超高性能 Web 服务器(htmlserver)

厉华

版本修订

文档版本号	修订日期	修订人	修订内容
v1.0.0	2016-08-08	厉华	创建
v1.0.1	2016-08-17	厉华	新增章节 压测
v1.0.2	2016-08-20	厉华	修改章节 虚拟主机 新增章节 内部实现

目录索引

1 概述 4

2 编译安装 5

2.1 编译 5

1.1 安装 5

1.2 用缺省配置第一次启动 5

3 配置文件 6

1.3 配置项列表及说明 6

1.4 虚拟主机 10

4 服务器管理 12

1.5 用自带脚本管理 12

4.1 直接用命令管理 13

5 压测 14

5.1 压测环境 14

5.2 压测方案 15

5.3 压测过程 15

5.4 压测结果 23

6 内部实现 26

6.1 系统结构 26

6.2 函数调用关系图 27

6.2.1 启动与初始化 27

6.2.2 管理进程 27

6.2.3 工作进程 28

1 概述

htmlserver(简称 HS)是一款国人原创研发的开放源代码的 C 语言实现的支持高并发、超高性能 Web 服务器，使用高性能 HTTP 解析器 fasterhttp 作为其解析核心，在开启 Keep-Alive 和 gip 压缩时性能秒杀（5 倍于）nginx。如此高性能得益于轻巧的架构设计和采用 Inotify 文件变化主动通知缓存机制，把大量静态文件尽可能缓存在内存直接读取，比传统的轮询式检查文件机制避免了大量存储 IO 操作。

htmlserver 的设计理念是快速、稳定和小巧。没有完全采用 apache 或 nginx 纯模块化架构，因为大多数人使用 webserver 一般都会把所有模块都打上，除了动态内容模块（如 mod_php），很少见到有人特意去组装模块，那还不如直接全部编译在一起算了，使用简单，避免了管理员或运维人员面对过多选择带来的学习成本。当你需要本地定制化时，直接改代码吧，因为它就是开源的嘛。htmlserver 只有在动态内容上才设计了模块接口，以适应各种各样的语言架构和开发者。

htmlserver 目前只支持 GET 和 HEAD 方法，将来很快会支持 POST 动态网页。

htmlserver 目前只支持 Linux，将来很快会支持 WINDOWS。

htmlserver 功能：

- 支持 HTTP/1.0、HTTP/1.1
- 支持通讯超时控制
- 支持多侦听端口
- 支持多虚拟主机（基于域名）
- 支持自定义错误页面
- 支持自定义缺省 index 文件
- 支持自适应 Keep-Alive
- 支持自适应 gzip、deflate 压缩
- 支持工作进程绑定 CPU
- 支持工作进程崩溃后，管理进程自动重启工作进程

• 支持优雅重启/重载配置，重启期间完全不中断对外服务

2 编译安装

2.1 编译

从 http://git.oschina.net/calvinwilliams/htmlserver 或 https://github.com/calvinwilliams/htmlserver 上 git clone 或直接下载 zip 包到本地解开，进入 src 目录，执行编译命令，Linux 环境构造文件为 makefile.Linux

```
$ make -f makefile.Linux
```

没有报错的话就能编译出可执行文件 htmlserver。

1.1 安装

（以下为安装到系统用户中，如果要安装到其它目录，请自行调整命令）

手工复制 htmlserver 到可执行文件目录$HOME/bin/。

手工复制安装包中的 conf/htmlserver.conf 到你的配置文件目录，如 $HOME/etc/，作为缺省配置。

手工复制安装包中的 www/* 到你的网站根目录，如 $HOME/www/，error_pages 目录放着缺省的报错页面，index.html 是示例首页文件。

手工复制安装包中的 shbin/hs.do 到$HOME/shbin/，这个脚本用于 htmlserver 启停管理。

这样就安装好了!

1.2 用缺省配置第一次启动

执行以下命令以缺省配置启动

```
$ htmlserver ~/etc/htmlserver.conf
```

如果没有产生输出、没有产生~/log/error.log 以及该日志中没有出现 ERROR 行的话表示启动成功。注意：缺省配置文件中的侦听端口为 9527。

可以看到进程，htmlserver 进程结构由一个管理进程+n 个工作进程组成

```
$ ps -ef | grep htmlserver | grep -v grep
calvin      14122         1    0 23:17 ? 00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin      14123 14122    0 23:17 ? 00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
```

以及侦听端口

```
$ netstat -an | grep 9527
tcp        0          0 0.0.0.0:9527        0.0.0.0:*             LISTEN
```

自测一下

```
$ curl http://127.0.0.1:9527/index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb18030" />
<title>Welcome</title>
</head>
<body>
Hello htmlserver
</body>
</html>
```

