
error_page 404 =200 /callback.php; // 这样 404 被改写成 200 来响应中

压力测试:
模拟 前 0-10 万是热数据,
10-20 万是冷门数据

请求热数据 0-10,请求 9 次
请求准予数据 请求 1 次,　　-----100 万次的请求.

优化思路:
nginx 响应请求
1:建立 socket 连接
2: 打开文件,并沿 socket 返回.

排查问题,也要注意观察这两点,
主要从系统的 dmesg ,和 nginx 的 error.log 来观察

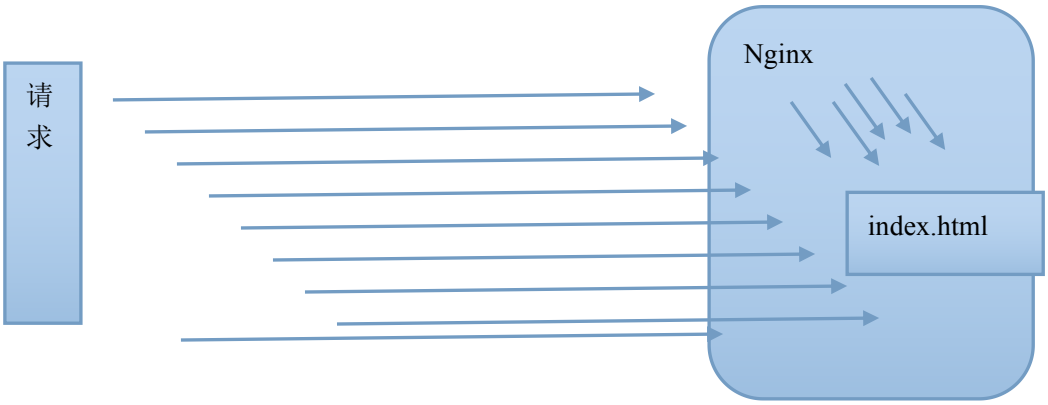
优化过程

1:判断 nginx 的瓶颈

1.1: 首先把 ab 测试端的性能提高,使之能高并发的请求.
易出问题: too many open files
原因 : ab 在压力测试时,打开的 socket 过多
解决: ulimit -n 30000 (重启失效)
观察结果: nginx 不需要特殊优化的情况下, 5000 个连接,1 秒内响应.
满足要求,但 wating 状态的连接过多.

1.2: 解决 waiting 进程过多的问题.
解决办法: keepalive_timeout = 0;
即: 请求结果后,不保留 tcp 连接.
在高并发的情况下, keepalive 会占据大量的 socket 连接.
结果: waiting 状态的连接明显减少.

1.3: 解决服务端 too many open files
分析: nginx 要响应,
1 是要建立 socket 连接,
2 是要读本地文件
这两个者限制.



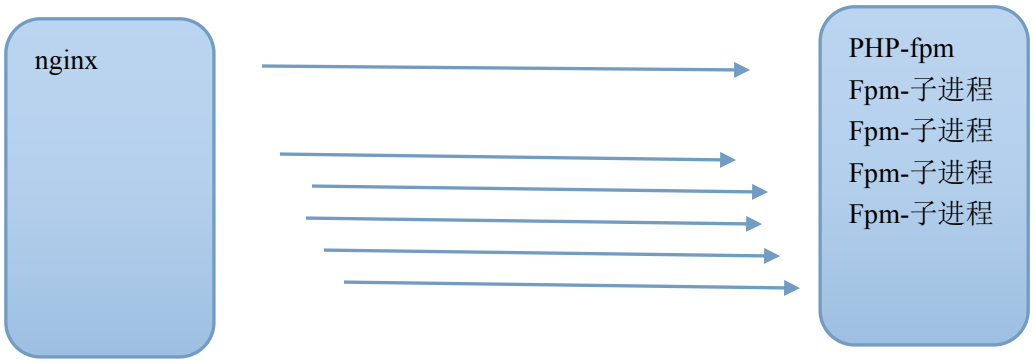
由上图可看出,nginx 的问题容易出在 2 点上:
1: nginx 接受的 tcp 连接多,能否建立起来?
2: nginx 响应过程,要打开许多文件 ,能否打开?

第 1 个问题: 在内核层面(见下)
第 2 个问题 (见下)

系统内核层面:
net.core.somaxconn = 4096 允许等待中的监听
net.ipv4.tcp_tw_recycle = 1 tcp 连接快速回收
net.ipv4.tcp_tw_reuse = 1 tcp 连接重用
net.ipv4.tcp_syncookies = 0 不抵御洪水攻击
ulimit -n 30000

Nginx 层面:
解决: nginx.conf 下面: work_connection 加大
worker_connections 10240;
Worker_rlimit_nofiles 10000;
Keepalive_timeout 0;

Nginx---->php-fpm 之间的优化



如上图,在很多个 nginx 来访问 fpm 时, fpm 的进程要是不够用, 会生成子进程.

生成子进程需要内核来调度,比较耗时,
如果网站并发比较大,
我们可以用静态方式一次性生成若干子进程,保持在内存中.

方法 -- 修改 php-fpm.conf
Pm = static 让 fpm 进程始终保持,不要动态生成
Pm.max_children= 32 始终保持的子进程数量

Php-mysql 的优化

Linux 机器下 ,php 通过 IP 连接其他 mysql 服务器时,容易出的问题
能 ping 能,但 connect 不到.

```
Warning: mysql_connect() [function.mysql-connect]: [2002] No route to host
(trying to connect via tcp://192.168.1.201:3306) in /var/www/phptomysql.php on
line 3

Warning: mysql_connect() [function.mysql-connect]: No route to host in
/var/www/phptomysql.php on line 3
```

一般是由:mysql 服务器的防火墙影响的.

并发 1 万连接,响应时间过长.

优化思路: 同上的 nginx
1: 内核层面,加大连接数,并加快 tcp 回收
2: mysql 层面,增大连接数
3: php 层面,用长连接,节省连接数
4: 用 memcached 缓存,减轻 mysql 负担

具体:
1.1 , PHP 服务器增大 ulimint -n 选项
1.2 mysql 服务器内核配置
添加或修改如下选项
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_syncookies = 0

sysctl -p 使修改立即生效

2.1 修改 mysql.cnf
Vi /etc/my.conf
service mysqld restart 重启 mysql

3.1 PHP 层面 ,用长连接
Mysql_connect ---> mysql_pconnect
注: pconnect 在 PHP 以 apache 模块的形式存在时,无效果.

Nginx+phjp+mysql+nginx
在引入 memcached 后,性能提升不明显,甚至还略有下降

memcached 使 50%的请求变快了,但是一部分,反倒慢了.
原因在于--PHP->memcached 也要建立 tcp 连接,代价挺高,
但缓存了数据之后,就省去了 mysql 的查询时间.

总结: memcached 适合存复杂的 sql,尤其是连接查询/模糊查询的 sql 结果

Memcached 服务器的优化(集中在内核的 ipv4 设置上,不再重复)