Nginx在永洪产品中的应用

前言

Nginx均不属于永洪标准产品的组成部分，其作为高并发场景下反向代理和负载均衡的一种通用的解决方案在许多的项目中得到了实践，验证了其可用性。就其在永洪中的具体应用做一个简单的说明。 通常情况下，在多C的环境中，客户只希望以一个统一的界面为业务人员提供服务，这个时候反向代理 多个C的工作就可以交由Nginx 来实现，但是这里存在一个问题。如果被代理到不同C上的报告或者数据集的资源不同，就会给用户带来混淆和困扰，所以一般建议做反向代理的多个C都使用数据库系统的方式配置统一的元数据库，这样可以保证每个用户的请求被代理到不同的C上所看到的内容是统一的。

- Linux下Nginx的基本安装

  测试环境：CentOS7.6，nginx1.19.9

  1. nginx 下 载 ：

     http://nginx.org/en/download.html

  2. 文件存放路径

     路径可自己指定

  3. 解压

     tar -zxvf nginx-1.19.9.tar.gz

  4. 解压完成后进入文件目录可以看到如下文件目录

     ```
     [root@localhost nginx-1.19.9]# ls
     auto  CHANGES  CHANGES.ru  conf  configure  contrib  html  LICENSE  man  README  src
     ```

  5. 安装Linux相关系统依赖

     yum -y install gcc pcre pcre-devel openssl openssl-devel zlib-devel gd gd-devel perl perl-ExtUtils-Embed

     ubantu下将yum改为apt-get即可

  6. 安装nginx

     - 在nginx下先安装配置必要的nginx依赖模块，Linux不会自动配置nginx的相关模块，需 要手动配置，如果这些模块不配置好，后续在使用时，nginx会报错，如果某些模块被遗漏了，后续在安装时会覆盖先前的安装，如果后续确实需要安装其他模块，要先做好备 份。

       ```
       ./configure --prefix=/usr/nginx/nginx-1.19.9/NGINX --with-http_stub_status_module --with-http_realip_module --with-http_ssl_module --with-http_gzip_static_module --with-mail_ssl_module --with-stream
       ```

       其中--prefix=/usr/nginx-1.19.9/NGINX为指定的相关配置文件路径，建议配置，后续会 用到

       *注：安装时必须配置相应的依赖，不然后续使用时会报错

     - 编译安装：make&&make install
     - 启动停止与重启（此安装方式需要在nginx解压目录下）

       如果配置了上文路径，启动命令为：./usr/nginx/nginx-1.19.9/NGINX/sbin/nginx -c /usr/nginx/nginx-1.19.9/conf/nginx.conf

如果未配置：nginx启动主程序将会被安装在默认路径：/usr/local/nginx/sbin/下，此时 执行：./usr/local/nginx/sbin/nginx -c /usr/nginx/nginx-1.19.9/conf/nginx.conf可启动nginx。-c为指定配置文件的路径，也可。

停止：./nginx -s stop

重启：./nginx -s reload

此时nginx已经基本安装完成。安

- 装注意事项

1. 联网时使用yum指令安装依赖的软件包
2. 无法联网则需要使用rpm来安装，没有系统光盘则需要单独下载对应系统版本的rpm文件。
3. rpm -ivh指令，rpm文件在系统光盘的Packages目录下。

7. nginx进程管理

程序主目录位于/usr/local/nginx/（或自行指定的安装目录下），该目录下的内容分别为conf(主配置文件目录)、html(网站根目录)、logs(日志文件目录)、sbin(主程序目录)。

- 启动主程序(目录下执行./nginx)
  /usr/local/nginx/sbin/nginx
- 指定配置文件启动主程序(首次启动)
  /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf
- 关闭主程序(目录下执行./nginx -s stop)
  /usr/local/nginx/sbin/nginx -s stop
- 重新加载设置(目录下执行./nginx -s reload)
  /usr/local/nginx/sbin/nginx -s reload kill指令
- 发送信号给nginx进程号
  Kill -QUIT cat /usr/local/nginx/sbin/nginx/logs/nginx.pid Kill -HUP cat /usr/local/nginx/sbin/nginx/logs/nginx.pid

