

第 11 章

Nginx 的非典型应用实例

Nginx 的用途非常多，在第 10 章已经介绍了 Nginx 作为负载均衡服务器、动态应用服务器、静态内容服务器的典型应用案例。而在本章中，主要介绍 Nginx 的一些实用的非典型应用实例。

11.1 用 HTTPS（SSL）构建一个安全的 Nginx Web 服务器

HTTPS（全称：Hypertext Transfer Protocol over Secure Socket Layer），是以安全为目标的 HTTP 通道，简单来讲是 HTTP 的安全版。即 HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。

它是一个 URI scheme（抽象标识符体系），句法类同 http:体系，用于安全的 HTTP 数据传输。https:URL 表明它使用了 HTTP，但 HTTPS 存在不同于 HTTP 的默认端口及一个加密/身份验证层（在 HTTP 与 TCP 之间）。这个系统的最初研发由网景公司进行，提供身份验证与加密通信方法，现在它被广泛用于万维网上安全敏感的通信，例如交易支付方面。

国内一些大型网站用户中心、邮箱、电子支付等业务，都支持 HTTPS（SSL）加密传输。

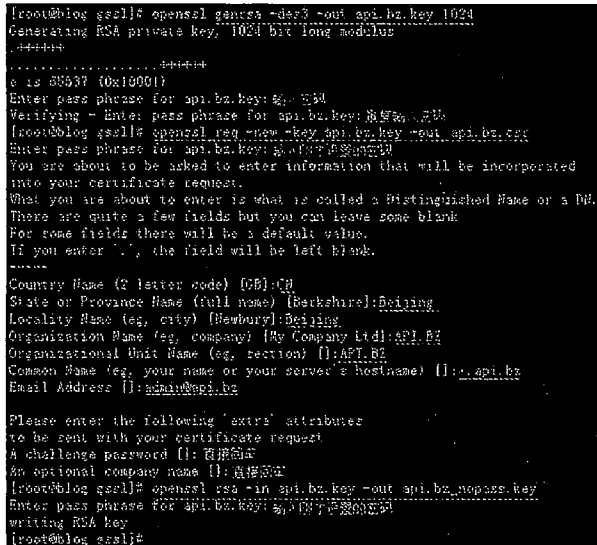
11.1.1 自行颁发不受浏览器信任的 SSL 证书

HTTPS 的 SSL 证书可以自行颁发，下面笔者以域名 api.bz 自行颁发一个 SSL 证书为例，介绍 SSL 证书的颁发步骤。

首先，创建一个私钥文件。api.bz.key 和 api.bz_nopass.key 都是私钥文件，不同的是前者须要输入密码，而后者不须要。假设在 Linux 的/etc/rc.local 文件中添加 Nginx 的启动脚本，重启服务器后，Nginx 会自动启动。但是，如果私钥 api.bz.key 用于 Nginx，那么在 Nginx 启动时会提示输入该私钥文件的密码，Nginx 则无法完成自动启动。使用 api.bz_nopass.key 在启动 Nginx 时无须输入密码，代码如下。

```
openssl genrsa -des3 -out api.bz.key 1024
openssl req -new -key api.bz.key -out api.bz.csr
openssl rsa -in api.bz.key -out api.bz_nopass.key
```

以上三条语句的操作步骤如图 11-1 所示。



```
[root@blog gssli]# openssl genrsa -des3 -out api.bz.key 1024
Generating RSA private key, 1024 bit long modulus
+++++
.....+++++
e is 65537 (0x10001)
Enter pass phrase for api.bz.key: api.bz
Verifying - Enter pass phrase for api.bz.key: api.bz
[root@blog gssli]# openssl req -new -key api.bz.key -out api.bz.csr
Enter pass phrase for api.bz.key: api.bz
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value.
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:CN
State or Province Name (full name) [Berkshire]:Beijing
Locality Name (eg, city) [Newbury]:Beijing
Organization Name (eg, company) [My Company Ltd]:API.BZ
Organizational Unit Name (eg, section) []:API.BZ
Common Name (eg, your name or your server's hostname) []:api.bz
Email Address []:admin@api.bz

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:直接回车
An optional company name []:直接回车
[root@blog gssli]# openssl rsa -in api.bz.key -out api.bz_nopass.key
Enter pass phrase for api.bz.key: api.bz
writing RSA key
[root@blog gssli]#
```

