Lab Report

Lab Name Content Delivery Network

Course	Computer Network
Major	Computer Science and Technology
Id	191220129
Name	Shangyu.Xing
Email	191220129@smail.nju.edu.cn
Date	2021.06

Contents

1 Objective

- Learn CDN and how to implement it;
- Get to know real network by deploying code to physical servers.

2 Requirements

This lab requires to implement a simplified dns server and caching server to achieve CDN.

- DNS server:

  Load DNS records table;

  reply DNS requests from client.

- caching server:

  reply GET requests from client;

  fetch files from remote server;

  maintain cache table.

3 Procedure

I completed all the tasks as required **including the optional task**. In this section, I will explain how I did my work in detail.

3.1 DNS Server

3.1.1 Load DNS Records Table

Just read the file and store the information into a list object for further querying. The method 'str.split' can convert a string to a list.

```
with open(dns_file, mode='r') as file:
    content = file.read()
    if(content[-1] == '\n'):
        content = content[:-1]
    for line in content.split('\n'):
        items = line.split(' ')
        for i in range(len(items)):
            if items[i][-1] == '.':
                items[i] = items[i][:-1]
        self.dns_table.append(items)
```

3.1.2 Match Domain

When the dns server receives a request from client, it should match the requested domain with the records table stored. Note that the table contains entry with * as beginning (which means the it can be matched with anything). However in this case it seems like a bit of overkill to use regular expression, so I adopted a policy to match the two string from end to beginning, and when the pointer moves to *, the match is successful.

```
@staticmethod
def match(dst, src) -> bool:
    i, j = len(dst) - 1, len(src) - 1
    while i >= 0 and j >= 0:
        if dst[i] == '*':
            return True
        if dst[i] != src[j]:
            return False
        i -= 1
        j -= 1
    return i == j
```

3.1.3 Select the Nearest IP

Simply traverse the matched ip list and select the nearest under Euclid distance.

```
def select_ip(self, lst, clientip):
        pc = IP_Utils.getIpLocation(str(clientip))
        if pc is None:
                return random.choice(lst)
        dist = math.inf
        for sip in lst:
                ps = IP_Utils.getIpLocation(sip)
                if ps is None:
                        return random.choice(lst)
                tmp_dist = self.calc_distance(pc[1], pc[0], ps
                    [1], ps[0])
                if tmp_dist < dist:
                        res = sip
                        dist = tmp_dist
        return res
```

3.2 Caching Server

When the cache server receives a get request, it should do the following:

1. check if the requested file is in cache;

2. if not, fetch the file from remote server and store it in cache;

3. return the file to the client.

The first and second steps are implemented in touchItem, and the last step is implemented in do_get.

3.2.1 Do Get

Do Get should also handle file not found error.

```
res = self.server.touchItem(self.path)
if res is None:
    self.send_error(HTTPStatus.NOT_FOUND, "'File not found'")
    return
headers, body = res
self.sendHeaders(headers)
self.sendBody(body)
```

3.2.2 TouchItem

```
if path in self.cacheTable and not self.cacheTable.expired(path)
    :
        return self.cacheTable.getHeaders(path), self.cacheTable
            .getBody(path)
response = self.requestMainServer(path)
if response is None:
        return None
headers, body = response.getheaders(), response.read()
self.cacheTable.setHeaders(path, headers)
self.cacheTable.appendBody(path, body)
return headers, body
```

3.2.3 Stream Forwarding

The workflow is as follows:

1. Upon receiving a request, the server search its cache for the requested file;

2. if found, return the whole body to client;

3. if not, fetch and send headers from remote server and repeat these steps until the whole body is received and sent:

   fetch 64KB data of body from remote server;

   send it to client.

To complete stream forwarding, I create a bytearray buffer as a member variable of CacheServer, along with some variables indicating the current state.

I also created a method get_body, which can be repeated called to get 64KB of body data. Every time it is called, it reads 64KB of response data into the buffer and return it. Note that the response object is stored as a member variable of CacheServer immediately after fetching and sending headers of the requested file.

```
def touchItem(self, path: str):
        assert(self.response is None)
        if path in self.cacheTable and not self.cacheTable.
           expired(path):
                return self.cacheTable.getHeaders(path), self.
                   cacheTable.getBody(path)
        response = self.requestMainServer(path)
        if response is None:
                return None
        headers = response.getheaders()
        self.cacheTable.setHeaders(path, headers)
        self.response = response
        self.path = path
        return headers, None # to be continue in get_body

def get_body(self):
        if self.response is None:
                raise StopIteration()
        length = self.response.readinto(self.buffer)
        if length < BUFFER_SIZE:
                self.response = None
        self.cacheTable.appendBody(self.path, self.buffer[:
           length])
        return self.buffer[:length]
```

The method do_GET should also be modified:

```
res = self.server.touchItem(self.path)
if res is None:
        self.send_error(HTTPStatus.NOT_FOUND, "'File not found'"
           )
        return
headers, body = res
self.sendHeaders(headers)
if body is None:
        try:
                while True:
                        self.sendBody(self.server.get_body())
        except StopIteration:
                pass
```

```
13 else:
14         self.sendBody(body)
```

But do_HEAD remains the same since the headers are not streamed.

