



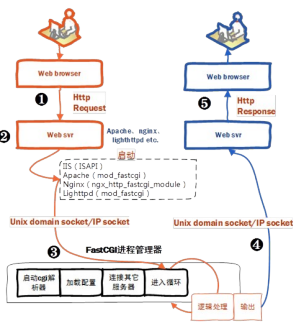
1. Web 服务器启动时，初始化FastCGI程序执行环境，加载apache的mod\_fastcgi模块，nginx的ngx\_http\_fastcgi\_module模块，libstdc++库，libhttpd2mod\_fastcgi模块。

2. FastCGI总控管理进程，启动fastcgi程序，向它使用spawn-fcgi管理器启动demo

cp/nginx-1.7.7/sbin/spawn-fcgi -a 127.0.0.1 -p 8088 -f /opt/nginx-1.7.7/cgi-bin/demo

fastcgi程序进行初始化，然后进入死循环，监听用户的请求

当进来一个请求时，Web 服务器把环境变量和这个页面请求通过一个unix domain socket(都位于同一物理服务器)或者一个IP Socket (FastCGI部署在其它物理服务器上)传送给FastCGI进程。



说明：在收到连接请求用户请求

1. web 服务器的fastcgi运行环境，通过socket（或socket即ipsocket），将数据传递给fastcgi程序进程。
2. fastcgi程序进程收到请求后，进行对应的逻辑处理。
3. 最后将处理结果通过socket（或socket即ipsocket），返回给web 服务器。
4. web 服务器通过Http Response响应包，返回给浏览器。

- step1. Web 服务器启动时嵌入初始化FastCGI执行环境，例如IIS ISAPI，apache mod\_fastcgi，nginx ngx\_http\_fastcgi\_module，lighttpd mod\_fastcgi
- step2. FastCGI进程管理器自身初始化。启动多个CGI解释器进程并等待来自Web 服务器的连接。启动FastCGI进程时，可以配置以ip和UNIX 域socket两种方式启动。
- step3. 当客户端请求到达Web 服务器时，Web 服务器将请求采用socket方式转发到 FastCGI主进程。FastCGI主进程选择并连接到一个CGI解释器，Web 服务器将CGI环境变量和标准输入及送到FastCGI子进程。

- step4. FastCGI子进程完成处理后将标准输出和错误信息从同一socket连接返回Web 服务器。当FastCGI子进程关闭连接时，请求便处理完成。
- step5. FastCGI子进程接着等待并处理来自Web 服务器的下一个连接。

由于 FastCGI 程序并不像不断的产生新进程，可以大大降低服务器的压力并且产生较高的应用效率，它的速度效率最少要比CGI 技术提高 5 倍以上，它还支持分布式的部署，即 FastCGI 程序可以在web 服务器以外的主机上执行。

**概念：**CGI 就是所谓的短生存期应用程序，FastCGI 就是所谓的长生存期应用程序。FastCGI像是一个常驻(long-live)型的CGI，它可以一直执行着，不会每次都花费时间去fork一次(这是CGI最为人们所诟病的fork-and-execute 模式)。

### 3.nginx cgi/fastcgi

nginx 不能像apache那样直接执行外部可执行程序，但nginx可以作为代理服务器，将请求转发给后端服务器，这也是nginx的主要作用之一，其中nginx就支持FastCGI代理，接收客户端的请求，然后将请求转发给后端fastcgi进程。下面介绍如何使用C/C++编写cgi/fastcgi，并部署到nginx中。

#### 3.1.nginx + fastcgi

通过前面的介绍知道，fastcgi进程由FastCGI进程管理器管理，而不是nginx，这样就需要一个FastCGI管理，管理我们编写fastcgi程序，本文使用spawn-fcgi作为FastCGI进程管理器。

##### 3.1.1. spawn-fcgi

spawn-fcgi是一个通用的FastCGI进程管理器，简单小巧，原先是属于lighttpd的一部分，后来由于使用比较广泛，所以就迁移出来作为独立项目了，spawn-fcgi使用pre-fork 模型，功能主要是打开监听端口，绑定地址，然后fork-and-exec创建我们编写的fastcgi应用程序进程，退出完成工作，fastcgi应用程序初始化，然后进入死循环监听socket的连接请求。

安装spawn-fcgi:

- 获取spawn-fcgi编译安装包。在<http://redmine.lighttpd.net/projects/spawn-fcgi/wiki>上可以获取当前最新的版本。
- 解压缩spawn-fcgi-x.x.x.tar.gz包。
- 进入解压缩目录，执行./configure。
- make & make install

