In-network Caching Using CDN

Zhicheng Chen, Cheng Han

1 Abstract

In order to better understand how CDN can improve the performance of a website, we first need to build a website for testing. Github Pages provides us the ideal way to get a test website efficiently. We then replace the original website content by our own pages in order to test video, image and some basic website functions.
After building the website with test video and images. We first try to get a faster speed through putting our jQuery through CDN. Jsdelivr is providing a free CDN for open Source and jQuery remains the same by having the same version on website. Therefore, we can replace the local jquery.min.js into the jquery file through CDN and fasten the speed. The detailed information will be provided in the following section.
This is only the first step. After replacing jquery, we can find significant increase in loading speed in some cities we tested such as Beijing and Berlin. But still, there is just a little increase in loading my pages. Still efforts can be done to get a better solution since now we only have the jQuery hosted on CDN. If we can put more content on cdn, the loading speed could be decent.
The next step we did is to put the pictures and videos on CDN. By doing this we expected to have a faster speed when loading the website. We realize it through Amazon Web Services (AWS), which is the world's most comprehensive and broadly adopted cloud platform. We use Simple Storage Service (S3) bucket, which is a public cloud storage resource available in (AWS). We first store the images and videos on S3, and use Amazon CloudFront, a fast CDN service that securely delivers data, videos, applications, and APIs to customers globally with low latency, high transfer speeds, all within a developer-friendly environment, to deploy the images and videos worldwide. By replacing the URLs to our website application, we can significantly lower the delay. The result will be shown in the following part.

2 Introduction to Content Delivery Network (CDN)

A content delivery network (CDN) refers to a geographically distributed group of servers which work together in order to provide fast delivery of Internet content. For improving speed and connectivity, a CDN places servers at the exchange points between different networks. These Internet exchange points (IXPs) are the original places that different Internet providers connect in order to provide each other access to traffic originating on their own different networks. By connecting to such high speed and highly connected locations, CDN is able to reduce costs and transmit times in high speed data delivery.

3 CDN Simulation

3.1 Content delivery tactics research

CDN were born to improve accessibility, while maintaining correctness. As the critical components in CDN, the edge cache servers storage the data cache from the provider server, which shorten the access delay and reduce the network load flow. Also, the edge cache knows whether the data is updated and updates them. One approach to improve the capability and efficiency of CDN is managing the edge cache servers with the content delivery tactics. While managing the edge cache servers, the tactic should be based on the visitor volume (VV) of each object (like page, image, video, etc.). There are two content delivery tactics based on different VVs, which can help improve CDN. Assume that there are N edge cache servers in CDN, and each server contains $N(i)$ different objects, where $i = 1, 2, \ldots\ldots$ and $N(i) \geq 0$.

3.2 The content delivery tactics based on the certain VV

The certain VV of the object means the number of hosts that visit this object. Assume that there are N edge cache servers in CDN, and each server contains N(i) different objects, where $i = 1, 2, \ldots\ldots$. and $N(i) \geq 0$.

During the fixed time T, if VV of the object j in $N(i)$ is written as $V(i,\ j)$, its certain VV is also $V(i,\ j)$. Based on the matrix of V, the minimum cost of distributing the objects can be designed by Dijkstra's algorithm. The chart flow is shown as Figure 2.1.

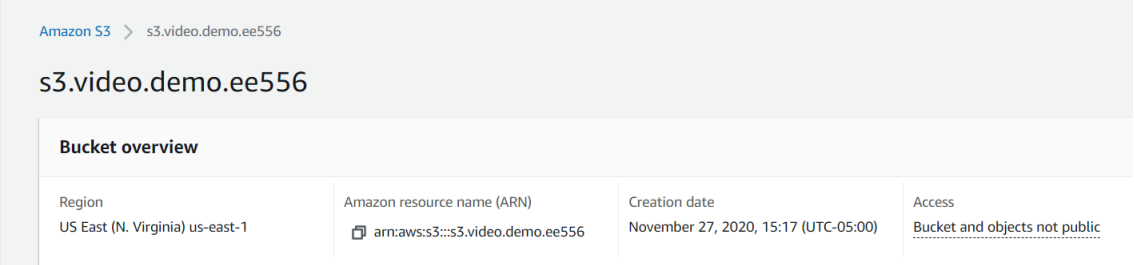

Fig. 1: the content delivery tactics based on the relative VV

Figure 2.1 the content delivery tactics based on the certain VV
2.2 the content delivery tactics based on the relative VV The relative VV of the object is the increment of hosts that visit this object during time T. If VV of the object j in N(i) is written as $V(i,\ j,\ t)$ at time t and $V(i,\ j,\ t-T)$ at time $t-T$, its relative VV is shown as

$$V(i,j) = V(i,j,t) - V(i,j,t-T)$$

3.3 Results on simulation

Through the decades, several network simulators, like CDNsim, ns-2 and OMNeT++, are proposed for developing the network architecture. The ns-2 is a comprehensive network simulator which is good at simulating web cache scenarios. Compared with other simulators, ns-2 is the only one that make the HTTP traffic be virtual, which means that it is possible for us to reproduce the real HTTP data. So, ns-2 is used as the simulator for this project. The experiment is based on the Ubuntu, while the simulator is ns-2. The number of edge server is 5 and each server has 5 objects. Each edge cache server has three hosts. The topology graph is shown in Figure 3.1.

