Nginx 实战篇

安装部署与配置全解析

TRS 王纯刚 2011/03/25 出品

目录

1 写在正文之前

最近质保在计划做 Portal 的性能测试，考虑到在国家统计局项目前期规划到 2000 多并发的 PV 情况下面，静态文件的压力会超过 10000。

根据对 Nginx 的介绍，Nginx 的性能和 Apache 相比，会有 100%的提升。原因：

得益于 Nginx 使用了最新的 epoll（Linux 2.6 内核）和 kqueue（freebsd）网络 I/O 模型，而 Apache 则使用的是传统的 select 模型。 目前 Linux 下能够承受高并发访问的 Squid、Memcached 都采用的是 epoll 网络 I/O 模型。

所以计划对主流的 Web Server 进行对比性能测试，特引入 Nginx 进行配置，在测试环境中现场实战了一次，特将过程记录总结如下。

注：在真正性能测试结束时，再给出各个 Web Server 的性能对比结果，欢迎大家持续关注。

2 部署步骤

2.1 下载介质

NgInx 部署之前，首先根据项目的需要选择需要安装的组件，实际环境一般会考虑需要支持 gzip 压缩和 rewrite 模块。所以安装的第一步是下载 Ngix 及 Ngix 的相关组件。

- **Nginx 本身**

  下载地址：http://nginx.org/en/download.html
  建议下载最新版本介质，目前最新的是：0.9.6

- **gzip 压缩依赖库：zlib**

  下载地址：http://www.zlib.net/
  我下载的版本是：Version 1.2.5

- **Rewrite 模块的正则表达式依赖库：pcre**

  pcre 库简称：Perl 兼容正则表达式 (Perl Compatible Regular Expressions)库。关于正则表达式的写法可以参考附件《perlre - perldoc_perl_org.mht》
  下载地址：http://www.pcre.org/
  我下载的版本是：pcre-8.02

2.2 编译部署介质

2.2.1 部署 zlib 库

执行 Linux 命令解压缩后编译安装：

```
tar -zxvf zlib-1.2.5.tar.gz
./configure
make
make install
```

默认安装到/usr/local/lib 下即可。
安装完成后可以 ls -l /usr/local/lib/libz.so 查询是否安装成功，安装成功后，通过 ls 可以提示文件已经存在。

2.2.2 部署 pcre 库

执行 Linux 命令解压缩后编译安装：

```
tar -zxvf pcre-8.02.tar.gz
./configure
make
make install
```

默认安装到/usr/local/lib 下即可，安装完成后可以 ls -l /usr/local/lib/libpcre.so 查询是否安装成功

2.2.3 部署 nginx

1. 解压缩介质：tar -zxvf nginx-0.9.6.tar.gz
2. 编 译 ./configure --prefix=/opt/nginx --with-poll_module --with-http_stub_status_module
   需要关注编译的日志
   - checking for epoll ... found 代表找到了高效的 poll 模式
   - checking for PCRE library ... found 代表找到了 pcre 库
   - checking for zlib library ... found 代表找到了 zlib 库
   - 如果后面需要用到状态监控，需要加上 -with-http_stub_status_module
3. 用 make 来安装：
   ```
   make
   make install
   ```
4. 验证 nginx 是否部署成功
   ```
   cd /opt/nginx/sbin
   ./nginx
   ```

netstat -ano | grep 80 | grep nginx，看到 nginx 在 80 端口处于监听状态，代表配置成功。

部署的两点注意：

1.在 kernel>2.6 的情况下面，推荐使用 poll 模式

2.偷懒的情况，可以创建一个软链接到/usr/sbin 下

ln -s /opt/nginx/sbin/nginx /usr/sbin/nginx

2.2.4 启动和停止 nginx

- 启动

```
cd /opt/nginx/sbin
./nginx
```

- 停止

./nginx -s stop

- 重新启动

./nginx –s reload

一点小知识：

查看 Web 请求的部分信息（推荐用 httpwatch 某些情况用不了的时候直接 Linux 命令）

curl --head 127.0.0.1:81

HTTP/1.1 200 OK

Server: nginx/0.9.6

Date: Wed, 23 Mar 2011 07:29:24 GMT

Content-Type: text/html

Content-Length: 151 Last-Modified: Wed, 23 Mar 2011 06:11:44 GMT

Connection: keep-alive

Accept-Ranges: bytes

2.3 配置 nginx

nginx 配置文件存在安装目录的 conf，如/opt/nginx/conf/nginx.conf 下。

配置文件修改后，可用 nginx -t 来进行测试

2.3.1 配置 gzip 压缩

配置 Gzip 压缩，在 nginx.conf 配置文件中加入：

```
gzip on;
gzip_min_length 1024;
gzip_buffers 4 8K;
gzip_types text/plain application/x-javascript text/xml text/css text/html application/xml;
```

