Computer Network lab7实验报告

姓名	学号
张洋彬	191220169
邮箱	完成日期
1016466918@qq.com	2021.6.14

1、实验名称

Lab 7: Content Delivery Network

2、实验目的

1、实现存储内容的缓存服务器。

2、实现能找到最近的缓存服务器的 DNS 服务器。

3、实验进行

3.1 Preparation

文件结构如下图所示：

安装我们需要的库：

Install packages we need:

```
$ python3 -m pip install -r requirements.txt
```

3.2 DNS server

1、首先从dns_table.txt将数据写入

```
fo = open("dnsServer/dns_table.txt","r+")
lines = fo.readlines()
for line in lines :
  self._dns_table.append(line.split())#list type
  fo.close()
```

2、用户请求的域名在表中找不到对应的记录，此时需要返回(response_type, response_val)，值为(None, None)。

3、用户请求的域名作为CNAME类型记录在表中找到，此时需要直接返回("CNAME", "xxx.xxx.xxx")。

4、用户请求的域名可以在带有类型 A 记录列表的表中找到。

- 如果列表中只有一条记录，则直接返回("A"，那个IP地址)。
- 如果列表中有多个记录。您需要考虑客户端 IP 和服务器 IP 的地理位置。您可以使用 IP_Utils.getIpLocation(ip_str) 获取 IP 地址的纬度和经度信息。

> 如果在我们的数据库中找不到客户端 IP 地址的位置信息（即 IP_Utils.getIpLocation(ip_str) 返回 None），则需要对多个服务器采用随机负载均衡策略。
> 如果找到客户端 IP 地址的位置，则需要从记录列表中选择最近的缓存节点（CDN 节点）作为 response_val。

对带*的判断：（sub_domain）

```
flag=0
str=entry[0]
if entry[0][0]=='*':
  str=str.strip('*')
  str=str.strip('.')
  flag=1
  #entry[0].strip('.')
  if flag==1:
    if str in request_domain_name:
```

对不带*的判断：

```
str=str.strip('.')
#print(str)
if str ==request_domain_name:
```

按之前的逻辑写的代码如下：

```
response_type=entry[1]
                    length=len(entry)-2
                    if response_type=='A':
                        if length==1:
                            response_val=entry[2]
                        else:
                            if IP_Utils.getIpLocation(client_ip):

 client_ip_latitude,client_ip_longtitude=IP_Utils.getIpLocation(client_ip)
                                min=sys.maxsize
                                nearest_ip=None
                                num=0
                                while num<length:

 ip_latitude,ip_longtitude=IP_Utils.getIpLocation(entry[2+num])
                                    if ((ip_latitude-client_ip_latitude)*
(ip_latitude-client_ip_latitude)+(ip_longtitude-client_ip_longtitude)*
(ip_longtitude-client_ip_longtitude)<min):
                                        min=(ip_latitude-client_ip_latitude)*
(ip_latitude-client_ip_latitude)+(ip_longtitude-client_ip_longtitude)*
(ip_longtitude-client_ip_longtitude)
                                        nearest_ip=entry[2+num]
                                    num=num+1
                                response_val=nearest_ip#return nearest
                            else:#not in our database
                                i=random.randint(0,length-1)
                                response_val=entry[2+num]#random

                    elif response_type== 'CNAME':
                        response_val=entry[2]
```

先打开dns server，然后输入`python3 test_entry.py dns`,结果如下：

```
njucs@njucs-VirtualBox:~/lab-7-bbzunyi$ python3 test_entry.py dns
2021/06/14-16:10:37| [INFO] DNS server started
test_cname1 (testcases.test_dns.TestDNS) ... ok
test_cname2 (testcases.test_dns.TestDNS) ... ok
test_location1 (testcases.test_dns.TestDNS) ... ok
test_location2 (testcases.test_dns.TestDNS) ... ok
test_non_exist (testcases.test_dns.TestDNS) ... ok

----------------------------------------------------------------------
Ran 5 tests in 0.010s

OK
2021/06/14-16:10:38| [INFO] DNS server terminated
```

可见，通过了全部测试样例

3.3 Caching server

> 缓存服务器是 CDN 的核心。 缓存的工作原理是有选择地将网站文件存储在 CDN 的缓存代理服务器上，从附近位置浏览的网站访问者可以快速访问这些文件。 它维护一个本地缓存表（例如一个数据库）来存储所有缓存的内容。在本节中，我们将实现一个简单的 CDN 缓存服务器。

1、完成sendHeaders

从headers中读取header,然后send_header

```
self.send_response(HTTPStatus.OK,"'File is found'")
for header in self.headers:
  self.send_header(header[0],header[1])
  self.end_headers()
```

