OpenResty 入门实验

目录

一、实验环境准备

从 ftp://192.168.1.103/pub/下载工具, ftp 地址改为实际值
安装 virtualbox
导入模板 centos7-lab.ova(已经安装了 mysql, redis, ppt 上的代码也在其中)
root 用户密码 rain1989
PPT 代码已经放在/home/demo 目录下

二、实验

1. OpenResty 源码编译安装、配置文件修改、熟悉常用 nginx 命令

1) OpenResty 安装(centos, root 用户)

a) 安装包下载
wget https://openresty.org/download/openresty-1.9.15.1.tar.gz
b) 依赖安装(提供的环境已经安装过)
yum install readline-devel pcre-devel openssl-devel gcc
c) 解压
tar zxvf openresty-1.9.15.1.tar.gz
cd openresty-1.9.15.1
d) configure
./configure --prefix=/opt/openresty --with-luajit --with-http_iconv_module -j2
e) make
make -j2
f) make install
make install

2) 常用命令

将 nginx 加到 PATH 环境变量
a) configure 选项
./configure --help
b) 重启
nginx -s reload
c) 停止
nginx -s stop
4) 检查配置文件
nginx -t
d) 查看安装的组件
nginx -V
e) 启动 nginx 的几种方式
nginx

nginx -c /xx/xxx/nginx.conf

nginx -p /xx/xxx/nginx

f) 查看 nginx 是否启动成功

curl http://127.0.0.1

g) 查看进程

ps -ef|grep nginx

3) 修改配置文件

```
user  demo;
worker_processes  auto;
worker_cpu_affinity auto;

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
error_log  logs/error.log  info;

pid        logs/nginx.pid;

worker_rlimit_nofile 65535;

events {
    worker_connections  65535;
}


http {
```

重启 nginx -s reload

2. hello world

修改 nginx.conf

```
server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

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

    #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;
    }

    location /helloworld {
        content_by_lua_block {
            ngx.say("HelloWorld")
        }
    }
```

nginx -s reload

curl http://127.0.0.1/helloworld

ab -k -c 100 -n 1000000 http://127.0.0.1/helloworld

3. OpenResty 的分阶段处理

1) 执行阶段

修改 nginx.conf

```
location /helloworld {
    content_by_lua_block {
        ngx.say("HelloWorld")
    }
}

location /mixed {
    set_by_lua $a 'ngx.log(ngx.INFO, "set_by_lua")';
    rewrite_by_lua 'ngx.log(ngx.INFO, "rewrite_by_lua")';
    access_by_lua 'ngx.log(ngx.INFO, "access_by_lua")';
    header_filter_by_lua 'ngx.log(ngx.INFO, "header_filter_by_lua")';
    body_filter_by_lua 'ngx.log(ngx.INFO, "body_filter_by_lua")';
    log_by_lua 'ngx.log(ngx.INFO, "log_by_lua")';
    content_by_lua 'ngx.log(ngx.INFO, "content_by_lua")';
}
```

nginx -s reload

curl http://127.0.0.1/mixed

可以查看日志里面的打印信息

/opt/openresty/nginx/logs/error.log

2) 阶段之间传递变量

```
location /mixed {
    set_by_lua $a 'ngx.log(ngx.INFO, "set_by_lua")';
    rewrite_by_lua 'ngx.log(ngx.INFO, "rewrite_by_lua")';
    access_by_lua 'ngx.log(ngx.INFO, "access_by_lua")';
    header_filter_by_lua 'ngx.log(ngx.INFO, "header_filter_by_lua")';
    body_filter_by_lua 'ngx.log(ngx.INFO, "body_filter_by_lua")';
    log_by_lua 'ngx.log(ngx.INFO, "log_by_lua")';
    content_by_lua 'ngx.log(ngx.INFO, "content_by_lua")';
}

location /ngx_ctx {
    rewrite_by_lua '
        ngx.ctx.foo = 76
    ';
    access_by_lua '
        ngx.ctx.foo = ngx.ctx.foo + 3
    ';
    content_by_lua_block {
        ngx.say(ngx.ctx.foo)
    }
}
```

nginx -s reload

curl http://127.0.0.1/ngx_ctx

4. 异步非阻塞操作 mysql

在目录/opt/openresty/nginx/下面创建一个目录 lua 用于放后面用到的 lua 脚本

修改 nginx.conf

```
http {
    include       mime.types;
    default_type  application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    #gzip  on;

    lua_package_path 'lua/?.lua;;';

    server {
        listen       80;
        server_name  localhost;
```

```
location /mysql_create {
    content_by_lua_file lua/mysql_create.lua;
}

location /mysql_insert {
    content_by_lua_file lua/mysql_insert.lua;
}

location /mysql_select {
    content_by_lua_file lua/mysql_select.lua;
}
```

1) 建表

代码 mysql_create.lua

curl http://127.0.0.1/mysql_create

2) 插入

代码 mysql_insert.lua

curl 'http://127.0.0.1/mysql_insert?name=jack&email=jack@163.com&password=iefioweio'

3) 查询

代码 mysql_select.lua

curl http://127.0.0.1/mysql_select?name=zhuyu

ab -k -c 100 -n 100000 http://127.0.0.1/mysql_select?name=zhuyu

5. 异步非阻塞操作 redis

```
location /redis_set {
    content_by_lua_file lua/redis_set.lua;
}

location /redis_get {
    content_by_lua_file lua/redis_get.lua;
}
```

1) redis set

redis_set.lua

curl 'http://127.0.0.1/redis_set?key=ad0001&value=插屏广告'

ab -n 500000 -c 100 -k 'http://127.0.0.1/redis_set?key=ad0001&value=插屏广告'

2) redis get

redis_get.lua

curl 'http://127.0.0.1/redis_get?key=ad0001'

ab -n 500000 -c 100 -k 'http://127.0.0.1/redis_get?key=ad0001'

6. OpenResty 的缓存、共享内存的使用

1) 缓存、共享内存

```
lua_package_path 'lua/?.lua;;';

lua_shared_dict cache_ngx 128m;
lua_shared_dict my_locks 100k;

server {
    listen       80;
    server_name  localhost;
```

```
location /iredis_get {
    content_by_lua_file lua/iredis_get.lua;
}

location /iredis_get2 {
    content_by_lua_file lua/iredis_get2.lua;
}
```

这里面用到了 iredis.lua(基于官方 redis.lua 封装，简化代码)，需要将其放在 resty 目录下，否则会报错

iredis_get 是未加缓存的

curl http://127.0.0.1/iredis_get

ab -n 500000 -c 100 -k http://127.0.0.1/iredis_get

iredis_get2 是使用缓存的

curl http://127.0.0.1/iredis_get2

ab -n 500000 -c 100 -k http://127.0.0.1/iredis_get2

2) 缓存失效风暴

```
lua_package_path 'lua/?.lua;;';

lua_shared_dict cache_ngx 128m;
lua_shared_dict my_locks 100k;

server {
    listen       80;
```

```
location /iredis_get3 {
    content_by_lua_file lua/iredis_get3.lua;
}
```

代码 iredis_get3.lua

curl http://127.0.0.1/iredis_get3