在Nginx使用Lua扩展功能

【问题背景】

在我们的线上系统中，使用Nginx作为反向代理服务器，记录了请求的日志，同时在后端的upstream server上也打开了Tomcat日志。有时候在排查性能抖动的问题时，两边的数据有时往往对不上，因为没有一个唯一的ID可以使用，而业务系统中SeqId是在应用逻辑中生成的无法直接在请求一开始进入我们的服务端时就记录下来，为排查问题带了诸多不便。因此想直接在Nginx中生成SeqId，这样请求一进来就生成并贯穿始终，对后续排查问题非常有帮助。

【想法】

【意义】

网上搜索到比较多的就是使用Lua脚本来对Nginx进行功能扩展。但最初稍微看了一下，觉得有点麻烦还要重新学习一门新的语言，于是想找一下看有没有其它语言的组件，最好是我熟悉的Java语言。没想到还被我从Nginx的扩展列表中找到了，叫作nginx-clojure，不过看了一下有点重，使用了Ring框架，虽然之前玩过一段Clojure，但还是感觉比较麻烦，最终还是转回了Lua，其实它还是挺简单的，很快就能上手。

第一步是先安装Nginx需要的一些类库：

```
yum install gcc
yum install gcc-c++
```

然后，编译安装pcre-8.34和zlib-1.2.8，接着安装Lua编译库LuaJit-2.0.3：

```
make PREFIX=/usr/install/luajit
make install PREFIX=/usr/install/luajit
```

在 `/etc/profile` 文件中增加环境变量，并执行 `source /etc/profile` 使之生效：

```
export LUAJIT_LIB=/usr/install/luajit/lib
export LUAJIT_INC=/usr/install/luajit/include/luajit-2.0
```

下载 ngx_devel_kit 和 lua-nginx-module，最后编译Nginx：

```
# nginx-1.6.0
./configure –prefix=/usr/install/nginx –add-module=/usr/install/ngx_devel_kit-0.2.

ln -s /usr/local/lib/libpcre.so.1 /lib64/libpcre.so.1
ln -s /usr/install/luajit/lib/libluajit-5.1.so.2 /lib46/libluajit-5.1.so.2
```

启动Nginx `sbin/nginx` ，浏览器输入http://localhost测试

在nginx.conf文件的server部分增加Lua脚本进行测试

```
    location /lua {
     set $resp "Hello,Nginx & Lua !";
     content_by_lua '
       ngx.header.content_type = "text/plain";
       ngx.say(ngx.var.resp);
     ';}
```

测试执行结果：

配置热加载

```
> sbin/nginx -s reload
> curl http://localhost/lua
Hello,Nginx & Lua !
```

生成SeqId： 请求唯一序列号

```
location / {
rewrite_by_lua '
              -- generate seq id
              if ngx.var.uri == "/riskService" then
                local ts=ngx.now()*1000;
                local seq=math.random(10000000,100000000);
                local args=ngx.req.get_uri_args();
                args["nginx_seq_id"]=ts.."-"..seq;
                ngx.req.set_uri_args(args);
              end
            ';

proxy_pass http://backup;
}
```

Nginx的执行过程有十一个，主要的执行过程包括：

- **rewrite** ：重写URI及参数，我们的实现就是在这个过程中，Lua脚本对应的是rewrite_by_lua，它会在Nginx的rewrite内容的最后面执行
- **access** ：主要做一些权限校验和控制，Lua脚本对应的是access_by_lua
- **content** ：对内容的处理，Lua脚本对应的是content_by_lua

因为Lua脚本里的时间戳函数 `os.time()` 只能精确到秒，如果要想到毫秒必须自己安装另外的三方库luasocket，为了取一个毫秒数不太值得，查了一下文档发现Nginx内置的 `ngx.now()` 也可以得到毫秒数。用 `math.random(10000000,100000000)` 得到一个8位随机数，以避免同一时刻生成的SeqId会重复。调用 `ngx.req.get_uri_args()` 可以得到GET请求中URL里的所有参数，返回结果是一个Table，和Java中的Map很类似，然后直接新增一个KV值，将新生成的SeqId添加进行，再调用 `ngx.req.set_uri_args(args)` 进行重设，就完成了SeqId的生成。

【途中遇到的问题】

完成上述配置后，Nginx再把请求转发到Tomcat时，就会带上这个SeqId的参数，但是你会发现Nginx自己的access日志中并没有生成，这里需要自己获取一下这个参数 `$arg_nginx_seq_id` ，因为日志中的 `$request` 取的应该是原始请求URI，是无法显示出自己在rewrite阶段添加的参数。

```
log_format  main  '$remote_addr - [$time_local] "$request seq_id=$arg_nginx_seq_i
                  '$status $body_bytes_sent '
                  '"$request_time" "$upstream_response_time" "$upstream_addr
```

`arg_XXX` 是系列变量，XXX指的是提交的参数名字，可以使用这样的方式方便地获取请求值，类似的还有 `cookie_XXX` ，更多请参考第一篇参考资料。

```
log_format  main  '$remote_addr - [$time_local] "$request seq_id=$arg_nginx_seq_id" '
                  '$status $body_bytes_sent '
                  '"$request_time" "$upstream_response_time" "$upstream_addr" "$request_body "';
```

参考资料

- agentzh 的 Nginx 教程
- HttpLuaModule文档说明
- Lua 5.2 Reference Manual

yikebocai / 2014-11-13

Published under (CC) BY-NC-SA in categories tech tagged with nginx lua