基于动态链接技术的web服务器动态扩展功能接口的设计与实现

黄丛宇
06161032
指导老师：马瑞芳

西安交通大学软件学院软件62班

June 21, 2010

目录

一、背景和意义

背景和意义

Web服务器：

- 互联网的核心组成部分，支撑整个互联网应用服务。
- 适应互联网应用的不断更新变化。
- 必须保证7*24小时的运行。

Web服务器现状：

* 大部分都不支持功能的动态增加。
* 必须重启或重新编译。

- 针对以上问题，本课题将基于动态链接库技术，使服务器在运行期间，可以动态的获知模块的增加并加载模块。
- 本系统实现了服务器的基本功能，重点实现模块动态加载特性。

背景和意义

Web服务器：

- 互联网的核心组成部分，支撑整个互联网应用服务。
- 适应互联网应用的不断更新变化。
- 必须保证7*24小时的运行。

Web服务器现状：

* 大部分都不支持功能的动态增加。
* 必须重启或重新编译。

- 针对以上问题，本课题将基于动态链接库技术，使服务器在运行期间，可以动态的获知模块的增加并加载模块。
- 本系统实现了服务器的基本功能，重点实现模块动态加载特性。

背景和意义

Web服务器：

- 互联网的核心组成部分，支撑整个互联网应用服务。
- 适应互联网应用的不断更新变化。
- 必须保证7*24小时的运行。

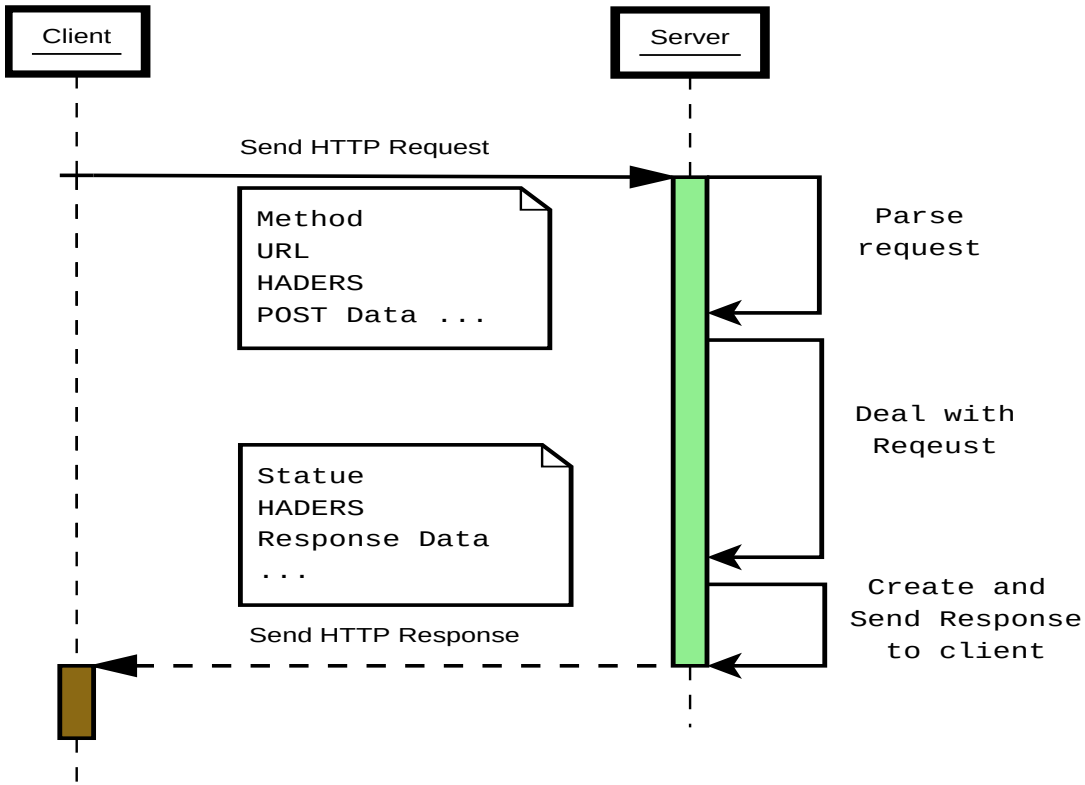
Web服务器现状：

* 大部分都不支持功能的动态增加。
* 必须重启或重新编译。

- 针对以上问题，本课题将基于动态链接库技术，使服务器在运行期间，可以动态的获知模块的增加并加载模块。
- 本系统实现了服务器的基本功能，重点实现模块动态加载特性。