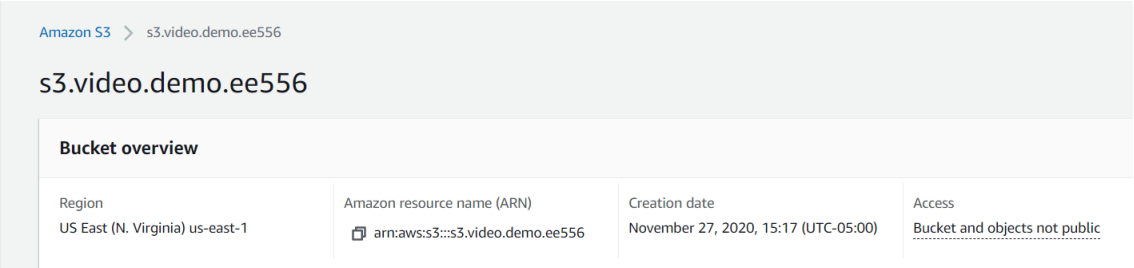

Fig. 2: Topology graph of simulation

We have modeled each server as an $M/M/1$ queue with service rate . The frequency of requests generated by the clients is a Poisson process with rate . The server's parameters are shown in Table 3.1.

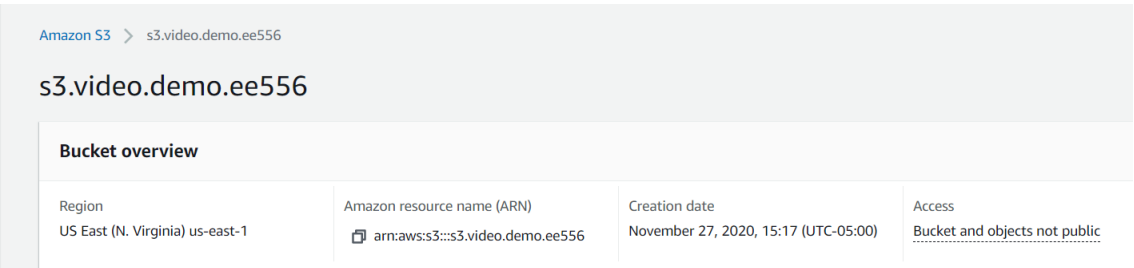

Fig. 3: Topology graph of simulation

Table 3.1
Figure 3.2 to 3.4 show the queue behaviors over time for Rand, LL, and 2RC algorithms, respectively. These graphs match the theoretical expectation for the queue's length, and they represent a quick way to compare

the effectiveness of the balancing mechanisms. As expected, server 8 is unloaded since the incoming request rate is lower than the service rate. For server 7 the queue length has an intermediate behavior since arrival and service rates are comparable. For the LL algorithm 3.3 the queue length follows the attended oscillations due to the time interval required for updating the status of the servers' load. For the 2RC algorithm 3.4 we observe that the queue lengths at servers 2 and 4 have a similar trend, due to the comparable traffic rates; server 7 is subject to a traffic peak, which propagates to server 8, which is close to it; the behavior of such servers is quite similar due to their isolation in a stub of the topology. In this case, the flash crowd effects are quite visible on the involved elements.

