超高性能 Web 服务器(hetao)

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
v1.0.3	2016-08-28	厉华	新增章节 扩大系统限制 重新压测
v1.0.4	2016-09-01	厉华	补充 配置通览和说明
v1.0.5	2016-09-04	厉华	重写 网站配置
v1.0.6	2016-09-06	厉华	重写 配置通览和说明 重写 网站配置
v1.0.7	2016-09-07	厉华	重写 配置通览和说明
v1.0.8	2016-09-08	厉华	重写 配置文件
v1.0.9	2016-09-10	厉华	修改 编译安装 修改 配置文件

目录索引

1 前言

2010 年，我给行里新核心项目研发了核心后台应用服务平台，采用了定制通讯协议，几年使用下来无论与第三方业务系统（大多数是 JAVA 体系）对接、还是协议效率等方面都感受不好，趁着今年发起研发新一代核心后台应用服务平台契机，重新审视通讯协议的设计，最终选择了 HTTP/1.1。

于是我花时间研发了高性能 HTTP 解析器 fasterhttp，在编写示例时想，既然有了 HTTP 解析器为何不研发一个静态页面 Web 服务器呢？于是结合文件系统主动通知机制 inotify 研发了 htmlserver，改善了传统的被动轮询更新的缓存设计，性能比号称世界最快的 Nginx 还要快好几倍，我备受鼓舞。

htmlserver 发布后受到了广大网友的巨大反响，除了攻击名字幼稚、版本号和认为我压测数据作弊的喷子外，还是有不少网友提出了中肯的意见和建议，当然避免不了和 Nginx 的功能比较，于是，原只是支持静态页面的研发目标又一次“被逼”扩展为还要支持动态页面、反向代理负载均衡。（好深的坑啊）

原名字已不适合，于是我重新创建了一个项目 hetao，hetao V0.1.0 从 htmlserver V1.0.0 移过来继续研发，计划加入反向代理负载均衡、动态页面接口等一个 Web 服务器应具备的功能。

故事还在继续...

2 概述

hetao 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 gzip 压缩时性能比 nginx 约快 3 倍。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。

hetao 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。hetao 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。

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

hetao 目前只支持 Linux，将来很快会支持 WINDOWS。

hetao 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

* 支持反向代理负载均衡（目前支持轮询、最少连接数算法），支持 HTTP 与 HTTPS 互转

* 支持 rewrite

（以上应该就是一个 Web 服务器的主要功能了吧）

3 编译安装

3.1 编译

从 http://git.oschina.net/calvinwilliams/hetao 或 https://github.com/calvinwilliams/hetao 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
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c ListenSession.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c LOGC.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c rbtree.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c list.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c fasterjson.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c fasterhttp.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c IDL_hetao_conf.dsc.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c Util.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c Config.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c Envirment.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c main.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c MonitorProcess.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c WorkerProcess.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c WorkerThread.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c TimerThread.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c OnAcceptingSocket.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c OnReceivingSocket.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c OnSendingSocket.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c VirtualHostHash.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c ProcessHttpRequest.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c OnConnectingForward.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c OnSendingForward.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c OnReceivingForward.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing    -c HttpSession.c
```

```
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing      -c HtmlCacheSession.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing      -c HtmlCacheEventHander.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing      -c HtmlCacheWdTree.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing      -c HtmlCachePathfilenameTree.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing      -c HttpSessionTimeoutTree.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing      -c LeastConnectionCountTree.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing      -c MimeTypeHash.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing      -c RewriteUrl.c
gcc -g -fPIC -O2 -Wall -Werror -fno-strict-aliasing -o hetao ListenSession.o LOGC.o rbtree.o list.o
fasterjson.o fasterhttp.o IDL_hetao_conf.dsc.o Util.o Config.o Envirment.o main.o
MonitorProcess.o WorkerProcess.o WorkerThread.o TimerThread.o OnAcceptingSocket.o
OnReceivingSocket.o OnSendingSocket.o VirtualHostHash.o ProcessHttpRequest.o
OnConnectingForward.o OnSendingForward.o OnReceivingForward.o HttpSession.o
HtmlCacheSession.o HtmlCacheEventHander.o HtmlCacheWdTree.o
HtmlCachePathfilenameTree.o HttpSessionTimeoutTree.o LeastConnectionCountTree.o
MimeTypeHash.o RewriteUrl.o      -lpcre -lpthread -lssl -lz
INSTALL CONFIG :
            mkdir -p /var/hetao
            mkdir -p /var/hetao/log
            hetao -> /usr/local/bin
            ../bin/hetao.sh -> /usr/local/bin
            ../conf/hetao.conf -> /etc/hetao
            ../certs/* -> /etc/hetao/certs
            ../www/* -> /var/hetao/www
Execute the command to install : sudo make -f makefile.Linux install
```

没有报错的话就能编译出可执行文件 hetao。也可以加上参数-j 10 以加快编译速度。

编译输出最后一段提示本次配置预安装目标，确认后执行后面的安装命令。

3.2 安装

以下为安装到系统用户中，如果要安装到其它目录，请修改 makefile.Linux 中的

```
...
###### 目标文件、安装目录配置区
NOCLEAN_DIRINST_NOCOVER=            /var/hetao
NOCLEAN_DIRINST2_NOCOVER=           /var/hetao/log
BIN                                 =           hetao
BININST                             =           /usr/local/bin
NOCLEAN_OBJ                         =           ../bin/hetao.sh
NOCLEAN_OBJINST                     =           /usr/local/bin
```

```
NOCLEAN_OBJ_NOCOVER         =           ../conf/hetao.conf
NOCLEAN_OBJINST_NOCOVER =           /etc/hetao
NOCLEAN_OBJ2_NOCOVER        =           ../certs/*
NOCLEAN_OBJINST2_NOCOVER=           /etc/hetao/certs
NOCLEAN_OBJ3_NOCOVER        =           ../www/*
NOCLEAN_OBJINST3_NOCOVER=           /var/hetao/www
...
```

执行安装命令：