二、动态扩展功能接口和服务器的分析和设计

HTTP协议分析

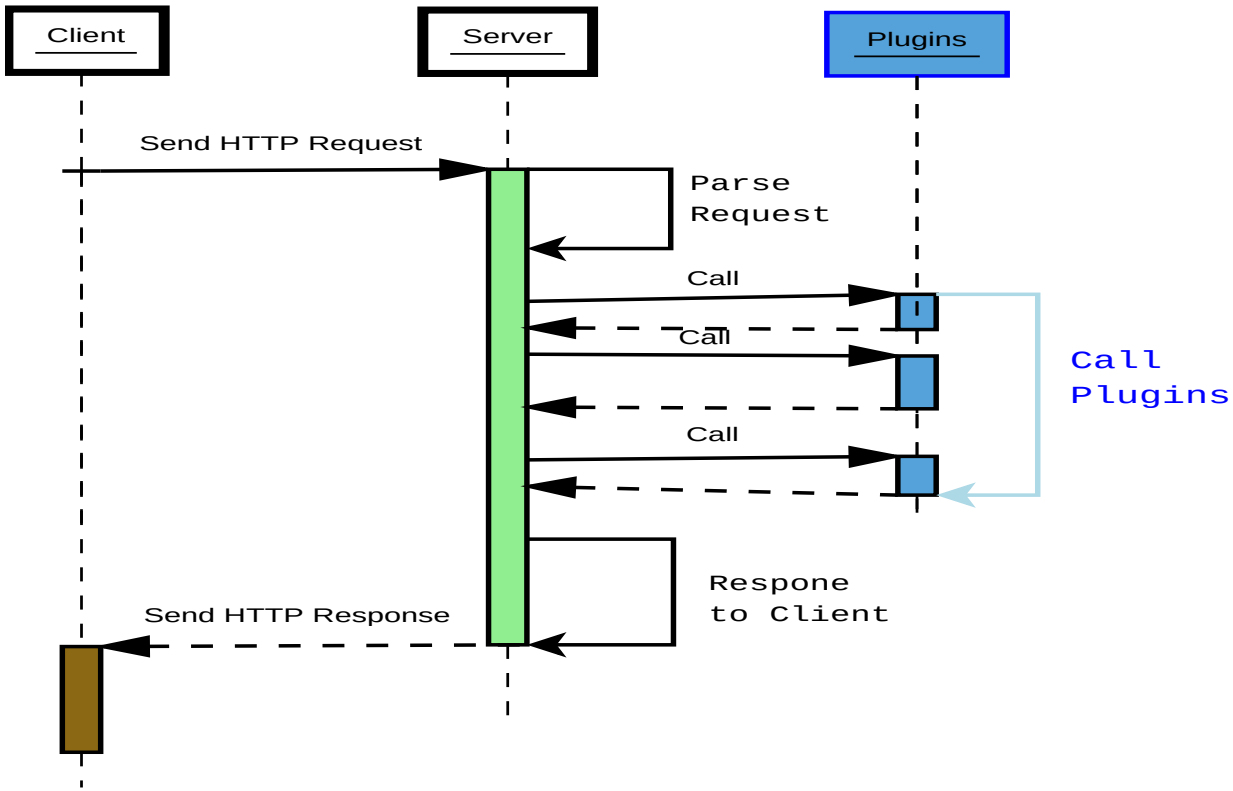


图： HTTP协议处理过程

服务器的处理分为三部分：解析Request，处理请求，返回response。

动态扩展功能接口调用过程的设计

在处理请求的过程中，调用接口函数。



图： 接口调用过程

服务器调用所有功能插件的接口函数。根据函数的返回结果确定执行步骤。

动态扩展功能接口函数设计：

接口定义了一些列函数原型。所有函数都有明确的声明和调用时机。
功能插件选择性的实现这些函数。

接口函数定义：

- `init:` 初始化插件。
- `set_default:` 设置插件的配置为默认值。
- `cleanup:` 清理插件。
- `trigger:` 每秒钟调用一次。相当于计时器。
- `sighup:` 处理挂断信号。

动态扩展功能接口函数设计：

接口定义了一些列函数原型。所有函数都有明确的声明和调用时机。
功能插件选择性的实现这些函数。

接口函数定义：

- `init:` 初始化插件。
- `set_default:` 设置插件的配置为默认值。
- `cleanup:` 清理插件。
- `trigger:` 每秒钟调用一次。相当于计时器。
- `sighup:` 处理挂断信号。