图 11-1 自行生成 SSL 私钥

然后，创建一个自签署的 CA 证书：

```
openssl req -new -x509 -days 3650 -key api.bz_nopass.key -out api.bz.crt
```

按照提示，填写国家、省份、城市、公司、部门、域名、邮箱等内容：

```
Country Name (2 letter code) [GB]:CN
State or Province Name (full name) [Berkshire]:Beijing
```

```

Locality Name (eg, city) [Newbury]:Beijing
Organization Name (eg, company) [My Company Ltd]:API.BZ
Organizational Unit Name (eg, section) []:API.BZ
Common Name (eg, your name or your server's hostname) []:*.api.bz
Email Address []:admin@api.bz

```

通过自行颁发私钥文件 `api.bz_nopass.key` 和 CA 证书 `api.bz.crt`, 就可以搭建安全的 Nginx Web 服务器了。

编译安装 Nginx 服务器时, 须要加上 SSL 模块, 例如:

```

./configure --user=www --group=www --prefix=/usr/local/webserver/nginx
--with-http_ssl_module
make && make install

```

然后, 可以按照代码 11-1 示例内容, 在 `nginx.conf` 配置文件中配置 HTTP (SSL) 虚拟主机:

代码 11-1

```

.....
server
{
    server_name sms.api.bz;
    listen 443;
    index index.html index.htm index.php;

    root /data0/htdocs/api.bz;

    ssl on;
    ssl_certificate api.bz.crt;
    ssl_certificate_key api.bz_nopass.key;
    .....
}
.....

```

自行颁发的 SSL 证书虽然能够实现加密传输功能, 但得不到浏览器的信任, 会出现如图 11-2 所示提示。

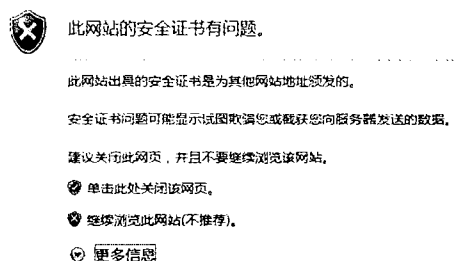


图 11-2 自行颁发的 SSL 证书在 IE7、IE8 浏览器下的提示

要解决浏览器的信任问题, 需要由正规的 CA 机构来颁发证书。

11.1.2 向 CA 机构申请颁发受浏览器信任的 SSL 证书

能够颁发受浏览器信任的 SSL 证书的 CA 机构有很多家, 国外著名的 CA 机构有 VeriSign、GlobalSign、GeoTrust、Thawte、Visa、Microsoft 等, 国内的 CA 机构有中国互联网络信息中心 (CNNIC)。

向 CA 机构申请 SSL 证书, 同样要按照以下步骤操作, 先生成自己的私钥文件和 CSR 文件。然后把 CSR 文件 (api.bz.csr) 提交给 CA 机构, 由 CA 机构生成 CRT 证书文件给你。

```
openssl genrsa -des3 -out api.bz.key 1024
openssl req -new -key api.bz.key -out api.bz.csr
openssl rsa -in api.bz.key -out api.bz_nopass.key
```

CA 机构颁发的 SSL 证书, 通常单域名和泛域名的价格不一样, 所以在使用 openssl 生成 CSR 文件时, 须要注意 Common Name 的填写, 如果你购买的是单域名 SSL 证书, Common Name 填写的内容就必须和你购买的 SSL 证书域名一致。例如你购买了 sms.api.bz 域名的 SSL 证书, Common Name 就必须填写 sms.api.bz:

```
Common Name (eg, your name or your server's hostname) []:*.api.bz
```

在国内的互联网企业中, 对于安全性要求较高的网站, 例如招商银行网上银行、中国工商银行网上银行、中国建设银行网上银行、支付宝、百度的百付宝、腾讯的财付通, 可选择 VeriSign 作为其 SSL 证书供应商。其他一些网站, 例如网易邮箱、金山逍遥网用户中心, 采用的是 CNNIC 颁发的 SSL 证书。