```
$ sudo make -f makefile.Linux install
mkdir -p /var/hetao
mkdir -p /var/hetao/log
cp -rf hetao /usr/local/bin/
cp -rf ../bin/hetao.sh /usr/local/bin/
mkdir -p /etc/hetao
cp -rf ../conf/hetao.conf /etc/hetao/
mkdir -p /etc/hetao/certs
cp -rf ../certs/gencert.sh /etc/hetao/certs/
cp -rf ../certs/server.crt /etc/hetao/certs/
cp -rf ../certs/server.csr /etc/hetao/certs/
cp -rf ../certs/server.key /etc/hetao/certs/
cp -rf ../certs/server.pem /etc/hetao/certs/
mkdir -p /var/hetao/www
cp -rf ../www/error_pages /var/hetao/www/
cp -rf ../www/index.html /var/hetao/www/
```

安装过程做了如下事情：

自动创建日志目录/var/hetao/log

自动复制主执行程序 hetao 到/usr/local/bin/

自动复制管理脚本 hetao.sh 到/usr/local/bin/

自动复制缺省配置文件 hetao.conf 到/etc/hetao/

自动复制示例证书文件到/etc/hetao/certs/

自动复制示例首页文件到/var/hetao/www/

自动复制自定义出错页面文件到/var/hetao/www/error_pages/

这样就安装好了!

3.3 用缺省配置第一次启动

执行以下命令以缺省配置启动

```
$ sudo hetao.sh start
```

如果没有产生输出、没有产生/var/hetao/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 80。

可以看到进程，hetao 进程结构由一个管理进程+n 个工作进程组成

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

以及侦听端口

```
$ netstat -an | grep -w 80
tcp        0      0 0.0.0.0:80          0.0.0.0:*               LISTEN
```

自测一下

```
$ curl http://127.0.0.1:80/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 HETAO
</body>
</html>
```

恭喜您，启动成功！

直接发送 TERM 信号到父进程可停止 hetao

```
$ kill 14122
```

3.4 扩大系统限制

默认系统中单个进程最大可打开描述字只有 1024 个，肯定不能满足一个正式的 Web 服务器的需要，那么作为生产环境，一定要扩大系统限制。

一些推荐的系统限制设置放在 conf/*，把文件内容追加到系统配置中，需要 root 权限。

conf/limits.conf.add -> /etc/security/limits.conf

```
*          soft       nofile    65536
*          hard       nofile    65536
*          soft       nproc     unlimited
*          hard       nproc     unlimited
```

conf/sysctl.conf.add -> /etc/sysctl.conf

```
fs.file-max=65536
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_tw_recycle=1
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_keepalive_time = 1200
net.ipv4.ip_local_port_range = 1024 65000
net.ipv4.tcp_max_tw_buckets = 5000
```

执行以下命令生效

```
sysctl -p
```

4 配置文件

4.1 配置通览和说明

安装时复制的配置文件为缺省配置，可根据实际情况调整，如侦听端口、server 配置等。

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

    "error_log" : "/var/hetao/log/error.log" ,
    "log_level" : ERROR ,

    "user" : "nobody" ,

    "limits" :
    {
        "max_http_session_count" : 100000 ,
        "max_file_cache" : 1024000
    } ,

    "listen" :
    {
        "ip" : "" ,
        "port" : 80 ,
        "website" :
        {
```

```
            "domain" : "" ,
            "wwwroot" : "/var/hetao/www" ,
            "index" : "/index.html,/index.htm" ,
            "access_log" : "/var/hetao/log/access.log"
        }
    } ,

    /*
    "listen" :
    {
        "ip" : "www.test.com" ,
        "port" : 80 ,
        "website" :
        {
            "domain" : "" ,
            "wwwroot" : "/var/hetao/www" ,
            "index" : "/index.html,/index.htm" ,
            "access_log" : "/var/hetao/log/access.log"
        }
    } ,
    */

    /*
    "listen" :
    {
        "ip" : "" ,
        "port" : 80 ,
        "website" :
        {
            "domain" : "" ,
            "wwwroot" : "/var/hetao/www" ,
            "index" : "/index.html,/index.htm" ,
            "access_log" : "/var/hetao/log/access.log" ,
            "rewrite" : { "pattern":"/(.+)/(.+)" , "template":"/(2)/(1)" }
        }
    } ,
    */

    /*
    "listen" :
    {
        "ip" : "" ,
        "port" : 80 ,
        "website" :
```

```
        {
                "domain" : "" ,
                "wwwroot" : "/var/hetao/www" ,
                "index" : "/index.html,/index.htm" ,
                "access_log" : "/var/hetao/log/access.log" ,
                "forward" :
                {
                        "forward_type" : "php" ,
                        "forward_rule" : "R" ,
                        "forward_server" : { "ip" : "192.168.6.111" , "port" : 8080 } ,
                        "forward_server" : { "ip" : "192.168.6.111" , "port" : 8080 } ,
                        "forward_server" : { "ip" : "192.168.6.111" , "port" : 8080 }
                }
        }
} ,
*/

/*
"listen" :
{
        "ip" : "" ,
        "port" : 80 ,
        "website" :
        {
                "domain" : "" ,
                "wwwroot" : "/var/hetao/www" ,
                "index" : "/index.html,/index.htm" ,
                "access_log" : "/var/hetao/log/access.log" ,
                "forward" :
                {
                        "forward_type" : "php" ,
                        "forward_rule" : "R" ,
                        "ssl" :
                        {
                                "certificate_file" : "/etc/hetao/certs/server.crt"
                        } ,
                        "forward_server" : { "ip" : "192.168.6.111" , "port" : 1443 } ,
                        "forward_server" : { "ip" : "192.168.6.111" , "port" : 1443 } ,
                        "forward_server" : { "ip" : "192.168.6.111" , "port" : 1443 }
                }
        }
} ,
*/
```

```
/*
"listen" :
{
	"ip" : "" ,
	"port" : 443 ,
	"ssl" :
	{
		"certificate_file" : "/etc/hetao/certs/server.pem" ,
		"certificate_key_file" : "/etc/hetao/certs/server.key"
	} ,
	"website" :
	{
		"domain" : "" ,
		"wwwroot" : "/var/hetao/www" ,
		"index" : "/index.html,/index.htm" ,
		"access_log" : "/var/hetao/log/access.log"
	}
} ,
*/

"tcp_options" :
{
	"nodelay" : 1 ,
	"nolinger" : -1
} ,

"http_options" :
{
	"compress_on" : 1 ,
	"timeout" : 60 ,
	"forward_disable" : 60
} ,

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