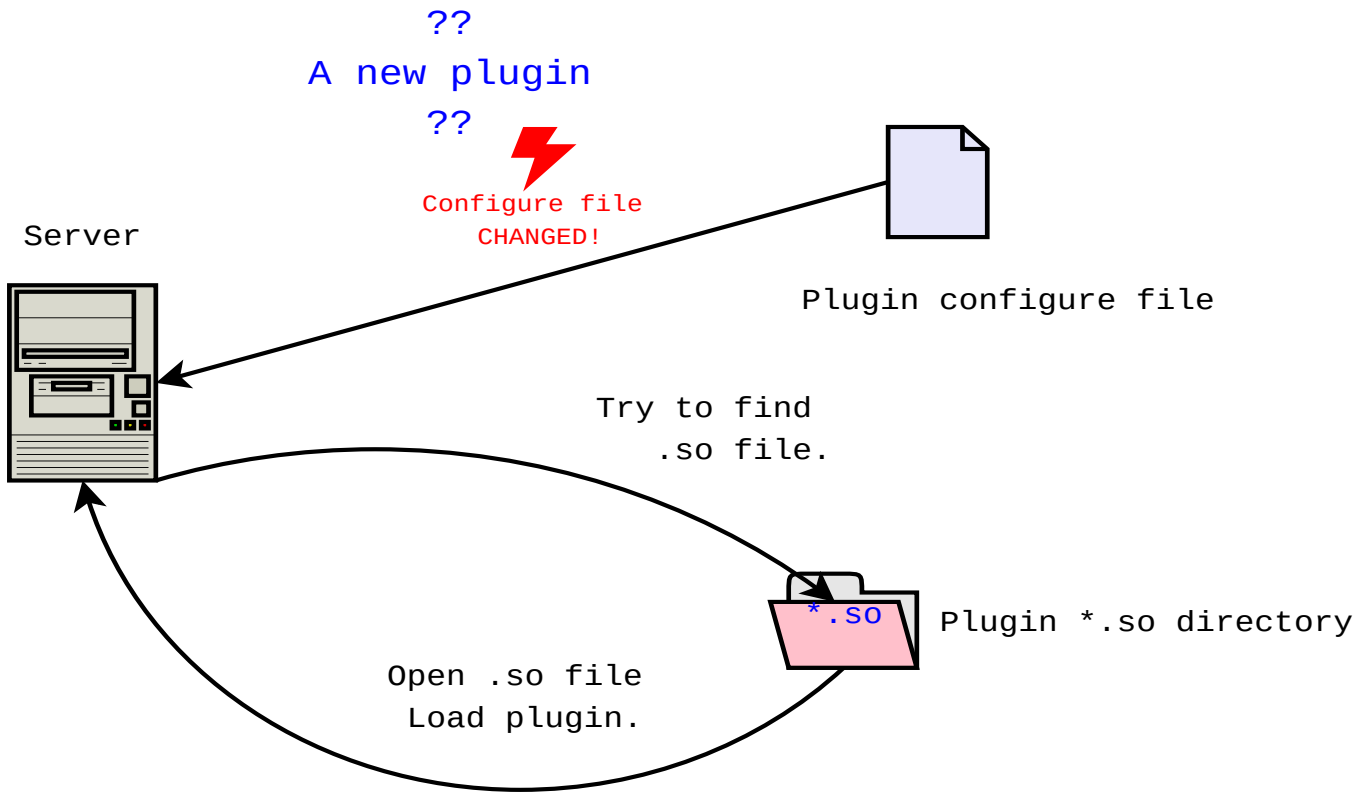
动态扩展功能接口函数设计：

接口函数定义（续）：

- url_raw: 获得未解码的URL地址后调用。
- url_clean: 获得已解码的URL地址后调用。
- docroot: 设置插件工作的根目录。
- physical: 获得请求资源对应的物理地址后调用。
- connection_close: 连接关闭时调用。
- connection_reset: 连接重置时调用。
- joblist: 连接被加入joblist时调用嗯。
- subrequest_start: 子请求开始。
- handle_subrequest: 处理子请求。
- subrequest_end: 子请求结束。