当然, 上述这些 CA 机构颁发的 SSL 证书是要付费的。目前只有一家 CA 机构, 颁发的 SSL 证书是免费的。

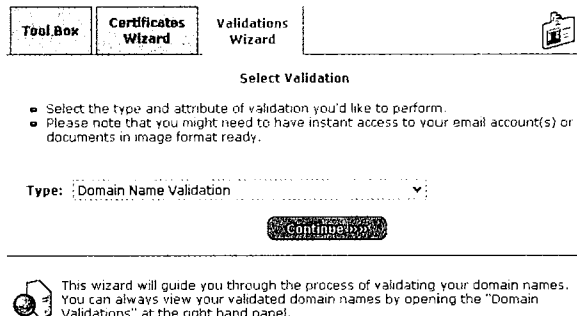
跟 VeriSign 一样, StartSSL (网址: <http://www.startssl.com>, 公司名: StartCom) 也是一家 CA 机构, 它的根证书很早之前就被一些具有开源背景的浏览器支持 (Firefox 浏览器部分版本、谷歌 Chrome 浏览器、苹果 Safari 浏览器等)。

2009 年 9 月, 微软在 Windows 升级补丁中, 更新了通过 Windows 根证书认证程序 (Windows Root Certificate Program) 的厂商清单, 并首次将 StartCom 公司列入了该认证清单, 这是微软首次将提供免费数字验证技术的厂商加入根证书认证列表中。现在, 在 Windows 7 或安装了升级补丁的 Windows Vista 或 Windows XP 操作系统中, 系统会完全信任由 StartCom 这类免费数字认证机构认证的数字证书, 从而让 StartSSL 也获得 IE 浏览器的支持。

注册成为 StartSSL (<http://www.startssl.com>) 用户, 并通过邮件验证后, 就可以申请免费的可信任的 SSL 证书了。步骤比较复杂, 就不详细介绍了, 申请向导的主要步骤如下:

(1) 在申请向导中选择验证类型为域名验证。StartSSL 将通过发邮件到你的域名 Whois 信

息中的联系人 E-mail 地址, 来验证您是否为该域名的拥有者。只有验证通过, 才会颁发证书给你, 如图 11-3 所示。



Tool Box **Certificates Wizard** **Validations Wizard**

Select Validation

- Select the type and attribute of validation you'd like to perform.
- Please note that you might need to have instant access to your email account(s) or documents in image format ready.

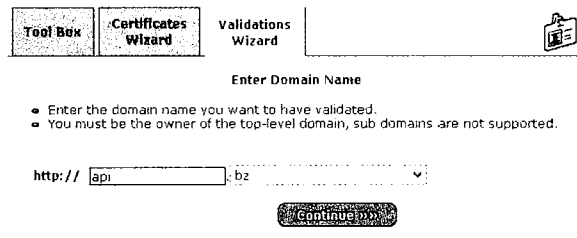
Type: Domain Name Validation

Continue

This wizard will guide you through the process of validating your domain names. You can always view your validated domain names by opening the "Domain Validations" at the right hand panel.

图 11-3 StartSSL 免费 SSL 证书申请向导 (1)

(2) 输入你要申请 SSL 证书的域名, 如图 11-4 所示。



Tool Box **Certificates Wizard** **Validations Wizard**

Enter Domain Name

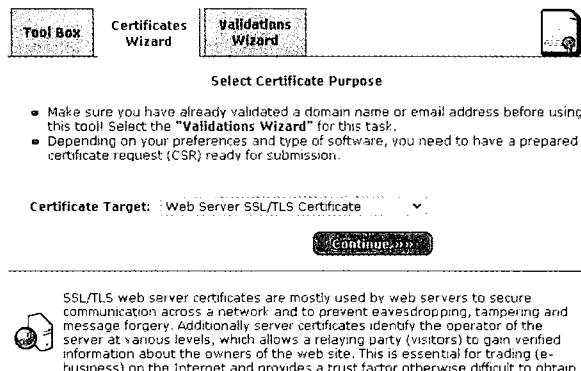
- Enter the domain name you want to have validated.
- You must be the owner of the top-level domain, sub domains are not supported.

http:// api bz

Continue

图 11-4 StartSSL 免费 SSL 证书申请向导 (2)

(3) 选择证书使用目的: Web Server SSL, 如图 11-5 所示。



Tool Box **Certificates Wizard** **Validations Wizard**

Select Certificate Purpose

- Make sure you have already validated a domain name or email address before using this tool. Select the "Validations Wizard" for this task.
- Depending on your preferences and type of software, you need to have a prepared certificate request (CSR) ready for submission.