```
    "mime_types" :
    {
        "mime_type" : { "type":"html htm shtml" , "mime":"text/html" ,
"compress_enable":"1" } ,
        "mime_type" : { "type":"css" , "mime":"text/css" , "compress_enable":"0" } ,
        "mime_type" : { "type":"xml" , "mime":"text/xml" , "compress_enable":"1" } ,
        "mime_type" : { "type":"txt" , "mime":"text/plain" , "compress_enable":"1" } ,
        "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" ,
"compress_enable":"1" } ,
        "mime_type" : { "type":"jar war ear" , "mime":"application/java-archive" } ,
        "mime_type" : { "type":"json" , "mime":"application/json" ,
"compress_enable":"1" } ,
        "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" ,
"compress_enable":"1" } ,
        "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" }
    }
}
```

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，枚举有 DEBUG、INFO、WARN、ERROR、FATAL

user　　启动后以该用户身份（可选配置）

limits　　限制设置

max_http_session_count 最大 HTTP 通讯会话并发数量

max_file_cache　　最大缓存文件大小

listen　　侦听配置

ip　　本地侦听端口，填空则为 0.0.0.0

port　　本地侦听端口

website　　网站配置

domain　　网站域名，用于匹配 HTTP 请求头选项 Host 区分虚拟主机。如果填空则统配所有

wwwroot　　网站本地根目录

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，一个 HTTP 请求写一条事件日志

ssl 服务端安全加密规则（可选配置块）

certificate_file 公钥证书文件名

certificate_key_file 私钥文件名

rewrite 改写 URI 规则（可选配置块）

pattern 正则匹配式

template 改写后模板

forward 反向代理规则（可选配置块）

forward_type 该文件扩展名的 URL 走反向代理

forward_rule 负载均衡算法，目前支持：R 轮询，L 最少连接数

ssl 客户端安全加密规则（可选配置块）

certificate_file 公钥证书文件名

forward_server 后端应用服务器地址

ip 后端侦听端口

port 后端侦听端口

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 是否响应浏览器端的压缩请求，有助于大幅减少通讯传输流量

timeout HTTP 超时时间，单位：秒

forward_disable 当反向代理连接后端失败后，暂禁时间，单位：秒

error_pages 出错页面配置（可选配置块）

error_page_??? HTTP 响应???时返回的页面文件，目前支持 400、401、

403、404、408、500、503、505

mime_types　　流类型配置集合。主要用于填充 HTTP 响应头选项 Content-Type

mime_type　　流类型配置

type　　文件扩展名

mime　　流类型描述，填充 HTTP 响应头选项 Content-Type

compress_enable　　是否压缩缓存，1 位压缩，不出现或 0 为不压缩

最后注意：json 元素之间有","以及最后一个元素后面没有","。

4.2　网站配置和示例

网站配置层次关系：

listen(侦听) - website(网站) - forward_server(反向代理转发服务器)

一个 hetao 运行实例里可以有多个 listen，每个 listen 为一个 ip、port 对，对应一个 TCP 服务端侦听。每个 listen 上可以配置多个网站 website，基于域名 domain 识别虚拟主机。每个 website 上可以配置成某一文件类型 forward_type 转发到后方应用服务器 forward_server，以及负载均衡算法 forward_rule。

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"/"之间的部分）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"

4.2.1　一个简单的网站配置

```
"listen" :
{
    "ip" : "" ,
    "port" : 80 ,
    "website" :
    {
        "domain" : "" ,
```

```
            "wwwroot" : "/var/hetao/www" ,
            "index" : "/index.html,/index.htm" ,
            "access_log" : "/var/hetao/log/access.log"
        }
    },
```

4.2.2 一个带域名的虚拟主机网站配置

```
    "listen" :
    {
        "ip" : "www.test.com" ,
        "port" : 80 ,
        "website" :
        {
            "domain" : "" ,
            "wwwroot" : "/var/hetao/www" ,
            "index" : "/index.html,/index.htm" ,
            "access_log" : "/var/hetao/log/access.log"
        }
    },
```

4.2.3 一个需要改写 URI 的网站配置（/xxx/yyy 改写为 /yyy/xxx）

```
    "listen" :
    {
        "ip" : "" ,
        "port" : 80 ,
        "website" :
        {
            "domain" : "" ,
            "wwwroot" : "/var/hetao/www" ,
            "index" : "/index.html,/index.htm" ,
            "access_log" : "/var/hetao/log/access.log" ,
            "rewrite" : { "pattern":"/(.+)/(.+)" , "template":"/(2)/(1)" }
        }
    },
```

4.2.4 一个反向代理配置，针对文件类型 php，轮询算法

```
    "listen" :
```

```
    {
        "ip" : "" ,
        "port" : 80 ,
        "website" :
        {
            "domain" : "" ,
            "wwwroot" : "/var/hetao/www" ,
            "index" : "/index.html,/index.htm" ,
            "access_log" : "/var/hetao/log/access.log" ,
            "forward" :
            {
                "forward_type" : "php" ,
                "forward_rule" : "R" ,
                "forward_server" : { "ip" : "192.168.6.111" , "port" : 8080 } ,
                "forward_server" : { "ip" : "192.168.6.111" , "port" : 8080 } ,
                "forward_server" : { "ip" : "192.168.6.111" , "port" : 8080 }
            }
        }
    } ,
```

4.2.5 一个反向代理配置，针对文件类型 php，轮询算法，转发时装载证书变成 HTTPS

```
    "listen" :
    {
        "ip" : "" ,
        "port" : 80 ,
        "website" :
        {
            "domain" : "" ,
            "wwwroot" : "/var/hetao/www" ,
            "index" : "/index.html,/index.htm" ,
            "access_log" : "/var/hetao/log/access.log" ,
            "forward" :
            {
                "forward_type" : "php" ,
                "forward_rule" : "R" ,
                "ssl" :
                {
                    "certificate_file" : "/etc/hetao/certs/server.crt"
                } ,
                "forward_server" : { "ip" : "192.168.6.111" , "port" : 1443 } ,
                "forward_server" : { "ip" : "192.168.6.111" , "port" : 1443 } ,
```

