第 1 章 轻量级的 HTTP 服务器 Nginx

1. Nginx 的模块从结构上分为核心模块、基础模块和第三方模块。HTTP 模块、MAIL 模块等属于核心模块，HTTP FastCGI 模块属于基础模块，Notice 模块属于第三方模块。
2. HttpGzip 模块支持在线实时压缩输出数据流（也就是 my.cnf 中的 gzip on 内容）。
3. 编译 Nginx 时，默认以 debug 模式进行，而在 debug 模式下会插入很多跟踪和 ASSERT 之类的信息。编译前在源码目录下的 auto/cc/gcc 文件找到如下几行并删除，即可取消 debug 模式，减少编译文件大小：

```
# debug
CFLAGS=" $CFLAGS -g"
```

4. TCMalloc 全称为 THread-Caching Malloc，与标准的 glibc 库的 Malloc 相比，TCMalloc 库在内存分配效率和速度上要高很多，这在很大程序上提高了服务器的高并发情况下的性能，从而降低了系统的负载。这个库应该不用配置，只安装就可以了。
5. Nginx 不支持对外部程序的直接调用和解析，所有的外部程序（包括 PHP）都必须通过 FastCGI 接口来调用。

第 2 章 高性能 HTTP 加速器 Varnish

1. Varnish 是一款高性能且开源的反向代理服务器和 HTTP 加速器。
2. pcre 库是为了兼容正则表达式。
3. VCL 是 Varnish 是配置语法，不是真正的编程语言，所以没有循环，没有自定义变量。
4. VCL 内置函数：

(1) vcl_recv 函数：用于接收和处理请求。当请求到达并被成功接收后调用，通过判断请求的数据来决定如何处理请求。

此函数一般以如下几个关键字结束：

- pass：表示进入 pass 模式，把请求控制权交给 vcl_pass 函数。
- pipe：表示进入 pipe 模式，把请求控制权交给 vcl_pipe 函数。
- Error code [reason]：表示返回“code”给客户端，并放弃处理该请求。“code”是错误标识，例如 200。“reason”是错误提示信息。

(2) vcl_pipe 函数：此函数在进入 pipe 模式时被调用，用于将请求直接传递至后端主机，在请求和返回的内容没有改变的情况下，将不变的内容返回给客户端，直到这个连接被关闭。

此函数一般以如下几个关键字结束：

- Error code [reason]。
- pipe。

(3) vcl_pass 函数：此函数在进入 pass 模式时被调用，用于将请求直接传递到后端主机。后端主机在应答数据后将应答数据发送给客户端，但不进行任何缓存，在当前连接下每次都返回最新的内容。