动态加载过程的设计

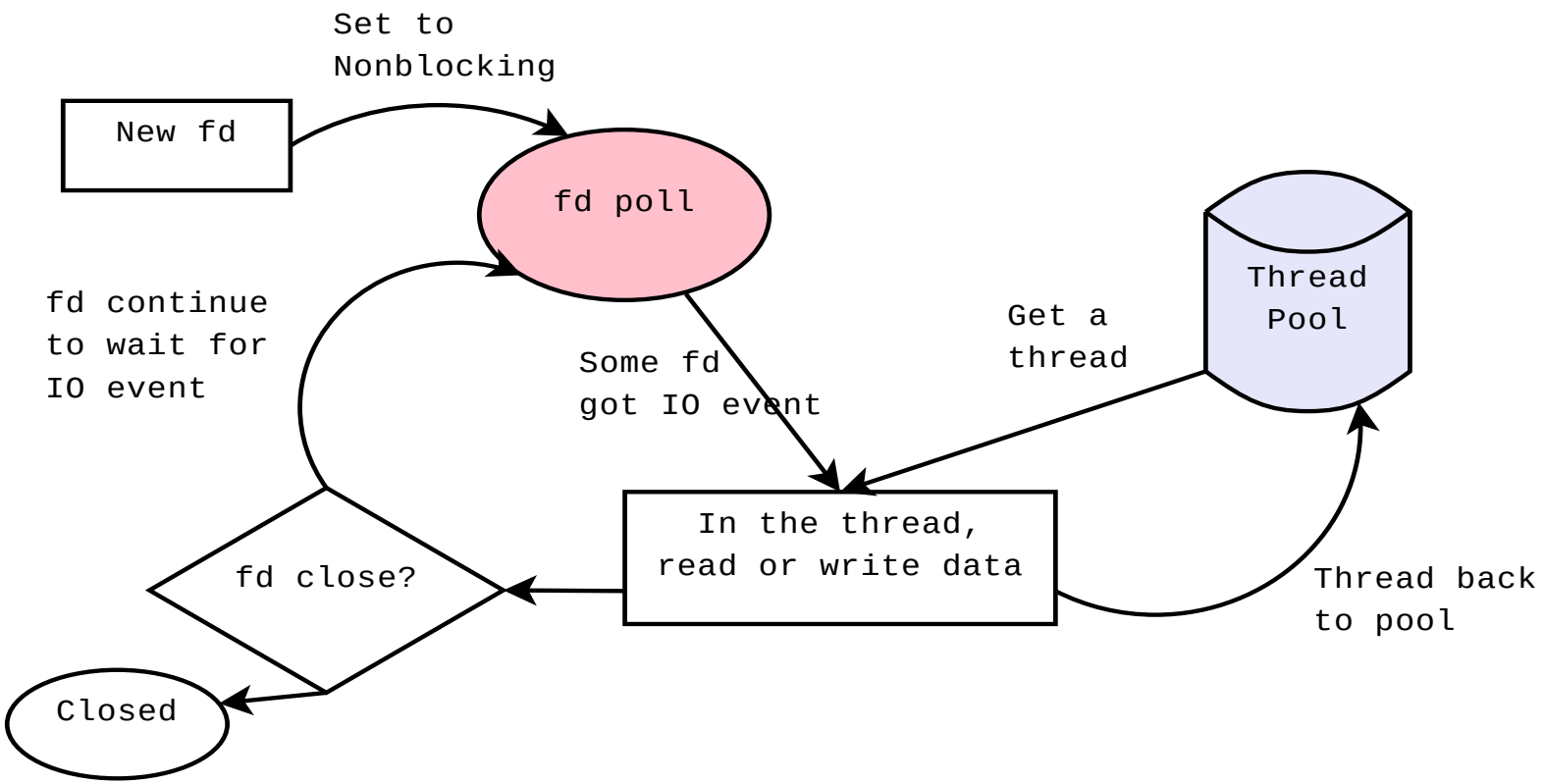
服务器实时的监测插件配置文件是否修改。一旦监测到修改，立即重新加载功能插件。



图： 处理过程

I/O分析和设计

Nonblocking IO + IO multiplexing + Thread pool



图： I/O结构图

状态机和线程池

状态机

使用状态机对连接进行处理。
对连接定义一系列状态。在整个生命周期中，连接处于某一个状态。根据连接的当前状态和所发生的事件，使连接进入下一状态。

线程池

预先创建一部分线程。如果线程不够，再创建新的线程。线程处理完I/O事件不销毁，放入线程池中以备下次继续使用。
降低线程创建和销毁的开销。

状态机和线程池

状态机

使用状态机对连接进行处理。
对连接定义一系列状态。在整个生命周期中，连接处于某一个状态。根据连接的当前状态和所发生的事件，使连接进入下一状态。

线程池

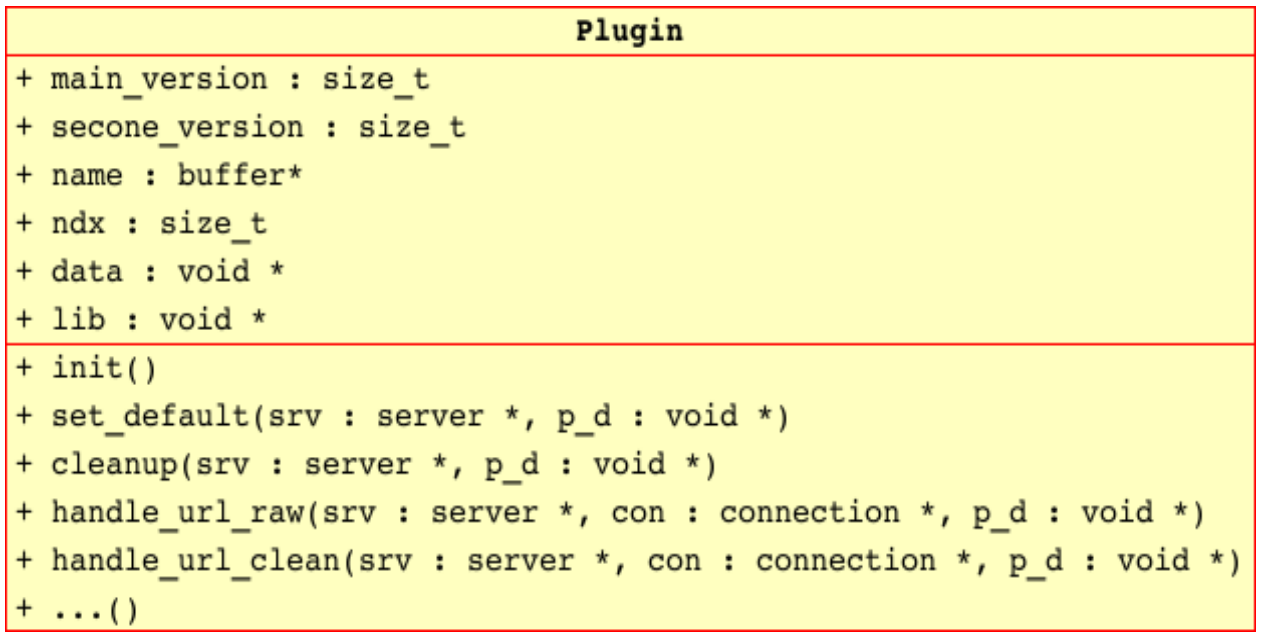
预先创建一部分线程。如果线程不够，再创建新的线程。线程处理完I/O事件不销毁，放入线程池中以备下次继续使用。
降低线程创建和销毁的开销。

三、动态扩展功能接口和服务器的实现

接口实现：plugin结构体

plugin结构体包含一个功能插件的所有信息。包括版本，名称以及一系列接口函数的地址。plugin结构体类似于一个类。服务器通过调用其成员方法（函数指针）来执行插件的功能。

定义如下：



图： plugin结构体

接口实现：plugin_slot数组

plugin_slot是一个二维数组。相当于plugin的登记表。服务器通过这个表中的信息对插件进行调用。表的形式如下：

表：plugin_slot示例：

名称	插件1	插件2	插件3
PLUGIN_SLOT_URL_RAW	p1	p2	p3
PLUGIN_SLOT_URL_CLEAN	p1	NULL	p3
PLUGIN_SLOT_DOCROOT	p1	NULL	NULL

表中存放的是plugin结构体指针。NULL表示此插件没有实现这个函数。服务器通过这张表来调用所有的插件。

接口实现：配置文件

服务器中包含一个定义插件的配置文件：swiftd-plugin.conf。

配置文件的形式如下：

```
#swiftd-plugin.conf #配置文件的形式为 -- 插件名称：插件动态库文件所在目录 $
#每个插件一行，每个插件配置信息必须以"$"结尾。
dir_index:/home/hcy/plugins/$
...
```