2、实现do_get和do_head

利用touchitem函数得到相应的headers和body，然后调用send_header和sendbody

```
item=None
item=self.server.touchItem(self.path)
self.headers=item[0]
body=item[1]
if self.headers is None:
  self.send_error(HTTPStatus.NOT_FOUND, "'File not found'")
  else:
    self.sendHeaders()
    self.sendBody(body)
```

do_head函数不用self.sendBody(body)

3、实现touchitem

检查path是否在cachetable中，如果在就直接返回headers和body，如果不在则调用requestMainServer，将返回值加入cachetable中

```
  if path in self.cacheTable.data:
    if self.cacheTable.expired(path)==False:
      return (self.cacheTable.getHeaders(path),self.cacheTable.getBody(path))
  else:
    response = self.requestMainServer(path)
    if response != None:
      headers = self._filterHeaders(response.headers)
      body = response.read()
      self.cacheTable.setHeaders(path,headers)
      self.cacheTable.appendBody(path,body)
      return (headers,body)
    return (None,None)
```

通过以下指令进行测试：

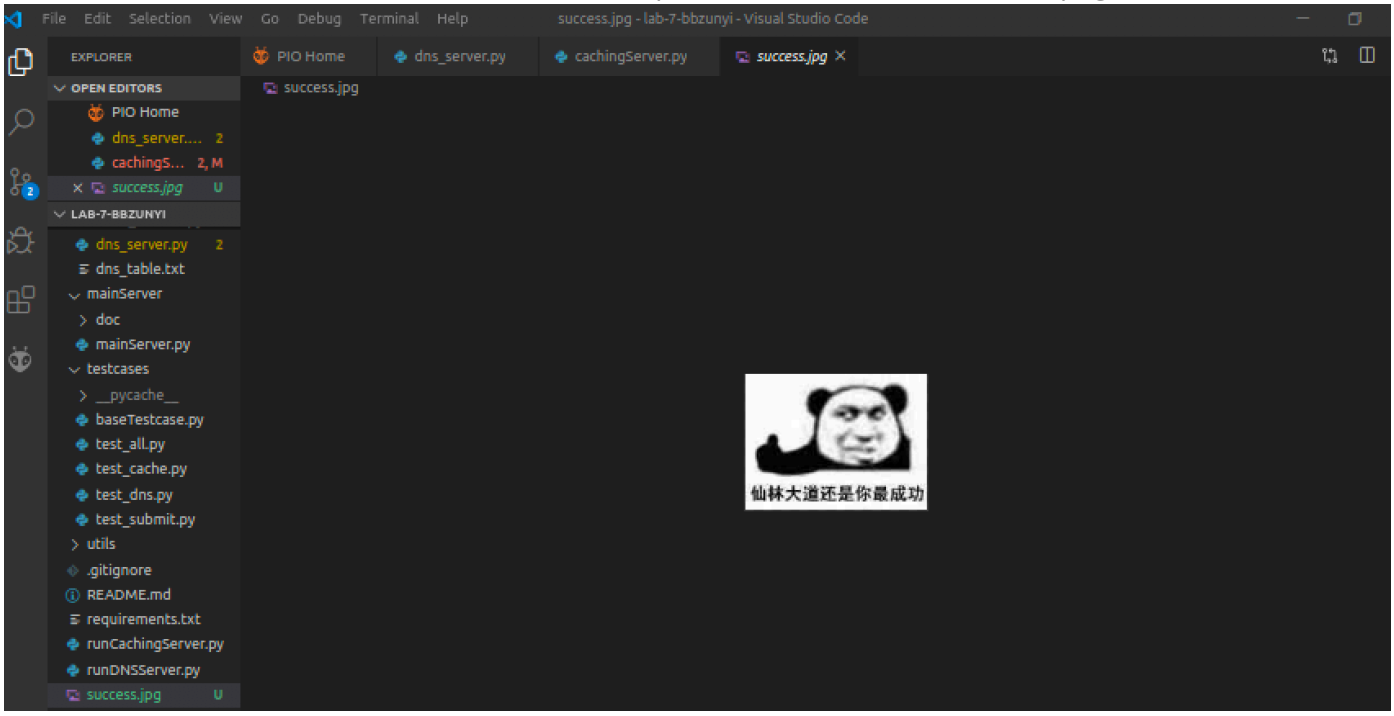
```
$ python3 mainServer/mainServer.py -d mainServer/
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
$ python3 runCachingServer.py localhost:8000
$ curl -O http://localhost:1222/doc/success.jpg
$ curl http://localhost:1222/nonexist
$ curl -I http://localhost:1222/doc/success.jpg
```