Certificate Target: Web Server SSL/TLS Certificate

Continue

SSL/TLS web server certificates are mostly used by web servers to secure communication across a network and to prevent eavesdropping, tampering and message forgery. Additionally server certificates identify the operator of the server at various levels, which allows a relaying party (visitors) to gain verified information about the owners of the web site. This is essential for trading (e-business) on the Internet and provides a trust factor otherwise difficult to obtain.

图 11-5 StartSSL 免费 SSL 证书申请向导 (3)

并非所有浏览器都能信任 StartSSL 的证书，如果您的网站比较重要，还是建议您购买 VeriSign、GlobalSign、CNNIC 等商业 CA 机构的 SSL 证书。

11.2 采用 Nginx 搭建 FLV 视频服务器

国内外著名的视频分享网站，例如 YouTube、优酷网、土豆网、新浪播客等，采用的都是 Flash 播放器播放 FLV/MP4 视频文件的技术。

FLV 视频可以采用两种方式发布，一种是普通的 HTTP 下载方式，另一种是基于 Flash Media Server 或 Red5 服务器的 RTMP/RTMPT 流媒体方式。YouTube、优酷网、土豆网、新浪播客等，采用的是 HTTP 下载方式，金山逍遥网的部分游戏视频采用的是 RTMP 流媒体方式。流媒体方式无须下载视频文件到本地，可以实时播放服务器上的 FLV 文件，可以任意拖曳 Flash 播放器播放进度条，用户体验较好，但是比较消耗服务器资源。而 HTTP 下载方式须要下载 FLV 视频文件到本地播放，一旦 FLV 视频下载完成，就不消耗服务器资源和带宽了，虽然也可以实现拖动，但是拖动功能不如流媒体方式强大。

对基于 HTTP 下载方式的 FLV 视频发布，可以通过 Nginx 的 Flv Stream 模块，实现 FLV 视频播放的进度条拖动。对基于 RTMPT 流媒体协议的 FLV 视频发布，Nginx 可以用来实现对后端 Flash Media Server 或 Red5 服务器 RTMPT 的反向代理。

11.2.1 采用 Nginx 的 Flv Stream 模块搭建 HTTP 下载方式的 FLV 视频服务器

Flv Stream 模块是 Nginx 的可选模块，需要在编译安装 Nginx 服务器时，把 Flv Stream 模块加上，例如：

```
./configure --user=www --group=www --prefix=/usr/local/webserver/nginx
--with-http_flv_module
make && make install
```

然后，按照代码 11-2 所示示例方式，在 nginx.conf 配置文件中配置存储 FLV 视频文件的虚拟主机：

代码 11-2

```
.....
server
{
    listen      80;
    server_name flv.domain.com;
    index index.shtml index.html index.htm;
```



```
root /data0/htdocs/flv_files;

limit_rate_after 3m;
limit_rate 512k;

location ~ /\.flv
{
    flv;
}

access_log off;
}
.....
```

重启 Nginx 后, Nginx 配置部分就算 OK 了。limit_rate_after 3m 和 limit_rate 512k 两项指令设置了一开始不限速, 在客户端下载 FLV 视频大小超过 3MB 后, 开始限制下载速度为 512KB/秒。一开始不限速可以使得刚播放视频时, 下载到客户端的视频文件字节数尽量多, 用户播放无须长时间等待缓冲。大小超过 3MB 的视频文件一般都比较大会比较大, 可以限速让用户边观看边下载, 在不影响用户体验的情况下, 可以节省不少服务器带宽资源。

当用户拖动 Flash 播放器的进度条时, 会发起一个类似 `http://flv.domain.com/test.flv?start=12345` 的请求 URL 到 Nginx 服务器, 告诉 Nginx 服务器下载 start 参数值附近的关键帧之后的那部分 FLV 文件, 从而实现拖动播放。