恭喜您，启动成功！

直接发送 TERM 信号到父进程可停止 htmlserver

```
$ kill 14122
```

3 配置文件

1.3 配置项列表及说明

安装时复制的配置文件为缺省配置

```
$ cat ~/etc/htmlserver.conf
{
        "worker_processes" : 1 ,
        "cpu_affinity" : 1 ,
        "accept_mutex" : 1 ,

        "error_log" : "$HOME$/log/error.log" ,
        "log_level" : ERROR ,
```

```
"limits" :
{
        "max_http_session_count" : 100000
},

"listen" :
{
        "ip" : "" ,
        "port" : "9527"
},

"server" :
{
        "domain" : "" ,
        "wwwroot" : "$HOME$/www" ,
        "index" : "/index.html,/index.htm" ,
        "access_log" : "$HOME$/log/access.log"
},
"tcp_options" :
{
        "nodelay" : 1 ,
        "nolinger" : -1
},

"http_options" :
{
        "compress_on" : 1 ,
        "connection_timeout" : 10 ,
        "timeout" : 60
},

"error_pages" :
{
        "error_page_400" : "$HOME$/www/error_pages/error_page_400.html" ,
        "error_page_401" : "$HOME$/www/error_pages/error_page_401.html" ,
        "error_page_403" : "$HOME$/www/error_pages/error_page_403.html" ,
        "error_page_404" : "$HOME$/www/error_pages/error_page_404.html" ,
        "error_page_408" : "$HOME$/www/error_pages/error_page_408.html" ,
        "error_page_500" : "$HOME$/www/error_pages/error_page_500.html" ,
        "error_page_503" : "$HOME$/www/error_pages/error_page_503.html" ,
        "error_page_505" : "$HOME$/www/error_pages/error_page_505.html"
},
```

```
    "mime_types" :
    {
        "mime_type" : { "type":"html htm shtml" , "mime":"text/html" } ,
        "mime_type" : { "type":"css" , "mime":"text/css" } ,
        "mime_type" : { "type":"xml" , "mime":"text/xml" } ,
        "mime_type" : { "type":"txt" , "mime":"text/plain" } ,
        "mime_type" : { "type":"gif" , "mime":"image/gif" } ,
        "mime_type" : { "type":"jpeg jpg" , "mime":"image/jpeg" } ,
        "mime_type" : { "type":"png" , "mime":"image/png" } ,
        "mime_type" : { "type":"tif tiff" , "mime":"image/tiff" } ,
        "mime_type" : { "type":"ico" , "mime":"image/x-ico" } ,
        "mime_type" : { "type":"jng" , "mime":"image/x-jng" } ,
        "mime_type" : { "type":"bmp" , "mime":"image/x-ms-bmp" } ,
        "mime_type" : { "type":"svg svgz" , "mime":"image/svg+xml" } ,
        "mime_type" : { "type":"jar war ear" , "mime":"application/java-archive" } ,
        "mime_type" : { "type":"json" , "mime":"application/json" } ,
        "mime_type" : { "type":"doc" , "mime":"application/msword" } ,
        "mime_type" : { "type":"pdf" , "mime":"application/pdf" } ,
        "mime_type" : { "type":"rtf" , "mime":"application/rtf" } ,
        "mime_type" : { "type":"xls" , "mime":"application/vnd.ms-excel" } ,
        "mime_type" : { "type":"ppt" , "mime":"application/vnd.ms-powerpoint" } ,
        "mime_type" : { "type":"7z" , "mime":"application/x-7z-compressed" } ,
        "mime_type" : { "type":"rar" , "mime":"application/x-rar-compressed" } ,
        "mime_type" : { "type":"swf" , "mime":"application/x-shockwave-flash" } ,
        "mime_type" : { "type":"xhtml" , "mime":"application/xhtml+xml" } ,
        "mime_type" : { "type":"bin exe dll iso img msi msp msm" ,
"mime":"application/octet-stream" } ,
        "mime_type" : { "type":"zip" , "mime":"application/zip" } ,
        "mime_type" : { "type":"docx" ,
"mime":"application/vnd.openxmlformats-officedocument.wordprocessingml.document" } ,
        "mime_type" : { "type":"xlsx" ,
"mime":"application/vnd.openxmlformats-officedocument.spreadsheetml.sheet" } ,
        "mime_type" : { "type":"pptx" ,
"mime":"application/vnd.openxmlformats-officedocument.presentationml.presentation" } ,
        "mime_type" : { "type":"mid midi kar" , "mime":"audio/midi" } ,
        "mime_type" : { "type":"mp3" , "mime":"audio/mpeg" } ,
        "mime_type" : { "type":"ogg" , "mime":"audio/ogg" } ,
        "mime_type" : { "type":"m4a" , "mime":"audio/x-m4a" } ,
        "mime_type" : { "type":"ra" , "mime":"audio/x-realaudio" } ,
        "mime_type" : { "type":"3gpp 3gp" , "mime":"video/3gpp" } ,
        "mime_type" : { "type":"ts" , "mime":"video/mp2t" } ,
        "mime_type" : { "type":"mp4" , "mime":"video/mp4" } ,
        "mime_type" : { "type":"mpeg mpg" , "mime":"video/mpeg" } ,
        "mime_type" : { "type":"mov" , "mime":"video/quicktime" } ,
```

```
        "mime_type" : { "type":"webm" , "mime":"video/webm" } ,
        "mime_type" : { "type":"flv" , "mime":"video/x-flv" } ,
        "mime_type" : { "type":"m4v" , "mime":"video/x-m4v" } ,
        "mime_type" : { "type":"mng" , "mime":"video/x-mng" } ,
        "mime_type" : { "type":"asx asf" , "mime":"video/x-ms-asf" } ,
        "mime_type" : { "type":"wmv" , "mime":"video/x-ms-wmv" } ,
        "mime_type" : { "type":"avi" , "mime":"video/x-msvideo" }
    }
}
```

