Computer Networks: lab07

Content Delivery Network

任课教师：田臣　　助教：方毓楚， 郑浩等

学院	计算机科学与技术系	专业	计算机科学与技术
学号	201220154	姓名	王紫萁
Email	201220154@smail.nju.edu.cn	开始/完成时间	2022. 5.20/2022.5.23

1. 实验目的

实现两个在CDN网络中的重要设备：DNS服务器和Caching服务器。

2. 实验原理

2.1. DNS服务器

DNS服务器负责CDN网络中的域名解析。服务器会根据记录的Caching Server的开销（综合考虑距离，跳数，链路状态等信息），选择开销最小的Cacing Server的IP地址作为用户实际访问的IP地址将用户定向到与该服务器的通讯。其次，域名解析的重要步骤就是匹配当前的域名和服务器中记录的域名。而记录域名的表项中又有两种类型的重定位地址：`CNAME`类型表示收到的域名需要重定向到另一个域名去，用户需要用这个新域名再次询问真实的IP地址；`A`类型表示收到的域名服务器可以选择记录的最近的Caching Server的IP地址返回给用户，用户和服务器的数据传输就此开始。

2.2. 缓存服务器

Caching服务器负责CDN网络中的内容分发。由于中心服务器可能距离服务地很远，因此缓存服务器将中心服务器的内容做一级缓存，使得除第一次请求之外的HTTP请求能够快速获得请求。而第一次请求在使用边请求边传输的技术后（本实验中的选作实现），也能让请求速度接近Caching服务器和中心服务器之间通讯的速度。

3. 实验步骤

3.1. DNS服务器

作为提供CDN网络基础功能——域名解析的重要设备，DNS服务器在解析域名时经历了以下几个步骤

- 查找匹配域名表项
- `A` 类型选择最近IP，如果能获得用户I和记录的Caching服务器的地理距离那么返回最近的Caching服务器，否则返回一个任意服务器的IP地址。 `CNAME` 类型直接返回记录的二次域名

另外，在初始化DNS域名表项时，还会判断当前域名是专属域名还是所有以该域名结尾的任意域名都可以重定向到目标IP或者二次域名中去。

DNS Table初始化

函数会通过正则表达式检查表项是否以*.开头，如果是就将 `mode` 设置为 `ANY` ，否则设置为 `UNI` ，再依次将每一个字段按类型填入到 `DNSEntry` 中。 `match` 方法会根据 `mode` 和 `type` 类型检查待查询的域名和当前表项的域名是否匹配，关于解析dns_table文件在此不做赘述，正则表达式的模式串为 `r'^\*\.[a-zA-Z\d.]+'`

```
class DNSEntry(object):
    def __init__(self, mode: str, nameType: str, domain_name: str,
transList: list):
        # hit when name is completely matched or is the postfix of the key
        # 'UNI'/'ANY'
        self.mode = mode
        # 'CNAME'/'A'
        self.nameType = nameType
        # entry's domain name
        self.domain_name = domain_name
        # entry's transmitting IP/name list
        self.transList = transList

    def match(self, targetDomain: str):
        bare_target = targetDomain.strip('.')
        bare_domain = self.domain_name.strip('.')
        # completely matched
        if bare_target == bare_domain:
            return True
        # matched as postfix
        else:
            if bare_target.endswith(bare_domain):
                if self.mode == 'ANY':
                    return True
        return False
```

获取最佳IP

获取最佳IP的过程分为以下几步：

- 检查用户IP距离是否可知，如不可知直接随机返回IP列表中的某一项
- 遍历IP列表，调用 `cal_distance` 找到与用户距离最短的IP并返回。
- 返回最短距离IP，如果所有IP距离都不可知就返回一个随机IP