如果遇到以下错误：“autogen.sh: x: autoreconf: not found”，因为没有安装automake 工具，ubuntu下下面的命令安装好就可以了：sudo apt-get install autoconf automake libtool。

spawn-fcgi的帮助信息可以通过man spawn-fcgi或spawn-fcgi -h获取，下面是部分常用spawn-fcgi参数信息：

```
-f <fcgiapp> 指定调用FastCGI的进程的执行程序位置
-a <addr> 绑定到地址addr。
-p <port> 绑定到端口port。
-s <path> 绑定到unix domain socket
-C <chlds> 指定产生的FastCGI的进程数，默认为5。（仅用于PHP）
-P <path> 指定产生的进程的PID文件路径。
-f <chlds> 指定产生的FastCGI的进程数（C的CGI用这个）
-u和-g FastCGI使用什么身份（-u 用户 -g 用户组）运行。CentOS下可以使用apache用户，其他的根据情况配置，如nobody，www-data等。
```

##### 3.1.2. 编写fastcgi应用程序

使用C/C++编写fastcgi应用程序，可以使用FastCGI软件开发套件或者其它开发框架，如fastcgi++。

本文使用FastCGI软件开发套件——fcgi (<http://www.fastcgi.com/drupal/rv8q/node21>)，通过此套件可以轻松编写fastcgi应用程序，安装fcgi:

- 获取fcgi编译安装包。在<http://www.fastcgi.com/drupal/rv8q/node21>上可以获取当前最新的版本。
- 解压缩fcgi-x.x.x.tar.gz包。
- 进入解压缩目录，执行./configure。
- make & make install

如果编译提示一下错误：

```
fcgi.cpp: In destructor 'virtual fcgi_streambuf::~fcgi_streambuf()':
```

```
fcgi.cpp:50: error: 'EOF' was not declared in this scope
```

```
fcgi.cpp: In member function 'virtual int fcgi_streambuf::overflow(n)':
```

```
fcgi.cpp:70: error: 'EOF' was not declared in this scope
```

```
fcgi.cpp:75: error: 'EOF' was not declared in this scope
```

```
fcgi.cpp: In member function 'virtual int fcgi_streambuf::sync()':
```

```
fcgi.cpp:86: error: 'EOF' was not declared in this scope
```

```
fcgi.cpp:87: error: 'EOF' was not declared in this scope
```

```
fcgi.cpp: In member function 'virtual int fcgi_streambuf::underflow()':
```

```
fcgi.cpp:113: error: 'EOF' was not declared in this scope
```

```
make[2]: *** [fcgi.o] Error 1
```

```
make[2]: Leaving directory '/root/downloads/fcgi-2.4.1-SNAP-0910052249/libfcgi'
```

```
make[1]: *** [all-recursive] Error 1
```

```
make[1]: Leaving directory '/root/downloads/fcgi-2.4.1-SNAP-0910052249'
```

```
make: *** [all] Error 2
```

解决办法:在./include/fcgi.h文件中加上#include <stdio>, 然后再编译安装就通过了。

如果提示找不到头文件，请在LD\_LIBRARY\_PATH或/etc/ld.so.conf中添加fcgi的安装路径，如/usr/local/lib，并执行ldconfig更新一下。

```
#include "fcgi_stdio.h"
#include <stdlib.h>
```

```
int main(void)
{
    int count = 0;
    while (FCGI_Accept() >= 0)
    {
        printf("Content-type: text/html\n"
            "\n\n"
            "<title>FastCGI Hello!</title>"
            "<h1>FastCGI Hello!</h1>"
            "Request number %d running on host %s</h1>\n",
            ++count, getenv("SERVER_NAME"));
    }
    return 0;
}
```

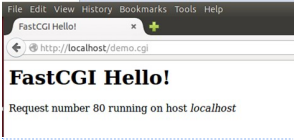
编译g++ main.cpp -o demo -lfcgi, 并将demo部署到/opt/nginx-1.7.7/cgi-bin/目录  
通过spawn-fcgi调用c/c++编写好的fastcgi程序 ./opt/nginx-1.7.7/sbin/spawn-fcgi -a 127.0.0.1 -p 8081 -f /opt/nginx-1.7.7/cgi-bin/demo

### 3.1.3. nginx fastcgi配置

关于nginx的几个配置文件解析, 可以参阅([nginx安装与使用](#))<http://www.cnblogs.com/skynet/p/4146083.html>, 在上置的nginx.conf基础上增加下面的fastcgi配置。

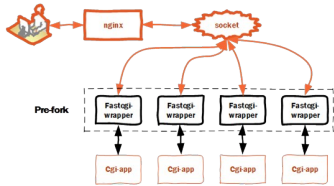
```
server {  
    listen 80;  
    server_name localhost;  
  
    #charset koi8-r;  
  
    #access_log logs/host.access.log main;  
  
    location / {  
        root html;  
        index index.html index.htm;  
    }  
  
    error_page 404 /404.html;  
    # redirect server error pages to the static page /50x.html  
    #  
    #error_page 500 502 503 504 /50x.html;  
    location = /50x.html {  
        root html;  
    }  
  
    location = /demo.cgi {  
        fastcgi_pass 127.0.0.1:8081;  
        fastcgi_index index.cgi;  
        include fastcgi.conf;  
    }  
}
```

这样nginx收到<http://localhost/demo.cgi>请求时, 会匹配到location = /demo.cgi块, 将请求传到后端的fastcgi应用程序处理, 如下所示: (注意其中number为80, 是因为该请求了80次)