- Nginx相关配置

```
#设置用户与组
#user  nobody;
#user nginx nginx;

#工作进程：数目。根据硬件调整，通常等于CPU数量或者2倍于CPU。
#可以通过ps aux |grep nginx查看
worker_processes  2;

#全局错误日志、日志级别及PID文件
#error_log   logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid（进程标识符）：存放路径。
pid        logs/nginx.pid;

#指定进程可以打开的最大描述符：数目。
worker_rlimit_nofile 65535;
#补充说明：
#这个指令是指当一个nginx进程打开的最多文件描述符数目，
#理论值应该是最多打开文件数（ulimit -n）与nginx进程数相除，
#但是nginx分配请求并不是那么均匀，所以最好与ulimit -n 的值保持一致。
# 现在在linux 2.6内核下开启文件打开数为65535，
#worker_rlimit_nofile就相应应该填写65535。
#这是因为nginx调度时分配请求到进程并不是那么的均衡，
```

```
#所以假如填写10240，总并发量达到3-4万时就有进程可能超过10240了，这时会返回502错误。

events {

                        #使用epoll的I/O 模型。linux建议epoll，FreeBSD建议采用kqueue，window下不指定。
        use epoll;
        #补充说明:
        #与apache相类，nginx针对不同的操作系统，有不同的事件模型
        #A）标准事件模型
        #Select、poll属于标准事件模型，如果当前系统不存在更有效的方法，nginx会选择select或poll
        #B）高效事件模型
                        #Kqueue：使用于FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 和 MacOS X.使用双处理器的MacOS X系统使用kqueue可能会造成内核崩溃。
        #Epoll：使用于Linux内核2.6版本及以后的系统。
        #/dev/poll：使用于Solaris 7 11/99+，HP/UX 11.22+ (eventport)，IRIX 6.5.15+ 和 Tru64 UNIX 5.1A+。
                #Eventport：使用于Solaris 10。 为了防止出现内核崩溃的问题， 有必要安装安全补丁

        #每个工作进程的最大连接数量。
        worker_connections              65535;
        #根据硬件调整，和前面工作进程配合起来用，尽量大，
        #但是别把cpu跑到100%就行。每个进程允许的最多连接数，
        #理论上每台nginx服务器的最大连接数为。worker_processes*worker_connections

        #multi_accept on;
}


http {
        #mime.types为文件类型定义文件
        include                 mime.types;
        #默认文件类型
        default_type            application/octet-stream;

        #日志格式设置。
        log_format      main    '$remote_addr - $remote_user[$time_local] "$request" '
                                '$status $body_bytes_sent "$http_referer" '
                                '"$http_user_agent" "$http_x_forwarded_for"'
                                        ' ->$upstream_addr [$request_time $upstream_response_time]';
        #$remote_addr 与$http_x_forwarded_for 用以记录客户端的ip地址；
        #$remote_user：用来记录客户端用户名称；
        #$time_local： 用来记录访问时间与时区；
        #$request： 用来记录请求的url与http协议；
        #$status： 用来记录请求状态；成功是200，
        #$body_bytes_sent ：记录发送给客户端文件主体内容大小；
        #$http_referer：用来记录从那个页面链接访问过来的；
        #$http_user_agent：记录客户浏览器的相关信息；
```

```
71          #通常web服务器放在反向代理的后面，这样就不能获取到客户的IP地址了，
72          #通过$remote_add 拿到的IP地址是反向代理服务器的iP地址。
73          #反向代理服务器在转发请求的http头信息中，可以增加 x_forwarded_for 信息，
74          #用以记录原有客户端的IP地址和原来客户端的请求的服务器地址。
```

```
75
76          #需要用access_log指令指定日志文件的存放路径;
77          access_log          logs/access.log                main;
78          error_log           logs/error.log;
79          #access_log          /dev/null;
80          #error_log           /dev/null;
81
82          #sendfile指令指定 nginx 是否调用sendfile 函数（zero copy 方式）来输出文
    件，
83          #对于普通应用，必须设为on。如果用来进行下载等应用磁盘IO重负载应用，
84          #可设置为off，以平衡磁盘与网络IO处理速度，降低系统uptime。
85          sendfile                    on;
86
87          #此选项允许或禁止使用socke的TCP_CORK的选项，此选项仅在使用sendfile的时候使用
88          tcp_nopush                  on;
89          #tcp_nodelay                on;
90
91          #隐藏Nginx的版本号
92          server_tokens off;
93
94          #keepalive超时时间。
95          keepalive_timeout 300s;
96
97          #可以防止经过代理或者负载均衡服务器时丢失源IP。
98          #set_real_ip_from指定代理或者负载均衡服务器的IP，可以指定复数个IP。
99          #real_ip_header指定从哪个header头检索出要的IP地址。
100         #set_real_ip_from 192.168.1.0/24;