光配置好 Nginx 服务器不管用, 还需要 FLV 文件 MetaData 中含有关键帧信息, 才能实现拖动播放。一般从其他视频格式转换成 FLV 视频格式的文件 MetaData 中是不含关键帧信息的, 可以在 Linux 下使用开源软件自动为 FLV 文件添加关键帧信息。

Linux 下为 FLV 文件添加关键帧的常用软件有两种。

(1) 开源软件 Yamdi:

我们可以按照代码 11-3 所示方式编译安装 Yamdi:

代码 11-3

```
wget
http://sourceforge.net/projects/yamdi/files/yamdi/1.4/yamdi-1.4.tar.gz/download
tar zxvf yamdi-1.4.tar.gz
cd yamdi-1.4/
make
make install
cd ../
```

安装完成 Yamdi 之后, 可以按照以下示例为 FLV 文件添加关键帧信息:

```
yamdi -i test.flv -o test.flv.new
mv -f test.flv.new test.flv
```

(2) 开源软件 FlvTool2:

在 CentOS 下, 可以按照以下方式安装 FlvTool2:

```
yum install ruby
wget http://blog.s135.com/soft/linux/flvtool2/flvtool2-1.0.6.tgz
tar zxvf flvtool2-1.0.6.tgz
cd flvtool2-1.0.6/
ruby setup.rb
cd ../
```

安装完成 Yamdi 之后, 可以按照以下示例为 FLV 文件添加关键帧信息:

```
flvtool2 -U test.flv
```

采用 FlvTool2 为 FLV 文件添加的关键帧为每两秒钟一个, 比 Yamdi 添加的关键帧要多, 因此视频拖动播放时要更流畅。但是, FlvTool2 为 FLV 文件添加关键帧所耗费的时间要比 Yamdi 多几倍到几十倍。

为 FLV 视频文件添加完关键帧后, 就要找一个支持 HTTP 进度条拖动的 Flash 播放器来播放 FLV 视频了。开源的 JW Player 播放器是个不错的选择, 可以通过其官网网站 (<http://www.longtailvideo.com/>) 下载。

我们可以通过 JW Player, 测试播放 Nginx 服务器上的 FLV 视频, 如果不出意外, 应该可以正常支持播放进度条拖动:

```
http://player.longtailvideo.com/player.swf?type=http&file=
http://flv.domain.com/test.flv
```

11.2.2 采用 Nginx 实现 FMS/Red5 流媒体视频服务器的负载均衡

Adobe 公司官方的 FMS (Flash Media Server) 和第三方的开源 Red5 流媒体服务器都支持 RTMP 协议、RTMPT 协议的 FLV 视频流媒体播放。

RTMP 全称 Real Time Messaging Protocol (实时消息传送协议), 是 Adobe Systems 公司为 Flash 播放器和服务器之间音频、视频和数据传输开发的私有协议。

RTMPT 协议是一个包装了 RTMP 的 HTTP 协议, 它从客户端发送 POST 请求到服务器。由于 HTTP 连接的非持久性本质, 为了及时更新状态和数据, RTMPT 需要客户端周期性向服务器轮询, 取得服务器或其他客户端产生的通知事件。

通过 Nginx, 可以实现多台 FMS/Red5 流媒体服务器 RTMPT 协议流媒体视频播放的负载均衡。nginx.conf 配置示例如代码 11-4 所示:

代码 11-4

```

.....
upstream fms_server_pool {
    ip_hash;
    server 192.168.1.3:80 max_fails=2 fail_timeout=30s;
    server 192.168.1.4:80 max_fails=2 fail_timeout=30s;
    server 192.168.1.5:80 max_fails=2 fail_timeout=30s;
}

server
{
    listen      80;
    server_name flv.domain.com;
    index index.html index.htm;
    root /data0/htdocs/flv_files;

    proxy_redirect    off;
    proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;

    proxy_connect_timeout    120;
    proxy_send_timeout       120;
    proxy_read_timeout       120;

    proxy_buffer_size        4k;
    proxy_buffers             4 32k;
    proxy_busy_buffers_size   64k;
    proxy_temp_file_write_size 64k;

    location /open/ {
        proxy_pass      http://fms_server_pool;
    }

    location /close/ {
        proxy_pass      http://fms_server_pool;
    }

    location /idle/ {
        proxy_pass      http://fms_server_pool;
    }

    location /send/ {
        proxy_pass      http://fms_server_pool;
    }

    access_log /data1/logs/rtmpt.log;
}
.....

