CS655 GENI Mini Project Final Report
Project Name: Web Caching

Juntao Wang U18801732, Yifei Fang U78572119
Tian Ding U90706530, Yanshan Song U93089867

Github Link: https://github.com/DaoDaoNoCode/cs655-mini-project
Project on GENI: Mini-Project

1. Problem Statement

1.1 Definition

Design an experiment on GENI nodes where one can show the benefit of having a web cache and study different aspects of web caching. Specifically, we measure latency with a cache and without a cache, study the function of max-age & expires, and gauge the performance improvement of retrieving objects through a web cache under the LRU eviction policy.

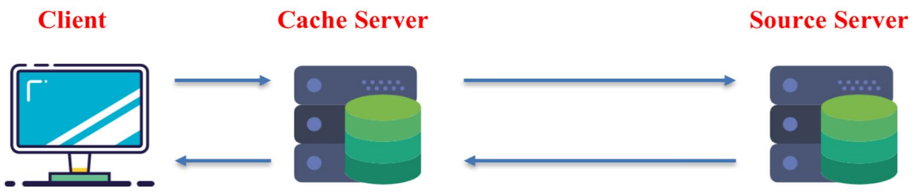
1.2 Motivation

In large network projects, CDN technology is frequently used to improve the objects retrieving performance. The core of CDN technology is setting up web caches in proper locations. So in the project, we will study the benefit of having a web cache.

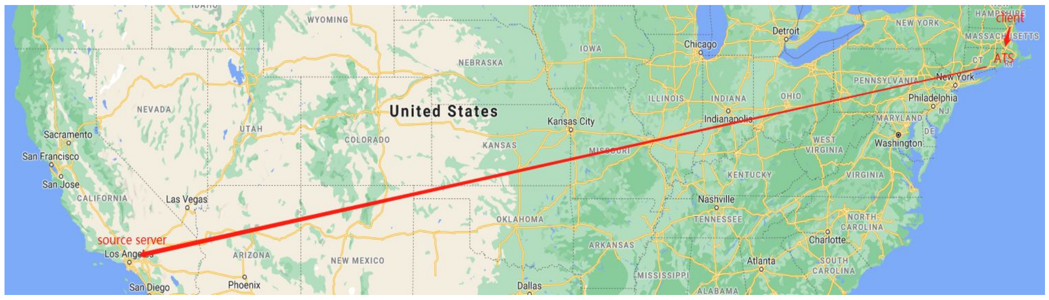
2. Design/setup

2.1 Setup diagram

In general, we use a 'Client-Cache Server-Source
Server' architecture. In this architecture, the client sends http requests to the cache server to retrieve objects. The cache server searches its cache for target objects firstly, if it can find, it will directly return target objects to the client. If it can't find or target objects are stale, it will send http requests to the source server to retrieve target objects, push them to its cache and return them to the client.

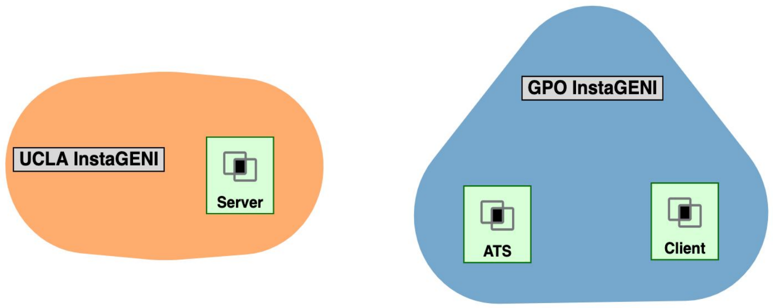


In the project, to simulate the real world scenario, we set up the client and the cache server in Boston, and set up the source server in Los Angeles, like the following image shows.



2.2 Environment/resources

We create three nodes in the GENI slice. Then we install Apache2 in the source server and Apache Traffic Server(ATS) playing the role of reverse proxy and caching in the cache server. In the client, we use the 'curl' command to send http requests and analyze data.



3. Execution/results

In the project, source server IP is 164.67.126.36, cache server IP is 192.1.242.153, client IP is 192.1.242.154.

3.1 Latency with a cache and without a cache

1) Add an 87MB video to the source server.

2) Use the command ***curl -o test.flv -w 'Total: %{time_total}s\n' 164.67.126.36/test.flv*** to measure the total time of retrieving an object from the source server.

3) Use the command ***curl 192.1.242.153/test.flv*** to make sure this video is stored in the web cache.

4) Use the command ***curl -o test.flv -w 'Total: %{time_total}s\n' 192.1.242.153/test.flv*** to measure the total time of retrieving an cached object.