动态库文件的名称为： 插件名称 + .so。
服务器通过监测这个配置文件，确定是否有插件需要加载或删除。

接口实现：配置文件

服务器中包含一个定义插件的配置文件：swiftd-plugin.conf。

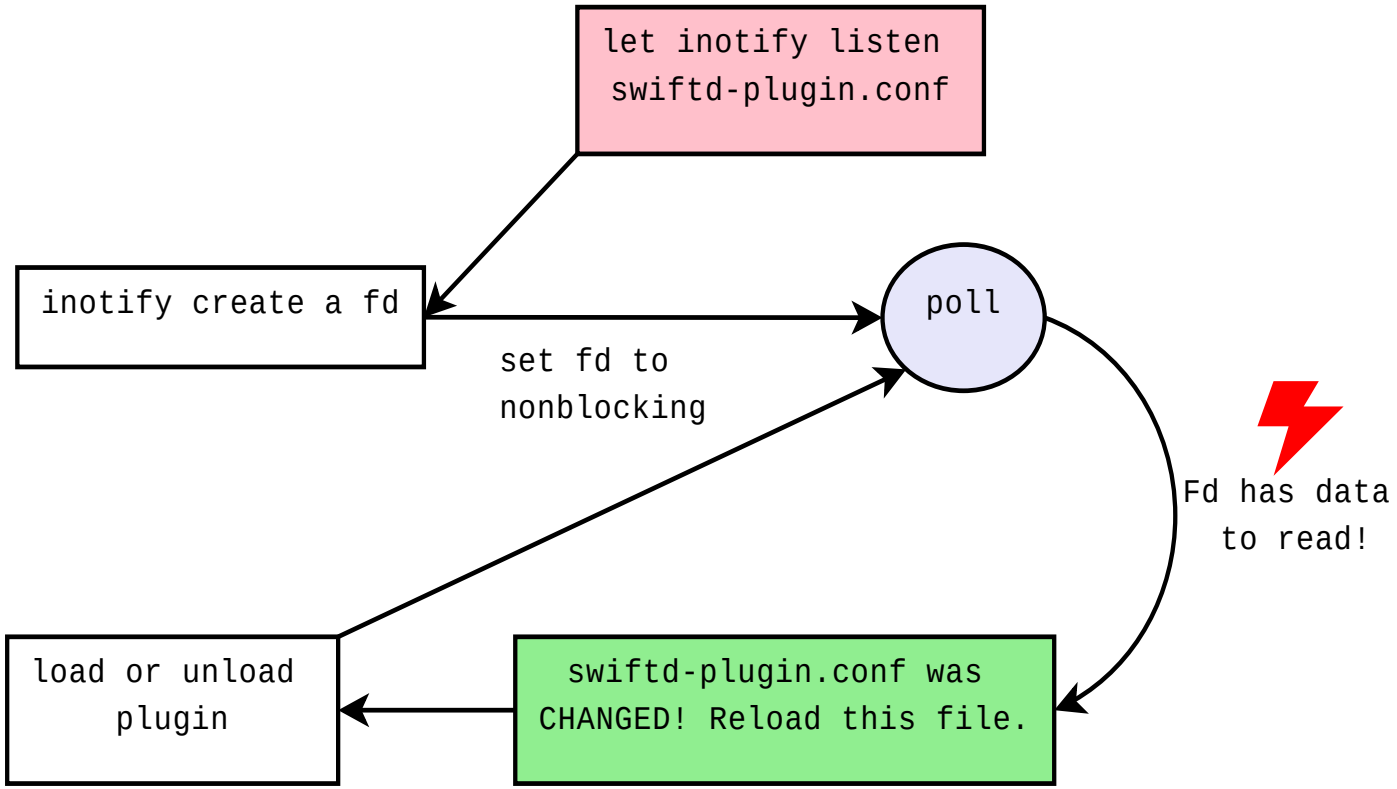
配置文件的形式如下：

```
#swiftd-plugin.conf #配置文件的形式为 -- 插件名称：插件动态库文件所在目录 $
#每个插件一行，每个插件配置信息必须以"$"结尾。
dir_index:/home/hcy/plugins/$
...
```

动态库文件的名称为： 插件名称 + .so。
服务器通过监测这个配置文件，确定是否有插件需要加载或删除。

接口实现：加载过程

服务器使用inotify监测插件配置文件。inotify产生一个文件描述符，通过监测这个文件描述符是否有数据可读即可监测到文件的改变。



图： 监测的过程

服务器实现

I/O

使用epoll对文件描述符的IO事件进行监测。
epoll的效率高于select/poll。

线程池

使用pthread的条件变量对线程进行唤醒和睡眠操作。
每个线程的主循环中，调用传递进来的函数。（包括参数）

状态机

对于每个连接，在其生存周期中，共定义了11个状态。包括connection,requeststart,read,handlerequest,requestend,readpose,responsestart,write,responseend,error和close。