```
                "forward_server" : { "ip" : "192.168.6.111" , "port" : 1443 }
            }
        }
    },
```

4.2.6 一个简单的 HTTPS 网站配置

```
    "listen" :
    {
        "ip" : "" ,
        "port" : 443 ,
        "ssl" :
        {
            "certificate_file" : "/etc/hetao/certs/server.pem" ,
            "certificate_key_file" : "/etc/hetao/certs/server.key"
        } ,
        "website" :
        {
            "domain" : "" ,
            "wwwroot" : "/var/hetao/www" ,
            "index" : "/index.html,/index.htm" ,
            "access_log" : "/var/hetao/log/access.log"
        }
    },
```

5 服务器管理

5.1 直接用命令管理

启动 hetao

```
$ hetao ~/etc/hetao.conf
```

查询 hetao 进程

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

优雅的重启 hetao，或者重载配置文件

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

```
calvin    14877 14876    0 00:10 ?      00:00:00 hetao /home/calvin/etc/hetao.conf
$ kill -USR2 14876
$    ps -ef | grep hetao | grep -v grep
calvin    14876        1    0 00:10 ?      00:00:00 hetao /home/calvin/etc/hetao.conf
calvin    14877 14876    0 00:10 ?      00:00:00 hetao /home/calvin/etc/hetao.conf
calvin    14889        1    0 00:12 ?    00:00:00 hetao /home/calvin/etc/hetao.conf
calvin    14890 14889    0 00:12 ?      00:00:00 hetao /home/calvin/etc/hetao.conf
$ kill 14876
$ ps -ef | grep hetao | grep -v grep
calvin    14889        1    0 00:12 ?    00:00:00 hetao /home/calvin/etc/hetao.conf
calvin    14890 14889    0 00:12 ?      00:00:00 hetao /home/calvin/etc/hetao.conf
```

向 hetao 发送重新打开日志文件信号

```
$ kill -USR1 14889
```

停止 hetao

```
$ kill 14889
```

5.2 用自带脚本管理

启动 hetao（默认配置文件路径~/etc/hetao.conf）

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

查询 hetao 进程

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

重启 hetao

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

优雅的重启 hetao，或者重载配置文件

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

```
calvin    14796     1   0 00:06 ?    00:00:00 hetao /home/calvin/etc/hetao.conf
calvin    14797 14796   0 00:06 ?    00:00:00 hetao /home/calvin/etc/hetao.conf
```

向 hetao 发送重新打开日志文件信号

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

停止 hetao

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

6 压测

6.1 压测环境

6.1.1 压测平台

压测发起端为台机 PC(192.168.6.17)，配置如下：

CPU ：Intel Core i3-3240 3.40GHz 3.40GHz

内存 ：512MB

WindowsXP 里面装了 VMWARE 10 里面装了 RedHat Enterprise Linux Server release 5.4 (32BITS)

压测网络为百兆有线

压测服务端为台机 PC(192.168.6.111)，配置如下：

CPU ：AMD E-350 1.60GHz 1.60GHz

内存 ：4GB

RedHat Enterprise Linux Server release 5.4 (32BITS)

6.1.2 压测客户端

压测客户端采用 Apache 自带工具 ab。

因 ab 只支持 HTTP/1.0 而不支持 HTTP/1.1，会影响 Nginx 的压缩和 Keep-Alive 不能同时开启的 BUG，故修改了 ab.c 中填充 HTTP 请求版本的代码，重新编译成 ab2 供压测使用。hetao 和 Apache 不受影响。

httpd-2.2.17/support/ab.c

```
1609        /* setup request */
1610        if (posting <= 0) {
1611            snprintf_res = apr_snprintf(request, sizeof(_request),
1612                "%s %s HTTP/1.1\r\n"
1613                "%s" "%s" "%s"
1614                "%s" "\r\n",
1615                (posting == 0) ? "GET" : "HEAD",
1616                (isproxy) ? fullurl : path,
1617                keepalive ? "Connection: Keep-Alive\r\n" : "",
1618                cookie, auth, hdrs);
1619        }
1620        else {
1621            snprintf_res = apr_snprintf(request,    sizeof(_request),
1622                "%s %s HTTP/1.1\r\n"
1623                "%s" "%s" "%s"
1624                "Content-length: %" APR_SIZE_T_FMT "\r\n"
1625                "Content-type: %s\r\n"
1626                "%s"
1627                "\r\n",
1628                (posting == 1) ? "POST" : "PUT",
1629                (isproxy) ? fullurl : path,
1630                keepalive ? "Connection: Keep-Alive\r\n" : "",
1631                cookie, auth,
1632                postlen,
1633                (content_type[0]) ? content_type : "text/plain", hdrs);
1634        }
```

6.1.3 压测服务端

选用以下 Web 服务器软件做横向压测，版本和配置侦听端口如下：

hetao/0.2.0，侦听端口为 9527

Nginx/1.9.13，侦听端口为 9528

Apache/2.2.14，侦听端口为 9529

Tengine/2.1.2，侦听端口为 9530

（原计划还有 kangle/3.4.8，但是从官网上下载的源代码编译安装始终报错，猜可能是我的 Linux 编译器 gcc 版本过低，不支持__sync_原子操作，但 rhel5.4 也不低啊，算了不用它了）