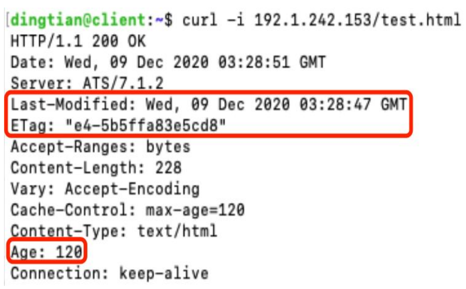
	1	2	3	average
without cache(s)	11.273102	8.905976	10.915969	10.365
with cache(s)	7.459160	7.443174	7.554811	7.4857

As the table shows, the average performance improvement is 27.95%

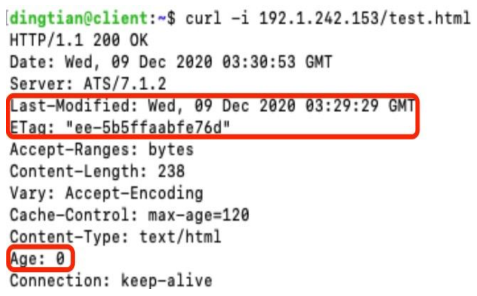
3.2 max-age & expires

(a) A cached object only has a max-age property

Modify the source server configuration to make sure the object returned by the source server only has the max-age property. Then use the *curl* command to test the update policy.



```
dingtian@client:~$ curl -i 192.1.242.153/test.html
HTTP/1.1 200 OK
Date: Wed, 09 Dec 2020 03:28:51 GMT
Server: ATS/7.1.2
Last-Modified: Wed, 09 Dec 2020 03:28:47 GMT
ETag: "e4-5b5ffa83e5cd8"
Accept-Ranges: bytes
Content-Length: 228
Vary: Accept-Encoding
Cache-Control: max-age=120
Content-Type: text/html
Age: 120
Connection: keep-alive
```

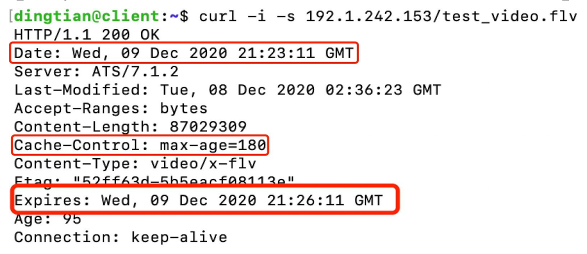


```
dingtian@client:~$ curl -i 192.1.242.153/test.html
HTTP/1.1 200 OK
Date: Wed, 09 Dec 2020 03:30:53 GMT
Server: ATS/7.1.2
Last-Modified: Wed, 09 Dec 2020 03:29:29 GMT
ETag: "ee-5b5ffaabfe76d"
Accept-Ranges: bytes
Content-Length: 238
Vary: Accept-Encoding
Cache-Control: max-age=120
Content-Type: text/html
Age: 0
Connection: keep-alive
```

The image shows ATS will judge whether the object is fresh by comparing the Age and max-age properties of the cached object. The Age property records the time from the last update of the object. If Age ≤ max-age, ATS will consider the cached object fresh otherwise it will update the object from the source server.

(b) A cached object only has an expires property

Modify the source server configuration to make sure the object returned by the source server only has an expires property. Then use the *curl* command to test the update policy.

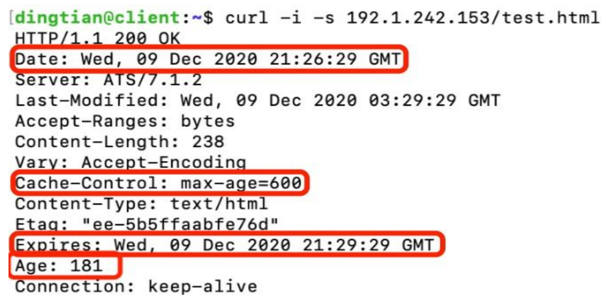


```
[dingtian@client:~$ curl -i -s 192.1.242.153/test_video.flv
HTTP/1.1 200 OK
Date: Wed, 09 Dec 2020 21:23:11 GMT
Server: ATS/7.1.2
Last-Modified: Tue, 08 Dec 2020 02:36:23 GMT
Accept-Ranges: bytes
Content-Length: 87029309
Cache-Control: max-age=180
Content-Type: video/x-flv
Etag: "52ff63d-5b5eacf08113e"
Expires: Wed, 09 Dec 2020 21:26:11 GMT
Age: 95
Connection: keep-alive
```

The image shows ATS will automatically calculate the max-age property according to the expires property.

(c) A cached object both have max-age & expires properties

Modify the source server configuration to make sure the object returned by the source server both have max-age & expires properties. Then use the *curl* command to test the update policy.