htmlserver 配置文件格式为 json。各配置项说明如下：

worker_processes　子进程数量。因为内部采用了多路复用，一般情况下一个子进程足矣。如果设置成-1，则创建与 CPU 核数量相等的子进程。

cpu_affinity　如果为 1，则子进程绑定在 CPU 上，如果为 0，不绑定。

accept_mutex　如果为 1，开启 accept 锁，防止多子进程因 epoll 惊群而引起的 CPU 稍稍高耗。

error_log　详细日志文件名。支持$...$环境变量展开。以下所有目录文件配置项都可以内嵌环境变量。

log_level　详细日志文件内的日志等级

limits　限制配置

　　max_http_session_count 最大 HTTP 通讯会话并发数量

listen　侦听配置

　　ip　本地侦听端口，不填则为 0.0.0.0

　　port　本地侦听端口，如果有多个端口，则格式为"port1,port2,..."。注意：前后有双引号。

server　主站点配置

　　domain　HTTP 头选项 Host 里的内容，用于区分虚拟主机。

　　wwwroot　网站根目录。

　　index　当浏览器请求的是目录，尝试的入口文件，格式为"/index.html"，如果有多个，则格式为"/index.html,/index.htm,..."。注意：入口文件名前有"/"。

　　access_log　事件日志文件名

servers　虚拟主机站点设置

server　　主站点配置

domain　　HTTP 头选项 Host 的值，用于区分虚拟主机。

wwwroot　　网站根目录。

index　　当浏览器请求的是目录，尝试的入口文件，格式为"/index.html"，如果有多个，则格式为"/index.html,/index.htm,..."。注意：入口文件名前有"/"。

access_log　事件日志文件名

tcp_options　　TCP 选项

nodelay　　当为 1 时，启用 TCP 选项 TCP_NODELAY，有助于提高响应速度；当为 0 时，关闭之。

nolinger　　当大于等于 0 时，启用 TCP 选项 SO_LINGER 并设置成其值；当为-1 时，不设置之。

http_options　　HTTP 选项

compress_on　启用服务端压缩，有助于大幅减少通讯数量量。

timeout　　HTTP 超时时间

error_pages　　出错页面配置

error_page_???　HTTP 响应???时返回的页面文件。目前支持的有 400、401、403、404、408、500、503、505。

mime_types　　流类型配置集合

mime_type　　流类型配置

type　　文件扩展名

mime　　HTTP 响应头 Content-Type 值

1.4 虚拟主机

server.domain 或 servers.server[].domain 需要匹配浏览器访问 Web 服务器请求头选项 Host 的值（URL 中"http://"与"/"之间部分）以确定服务器使用哪个虚拟主机来响应，如：

http://www.google.com/　　domain 为"www.google.com"

http://192.168.1.110:8080/　domain 为"192.168.1.110:8080"

当你只有一个站点 www.a.com 时，可以这样配置以准确匹配域名 a.com：

```
"server" :
{
    "domain" : "www.a.com" ,
    ...
},
/*
"servers" :
{
*/
```

当你有多个站点时，可以这样配置以准确匹配域名：

```
/*
"server" :
{
    ...
},
*/
"servers" :
{
    server
    {
        "domain" : "www.a.com" ,
        ...
    }
    server
    {
        "domain" : "www.a.com" ,
        ...
    }
```

```
    server
    {
        "domain" : "www.c.com" ,
        ...
    }
},
```

你也可以这样配置以不匹配域名，指向一个缺省的虚拟主机：

```
"server" :
{
    "domain" : "" ,
    ...
},
/*
"servers" :
{
*/
```

4 服务器管理

1.5 用自带脚本管理

启动 htmlserver

```
$ hs.do start
htmlserver start ok
calvin    14703     1  0 00:05 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin    14704 14703  0 00:05 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
```

查询 htmlserver 进程

```
$ hs.do status
calvin    14703     1  0 00:05 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin    14704 14703  0 00:05 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
```

重启 htmlserver

```
$ hs.do restart
calvin     14703        1    0 00:05 ?     00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14704 14703    0 00:05 ?     00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
htmlserver end ok
htmlserver start ok
calvin     14761        1    0 00:06 ?     00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14762 14761    0 00:06 ?     00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
```

优雅的重启 htmlserver，或者重载配置文件

```
$ hs.do restart_graceful
calvin     14761        1    0 00:06 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14762 14761    0 00:06 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
new htmlserver pid[14796] start ok
old htmlserver pid[14761] end ok
calvin     14796        1    0 00:06 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14797 14796    0 00:06 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
```

向 htmlserver 发送重新打开日志文件信号

```
$ hs.do relog
calvin     14796        1    0 00:06 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14797 14796    0 00:06 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
send signal to htmlserver for reopenning log
```

停止 htmlserver

```
$ hs.do stop
calvin     14796        1    0 00:06 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14797 14796    0 00:06 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
htmlserver end ok
```

4.1 直接用命令管理

启动 htmlserver

```
$ htmlserver ~/etc/htmlserver.conf
```

查询 htmlserver 进程

```
$ ps -ef | grep htmlserver | grep -v grep
calvin     14876        1    0 00:10 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14877 14876    0 00:10 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
```

优雅的重启 htmlserver，或者重载配置文件

```
$ ps -ef | grep htmlserver | grep -v grep
calvin     14876        1    0 00:10 ?     00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14877 14876    0 00:10 ?     00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
$ kill -USR2 14876
```

```
$    ps -ef | grep htmlserver | grep -v grep
calvin     14876        1    0 00:10 ?      00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14877 14876    0 00:10 ?      00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14889        1    0 00:12 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14890 14889    0 00:12 ?      00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
$ kill 14876
$ ps -ef | grep htmlserver | grep -v grep
calvin     14889        1    0 00:12 ?    00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
calvin     14890 14889    0 00:12 ?      00:00:00 htmlserver /home/calvin/etc/htmlserver.conf
```