结果如下图所示：

```
njucs@njucs-VirtualBox:~/lab-7-bbzunyi$ curl -O http://localhost:1222/doc/success.jpg
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2125    0  2125    0     0   691k      0 --:--:-- --:--:-- --:--:--  691k
```

```
njucs@njucs-VirtualBox:~/lab-7-bbzunyi$ python3 mainServer/mainServer.py -d mainServer/
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [15/Jun/2021 01:20:22] "GET /doc/success.jpg HTTP/1.1" 200 -
127.0.0.1 - - [15/Jun/2021 01:20:38] code 404, message File not found
127.0.0.1 - - [15/Jun/2021 01:20:38] "GET /nonexist HTTP/1.1" 404 -
```

![截屏2021-06-15 上午12.00.36](/Users/mac/Desktop/截屏2021-06-15 上午12.00.36.png



可见，可以成功下载图片

输入`python3 test_entry.py cache`,结果如下，正确：

```
test_01_cache_missed_1 (testcases.test_cache.TestCache) ...
[Request time] 409.78 ms
ok
test_02_cache_hit_1 (testcases.test_cache.TestCache) ...
[Request time] 89.26 ms
ok
test_03_cache_missed_2 (testcases.test_cache.TestCache) ...
[Request time] 90.50 ms
ok
test_04_cache_hit_2 (testcases.test_cache.TestCache) ...
[Request time] 88.26 ms
ok
test_05_HEAD (testcases.test_cache.TestCache) ...
[Request time] 103.15 ms
ok
test_06_not_found (testcases.test_cache.TestCache) ...
[Request time] 51.88 ms
ok

----------------------------------------------------------------------
Ran 6 tests in 4.925s

OK
```

其他两个测试结果如下：

```
njucs@njucs-VirtualBox:~/lab-7-bbzunyi$ python3 test_entry.py dns
2021/06/15-01:33:44| [INFO] DNS server started
test_cname1 (testcases.test_dns.TestDNS) ... ok
test_cname2 (testcases.test_dns.TestDNS) ... ok
test_location1 (testcases.test_dns.TestDNS) ... ok
test_location2 (testcases.test_dns.TestDNS) ... ok
test_non_exist (testcases.test_dns.TestDNS) ... ok

----------------------------------------------------------------------
Ran 5 tests in 0.011s

OK
2021/06/15-01:33:46| [INFO] DNS server terminated
```

```
njucs@njucs-VirtualBox:~/lab-7-bbzunyi$ python3 test_entry.py all
2021/06/15-01:34:04| [INFO] DNS server started
2021/06/15-01:34:04| [INFO] Main server started
2021/06/15-01:34:04| [INFO] RPC server started
2021/06/15-01:34:04| [INFO] Caching server started
test_01_cache_missed_1 (testcases.test_all.TestAll) ...
[Request time] 92.07 ms
ok
test_02_cache_hit_1 (testcases.test_all.TestAll) ...
[Request time] 90.24 ms
ok
test_03_not_found (testcases.test_all.TestAll) ...
[Request time] 56.06 ms
ok

----------------------------------------------------------------------
Ran 3 tests in 2.309s

OK
2021/06/15-01:34:07| [INFO] DNS server terminated
2021/06/15-01:34:07| [INFO] Caching server terminated
2021/06/15-01:34:07| [INFO] PRC server terminated
2021/06/15-01:34:07| [INFO] Main server terminated
```