```
...
g++ -I../module/access -I../module/whm -O2 -g -DNDEBUG    -D_REENTRANT -DLINUX
-D_LARGE_FILE -D_FILE_OFFSET_BITS=64 -D__USE_FILE_OFFSET64 -L../lib      -o kangle cache.o
KConfig.o forwin32.o garbage_c.o HttpCore.o KAccess.o KAcserver.o KAcserverManager.o
KBuffer.o KChain.o KConfigBuilder.o KConfigParser.o KContentType.o KDiskCache.o
KPortSelector.o KKqueueSelector.o KEpollSelector.o KFastcgiFetchObject.o KFastcgiUtils.o
KFetchObject.o KFileMsg.o KFileName.o KHtmlSupport.o KHtmlSupportException.o
KHttpKeyValue.o KHttpManage.o KHttpObject.o KHttpObjectHash.o KHttpObjectParserHook.o
KHttpProtocolParser.o KHttpProtocolParserHook.o KHttpProxyFetchObject.o KHttpRequest.o
KHttpServerParser.o KLang.o KLangParser.o KLogElement.o KReg.o KSelector.o
KSelectorManager.o KSequence.o KServerListen.o KSocket.o KSocketFetchObject.o KTable.o
KThreadPool.o KTimeMatch.o KUrlValue.o KVirtualHost.o KVirtualHostManage.o KWriteBack.o
KWriteBackManager.o KXmlContext.o KXml.o KXmlException.o KXmlSupport.o lib.o log.o
main.o malloc_debug.o md5.o work.o utils.o KAccessParser.o KString.o KRewriteMark.o
KSingleProgram.o KHttpTransfer.o KDeChunked.o KGzip.o KServer.o KSelectable.o KStream.o
KNsVirtualHost.o KContentMark.o KRedirectMark.o KLineFile.o KMultiHostAcl.o test.o
KHttpFieldValue.o KSingleAcserver.o KMultiAcserver.o KSockPoolHelper.o KEnvInterface.o
KRedirect.o KCgiRedirect.o KCgiFetchObject.o KPipeStream.o KCgi.o KCgiEnv.o KApiRedirect.o
KApiEnv.o HttpExt.o KApiFetchObject.o KHttpHeadPull.o KSockFastcgiFetchObject.o
KApiFastcgiFetchObject.o KPathRedirect.o KLogManage.o KBaseVirtualHost.o process.o
KContentTransfer.o KChunked.o KCacheStream.o KHttpField.o KHttpDigestAuth.o KHttpAuth.o
KHttpBasicAuth.o KAuthMark.o KObjectList.o KAjpMessage.o KAjpFetchObject.o
KExpressionParseTree.o KSSICommandCondition.o KSSICommandEcho.o
KSSICommandInclude.o KSSIContext.o KSSIRedirect.o KSSICommandSet.o KSSIProcess.o
KSSICommand.o KSSICommandPrintEnv.o KSSIFetchObject.o KServiceProvider.o
KISAPIServiceProvider.o directory.o KSSICommandExec.o KSSICommandConfig.o ssl_utils.o
KApiPipeStream.o KPoolableSocketContainer.o KProcessManage.o KCmdPoolableRedirect.o
KSubVirtualHost.o KIpVirtualHost.o KHttpPost.o KHtAccess.o KHtModule.o
KHtRewriteModule.o KRewriteMarkEx.o EdcodeUtils.o KProcess.o KApiProcess.o
KCmdProcess.o KVirtualHostProcess.o KExtendProgram.o KDynamicString.o kmysql.o
KCdnMysqlMark.o KCdnRewriteMark.o KCdnContainer.o KTempleteVirtualHost.o
KVirtualHostDatabase.o KDsoModule.o KList.o KListNode.o KLogHandle.o KRequestQueue.o
KContext.o KCdnRedirect.o time_utils.o rbtree.o KVirtualHostContainer.o KSocketBuffer.o
KAsyncFetchObject.o KSyncFetchObject.o KStaticFetchObject.o KDirectoryFetchObject.o
KApiDso.o KUwsgiFetchObject.o KScgiFetchObject.o KHmuxFetchObject.o KTempFile.o
KListenConfigParser.o KApacheVirtualHost.o KSSLSocket.o KAsyncWorker.o KInputFilter.o
KMultiPartInputFilter.o KReplaceContentMark.o KReplaceContentFilter.o
```

```
KConcatFetchObject.o KIpSpeedLimitMark.o KDynamicListen.o KCache.o KPerIpAcl.o
KDiskCacheIndex.o
KSqliteDiskCacheIndex.o ../module/whm/dllmain.o ../module/whm/WhmCallMap.o ../modul
e/whm/WhmCommand.o ../module/whm/WhmContext.o ../module/whm/whm.o ../module/
whm/WhmLog.o ../module/whm/WhmPackage.o ../module/whm/WhmPackageManage.o ../
module/whm/KWhmService.o ../module/whm/stdafx.o ../module/whm/WhmDso.o ../modul
e/whm/WhmExtend.o ../module/whm/WhmUrl.o ../module/whm/WhmShell.o ../module/wh
m/WhmShellProcess.o ../module/whm/WhmShellSession.o ../module/whm/whmdso/core/co
re.o KTimer.o KUrlParser.o KHttpFilterContext.o KHttpFilterDso.o KHttpFilterDsoManage.o
KHttpFilterHookCollectRequest.o KHttpFilterHook.o KHttpFilterManage.o KTempFileStream.o
KHttpFilterStream.o KHttpFilterHookCollectResponse.o KAccessDso.o KConnectionSelectable.o
KReadWriteBuffer.o KResponseContext.o KUpstreamSelectable.o KSimulateRequest.o
KCloudIpAcl.o      -lpthread -lpcre -lz -ldl
KConfig.o: In function `katom_cas':
/home/calvin/expack/kangle-3.4.8/src/katom.h:107: undefined reference to
`__sync_bool_compare_and_swap_4'
KConfig.o: In function `katom_inc':
/home/calvin/expack/kangle-3.4.8/src/katom.h:39: undefined reference to
`__sync_add_and_fetch_4'
HttpCore.o: In function `katom_inc':
/home/calvin/expack/kangle-3.4.8/src/katom.h:39: undefined reference to
`__sync_add_and_fetch_4'
HttpCore.o: In function `katom_dec':
/home/calvin/expack/kangle-3.4.8/src/katom.h:49: undefined reference to
`__sync_add_and_fetch_4'
HttpCore.o: In function `katom_dec':
/home/calvin/expack/kangle-3.4.8/src/KHttpRequest.h:353: undefined reference to
`__sync_add_and_fetch_4'
HttpCore.o: In function `katom_inc':
/home/calvin/expack/kangle-3.4.8/src/katom.h:39: undefined reference to
`__sync_add_and_fetch_4'
HttpCore.o:/home/calvin/expack/kangle-3.4.8/src/katom.h:49: more undefined references to
`__sync_add_and_fetch_4' follow
...
```

6.2 压测方案

考察较大量开启 HTTP 长连接 Keep-Alive、开启 gzip 压缩、中型大小网页的 GET 性能

并发 1000，共发起 HTTP 请求 5 万次，目标网页文件大小约 3.3KB

准备网页文件 press.html