4 Test & Result

4.1 Testcases

Firstly I tested my code using the given testcases.

```
~/Workspace/assignments/network/lab-7-xingshangyu(master*) » python3 test_entry.py dns
2021/06/04-11:11:33| [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 0.021s

OK
2021/06/04-11:11:34| [INFO] DNS server terminated
```

Figure 1: test dns

```
~/Workspace/assignments/network/lab-7-xingshangyu(master*) » python3 test_entry.py cache
2021/06/04-11:11:41| [INFO] Main server started
2021/06/04-11:11:41| [INFO] RPC server started
2021/06/04-11:11:41| [INFO] Caching server started
test_01_cache_missed_1 (testcases.test_cache.TestCache) ...
[Request time] 14.77 ms
ok
test_02_cache_hit_1 (testcases.test_cache.TestCache) ...
[Request time] 7.68 ms
ok
test_03_cache_missed_2 (testcases.test_cache.TestCache) ...
[Request time] 10.59 ms
ok
test_04_cache_hit_2 (testcases.test_cache.TestCache) ...
[Request time] 6.50 ms
ok
test_05_HEAD (testcases.test_cache.TestCache) ...
[Request time] 7.55 ms
ok
test_06_not_found (testcases.test_cache.TestCache) ...
[Request time] 8.85 ms
ok

----------------------------------------------------------------------
Ran 6 tests in 3.884s

OK
2021/06/04-11:11:45| [INFO] Caching server terminated
2021/06/04-11:11:45| [INFO] PRC server terminated
2021/06/04-11:11:45| [INFO] Main server terminated
```

Figure 2: test cache

```
~/Workspace/assignments/network/lab-7-xingshangyu(master*) » python3 test_entry.py all
2021/06/04-11:11:49| [INFO] DNS server started
2021/06/04-11:11:49| [INFO] Main server started
2021/06/04-11:11:49| [INFO] RPC server started
2021/06/04-11:11:49| [INFO] Caching server started
test_01_cache_missed_1 (testcases.test_all.TestAll) ...
[Request time] 18.90 ms
ok
test_02_cache_hit_1 (testcases.test_all.TestAll) ...
[Request time] 7.11 ms
ok
test_03_not_found (testcases.test_all.TestAll) ...
[Request time] 7.19 ms
ok

----------------------------------------------------------------------
Ran 3 tests in 1.761s

OK
2021/06/04-11:11:52| [INFO] DNS server terminated
2021/06/04-11:11:52| [INFO] Caching server terminated
2021/06/04-11:11:52| [INFO] PRC server terminated
2021/06/04-11:11:52| [INFO] Main server terminated
```

Figure 3: test all

4.2 Deployment

The test log is here:

```
test_01_cache_missed_1 (testcases.test_all.TestAll) ... ok
test_02_cache_hit_1 (testcases.test_all.TestAll) ... ok
test_03_not_found (testcases.test_all.TestAll) ... ok

----------------------------------------------------------------------
Ran 3 tests in 3.139s

OK

[Request time] 708.49 ms

[Request time] 2.86 ms

[Request time] 739.01 ms
```

Figure 4: client log

```
1  2021/06/04-03:17:07| [INFO] Caching server started
2  Caching server serving on http://0.0.0.0:8153
3  2021/06/04-03:17:10| [Info] Fetched '/doc/success.jpg' from main server '20.188.
   122.123:8888'
4  2021/06/04-03:17:10| [From 10.0.0.24:45164] "GET /doc/success.jpg HTTP/1.1" 200 -
5  2021/06/04-03:17:11| [From 10.0.0.24:45166] "GET /doc/success.jpg HTTP/1.1" 200 -
6  2021/06/04-03:17:12| [Error] File not found on main server '20.188.122.123:8888'
7  2021/06/04-03:17:12| [From 10.0.0.24:45168] code 404, message 'File not found'
8  2021/06/04-03:17:12| [From 10.0.0.24:45168] "GET /noneexist HTTP/1.1" 404 -
```

Figure 5: cache server log

```
2021/06/04-03:17:07| [INFO] DNS server started
DNS server serving on 0.0.0.0:8153
2021/06/04-03:17:09| [Info] Receving DNS request from '10.0.0.24' asking for 'stfw.localhost.computer.'
2021/06/04-03:17:11| [Info] Receving DNS request from '10.0.0.24' asking for 'stfw.localhost.computer.'
2021/06/04-03:17:11| [Info] Receving DNS request from '10.0.0.24' asking for 'stfw.localhost.computer.'
```

Figure 6: dns log

Explanation:

- In the first test the file was not cached, so caching server fetched the file from remote server, which consumed over 700ms.
- In the second test, the file had already been cached, so the server returned the file from its cache immediately, which consumed less than 3ms.
- In the first test the file was not cached, so caching server asked for the file from remote server and got a file not found error, which also consumed over 700ms.

We can learn that CDN greatly shorten the response time.

5 Summary

- Knowing how to effectively use debugging tools such as pdb will greatly enhance working efficiency;
- English reading and writing skills are important.