向 htmlserver 发送重新打开日志文件信号

```
$ kill -USR1 14889
```

停止 htmlserver

```
$ kill 14889
```

5 压测

5.1 压测环境

拿两款现在最流行的 Web 服务器软件做横向压测，采用版本如下：

nginx/1.9.13

Apache/2.2.3

htmlserver 版本为

htmlserver/1.0.0

压测客户端采用 Apache 自带工具 ab

压测发起端为笔记本电脑，配置如下：

CPU　: Intel Core i5-3320 2.60GHz 2.60GHz

内存 : 1GB

RedHat Enterprise Linux Server release 5.4 (32BITS)

压测网络为百兆有线

压测服务端为台式 PC，配置如下：

CPU　: Intel Core i3-3240 3.40GHz 3.40GHz

内存 ：512MB

RedHat Enterprise Linux Server release 5.4 (32BITS)

5.2 压测方案

方案 A

考察较大量 HTTP 短连接、小网页文件处理性能

并发 1000，共发起 HTTP 请求 10 万次，目标网页文件大小 2B

命令：ab -c 1000 -n 100000 http://192.168.6.17:9527/press2.html

方案 B

考察较大量 HTTP 长连接 Keep-Alive、中型网页、gzip 压缩处理性能

并发 1000，共发起 HTTP 请求 20 万次，目标网页文件大小约 4.3KB

命令：ab -kc 1000 -n 200000 -H "Accept-Encoding: gzip" http://192.168.6.17:9527/press.html

5.3 压测过程

先方案 A，先交替的各压一次热热身（可以预览一下性能）

```
$ ab -c 1000 -n 100000 http://192.168.6.17:9527/press2.html
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.6.17 (be patient)
Completed 10000 requests
Completed 20000 requests
Completed 30000 requests
Completed 40000 requests
Completed 50000 requests
Completed 60000 requests
Completed 70000 requests
Completed 80000 requests
Completed 90000 requests
Completed 100000 requests
Finished 100000 requests

Server Software:        htmlserver/1.0.0
```

```
Server Hostname:        192.168.6.17
Server Port:            9527

Document Path:          /press2.html
Document Length:        2 bytes

Concurrency Level:      1000
Time taken for tests:   18.324 seconds
Complete requests:      100000
Failed requests:        0
Write errors:           0
Total transferred:      9100273 bytes
HTML transferred:       200006 bytes
Requests per second:    5457.36 [#/sec] (mean)
Time per request:       183.239 [ms] (mean)
Time per request:       0.183 [ms] (mean, across all concurrent requests)
Transfer rate:          484.99 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   56 473.9      4    9018
Processing:     1   93 657.6     17    9043
Waiting:        0   91 657.5     15    9043
Total:          5  149 859.2     21    9099

Percentage of the requests served within a certain time (ms)
  50%     21
  66%     23
  75%     24
  80%     25
  90%     29
  95%     42
  98%   3037
  99%   3064
 100%   9099 (longest request)


$ ab -c 1000 -n 100000 http://192.168.6.17:9528/press2.html
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.6.17 (be patient)
Completed 10000 requests
```

```
Completed 20000 requests
Completed 30000 requests
Completed 40000 requests
Completed 50000 requests
Completed 60000 requests
Completed 70000 requests
Completed 80000 requests
Completed 90000 requests
Completed 100000 requests
Finished 100000 requests

Server Software:        nginx/1.9.13
Server Hostname:        192.168.6.17
Server Port:            9528

Document Path:          /press2.html
Document Length:        2 bytes

Concurrency Level:      1000
Time taken for tests:   21.201 seconds
Complete requests:      100000
Failed requests:        0
Write errors:           0
Total transferred:      23106466 bytes
HTML transferred:       200054 bytes
Requests per second:    4716.70 [#/sec] (mean)
Time per request:       212.013 [ms] (mean)
Time per request:       0.212 [ms] (mean, across all concurrent requests)
Transfer rate:          1064.32 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   68 846.2      4   21035
Processing:     1  104 844.4     16   21117
Waiting:        1  103 844.0     15   21105
Total:          6  172 1297.6    20   21170

Percentage of the requests served within a certain time (ms)
  50%     20
  66%     23
  75%     25
  80%     26
  90%     31
  95%     39
```

```
  98%      3019
  99%      9006
 100%    21170 (longest request)

$ ab -c 1000 -n 100000 http://192.168.6.17:9529/press2.html
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.6.17 (be patient)
Completed 10000 requests
Completed 20000 requests
Completed 30000 requests
Completed 40000 requests
Completed 50000 requests
Completed 60000 requests
Completed 70000 requests
Completed 80000 requests
Completed 90000 requests
Completed 100000 requests
Finished 100000 requests

Server Software:        Apache/2.2.3
Server Hostname:        192.168.6.17
Server Port:            9529

Document Path:          /press2.html
Document Length:        2 bytes

Concurrency Level:      1000
Time taken for tests:   45.163 seconds
Complete requests:      100000
Failed requests:        0
Write errors:           0
Total transferred:      26200262 bytes
HTML transferred:       200002 bytes
Requests per second:    2214.20 [#/sec] (mean)
Time per request:       451.631 [ms] (mean)
Time per request:       0.452 [ms] (mean, across all concurrent requests)
Transfer rate:          566.53 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   48 612.5      1   21024
```

```
Processing:         1    255 2494.2         26      45074
Waiting:            1    255 2494.2         25      45074
Total:             16    303 2602.5         27      45158

Percentage of the requests served within a certain time (ms)
  50%         27
  66%         30
  75%         32
  80%         33
  90%         42
  95%         54
  98%       3026
  99%       9022
 100%      45158 (longest request)
```