101         set_real_ip_from 192.168.2.1;
102         real_ip_header X-Forwarded-For;
103         real_ip_recursive on;
104
105         #open_file_cache_valid 1h;
106
107         ##cache##
108         #proxy_buffering on;
109         #proxy_buffer_size 128k;
110         #proxy_buffers 16 256k;
111         #proxy_busy_buffers_size 256k;
112         #proxy_temp_file_write_size 512k;
113         #proxy_temp_path /usr/local/nginx/temp_dir;
114         #proxy_cache_path /usr/local/nginx/cache levels=1:2
    keys_zone=cache_one:300m inactive=1d max_size=30g;
115         ##end##
116
117         #客户端请求读取超时时间
118         #client_header_timeout 10;
119         #客户端保持活动的超时时间
120         #client_body_timeout 15;
121
122         #响应客户端的超时时间
123         send_timeout 300s;
124         #reset_timedout_connection on;
125
126         client_max_body_size 100M;
127         client_body_buffer_size 128k;
128
129         #限制每个ip并发连接
```

```
130         #limit_conn_zone $binary_remote_addr zone=addr:1000m;
```

```
        #limit_conn addr 10000;

    gzip  on;
    #是否采用压缩功能，将页面压缩后传输更节省流量
    #gzip  on;
    #send_timeout 60s;
    #open_file_cache max=1000000 inactive=30s;
    #open_file_cache_valid 1h;

            #upstream名称，bi.com，名称可以随意配置，只会在配置文件中用到，服务使用者接触不到。
    upstream bi.com {
                #tengine支持的请求转发策略，每次会话都会生成唯一cookie，跟机器IP无关，没有ip_hash存在的问题。如果是Tengine请使用此配置。
            #session_sticky;
            #session_sticky cookie=bi domain=bi.com mode=insert fallback=off option=indirect path=/ option=indirect;
            server 192.168.0.122:8085 weight=1;
            server 192.168.0.122:8080 weight=1;
            #server 10.48.0.186:8080 weight=1;
                #请求转发策略名称，nginx支持多种请求转发策略，产品支持的目前只有这一种，会把同一来源的请求（同一C段的IP）转发到同一个C节点。但是如果是在局域网环境，如果IP是同一C段的，则所有请求都会访问同一台C节点，无法起到负载均衡作用。另外，如果客户请求到转发服务器之前还经过了其它代理，也起不到负载均衡的作用，因为做hash取模的ip都是同样的代理服务器的ip。
            #ip_hash;
    }


    server {

                #监听9000端口（建议配置），当访问该tengine服务器的9000端口时对请求进行拦截处理，转发到C节点1/C节点2
            listen          80;
                #接收客户请求的ip或者域名，一般都是nginx部署的服务器ip(127.0.0.1)或域名
            server_name 127.0.0.1 bi.com localhost;

            #配置防盗链接
            #location ~* ^.+\.(gif|jpg|png|swf|flv|rar|zip)$ {
                #valid_referers none blocked mobile.huaan.com.cn;
            #    valid_referers none blocked HT0x02.huaan.com.cn;
            #    if ($invalid_referer) {
            #        rewrite ^/ [img]http://mobile.huaan.com.cn/images/default/logo.gif[/img];
            #        return 403;
            #    }
            #}

            #代理服务器可以缓存C节点的静态文件，无需再转发请求到C节点获取静态文件。
            #location ~ .*\.(gif|jpg|htm|html|css|js|flv|ico|swf)(.*) {
                #缓存名称为bi.com的upstream对应的服务的静态文件
                #proxy_pass http://bi.com;
                #proxy_redirect off;
                #proxy_set_header Host $host;
                #30天内都代理的静态文件都从代理服务器获取；超过30天，代理服务器从C节点重新获取新的静态文件。如果产品jar更新比较频繁，可以设置较短的缓存时间。
```

```
177                     #expires 30d;
```

```
                                #websocket相关配置，产品移动端使用了websocket，需要配置这个才能正常使用。
                    #proxy_set_header Upgrade $http_upgrade;
                    #proxy_set_header Connection "upgrade";
                #}

            #拦截通过/bi/Viewer路径进来的请求
            location /bi {
                    #把请求随机转发到upstream bi.com配置下的server中其中的一个
                    proxy_pass http://bi.com/bi;
                    proxy_cookie_path /bi /;
                    proxy_set_header Host $host;