服务器实现

I/O

使用epoll对文件描述符的IO事件进行监测。
epoll的效率高于select/poll。

线程池

使用pthread的条件变量对线程进行唤醒和睡眠操作。
每个线程的主循环中，调用传递进来的函数。（包括参数）

状态机

对于每个连接，在其生存周期中，共定义了11个状态。包括connection,requeststart,read,handlerequest,requestend,readpose,responsestart,write,responseend,error和close。

服务器实现

I/O

使用epoll对文件描述符的IO事件进行监测。
epoll的效率高于select/poll。

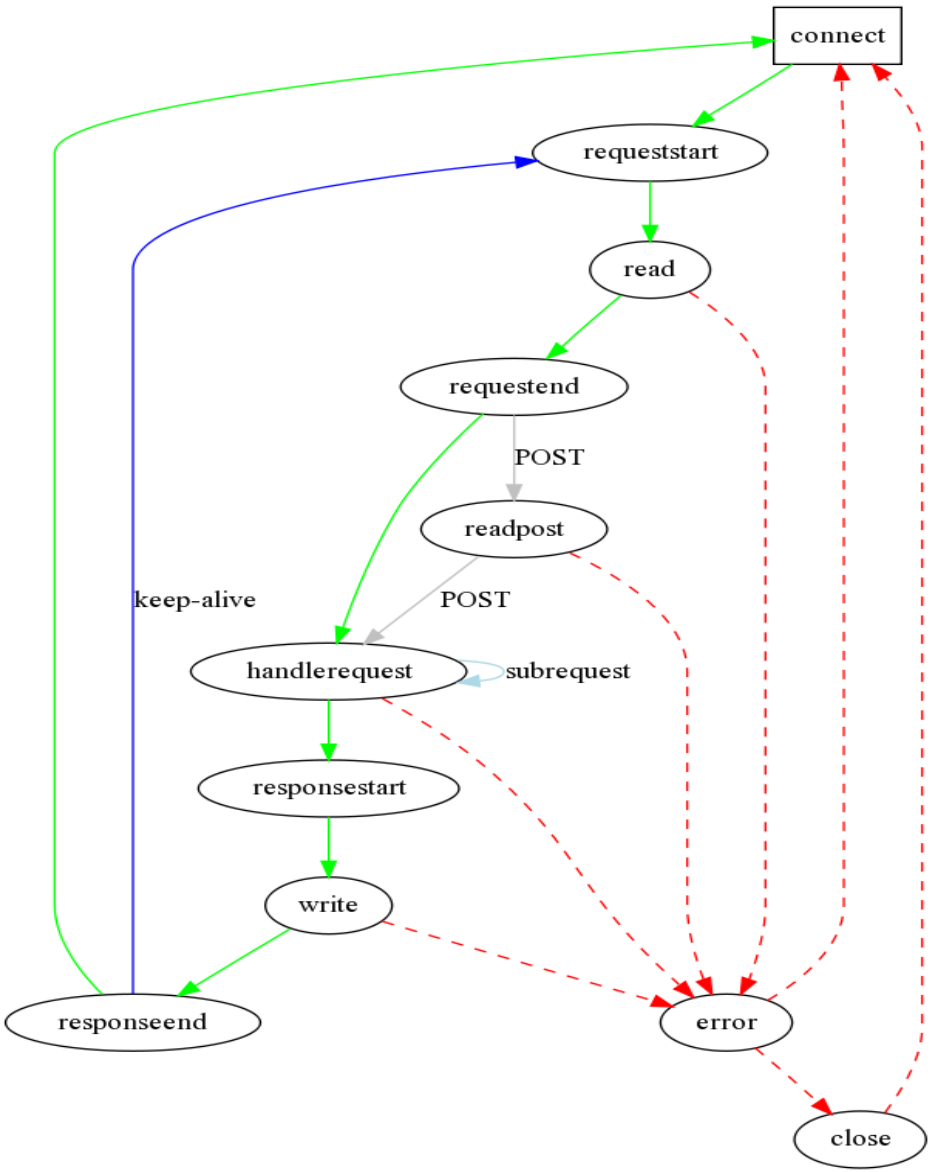
线程池

使用pthread的条件变量对线程进行唤醒和睡眠操作。
每个线程的主循环中，调用传递进来的函数。（包括参数）

状态机

对于每个连接，在其生存周期中，共定义了11个状态。包括connection,requeststart,read,handlerequest, requestend,readpose,responsestart,write,responseend, error和close。