3.4 Deployment

在terminal里测试的结果在3.3中展示，下面是在openNetlab中的测试结果如下：

191220169_dns.log

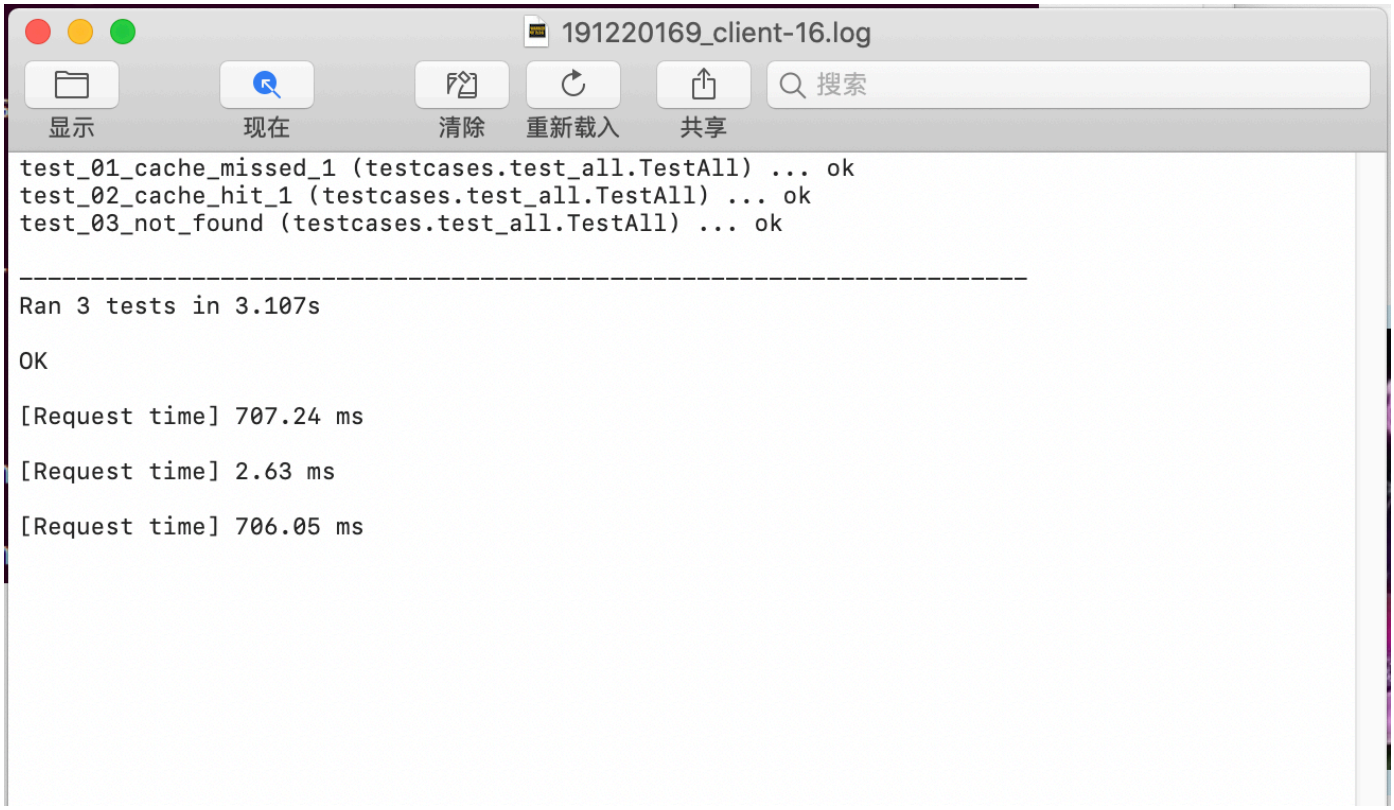
显示 现在 清除 重新载入 共享 搜索

```
2021/06/14-18:43:51| [INFO] DNS server started
DNS server serving on 0.0.0.0:8188
2021/06/14-18:43:53| [Info] Receving DNS request from '10.0.0.24' asking for
'stfw.localhost.computer.'
2021/06/14-18:43:55| [Info] Receving DNS request from '10.0.0.24' asking for
'stfw.localhost.computer.'
2021/06/14-18:43:55| [Info] Receving DNS request from '10.0.0.24' asking for
'stfw.localhost.computer.'
```

191220169_cache.log

显示 现在 清除 重新载入 共享 搜索

```
2021/06/14-18:43:51| [INFO] Caching server started
Caching server serving on http://0.0.0.0:8188
2021/06/14-18:43:54| [Info] Fetched '/doc/success.jpg' from main server
'20.188.122.123:8888'
2021/06/14-18:43:54| [From 10.0.0.24:54160] "GET /doc/success.jpg HTTP/1.1" 200 -
2021/06/14-18:43:55| [From 10.0.0.24:54162] "GET /doc/success.jpg HTTP/1.1" 200 -
2021/06/14-18:43:56| [Error] File not found on main server '20.188.122.123:8888'
2021/06/14-18:43:56| [From 10.0.0.24:54164] code 404, message 'File not found'
2021/06/14-18:43:56| [From 10.0.0.24:54164] "GET /noneexist HTTP/1.1" 404 -
```



可见cache_hit的话时间显著减少了，因为如果cache_hit的话，和其他两者不同的是，会减少requestMainServer，也就是去访问main server的时间，其余两种情况都会去访问main server，由于距离遥远，时间会显著增大。

4、实验感想

这次实验感觉和之前的一样，但是又不一样。需要自己去阅读大量的API，然后再根据教程来进行写代码。在过程中也遇到了很多麻烦，通过询问老师同学，这些麻烦也一一得到解决，总的来说还是挺有收获的，对HTTP、url、还有httpresponse这些东西也了解了很多。最后在3.4之前虽然通过了全部测试，但是在3.4失败了很多次（这就是16的原因），最后也找到了bug，更改了错误，最后一次实验OVER！希望这次文件夹名字没有加个s。