```

刚才已经介绍过的开源 JW Player 播放器，同样可以用来播放 RTMPT 协议的 FLV 视频，示例如下：

http://player.longtailvideo.com/player.swf?streamer=rtmpt://flv.domain.com/play&file=videos/test.flv

播放基于 RTMPT 协议的 FLV 流媒体视频时,可以通过 Nginx 的访问日志/data1/logs/rtmpt.log 来了解整个播放流程。RTMPT 用命令: OPEN、SEND、IDLE、CLOSE 来控制网络连接,当处理 RTMPT 的请求时,所有的 RTMPT 命令使用 POST 方法来处理请求和回应,此外其他命令则可以使用 GET 方法,如代码 11-5 所示。

代码 11-5

```
192.168.1.2 - - [14/Jan/2010:03:12:06 +0800] "POST /open/1 HTTP/1.1" 200 11 "-"
"Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:11 +0800] "POST /idle/1248426880/0 HTTP/1.1" 200
1 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:17 +0800] "POST /send/1248426880/1 HTTP/1.1" 200
3074 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:22 +0800] "POST /send/1248426880/2 HTTP/1.1" 200
289 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:28 +0800] "POST /send/1248426880/3 HTTP/1.1" 200
1 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:34 +0800] "POST /send/1248426880/4 HTTP/1.1" 200
116 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:41 +0800] "POST /send/1248426880/5 HTTP/1.1" 200
534741 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:45 +0800] "POST /send/1248426880/6 HTTP/1.1" 200
745276 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:49 +0800] "POST /send/1248426880/7 HTTP/1.1" 200
784077 "-" "Shockwave Flash"
.....
```

通过 Nginx 对 FMS/Red5 流媒体视频服务器的反向代理,既能够实现负载均衡,又能够实现自动故障转移,是一项不错的选择。

11.3 Nginx+PHP+MySQL 在小内存 VPS 服务器上的优化

VPS (全称 Virtual Private Server) 是利用最新虚拟化技术在一台物理服务器上创建多个相互隔离的虚拟私有主机。它们以最大化的效率共享硬件、软件许可证及管理资源。对其用户和应用程序来讲,每一个 VPS 平台的运行和管理都与一台独立主机完全相同,因为每个 VPS 均可独立进行重启并拥有自己的 root 访问权限、用户、IP 地址、内存、过程、文件、应用程序、系统函数库及配置文件。

目前提供 VPS 租用服务的提供商越来越多,大有取代虚拟主机之势。VPS 服务器最重要的指标就是内存大小,多个 VPS 服务器可以共享一颗 CPU,但不能共享同一块内存。内存越大,价格越贵。

本节针对 128MB 内存 Linux 操作系统的 VPS，对 Nginx + PHP + MySQL 优化进行介绍。

11.3.1 增加 swap 交换文件

如果您的 VPS 提供商没有为您的 VPS 划分 swap 交换分区，则可以自行创建 swap 交换文件来代替交互分区。

(1) 创建并激活 swap 交换文件。

```
cd /var/  
dd if=/dev/zero of=swapfile bs=1024 count=262144  
/sbin/mkswap swapfile  
/sbin/swapon swapfile
```