```
-rwxrwxr-x 1 calvin calvin 3321 08-27 21:03 press.html
```

命令：

```
$ ab2 -kc 1000 -n 50000 -H "Accept-Encoding: gzip" http://192.168.6.111:????/press.html
```

6.3 压测过程

先交替的各压一次热热身（可以预览一下性能）

```
$ ab2 -kc 1000 -n 50000 -H "Accept-Encoding: gzip" http://192.168.6.111:9527/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 192.168.6.111 (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:        hetao/0.2.0
Server Hostname:         192.168.6.111
Server Port:            9527

Document Path:          /press.html
Document Length:        281 bytes

Concurrency Level:      1000
Time taken for tests:   6.923 seconds
Complete requests:      100000
Failed requests:        0
Write errors:           0
Keep-Alive requests:    100000
Total transferred:      41709990 bytes
HTML transferred:       28242186 bytes
```

```
Requests per second:    14445.19 [#/sec] (mean)
Time per request:       69.227 [ms] (mean)
Time per request:       0.069 [ms] (mean, across all concurrent requests)
Transfer rate:          5883.87 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   13 198.9      0    3080
Processing:     1   31 102.2     15    3456
Waiting:        0   31 102.2     15    3456
Total:          1   44 242.5     15    3479

Percentage of the requests served within a certain time (ms)
  50%     15
  66%     31
  75%     36
  80%     43
  90%     64
  95%     85
  98%    108
  99%    126
 100%   3479 (longest request)

$ ab2 -kc 1000 -n 50000 -H "Accept-Encoding: gzip" http://192.168.6.111:9528/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 192.168.6.111 (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:        192.168.6.111
```

```
Server Port:                    9528

Document Path:                  /press.html
Document Length:                293 bytes

Concurrency Level:              1000
Time taken for tests:           23.928 seconds
Complete requests:              100000
Failed requests:                0
Write errors:                   0
Keep-Alive requests:            99004
Total transferred:              54195020 bytes
HTML transferred:               29300000 bytes
Requests per second:            4179.19 [#/sec] (mean)
Time per request:               239.281 [ms] (mean)
Time per request:               0.239 [ms] (mean, across all concurrent requests)
Transfer rate:                  2211.83 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   23 268.2      0    3167
Processing:     1  127 904.6     75   20671
Waiting:        0  127 904.6     75   20670
Total:          1  150 1058.7    75   23814

Percentage of the requests served within a certain time (ms)
  50%     75
  66%     86
  75%     87
  80%     87
  90%     92
  95%     96
  98%     96
  99%   2365
 100%  23814 (longest request)

$ ab2 -kc 1000 -n 50000 -H "Accept-Encoding: gzip" http://192.168.6.111:9529/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 192.168.6.111 (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/2.2.14
Server Hostname:        192.168.6.111
Server Port:            9529

Document Path:          /press.html
Document Length:        281 bytes

Concurrency Level:      1000
Time taken for tests:   39.800 seconds
Complete requests:      100000
Failed requests:        0
Write errors:           0
Keep-Alive requests:    99119
Total transferred:      65363814 bytes
HTML transferred:       28101124 bytes
Requests per second:    2512.58 [#/sec] (mean)
Time per request:       397.998 [ms] (mean)
Time per request:       0.398 [ms] (mean, across all concurrent requests)
Transfer rate:          1603.83 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1  48.4      0    3001
Processing:     1   97 774.0      2   25875
Waiting:        0   97 773.8      2   25875
Total:          1   98 780.3      2   25897

Percentage of the requests served within a certain time (ms)
  50%      2
  66%      3
  75%      3
  80%      4
  90%      6
```

```
  95%     82
  98%   1398
  99%   2352
 100%  25897 (longest request)

$ ab2 -kc 1000 -n 50000 -H "Accept-Encoding: gzip" http://192.168.6.111:9530/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 192.168.6.111 (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:        Tengine/2.1.2
Server Hostname:        192.168.6.111
Server Port:            9530