每个命令详解如下：

- **zip_min_length**

  设置允许压缩的页面最小字节数，页面字节数从 header 头中的 Content-Length 中进行获取。 默认值是 0，不管页面多大都压缩。 建议设置成大于 1k 的字节数，小于 1k 可能会越压越大。

- **gzip_types**

  配置需要压缩的请求的 Content-Type 类型，对符合指定类型的请求启用 gzip 压缩。

- **gzip_buffers**

  设置系统获取几个单位的缓存用于存储 gzip 的压缩结果数据流。例如 4 4k 代表以 4k 为单位，按照原始数据大小以 4k 为单位的 4 倍申请内存。 4 8k 代表以 8k 为单位，按照原始数据大小以 8k 为单位的 4 倍申请内存。

2.3.2 高性能配置

- 设置工作的进程数

  ```
  worker_processes  5;
  ```

  设置后重新启动 Nginx，可以查看到后台进程：

  ```
  [root@test217 html]# ps -ef | grep nginx
  root      31930     1  0 15:00 ?        00:00:00 nginx: master process nginx
  nobody    31931 31930  0 15:00 ?        00:00:00 nginx: worker process
  nobody    31932 31930  0 15:00 ?        00:00:00 nginx: worker process
  nobody    31933 31930  0 15:00 ?        00:00:00 nginx: worker process
  nobody    31934 31930  0 15:00 ?        00:00:00 nginx: worker process
  nobody    31935 31930  0 15:00 ?        00:00:00 nginx: worker process
  ```

- 指定事件响应模式为为高效的 poll 模式

  ```
  events {
  use epoll;
  worker_connections  1024;
  }
  ```

2.3.3 配置 nginx 状态监控

在 Nginx.conf 配置文件中增加：

```
location /NginxStatus {
    stub_status on;
}
```

重新启动 Nginx，通过浏览器访问 http://192.9.100.217/NginxStatus，就可以看到状态监控页面。

状态监控结果详细解读：

- Active connections: 2
  当前 nginx 正处理的活动连接数
- server accepts handled requests 3 3 10
  总共处理了 3 次连接，成功创建了 3 次连接，共请求了 10 次。总连接数-成功连接数为失败连接数
- Reading: 0 Writing: 1 Waiting: 1
  reading nginx 读取到客户端的 header 信息数
  Writing nginx 返回给客户端的 Header 信息数
  Waiting 开启 keep-alive 的情况下，这个值等于 active-(reading+writing)， 意思指 nginx 已经处理完正在等候下一次请求的驻留连接

2.3.4 反向代理实现动静结合(NgInx + Tomcat / WebLogic / WebSphere)

1. 通过 location 来区分动静文件；

静态文件，在配置文件中增加：

```
location /portal {
    root /home/trs/wchg/wwwhome/;
}
```

注意：这里有个陷阱，就是 root 的目录是指 portal 目录的父目录

2. 动态文件转发给应用服务器

动态文件通过正则表达式区分，所以需要配置 pcre 库，在配置文件中增加：

```
location ~ ^/wcm/.*\.(jsp|do){
proxy_pass http://192.9.100.213:9080;
}
```

根据应用的不同可能需要转发的请求也不同，只要在我的这个配置(jsp|do|XXX)，后面

的 XXX 类似替换为需要转发的后缀就可以了。

注意：和 Rewrite 的不同，匹配到的字符串后面不要再加上$1 了。

3. 反向代理的高级参数

一般在 nginx.conf 配置文件中增加：

proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

proxy_set_header X-Real-IP $remote_addr;

proxy_set_header Host $host;