连接处理状态机



绿色路径为一个正常的处理路径。

红色路径为出错路径。

蓝色路径，HTTP/1.1中保持连接。

四、运行结果

启动和访问

启动：

```
hcy@xjtuacm-desktop:~/src/swiftd/src$ ./swiftd -D
starting server...
close stdin and stdout.
set defaults configure.
open log.
init the network.
hcy@xjtuacm-desktop:~/src/swiftd/src$ starting thread pool.
starting fdevent system.
load plugins.
start server. OK!

hcy@xjtuacm-desktop:~/src/swiftd/src$
```

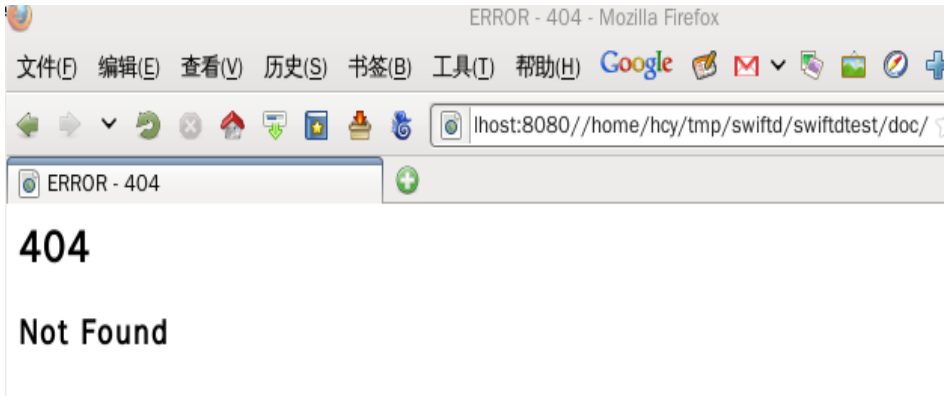
正常访问

运行结果

注释掉插件配置：

服务器不需要重启！

访问失败：

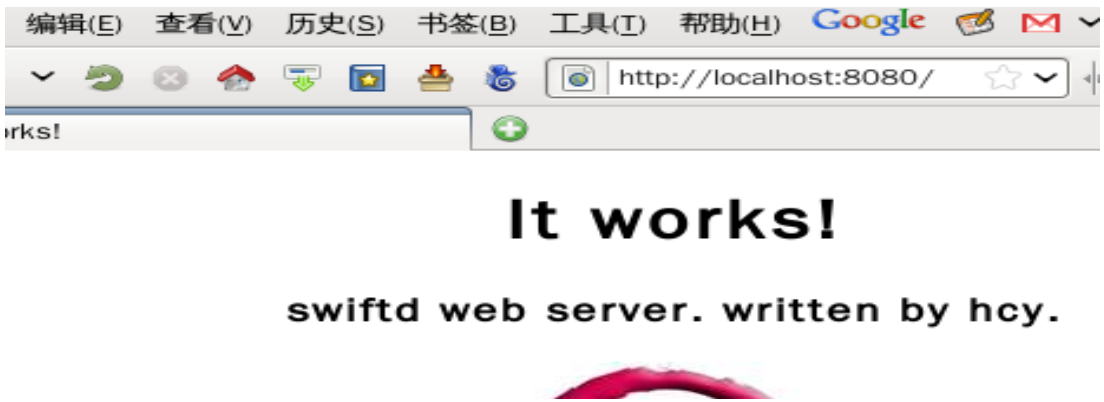


运行结果

增加插件配置：

服务器同样不需要重启！

访问成功：



五、总结

运行结果

1. 阅读关于HTTP协议的相关资料，理解了HTTP协议的处理过程。
2. 学习了在Linux下进行开发。熟悉了linux，gcc，gdb，vim等的使用。
3. 动态加载功能接口达到了课题的目标要求。可以正确的进行动态加载和卸载功能插件。
4. 服务器可以正确处理HTTP请求。包括HTTP/1.1和HTTP/1.0。能够正确的调用插件处理请求。
5. 总共完成11000行C代码的编写和调试。

结束

That's all!