Document Path:          /press.html
Document Length:        293 bytes

Concurrency Level:      1000
Time taken for tests:   25.203 seconds
Complete requests:      100000
Failed requests:        0
Write errors:           0
Keep-Alive requests:    99027
Total transferred:      51895135 bytes
HTML transferred:       29300000 bytes
Requests per second:    3967.81 [#/sec] (mean)
Time per request:       252.028 [ms] (mean)
Time per request:       0.252 [ms] (mean, across all concurrent requests)
Transfer rate:          2010.84 [Kbytes/sec] received
```

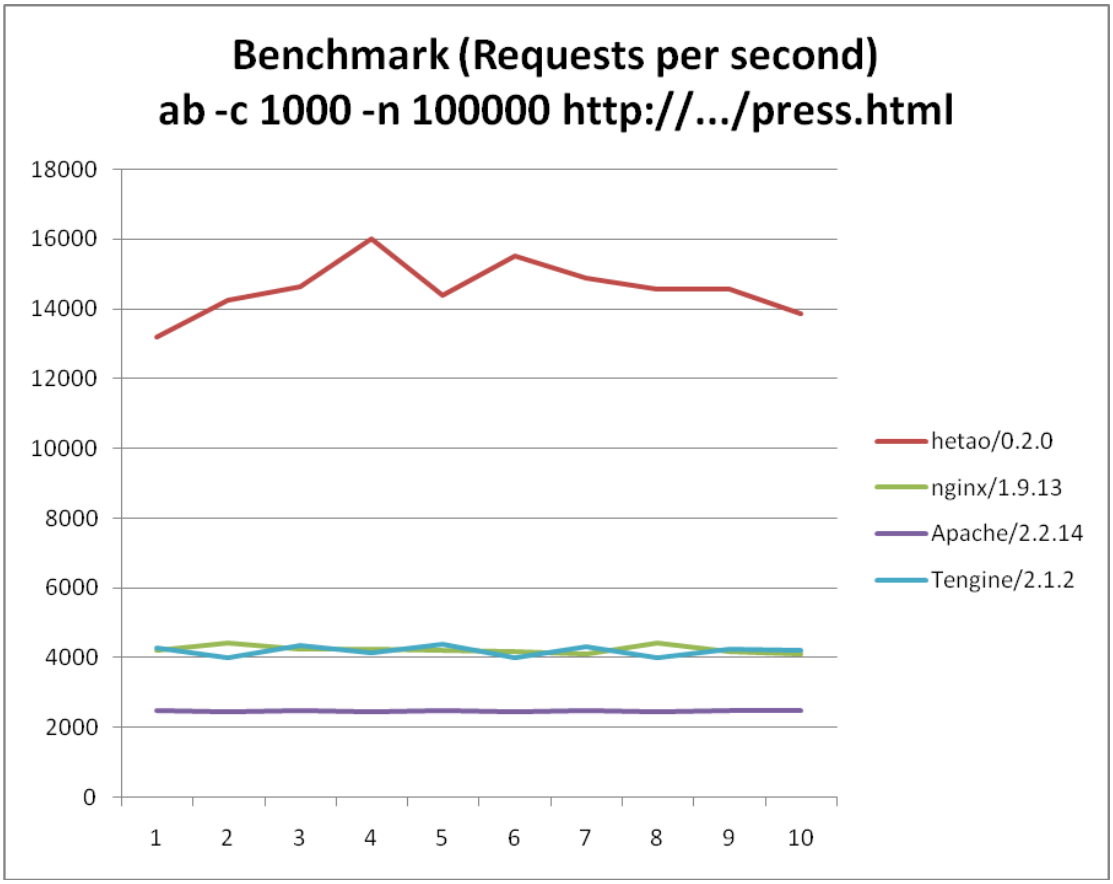
```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   16 233.6      0   21003
Processing:     1   91 485.5     33   21267
Waiting:        1   91 485.5     33   21267
Total:          1  107 637.0     33   24392