经纬距离的计算用$Haversine$公式解决

生成回复信号

生成回复信号的步骤如下：

- 遍历TABLE找到匹配的表项
- 根据nameType判断直接返回二次域名还是返回最佳IP
- 如果找不到返回None，由框架代码发送错误回应

3.2. 缓存服务器

基础版本的缓存服务器处理小文件时效果不错，但无法处理较大文件。我们先介绍基础版本，采用 `buffer` 处理大文件的版本只需要在基础版本上稍加改动

缓存查询touchItem

作为加速访问的最基础的步骤，最简单的情况就是查询到了缓存的表项，只需要直接返回 `header` 和 `body` 即可。而查询不到表项，会用当前路径向 `mainServer` 发送请求， `mainServer` 若也无法找到会返回到 `HTTPHandler` 进行发送404的错误码。查询到就将 `header` 和 `body` 加入到 `table` 中即可。

如果要处理大文件，在接收到 `mainServer` 的回复并且扩展 `body` 时会稍有不同，由于每一次将 `BUFFER_SIZE` 个字节通过 `readinto` 读取到 `buffer` 中所以可能无法一次读完，故先将这部分 `buffer` 的内容通过 `appendBody` 添加到数据库中，然后用 `yield` 返回当前接收到的 `header` 和 `body` 并由 `handler` 立即发送这一部分 `buffer` 从而实现边接收边发送。

改进版本实现的好处还有防止 `body` 太大一次性读入内存造成内存错误，而一个小内存的 `buffrer` 正好可以解决这样的问题

sendHeaders

来到 `senHeaders` 说明之前的检查都通过，可以发送header了，在发送header之前先发送一个OK的response表示可以正确传输。接着遍历传进来的 `headers` 列表一一加入到 `internal queue` 中加入完毕后执行 `end_headers` 将队列清空

doGET & doHEAD

基础版本的实现很简单：

- 调用 `touchItem()` 得到一个response
- 如果response为空返回 `NOT FOUND` 错误信息，并调用 `end_headers` 清空 `header` 的 `queue` 。
- 不为空先 `send_headers` ，再 `send_body`

使用 `buffer` 的版本在上述步骤之后又多了一个步骤， `touchItem` 本质上是一个迭代器，因此 `next` 到下一次 `yield` 返回 `body` ，我们将新的 `body` 发送即可。

```
    def do_GET(self):
        touch = self.server.touchItem(self.path)
        headers, body = next(touch)
        if headers is None:
            self.send_error(HTTPStatus.NOT_FOUND, "''Oops:-), File Not
Found!'")
```

```
            self.end_headers()  # clear headers' queue
        else:
            self.sendHeaders(headers)
            self.sendBody(body)

        while True:
            try:
                _, body = next(touch)
                self.sendBody(body)
            except StopIteration:
                break
```

而 `doHEAD` 就不用执行后面的 `while` 并且只执行 `sendHeaders` 即可

4. 实验结果和分析

manual test

```
(venv) njucs@njucs-VirtualBox:~/netlab/lab-07-wayne-ziqi$ 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 - - [23/May/2022 17:05:48] "GET /doc/success.jpg HTTP/1.1" 200 -
127.0.0.1 - - [23/May/2022 17:06:02] "GET /doc/index.html HTTP/1.1" 200 -
127.0.0.1 - - [23/May/2022 17:06:11] code 404, message File not found
127.0.0.1 - - [23/May/2022 17:06:11] "GET /nonexist HTTP/1.1" 404 -
```

```
(venv) njucs@njucs-VirtualBox:~/netlab/lab-07-wayne-ziqi$ python3 runCachingServer.py localhost:8000
Caching server serving on http://0.0.0.0:1222
2022/05/23-17:05:48| [Info] Fetched '/doc/success.jpg' from main server 'localhost:8000'
2022/05/23-17:05:48| [From 127.0.0.1:39024] "GET /doc/success.jpg HTTP/1.1" 200 -
2022/05/23-17:06:02| [Info] Fetched '/doc/index.html' from main server 'localhost:8000'
2022/05/23-17:06:02| [From 127.0.0.1:39028] "GET /doc/index.html HTTP/1.1" 200 -
2022/05/23-17:06:11| [Error] File not found on main server 'localhost:8000'
2022/05/23-17:06:11| [From 127.0.0.1:39032] code 404, message ''Oops:-), File Not Found!'
2022/05/23-17:06:11| [From 127.0.0.1:39032] "GET /nonexist HTTP/1.1" 404 -
```

```
(venv) njucs@njucs-VirtualBox:~/netlab/lab-07-wayne-ziqi$ 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  100  2125    0     0   691k      0 --:--:-- --:--:-- --:--:--  691k
(venv) njucs@njucs-VirtualBox:~/netlab/lab-07-wayne-ziqi$ curl -O http://localhost:1222/doc/index.html
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   352  100   352    0     0  70400      0 --:--:-- --:--:-- --:--:-- 70400
(venv) njucs@njucs-VirtualBox:~/netlab/lab-07-wayne-ziqi$ curl http://localhost:1222/nonexist
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
        <title>Error response</title>
    </head>
    <body>
        <h1>Error response</h1>
        <p>Error code: 404</p>
        <p>Message: ''Oops:-), File Not Found!'.</p>
        <p>Error code explanation: HTTPStatus.NOT_FOUND - Nothing matches the given URI.</p>
    </body>
</html>
(venv) njucs@njucs-VirtualBox:~/netlab/lab-07-wayne-ziqi$
```