                            proxy_set_header X-Real-IP $remote_addr;
                    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                                #websocket相关配置，产品移动端使用了websocket，需要配置这个才能正常使用。
                    proxy_http_version 1.1;
                    proxy_set_header Upgrade $http_upgrade;
                    proxy_set_header Connection "upgrade";
                    #client_max_body_size 100m;
                    #后端服务器连接的超时时间_发起握手等候响应超时时间，Nginx默认60秒，可以不配置
                    proxy_connect_timeout 300s;
                        #连接成功后_等候后端服务器响应时间_其实已经进入后端的排队之中等候处理（也可以说是后端服务器处理请求的时间），Nginx默认60秒，可以不配置
                    proxy_send_timeout 300s;
                        #后端服务器数据回传时间_就是在规定时间之内后端服务器必须传完所有的数据，Nginx默认60秒，可以不配置
                    proxy_read_timeout 300s;
                            session_sticky_hide_cookie upstream=bi.com;
                    #proxy_set_header Cookie $http_cookie;
            }

                #location ^~ /fund-mg-web/ {
                    #return 403;
                #}
                #location ^~ /console/ {
                #     return 403;
                #}

                #默认交易
              #location / {


                        #limit_conn addr 5;
                        #限制每个连接下载速度
                        #limit_rate 1000000k;

                                #proxy_set_header Host $host:$server_port;
                        #proxy_set_header Host $host;
                                #proxy_set_header X-Real-IP $remote_addr;
                        #proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                        #proxy_redirect off;
                        #proxy_buffering off;
                        #proxy_pass http://wlsserver;
```

```
228                 #proxy_connect_timeout 300s;
```

```
                        #proxy_send_timeout 300s;
                        #proxy_read_timeout 300s;

            #}

                location /nginx {
                    stub_status on;
                    access_log on;
                    auth_basic "NginxStatus";
                }
            #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;
            }
                                        # proxy the PHP scripts to Apache listening on 127.0.0.1:80
            #
            #location ~ \.php$ {
            #    proxy_pass   http://127.0.0.1;
            #}

                            # pass the PHP scripts to FastCGI server listeningon 127.0.0.1:9000
            #
            #location ~ \.php$ {
            #    root           html;
            #    fastcgi_pass   127.0.0.1:9000;
            #    fastcgi_index  index.php;
            #    fastcgi_param  SCRIPT_FILENAME /scripts$fastcgi_script_name;
            #    include        fastcgi_params;
            #}

                                            # deny access to .htaccess files, if Apache's document root
            # concurs with nginx's one
            #
            #location ~ /\.ht {
            #    deny  all;
            #}
        }

        # another virtual host using mix of IP-, name-, and port-based configuration
        #
        #server {
        #    listen       8000;
        #    listen       somename:8080;
        #    server_name  somename  alias  another.alias;

        #    location / {
        #        root   html;
        #        index  index.html index.htm;
        #    }
        #}