(2) 加到 fstab 文件中让系统引导时自动启动。

```
vi /etc/fstab
```

在末尾增加以下内容：

```
/var/swapfile swap swap defaults 0 0
```

11.3.2 Nginx 的主配置文件（nginx.conf）优化

Nginx 的主配置文件（nginx.conf）优化示例如代码 11-6 所示：

代码 11-6

```
user www www;  
  
#Nginx 每个进程耗费 10MB~12MB 内存，这里只开启一个 Nginx 进程，节省内存。  
worker_processes 1;  
  
error_log /data/logs/nginx_error.log crit;  
  
pid /usr/local/webserver/nginx/nginx.pid;  
  
#Specifies the value for maximum file descriptors that can be opened by this process.  
worker_rlimit_nofile 51200;  
  
events  
{  
    use epoll;  
    worker_connections 51200;  
}  
  
http  
{
```

```

include      mime.types;
default_type application/octet-stream;

#charset gb2312;

server_names_hash_bucket_size 128;
client_header_buffer_size 32k;
large_client_header_buffers 4 32k;

sendfile on;
tcp_nopush  on;

keepalive_timeout 60;

tcp_nodelay on;

fastcgi_connect_timeout 300;
fastcgi_send_timeout 300;
fastcgi_read_timeout 300;
fastcgi_buffer_size 64k;
fastcgi_buffers 4 64k;
fastcgi_busy_buffers_size 128k;
fastcgi_temp_file_write_size 128k;

#对网页文件、CSS、JS、XML 等启动 gzip 压缩，减少数据传输量，提高访问速度。
gzip on;
gzip_min_length 1k;
gzip_buffers 4 16k;
gzip_http_version 1.0;
gzip_comp_level 2;
gzip_types text/plain application/x-javascript text/css application/xml;
gzip_vary on;

server
{
    listen      80;
    server_name www.yourdomain.com;
    index index.html index.htm index.php;
    root /data0/htdocs/blog;

    location ~ .*\. (php|php5)?$
    {
        #将 Nginx 与 FastCGI 的通信方式由 TCP 改为 Unix Socket。TCP 在高并发访问下比 Unix Socket
        #稳定，但 Unix Socket 速度要比 TCP 快。
        fastcgi_pass unix:/tmp/php-cgi.sock;
        #fastcgi_pass 127.0.0.1:9000;
        fastcgi_index index.php;
        include fcgi.conf;
    }

    #图片更改较少，将它们在浏览器本地缓存 15 天，可以提高下次打开时的加载速度。
    location ~ .*\. (gif|jpg|jpeg|png|bmp|swf)$

```

```

{
    expires      15d;
}

#将JavaScript、CSS在浏览器本地缓存1天，访问者在看完一篇文章或一页后，再看另一篇文章或另一
#页的内容，无须从服务器再次下载相同的JavaScript、CSS，提高了页面显示速度。
location ~ .*\. (js|css)? $
{
    expires      1d;
}

log_format access '$remote_addr - $remote_user [$time_local] "$request" '
    '$status $body_bytes_sent "$http_referer" '
    '"$http_user_agent" $http_x_forwarded_for';
access_log /data/logs/access.log access;
}
}

```

11.3.3 PHP (FastCGI) 的配置优化

(1) php.ini 配置文件中关于 eAccelerator 的优化。只使用 1MB 共享内存，删除所有在最后 3 600 秒内无法存取脚本缓存，用磁盘辅助进行缓存，如代码 11-7 所示。

代码 11-7

```

[eaccelerator]
zend_extension="/usr/local/webserver/php/lib/php/extensions/no-debug-non-zts-
20060613/eaccelerator.so"
eaccelerator.shm_size="1"
eaccelerator.cache_dir="/usr/local/webserver/eaccelerator_cache"
eaccelerator.enable="1"
eaccelerator.optimizer="1"
eaccelerator.check_mtime="1"
eaccelerator.debug="0"
eaccelerator.filter=""
eaccelerator.shm_max="0"
eaccelerator.shm_ttl="3600"
eaccelerator.shm_prune_period="3600"
eaccelerator.shm_only="0"
eaccelerator.compress="1"
eaccelerator.compress_level="9"
eaccelerator.keys = "disk_only"
eaccelerator.sessions = "disk_only"
eaccelerator.content = "disk_only"

```

(2) 如果您按照第 5 章安装了 PHP (FastCGI)，则可以按照以下方式对 php-fpm.conf 配置进行优化。

修改两项，一是修改以下一行，将启动的 php-cgi 进程数改为 5 个：

```
<value name="max_children">5</value>
```

二是修改以下一行，将 TCP 模式改为 Unix Socket 模式：

```
<value name="listen_address">/tmp/php-cgi.sock</value>
```