- ✓ **带上客户原始请求 IP 等请求信息**

  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

  保留客户端传递过来的 XFF 信息;XFF 信息记录 Http 请求的真实的客户端， 也就是 HTTP 的请求端真实的 IP，只有在通过了 HTTP 代理或者负载均衡服务器时才会添加该项；

- ✓ **保留客户端请求的真实 IP 地址，用于某些访问统计**

  proxy_set_header X-Real-IP $remote_addr;

- ✓ **保留客户端请求的域名信息**

  proxy_set_header Host $host;

4. 反向代理的优化参数

proxy_connect_timeout 30;

nginx 跟后端服务器连接超时时间

proxy_send_timeout 30;

后端服务器数据回传时间(代理发送超时)

proxy_read_timeout 60;

连接成功后，后端服务器响应时间(代理接收超时)

proxy_buffer_size 16k;

设置代理服务器（nginx）保存用户头信息的缓冲区大小

proxy_buffers 4 32k;

proxy_buffers 缓冲区，网页平均在 32k 以下的话，这样设置

proxy_busy_buffers_size 64k;

高负荷下缓冲大小（proxy_buffers*2）

proxy_temp_file_write_size 256k;

设定缓存文件夹大小，大于这个值，将从 upstream 服务器传

当代理下载的文件超过该参数设置的大小时， nginx 会先将文件写入临时目录 (缺省为 nginx 安装目下/proxy_temp 目录),

注意：更多优化参数可以参考《HttpProxyModule.mht》

2.3.5 配置虚拟主机

很简单，Server 节点就是一个虚拟主机

1.复制自带的 Server 节点

2.在 Server 节点中的 Server_Name 改成虚拟主机名就可以了

```
server {
    listen          81;
    server_name     portal.test.cn;
}
```

2.3.6 配置静态文件超时时间

在配置文件中增加静态文件的超时时间：

```
location ~/wcm/*\.(jpg|css|html|htm|js|gif|txt) {
    root /home/trs/wchg/wwwhome/;
    expires 24h;
}
```

注意：location 匹配长的优先，匹配长的后就不匹配短的了，所以需要每个 location 都指定 root 或 alias

expires 指令可以控制 HTTP 应答中的“ Expires ”和“ Cache-Control ”的头标（起到控制页面缓存的作用）

具体的语法：expires [time|epoch|max|pff]
默认值：off
expires 指令控制 HTTP 应答中的“Expires”和“Cache-Control”Header 头部信息，启动控制页面缓存的作用
time:可以使用正数或负数。“Expires”头标的值将通过当前系统时间加上设定 time 值来设定。
time 值还控制"Cache-Control"的值：
负数表示 no-cache
正数或零表示 max-age=time

2.3.7 配置日志格式与按天轮换

1. 指定日志格式

由于 nginx 日志的默认格式不能被 awsats 所分析，所以经常需要更改 nginx 的 log 日志格式。具体的做法如下：

a) 自定义日志格式：

```
log_format wchglog '$remote_addr - $remote_user [$time_local]  ' '"$request" $status $body_bytes_sent ' '"$http_referer" "$http_user_agent"';
```

b) 指定访问日志和错误日志的格式和路径：（路径必须让 nginx 提供服务的用户，注意不是启动用户具有写的权限）

```
access_log  logs/access.log  wchglog; error_log logs/error.log
```

为提高日志文件读写性能，可以用下面的指令来缓存：

open_log_file_cache max=1000 inactive=20s min_uses=2 valid=1m;

2. 日志按天切换

方式很多，传统的可以借助 ocronolog 来做，这里采用 logrorate 来实现日志按天切换，但是 ocronolog 因为会附件进程，比较费性能，现在比较好的方式是采用 logrotate 来实现。主要有两个好处：

a) 操作系统安装的时候会自动安装 logrorate。用 Linux 命令可以查看是否安装 cat /etc/log*；
b) 不用 Kill 和 restart nginx，只要重新载入一下就可以了。

具体做法：

参考文件 nginx,将文件放到/etc/lograte.d 目录下面，文件内容如下：

```
/opt/nginx/logs/*.log {
rotate 99
compress
daily
missingok
notifempty
sharedscripts
    postrotate          /sbin/service nginx reload > /dev/null 2>/dev/null || true
endscript
}
```