```

```
    # HTTPS server 配置
    #
    #server {
    #    listen       443 ssl;
    #    server_name  localhost;

    #    ssl_certificate      cert.pem;
    #    ssl_certificate_key  cert.key;

    #    ssl_session_cache    shared:SSL:1m;
    #    ssl_session_timeout  5m;

    #    ssl_ciphers  HIGH:!aNULL:!MD5;
    #    ssl_prefer_server_ciphers  on;

    #    location / {
    #        root   html;
    #        index  index.html index.htm;
    #    }
    #}

}
# 每次修改nginx.conf配置文件后，x
```

- 附录表
  - **1.** nginx分内置模块和第三方模块，内置模块分主模块和事件模块。
    2. 表1给出的模块为默认自动编译的，可以使用--without参数禁用。
    3. 表2是附加模块，需要在编译时通过--with参数手动开启。
    4. 还可以通过--add-module=/path/module的方式编译第三方模块。
  - 表1

模块名称	描述	禁用选项
Core	Nginx核心功能	--without-http
Access	基于IP的访问控制	--without-http_access_module
Auth Basic	HTTP用户认证模块	--without-http_auth_basic_module
Auto Index	自动目录索引	--without-http_autoindex_module
Browser	描述用户代理	--without-http_browser_module
Charset	重新编码网页	--without-http_charset_module
Empty GIF	内存中存放一个图片	--without-http_empty_gif_module
FastCGI	FastCGI支持	--without-http_fastcgi_module
Geo	支持IP变量设置	--without-http_geo_module
Gzip	Gzip压缩	--without-http_gzip_module
Limit Requests	限制客户端连接频率	--without-http_limit_req_module
Limit Conn	会话的并发连接	--without-http_limit_conn_module
Map	设置变量	--without-http_map_module
Memcached	Memcache支持	--without-http-memcache_module
Referer	基于Referer头部信息过滤	--without-http_referer_module
Rewrite	使用正则表达式重写请求	--without-http_rewrite_module
SCGI	支持SCGI协议	--without-http_scgi_module
Upstream	负载均衡	--without-http_upstream_ip_hash_module
Headers	设置http响应的头部信息	-
Index	首页	-
Log	自定义日志	-

- 表2

模块名称	描述	开启选项
Embedded Perl	支持Perl	--with-http_perl_module
FLV	支持Flash视频	--with-http_flv_module
GeoIP	通过IP变量实现负载均衡	--with-http_geoip_module
Google Perftools	支持谷歌的性能优化工具	--with- google_perftools_module
Gzip Precompression	压缩静态文件	--with- http_gzip_static_module
Image Filter	转换图形的过滤器	--with- http_image_filter_module
MP4	支持MP4	--with-http_mp4_module
Real IP	使用Nginx作为后端服务器	--with-http_realip_module
Secure Link	使用密钥保护页面	--with- http_secure_link_module
SSL	支持HTTPS/SSL	--with-http_ssl_module
Stub Status	查看服务器状态	--with- http_stub_status_module
WebDAV	支持WebDAV	--with-http_dav_module
Core	邮件代理功能	--with-mail
	邮件代理功能	--without-mail_pop3_module
	邮件代理功能	--without-mail_imap_module
	邮件代理功能	--without-mail_smtp_module
SSL	支持SSL/TLS加密邮件协议	--with-mail_ssl_module

- 表3：kill指令发送信号给nginx进程号

信号名称	描述
TERM,INT	快速关闭
HUP	重启应用新的配置文件
USR2	升级程序
QUIT	优雅地关闭，保持现有的客户端连接
USR1	重新打开日志文件
WINCH	优雅地关闭工作进程

- 表4：http响应状态码

代码	含义
100	请求已接收，客服端可以继续发送请求
101	Switching Protocals服务器根据客户端的请求切换协议
200	一切正常
201	服务器已经创建了文档
202	已经接受了请求，但处理尚未完成
203	文档正常返回，但一些头部信息可能不正确
300	客户端请求的资源可以在多个位置找到
301	客户端请求的资源可以在其他位置找到
305	使用代理服务
400	请求语法错误
401	访问被拒绝
401.1	登录失败
403	资源不可用
403.6	IP地址被拒绝
403.9	用户数过多
404	无法找到指定资源
406	指定资源已找到，但MIME类型与客户端要求不兼容
407	要求进行代理身份验证
500	服务器内部错误
500.13	服务器太忙
501	服务器不支持客户端请求的功能
502	网关错误
503	服务不可用
504	网关超时，服务器处于维护或者负载过高无法响应
505	服务器不支持客户端请求的HTTP版本