### 3.2. nginx + cgi

nginx 不能直接运行外部都可执行程序, 并且cgi是接收请求时才会启动cgi进程, 不像fastcgi会在一开始就启动好, 这样nginx天生是不支持cgi的, nginx 从然不支持cgi, 但它支持 fastCGI, 所以, 我们可以考虑使用fastcgi包裹来支持 cgi, 原理大致如下而所示: pre-fork几个通用的代理fastcgi程序——fastcgi-wrapper, fastcgi-wrapper启动执行cgi然后得到cgi的执行结果返回给nginx(fork-and-exec)。



明白原理之后, 编写一个fastcgi-wrapper也比较简单, 网上流传比较多的一个解决方案是, 来自nginx wiki(<http://wiki.nginx.org/SimpleCGI>) 上的使用perl的fastcgi包裹脚本cgiwrap-fcgi.pl, 但使用perl不是很感冒, 下面给出一个C/C++写的fastcgi-wrapper,

#### 3.2.1. fastcgi-wrapper

其实编写C/C++的fastcgi-wrapper, 就是写一个C/C++的fastcgi, 多像和原理图前面的小节(nginx+fastcgi)一样, github上已经有人开源了, C写的fastcgi-wrapper: <https://github.com/gnoesk/fcgiwrap>,

安装fcgiwrap:

- 下载 (<https://github.com/gnoesk/fcgiwrap.git>)
- 解压缩fcgiwrap, 进入解压缩目录
- autoreconf -i
- ./configure
- make && make install

启动fastcgi-wrapper: ./opt/nginx-1.7.7/sbin/spawn-fcgi -f /usr/local/sbin/fcgiwrap -p 8081

#### 3.2.2. nginx fcgiwrap配置

在nginx.conf中增加下面的location配置块, 这样所有的xxx.cgi请求都会走到fcgiwrap, 然后fcgiwrap会执行cgi-bin目录下的cgi程序。

```
location ~ .*\.cgi$ {  
    root    cgi-bin;  
    fastcgi_pass 127.0.0.1:8081;  
    fastcgi_index index.cgi;  
    include fastcgi.conf;  
}
```

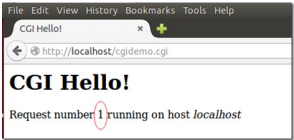
#### 3.2.3. 编写cgi应用程序

```
#include <stdio.h>  
#include <stdlib.h>
```

```
int main(void)  
{  
    int count = 0;  
    printf("Content-type: text/html\r\n"  
        "\r\n"  
        "<title>CGI Hello!</title>"  
  
        "<h1>CGI Hello!</h1>"  
        "Request number %d running on host %s</s>\r\n",  
        ++count, getenv("SERVER_NAME"));;  
    return 0;  
}
```

tyler@ubuntu:~/ClionProjects/HelloFastCGI\$ g++ cgi.cpp -o cgidemo -lfcgi

tyler@ubuntu:~/ClionProjects/HelloFastCGI\$ sudo cp cgidemo /opt/nginx-1.7.7/cgi-bin/



注意图中的请求次数一直都是1, 因为cgi的模式是fork-and-exec, 每次都是一个新的进程。

## 参考链接

- CGI, <http://www.daz.cn/yffaq>
- fastcgi, <http://www.daz.cn/yffaq>
- spawn-fcgi, <http://redmine.lighttpd.net/projects/spawn-fcgi/wiki>
- fcgi, <http://www.fastcgi.com/dropin/node/6?c=node/21>
- fcgiwrap, <https://github.com/gnoesk/fcgiwrap.git>

作者: 吴晨

出处: <http://www.cnblogs.com/skynet/>

本文基于[署名 2.5 中国大陆](#)许可协议发布, 欢迎转载, 演绎或用于商业目的, 但是必须保留本文的署名[吴晨](#)(包含链接)。



吴晨

关注: 18

粉丝: 4040

推荐阅读

+ 加关注

+ 上一篇: Nginx安装与使用


+ 下一篇: PyQGIS应用与实践

11

点赞

0

反对

 登录后才能查看或发表评论 | [立即注册](#) | [找回密码](#) | [帮助](#)

编辑推荐：

- 记一次 .NET 某企业级WEB网站 CPU 爆高事故分析
- 从 Mongo 到 ClickHouse 我到底经历了什么？
- 使用 Three.js 让二维图片获得3D效果
- 福原爱姐：运用 transform 导致文本模糊的视觉错觉
- ASP.NET Core 6框架揭秘(系列演示05)：依赖注入基本编程模式

 百度熊能记

开发者上云优惠专场 云服务器0元/月

最新资讯：

- 与互联网一起急速下沉，那些等待解救的“28岁”
- 字面时代就崩塌？
- 新剧吧搞了个年度盘点，结果被秒得连裤都不剩
- 威马排队 沈晖面瘫 “定档时别”
- CIPPAC新机，没有惊喜
- [更多新闻...](#)