主要内容的意义如下：

第一行，需要切换的文件
rotate 99 代表总共用 99 个文件来记录
compress 代表启用压缩
daily 代表按天切换日志

2.4 其它配置

2.4.1 禁止出错时泄露服务器的版本

在脚本中增加：

server_tokens off;

2.4.2 限制客户端 POST 提交的数据大小

在脚本中增加：

client_max_body_size 50m;

client_body_buffer_size 256k;

2.4.3 静态目录 root 和 alias 的区别

指定目录的属性：alias 指定的目录是准确的；root 是指定目录的上级目录，并且该上级目录要含有 location 指定名称的同名目录。
使用 alias 标签的目录块中不能使用 rewrite 的 break
注意：一般情况下，在 location /中配置 root，在 location /other 中配置 alias 是一个好习惯。

2.4.4 4.限制并发数和下载速率

```
limit_conn      one 1;
limit_rate 20k;
```

2.4.5 指定 nginx 提供服务的用户：

缺省为 nobody
#user nobody;

2.4.6 指定错误页

```
error_page      500 502 503 504    /portal/internal_error.html;
location = /portal/internal.html {
        alias /home/trs/wchg/wwwhome/portal;
}
```

2.5 高级配置

2.5.1 利用 Nginx 实现开源负载均衡的分发

1. 申明负载均衡有哪几台机器组成

```
upstream wchgCluster {
server localhost:82 weight=2 ;
server localhost:83 weight=3;
}
```

Weight 为权重，代表请求分发的百分比，默认为 1。如上述配置，第一台机器的权重为 2/5 即将 40%的请求分给 82 端口.

2. 定义两个 Server

Server1 监听在本机的 82 端口

```
server {
listen          82;
server_name    localhost;
location / {
  proxy_pass http://192.9.100.213:9080;
}
}
```

Server2 监听在本机的 83 端口

```
server {
listen          83;
server_name    localhost;
location / {
      proxy_pass http://192.9.100.214:9080;
}
}
```

3. 配置反向代理，将请求发给集群 wchgCluster

```
location ~ ^/portal/.*\.(jsp|do) {
      proxy_pass  http://wchgCluster ; #代理给集群的时候，这里的名字和上面的 cluster 的名字相同
}
```

通过测试，我们将 server2 的转发停止掉，然后刷新浏览器就可以看到一会儿正确，一会儿错误了，统计一下请求数，就可以看到转发的规律，基本上是按照 weight 的顺序进行轮流转发。Round-Robin 的模式。

2.5.2 利用 Nginx 实现静态文件的权限控制

1. 主要参考文章

来自于内网**肖锋**文章

《优化 TRS 二进制文件的读写效率，推荐 nginx 的 X-Accel-Redirect 访问》
地址：
http://intranet.trs.com.cn/portal/portaldoc/document_show.jsp?ChannelId=434&PageId=0&DocumentId=200239》

2. 原理补充说明

主要原理：nginx 反向代理得到后端请求发现带有 X-Accel-Redirect 的 header，那么根据这个头记录的路径信息打开磁盘文件

3. 具体做法

以控制某静态目录的权限为例

a) 将真正要控制静态权限的静态目录，如 realpub，禁止直接访问

```
location /realpub/ {
    alias /home/trs/wchg/wwwhome/realpub/;
    internal;
}
```

b) 增加一个虚的目录，如/pub，在虚的目录中配置 rewrite 规则

```
location /pub/ {
alias /home/trs/wchg/wwwhome/pub;
rewrite "/aaa/(.*)$" /portal/right_check.jsp?filename=$1 last;
}
```

c) 在应用中放置 right_check.jsp，根据 filename 参数中所携带的 信息进行权限验证，然后返回。JSP 中的逻辑可以参考下面的代码

```
String sFileName = request.getParameter("filename");

//在这里根据文件名插入业务逻辑

response.reset();
response.setHeader("Content-Type","application/octet-stream");
response.setHeader("X-Accel-Redirect", "/protectfile/"+sFileName);
```

补充两点：

- 注意：最好不要将 realpub 和 pub 指向同一个目录
- 补充：验证过 html、txt 和 zip 文件，完全可行