Percentage of the requests served within a certain time (ms)
  50%     33
  66%    101
  75%    112
  80%    112
  90%    132
  95%    182
  98%    213
  99%    337
 100%  24392 (longest request)
```

然后交替压 10 次，取“Requests per second”的值

6.4 压测结果

ROUND	hetao/0.2.0	nginx/1.9.13	Apache/2.2.14	Tengine/2.1.2
1	13191.46	4208.08	2472.8	4282.31
2	14237.18	4395.69	2466.09	4013.62
3	14650.89	4245.99	2471.7	4346.52
4	16023.53	4234.76	2454.81	4152.04
5	14409.31	4206.19	2469.85	4381.55
6	15535.74	4184.32	2458.29	4013.12
7	14893.44	4110.75	2471.7	4313.44
8	14581.95	4406.23	2467.72	4014.66
9	14572.83	4171.1	2481.25	4250.61
10	13868.79	4100.61	2479.77	4209.85



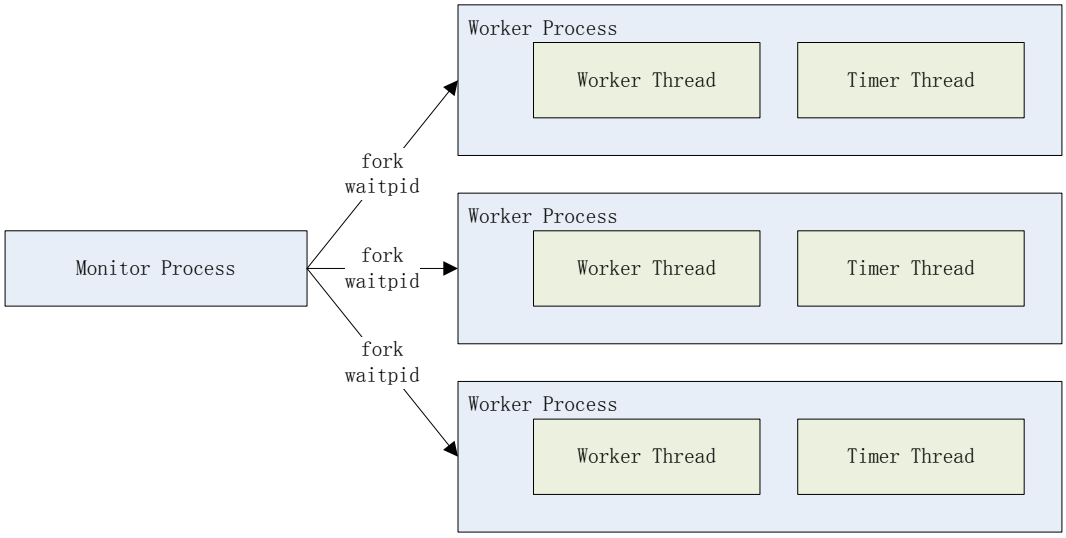
结论：

在开启 Keep-Alive 和 gzip 压缩、中型文件（约 3.3KB）的场景下，hetao 比 nginx 足足快了近 3.5 倍 ^_^

（现代浏览器一般都开启 Keep-Alive 和压缩，3.3KB 也算是普遍的网页大小）

7 内部实现

7.1 系统结构



hetao 进程结构：

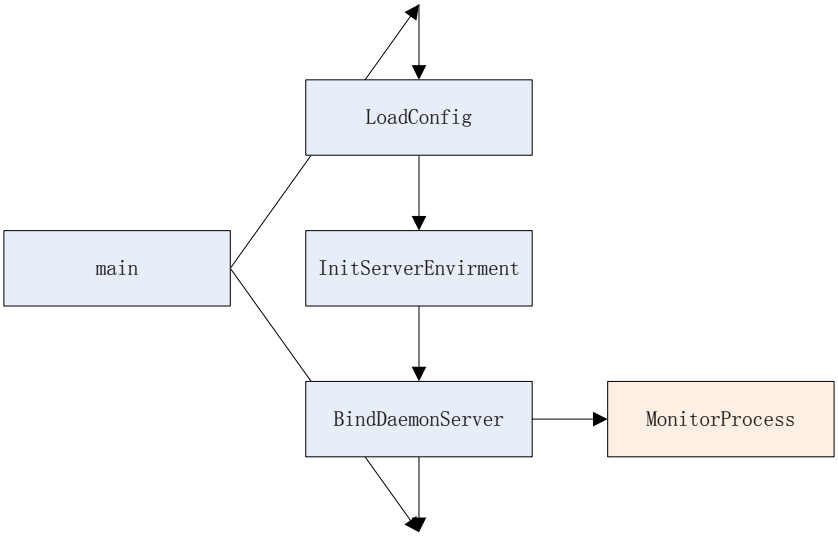
- 管理进程，负责创建、监管工作进程，负责传递 signal 管理命令。
- 工作进程

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。

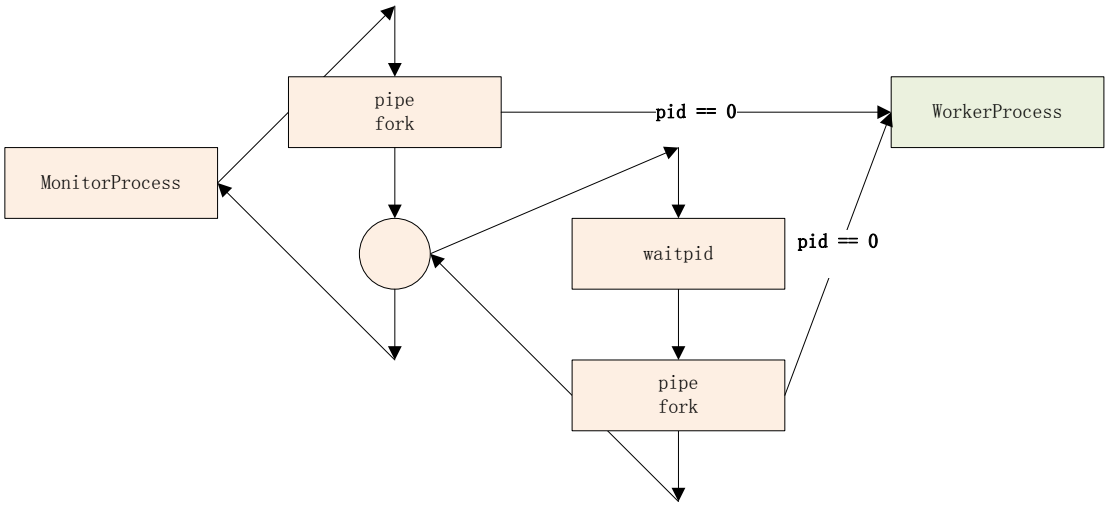
7.2 函数调用关系图

7.2.1 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。

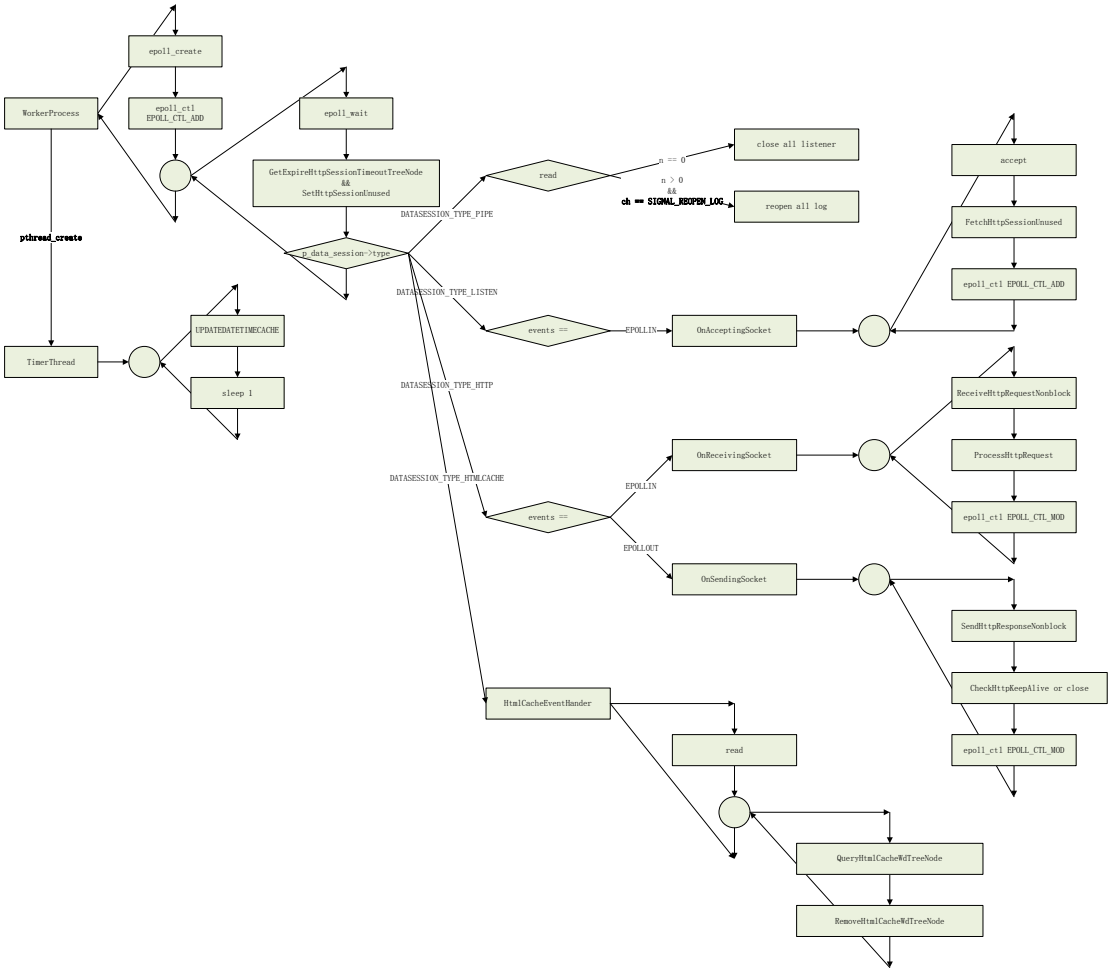
7.2.2 管理进程



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。

7.2.3 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。