脚本测试

```
2022/05/23-17:08:37| [INFO] DNS server started
2022/05/23-17:08:37| [INFO] Main server started
2022/05/23-17:08:37| [INFO] RPC server started
2022/05/23-17:08:37| [INFO] Caching server started
test_01_cache_missed_1 (testcases.test_all.TestAll) ...
[Request time] 4.65 ms

[Request time] 2.17 ms
ok
test_02_cache_hit_1 (testcases.test_all.TestAll) ... ok

[Request time] 2.45 ms
test_03_not_found (testcases.test_all.TestAll) ... ok

----------------------------------------------------------------------
Ran 3 tests in 1.593s

OK
2022/05/23-17:08:40| [INFO] DNS server terminated
2022/05/23-17:08:40| [INFO] Caching server terminated
2022/05/23-17:08:40| [INFO] PRC server terminated
2022/05/23-17:08:40| [INFO] Main server terminated

Process finished with exit code 0
```

OpenNetLab 测试

```
# cache
2022/05/23-08:38:09| [INFO] Caching server started
Caching server serving on http://0.0.0.0:8231
2022/05/23-08:38:11| [Info] Fetched '/doc/success.jpg' from main server '20.188.122.123:8888'
2022/05/23-08:38:11| [From 10.0.0.24:49544] "GET /doc/success.jpg HTTP/1.1" 200 -
2022/05/23-08:38:12| [From 10.0.0.24:49546] "GET /doc/success.jpg HTTP/1.1" 200 -
2022/05/23-08:38:14| [Error] File not found on main server '20.188.122.123:8888'
2022/05/23-08:38:14| [From 10.0.0.24:49548] code 404, message ''Oops:-), File Not Found!'
2022/05/23-08:38:14| [From 10.0.0.24:49548] "GET /noneexist HTTP/1.1" 404 -
```

```
# dns
2022/05/23-08:38:09| [INFO] DNS server started
DNS server serving on 0.0.0.0:8231
2022/05/23-08:38:11| [Info] Receving DNS request from '10.0.0.24' asking for 'stfw.localhost.computer.'
2022/05/23-08:38:12| [Info] Receving DNS request from '10.0.0.24' asking for 'stfw.localhost.computer.'
2022/05/23-08:38:13| [Info] Receving DNS request from '10.0.0.24' asking for 'stfw.localhost.computer.'
```

```
# client
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 2.892s

OK

[Request time] 462.76 ms

[Request time] 2.35 ms

[Request time] 737.48 ms
```

我们发现 `client` 在 `cache` 命中和不命中的时间相差460ms，说明 `cache` 加快用户请求的效果是显著的

5. 实验总结

了解了CDN的工作原理，掌握了DNS Server域名解析，重定向到Cache server，以及最终数据传输的整个流程。并熟悉了 `python` 的 `generator` 在处理大文件上的优势以及写法。体会了cache命中和cache缺失在时间上的巨大差异