此函数一般以如下几个关键字结束：

- Error code [reason]。

- pass。

(4) lookup：表示在缓存中查找被请求的对象，并且根据查找的结果把控制权交给函数 vcl_hit 或函数 vcl_miss。

(5) vcl_hit 函数：在执行 lookup 指令后，在缓存中找到请求的内容后将自动调用该函数。

此函数一般以如下几个关键字结束：

- deliver：表示将找到的内容发送给客户端，并把控制权交给函数 vcl_deliver。
- Error code [reason]。
- pass。

(6) vcl_miss 函数：在执行 lookup 指令后，在缓存中没有找到请求的内容时自动调用该方法。函数可用于判断是否需要从后端服务器获取内容。

此函数一般以如下几个关键字结束：

- fetch：表示从后端获取请求的内容，并把控制权交给 vcl_fetch 函数。
- Error code [reason]。
- pass。

(7) vcl_fetch 函数：在后端主机更新缓存并且获取内容后调用该方法，接着，通过判断获取的内容来决定是将内容放入缓存，还是直接返回给客户端。

此函数一般以如下几个关键字结束：

- deliver：表示将找到的内容发送给客户端，并把控制权交给函数 vcl_deliver。
- Error code [reason]。
- pass。

(8) vcl_deliver 函数：将在缓存中找到请求的内容发送给客户端前调用此方法。

此函数一般以如下几个关键字结束：

- deliver：表示将找到的内容发送给客户端，并把控制权交给函数 vcl_deliver。
- Error code [reason]。

(9) vcl_timeout 函数：在缓存内容到期前调用此函数。

此函数一般以如下几个关键字结束：

- discard：表示从缓存中清除该内容。
- fetch。

(10) vcl_discard 函数：在缓存内容到期后或缓存空间不够时，自动调用该函数。

此函数一般以如下几个关键字结束：

- discard：表示从缓存中清除该内容。
- keep：表示将内容继续保留在缓存中。

5. VCL 处理流程图：

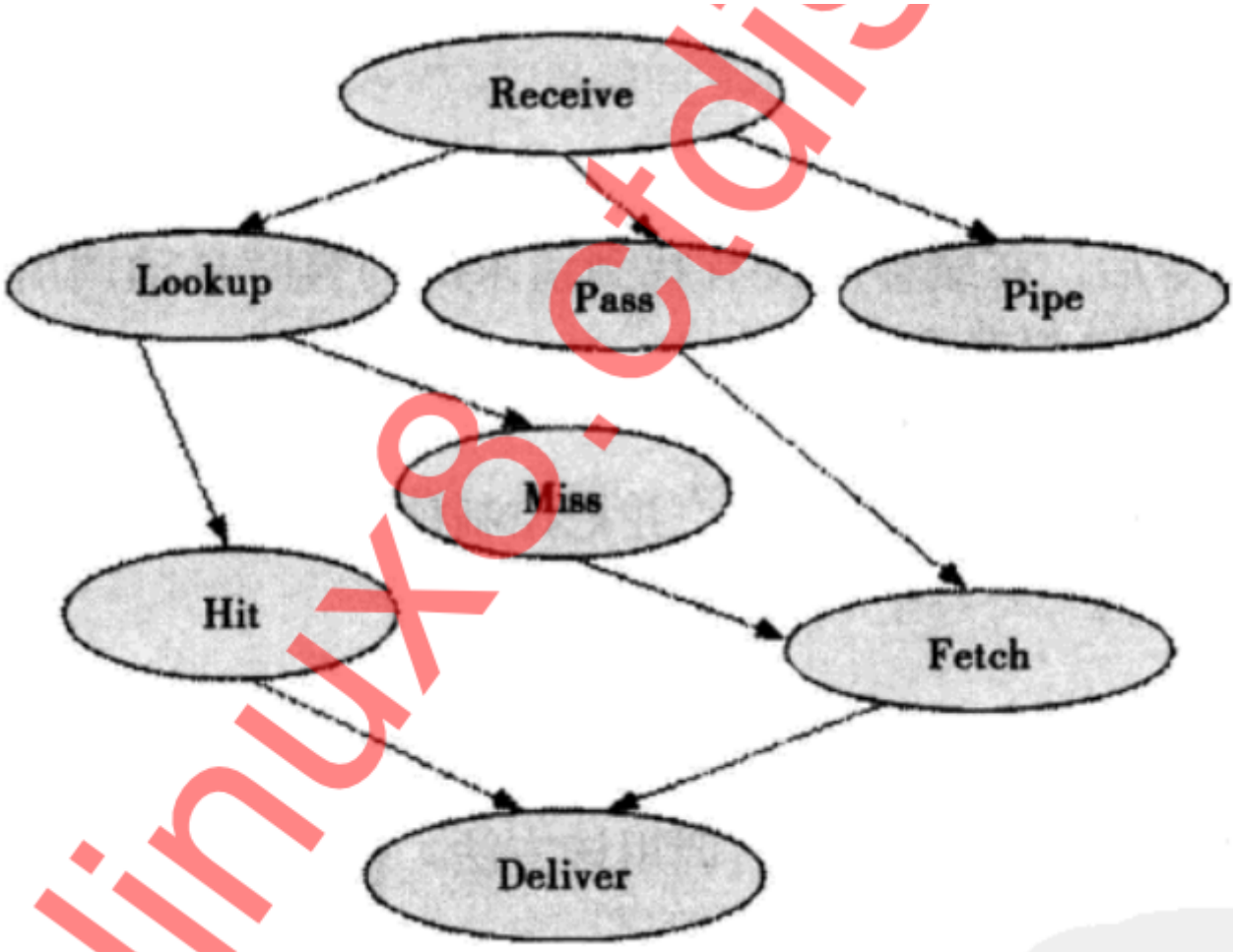

图 2-1　Varnish 处理 HTTP 请求的运行流程图

6. 内容的公用变量：

表 2-2　请求到达后可以使用的 VCL 内置的公用变量

公用变量名称	含　义
req.backend	指定对应的后端主机
server.ip	表示服务器端 IP
client.ip	表示客户端 IP
req.request	指定请求的类型，例如 GET、HEAD 和 POST 等
req.url	指定请求的地址
req.proto	表示客户端发起请求的 HTTP 协议版本
req.http.header	表示对应请求中的 HTTP 头部信息
req. restarts	表示请求重启的次数，默认最大值为 4

表 2-3　向后端主机请求时可以使用的 VCL 内置的公用变量

公用变量名称	含　义
beresp.request	指定请求的类型，例如 GET 合 HEAD 等
beresp.url	指定请求的地址
beresp .proto	表示客户端发起请求的 HTTP 协议版本
beresp .http.header	表示对应请求中的 HTTP 头部信息
beresp .ttl	表示缓存的生存周期，也就是 cache 保留多长时间，单位是秒

表 2-4　从 cache 或后端主机获取内容后可以使用的 VCL 内置的公用变量

公用变量名称	含　义
obj.status	表示返回内容的请求状态代码，例如 200、302 和 504 等
obj.cacheable	表示返回的内容是否可以缓存，也就是说，如果 HTTP 返回的是 200、203、300、301、302、404 或 410 等，并且有非 0 的生存期，则可以缓存
obj.valid	表示是否是有效的 HTTP 应答
obj.response	表示返回内容的请求状态信息
obj.proto	表示返回内容的 HTTP 协议版本
obj.ttl	表示返回内容的生存周期，也就是缓存时间，单位是秒
obj.lastuse	表示返回上一次请求到现在的间隔时间，单位是秒

表 2-5　对客户端应答时可以使用的公用变量

公用变量名称	含　义
resp.status	表示返回给客户端的 HTTP 状态代码
resp.proto	表示返回给客户端的 HTTP 协议版本
resp.http.header	表示返回给客户端的 HTTP 头部信息
resp.response	表示返回给客户端的 HTTP 状态信息

7.