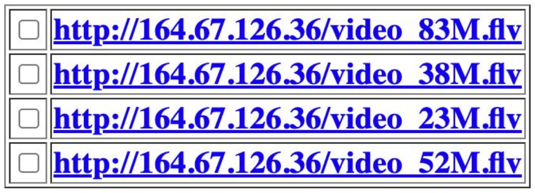
```
[dingtian@client:~$ curl -i -s 192.1.242.153/test.html
HTTP/1.1 200 OK
Date: Wed, 09 Dec 2020 21:26:29 GMT
Server: ATS/7.1.2
Last-Modified: Wed, 09 Dec 2020 03:29:29 GMT
Accept-Ranges: bytes
Content-Length: 238
Vary: Accept-Encoding
Cache-Control: max-age=600
Content-Type: text/html
Etag: "ee-5b5ffaabfe76d"
Expires: Wed, 09 Dec 2020 21:29:29 GMT
Age: 181
Connection: keep-alive
```

According to the expires property, the object will keep fresh in 3 minutes, but according to the max-age property, the object will keep fresh in 10 minutes. Now, ATS doesn't update the object from the source server when Age is 181, which indicates that when a max-age property conflicts with an expires property, the max-age property has the priority.

3.3 Eviction Policy

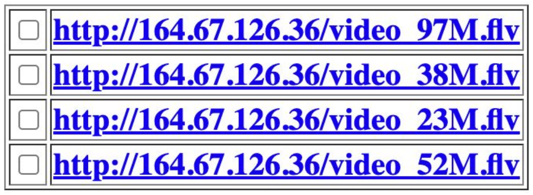
According to the ATS document, the traffic server cache only provides one eviction policy which is overwriting the least recently cached objects when the space of cache is already full. Following is the test we used to prove our traffic server does use the same policy:

1) We used 4 curl commands to add 4 objects into our cache with size 83M, 52M, 38M, 23M (The order of size is also the order we send the request). Here is the inspector of the cache provided by ATS:



☐	http://164.67.126.36/video_83M.flv
☐	http://164.67.126.36/video_38M.flv
☐	http://164.67.126.36/video_23M.flv
☐	http://164.67.126.36/video_52M.flv

2) We tried to add another object into our cache with size 97M. Since the storage space of our cache server is 256 MB, the cache server needs to remove the least recently cached objects. In the following picture, we can see that the cache object with size 83M has been removed by the cache server and a new object with size 97M has been added into our cache.



☐	http://164.67.126.36/video_97M.flv
☐	http://164.67.126.36/video_38M.flv
☐	http://164.67.126.36/video_23M.flv
☐	http://164.67.126.36/video_52M.flv

3) According to our experiment, we can prove that the traffic cache server does use the LRC(Least Recently Cached) policy. Then we started to test the performance of our cache server under different cases.

Since we know the eviction policy, we can now try to figure out what are the best application scenarios for the web cache. So we design the following experiment.

(a) We add objects with the following order: 83M->52M->38M->23M->97M and loop this action for 3 times, the download speed is listed as follows. (unit: Mbps)

83M	52M	38M	23M	97M	83M	52M	38M	23M	97M	83M	52M	38M	23M	97M	Avg
2.93	3.38	7.64	7.99	3.46	1.84	9.12	8.23	7.15	9.66	9.65	5.17	6.56	7.83	7.08	6.51

(b) Then we clear the cache and add objects as the following order: 83M for 3 times -> 52M for 3 times -> 38M for 3 times -> 23M for 3 times -> 97M for 3 times, get the following table.

83M	83M	83M	52M	52M	52M	38M	38M	38M	23M	23M	23M	97M	97M	97M	Avg
9.66	11.1	11.0	9.42	11.0	8.68	4.74	10.9	9.3	7.55	10.8	10.2	1.2	11.1	11.1	9.18

From the experiment above, we find that this eviction policy is best used when the objects are requested continuously, because we can always get the objects we want from the cache. However, when it comes to requesting different objects in order, it performs not as well as before because we need to always retrieve the objects that were just evicted from the cache.

4. Conclusion & Future work

1) In our project, performance improvement of retrieving objects with a cache is 27.95% .

2) ATS uses Age & max-age properties to judge whether an object in the cache is fresh or not. If an object only has an expires property, ATS will automatically calculate a max-age property according to the expires property. If a max-age property conflicts with an expires property, the max-age property will have the priority.

3) ATS uses LRC (Least-Recently-Cached) as its eviction policy. We cannot choose an eviction policy for the cache so we only validate it. However, we could set different eviction policies for RAM cache so it’s a research field in the future.