11.3.4 MySQL 5.1 配置优化

(1) 使用以下参数编译安装的 MySQL 5.1 默认支持 4 种存储引擎：CSV、MRG_MYISAM、MEMORY、MyISAM，不支持 InnoDB 存储引擎。由于内存有限，而 InnoDB 耗费的内存较大，这里推荐使用 MyISAM 存储引擎。

```
./configure --prefix=/usr/local/webserver/mysql/ --enable-asm
--with-extra-charsets=complex --enable-thread-safe-client --with-big-tables
--with-readline --with-ssl --with-embedded-server --enable-local-infile
make && make install
```

(2) MySQL 5.1 配置文件 (my.cnf) 优化示例如代码 11-8 所示。

代码 11-8

```
[client]
port      = 3306
socket    = /tmp/mysql.sock

[mysqld]
user      = mysql
port      = 3306
socket    = /tmp/mysql.sock
basedir   = /usr/local/webserver/mysql
datadir   = /usr/local/webserver/mysql/data
open_files_limit = 600
back_log  = 20
max_connections = 100
max_connect_errors = 200
table_cache = 60
external-locking = FALSE
max_allowed_packet = 16M
sort_buffer_size = 128K
join_buffer_size = 128K
thread_cache_size = 10
thread_concurrency = 8
query_cache_size = 0M
query_cache_limit = 2M
query_cache_min_res_unit = 2k
default_table_type = MyISAM
thread_stack = 192K
transaction_isolation = READ-UNCOMMITTED
tmp_table_size = 512K
max_heap_table_size = 32M
/usr/local/webserver/mysql/data/slow.log
```

```
/usr/local/webserver/mysql/data/error.log
long_query_time = 1
log_long_format
server-id = 1
#log-bin = /usr/local/mysql/data/binlog
binlog_cache_size = 2M
max_binlog_cache_size = 4M
max_binlog_size = 512M
expire_logs_days = 7
key_buffer_size = 4M
read_buffer_size = 1M
read_rnd_buffer_size = 2M
bulk_insert_buffer_size = 2M
myisam_sort_buffer_size = 4M
myisam_max_sort_file_size = 10G
myisam_max_extra_sort_file_size = 10G
myisam_repair_threads = 1
myisam_recover

[mysqldump]
quick
max_allowed_packet = 16M
```

11.4 采用 Nginx 搭建正向代理服务器

正向代理就是通常所说的代理，是某台电脑通过一台服务器来上 Internet 网的这种方式，其中这台电脑就叫客户机，这台服务器就叫正向代理服务器，也就是通常所说的代理服务器。一般情况下，客户机必须指定代理服务器（IE 浏览器可在工具→Internet 选项→连接→局域网设置→代理服务器中设置）。

Nginx 正向代理的 nginx.conf 配置文件如代码 11-9 所示：

代码 11-9

```
.....
server
{
    listen      8080;
    location / {
        #DNS 解析服务器的 IP 地址
        resolver 8.8.8.8;
        proxy_pass http://$host$request_uri;
    }

    access_log /data1/logs/proxy.log;
}
.....
```

配置完成后，重启 Nginx 使配置生效。然后，你就可以在 IE 浏览器菜单栏中的“工具”→“Internet 选项”→“连接”→“局域网设置”→“代理服务器”中设置代理服务器 IP 地址（假设为 61.1.1.1）和端口，如图 11-6 所示。

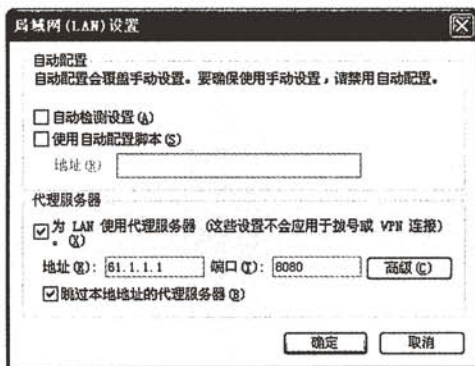


图 11-6 IE 浏览器中的代理服务器设置

然后，您的 IE 浏览器就可以在 Nginx 代理服务器访问 Internet 了。