然后交替压 8 次，取“Requests per second”的值

再方案 B，先交替的各压一次热热身（可以预览一下性能）

```
$ ab -kc 1000 -n 200000 -H "Accept-Encoding: gzip" http://192.168.6.17:9527/press.html
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.6.17 (be patient)
Completed 20000 requests
Completed 40000 requests
Completed 60000 requests
Completed 80000 requests
Completed 100000 requests
Completed 120000 requests
Completed 140000 requests
Completed 160000 requests
Completed 180000 requests
Completed 200000 requests
Finished 200000 requests

Server Software:        htmlserver/1.0.0
Server Hostname:        192.168.6.17
Server Port:            9527

Document Path:          /press.html
Document Length:        283 bytes

Concurrency Level:      1000
Time taken for tests:   10.543 seconds
```

```
Complete requests:      200000
Failed requests:        0
Write errors:           0
Keep-Alive requests:    200000
Total transferred:      84402532 bytes
HTML transferred:       56601698 bytes
Requests per second:    18969.70 [#/sec] (mean)
Time per request:       52.716 [ms] (mean)
Time per request:       0.053 [ms] (mean, across all concurrent requests)
Transfer rate:          7817.82 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    5 115.9      0    3007
Processing:     9   48 161.5     49    9066
Waiting:        9   48 161.5     49    9066
Total:          9   52 216.9     49    9124

Percentage of the requests served within a certain time (ms)
  50%     49
  66%     51
  75%     52
  80%     54
  90%     58
  95%     75
  98%     81
  99%     84
 100%   9124 (longest request)

$ ab -kc 1000 -n 200000 -H "Accept-Encoding: gzip" http://192.168.6.17:9528/press.html
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.6.17 (be patient)
Completed 20000 requests
Completed 40000 requests
Completed 60000 requests
Completed 80000 requests
Completed 100000 requests
Completed 120000 requests
Completed 140000 requests
Completed 160000 requests
Completed 180000 requests
```

```
Completed 200000 requests
Finished 200000 requests

Server Software:        nginx/1.9.13
Server Hostname:        192.168.6.17
Server Port:            9528

Document Path:          /press.html
Document Length:        308 bytes

Concurrency Level:      1000
Time taken for tests:   42.928 seconds
Complete requests:      200000
Failed requests:        0
Write errors:           0
Keep-Alive requests:    0
Total transferred:      105000525 bytes
HTML transferred:       61600308 bytes
Requests per second:    4659.01 [#/sec] (mean)
Time per request:       214.638 [ms] (mean)
Time per request:       0.215 [ms] (mean, across all concurrent requests)
Transfer rate:          2388.66 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   62 711.5      2   21025
Processing:     1  110 861.9     23   21038
Waiting:        1  109 861.9     22   21038
Total:          5  172 1225.1    25   21109

Percentage of the requests served within a certain time (ms)
  50%     25
  66%     26
  75%     27
  80%     27
  90%     30
  95%     38
  98%   3026
  99%   3051
 100%  21109 (longest request)

$ ab -kc 1000 -n 200000 -H "Accept-Encoding: gzip" http://192.168.6.17:9529/press.html
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
```

```
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.6.17 (be patient)
Completed 20000 requests
Completed 40000 requests
Completed 60000 requests
Completed 80000 requests
Completed 100000 requests
Completed 120000 requests
Completed 140000 requests
Completed 160000 requests
Completed 180000 requests
Completed 200000 requests
Finished 200000 requests

Server Software:        Apache/2.2.3
Server Hostname:        192.168.6.17
Server Port:            9529

Document Path:          /press.html
Document Length:        4301 bytes

Concurrency Level:      1000
Time taken for tests:   84.140 seconds
Complete requests:      200000
Failed requests:        0
Write errors:           0
Keep-Alive requests:    0
Total transferred:      913448180 bytes
HTML transferred:       860242062 bytes
Requests per second:    2376.98 [#/sec] (mean)
Time per request:       420.702 [ms] (mean)
Time per request:       0.421 [ms] (mean, across all concurrent requests)
Transfer rate:          10601.79 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0  128 778.9     13   55204
Processing:     2  262 1513.2    71   66072
Waiting:        1  232 1454.4    58   54037
Total:         30  390 1745.0    84   66119

Percentage of the requests served within a certain time (ms)
  50%     84
```

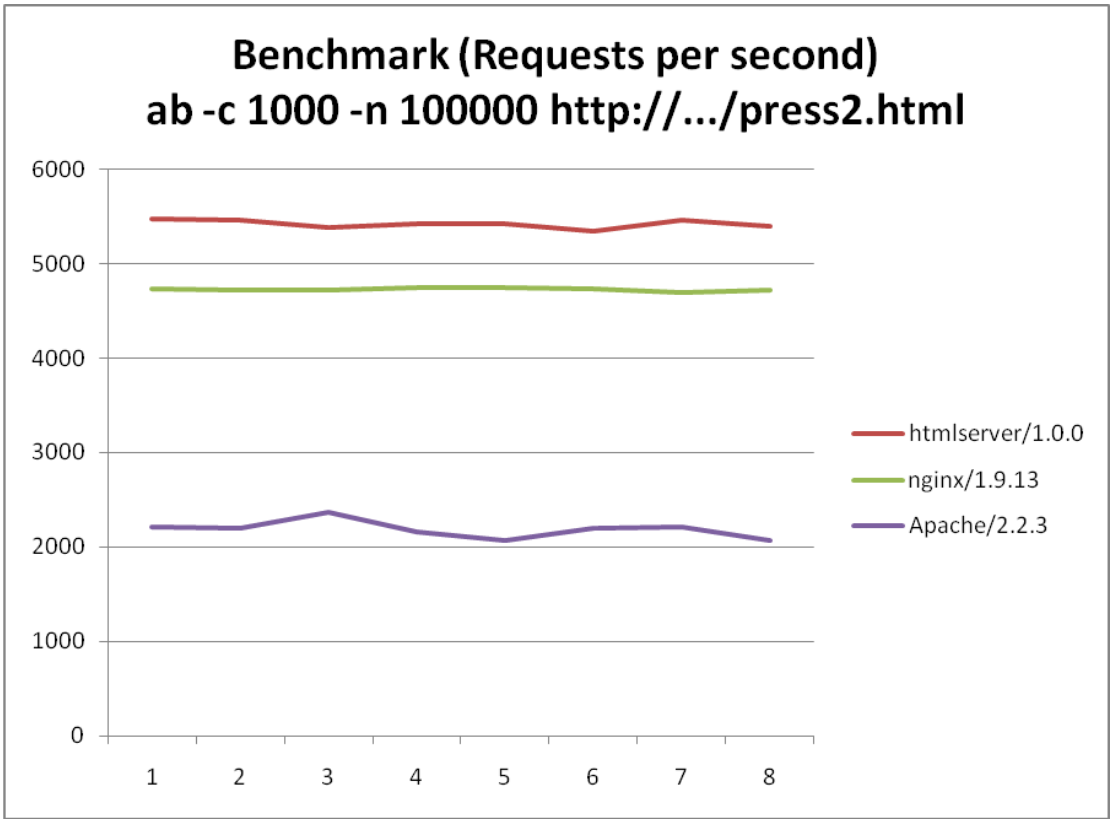
```
 66%     86
 75%     88
 80%     89
 90%    130
 95%   3082
 98%   3092
 99%   9038
100%  66119 (longest request)
```

然后交替压 8 次，取“Requests per second”的值

5.4 压测结果

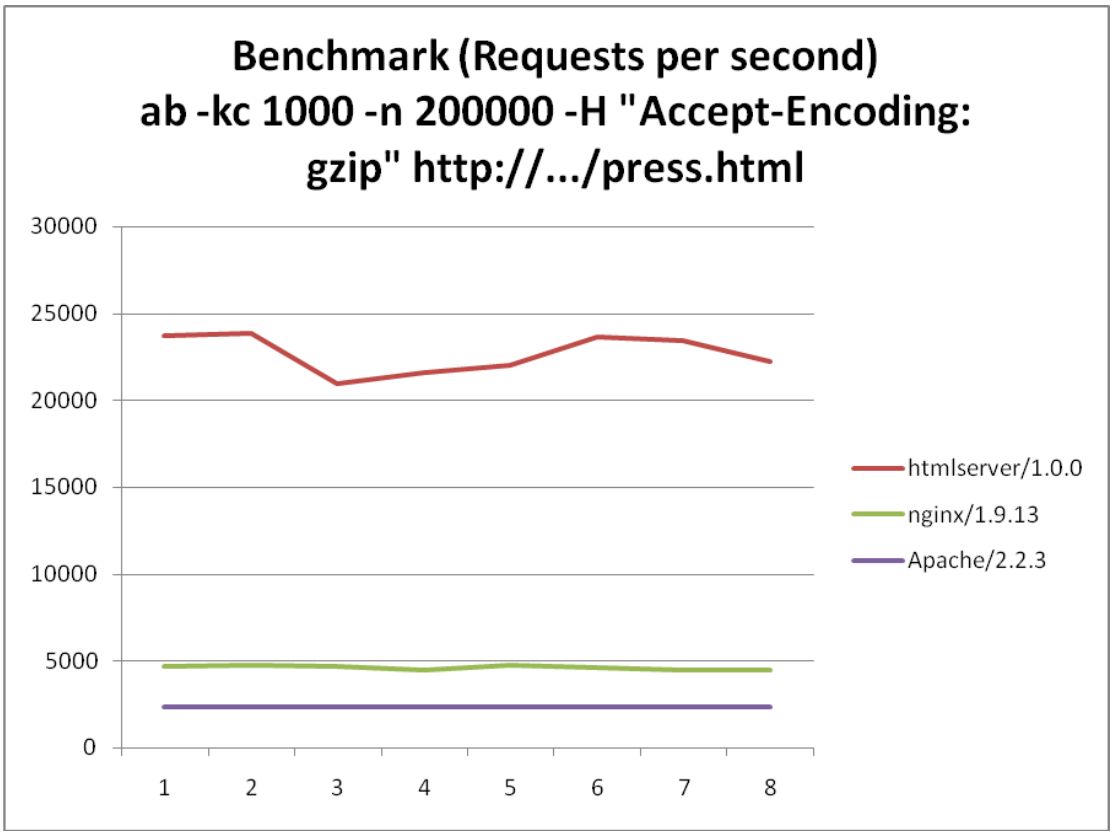
方案 A：

ROUND	htmlserver/1.0.0	nginx/1.9.13	Apache/2.2.3
1	5477.76	4732.07	2216.27
2	5459.26	4722.11	2209.36
3	5388.35	4715.31	2371.89
4	5421.64	4746.15	2160.36
5	5423.54	4743.78	2075.2
6	5347.08	4729.63	2204.47
7	5469.55	4695.48	2215.76
8	5405.51	4716.23	2076.46



方案 B：

ROUND	htmlserver/1.0.0	nginx/1.9.13	Apache/2.2.3
1	23678.5	4689.11	2337.84
2	23840.37	4733.28	2327.54
3	20989.13	4649.61	2328.36
4	21585.88	4434.38	2327.78
5	22030.33	4710.11	2357.61
6	23646.51	4601.01	2350.02
7	23414.56	4435.19	2332.41
8	22274.86	4436.41	2323.57



结论：

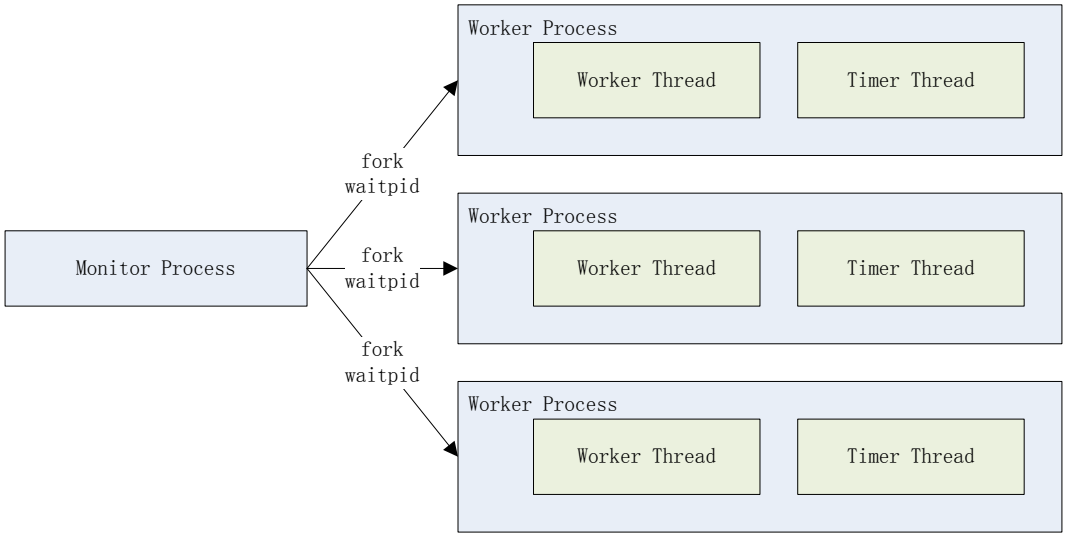
在未启用 Keep-Alive、小文件的场景下，htmlserver 比 nginx 大约快了 15%；

在开启 Keep-Alive、中型文件（约 4.3KB）、开启 gzip 压缩的场景下，htmlserver 比 nginx 足足快了 5 倍，秒杀目前世界上号称最快的 nginx！^_^

（现代浏览器一般都开启 Keep-Alive，4.3KB 也算是普遍的网页大小）

6 内部实现

6.1 系统结构



htmlserver 进程结构：

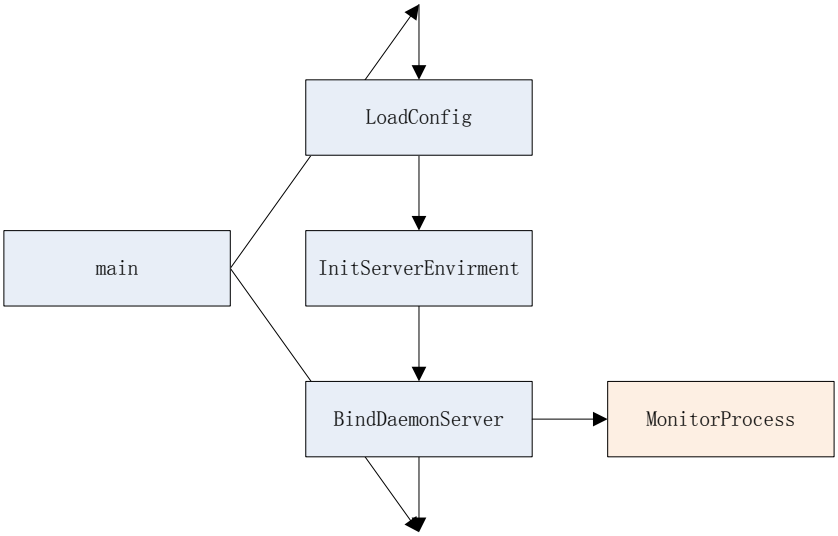
- 管理进程，负责创建、监管工作进程，负责传递 signal 管理命令。
- 工作进程

工作线程，负责多路复用 IO 管理，负责解析 HTTP，负责静态文件的响应和缓存。

定时器线程，负责定时更新用于日志输出的时间缓冲区。

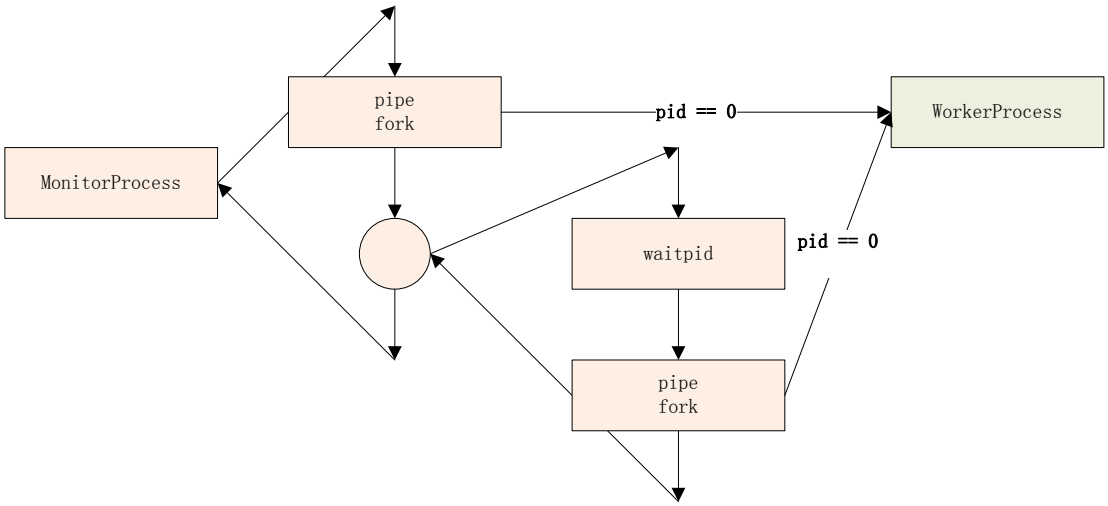
6.2 函数调用关系图

6.2.1 启动与初始化



启动后，经过装载配置和初始化环境后，函数 BindDaemonServer 转换进程为守护进程，切换到管理进程角色。

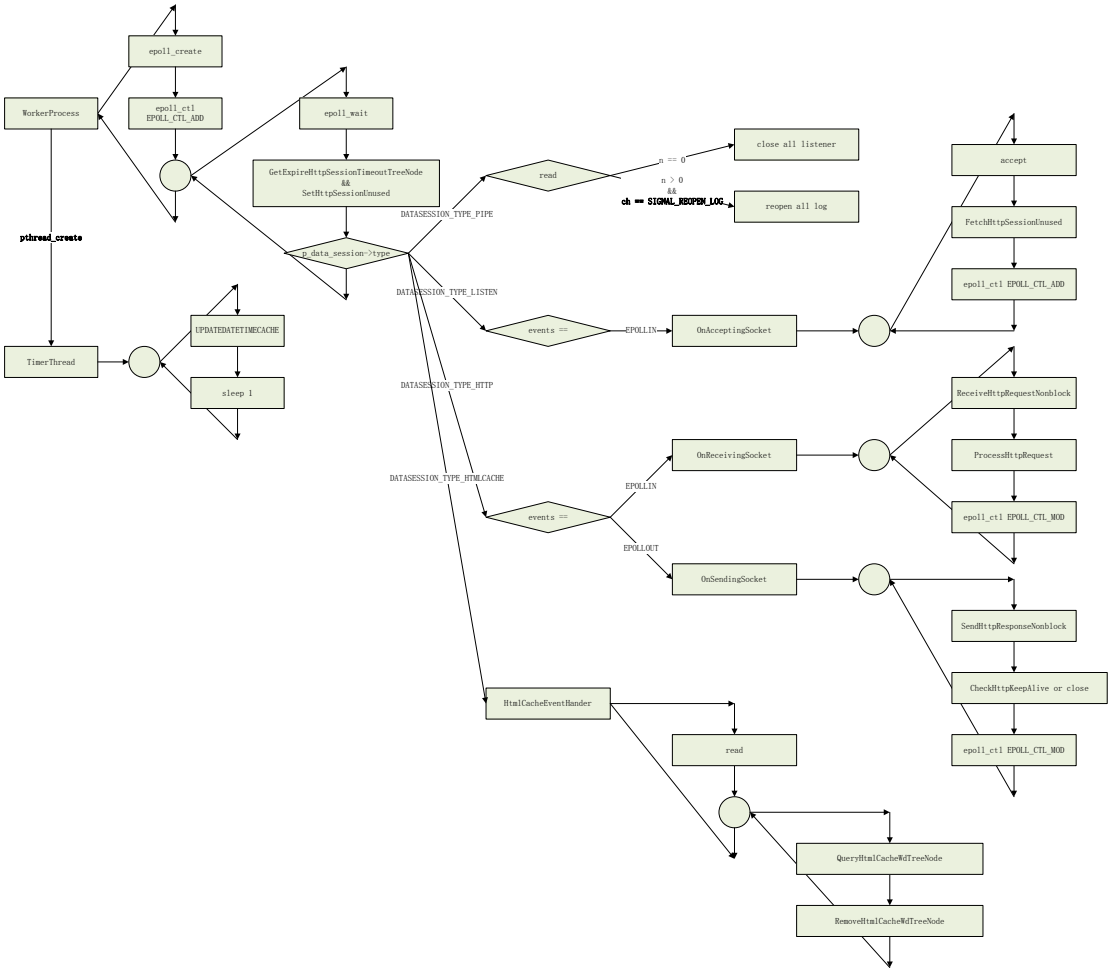
6.2.2 管理进程



创建所有管道和工作进程，然后监控工作进程结束事件，重启工作进程。

如果期间接收到 signal，通过管道传递命令给所有工作进程。

6.2.3 工作进程



创建多路复用 IO 池，加入管道、文件缓存句柄、侦听端口，然后进入主循环，等待 IO 事件。

如果是侦听端口事件，接受连接放入多路复用 IO 池。

如果是通讯会话事件，收发数据，处理 HTTP 请求，加入文件监控句柄，并修改多路复用 IO 等待事件掩码。

如果是文件缓存事件，清理该文件监控句柄。

如果是管道事件，处理管理进程传递过来的事件。