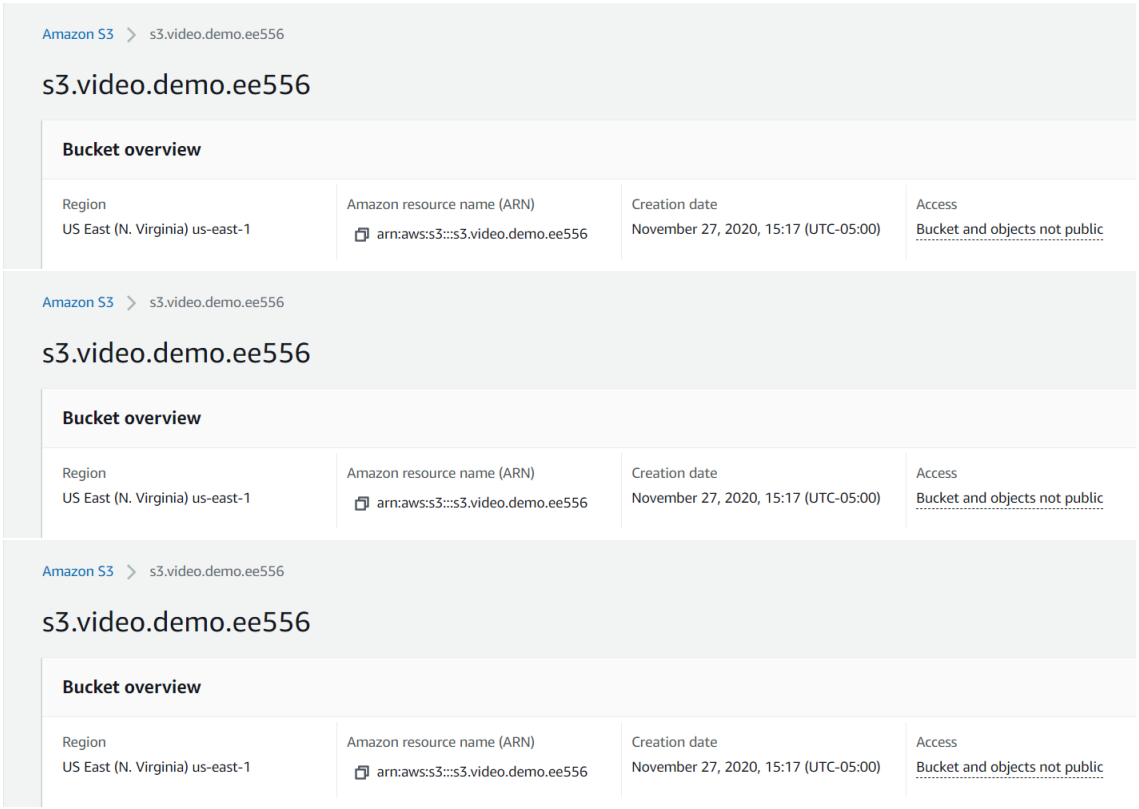

Fig. 4: Topology graph of simulation

4 CDN for website improvement

During this section, we will implement CDN in a test website and find out how exactly CDN could be done to improve the performance of a website.

4.1 Creating a test website

Creating a website can be hard, we might lose ourselves on the web trying to find where to begin. GitHub Pages, however, can easily solve the problem. Github handles both the versioning and deployment. Github pages can be an ideal solution for the test website. By replacing the Github provided homepage into our own website, we can easily set up a website with test content. The website is available in the following link: https://chenghan111.github.io/ChengHanCSE514.github.io/

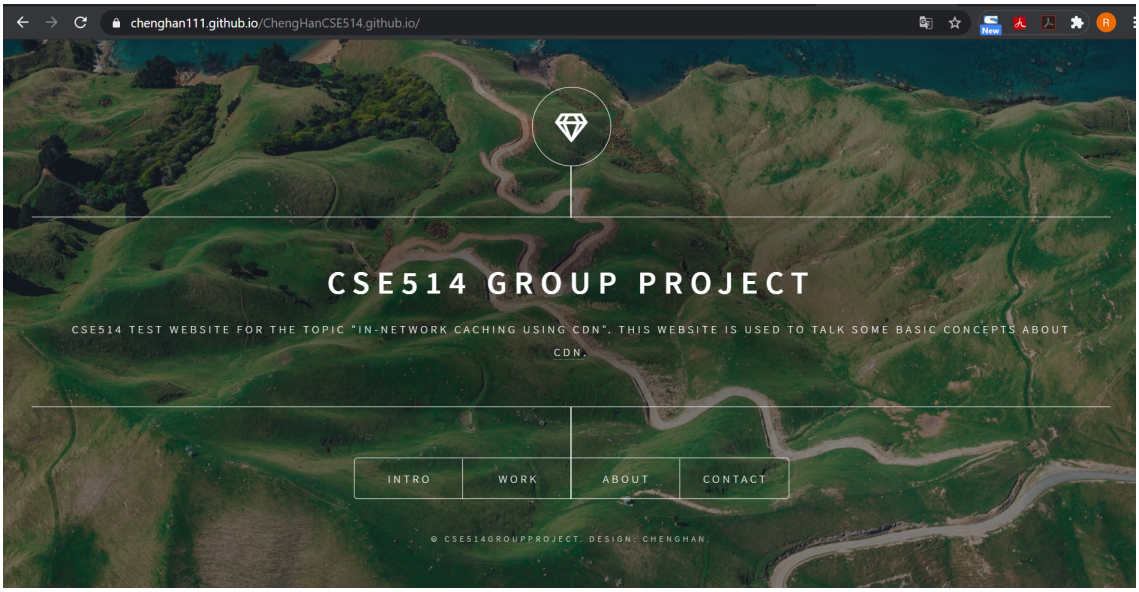

Fig. 5: Homepage of the test website

One thing should be mentioned again is that we replaced the original website provided by Github pages into our own website since we need more test content in the pages and the single Github page cannot satisfy our needs. There are videos and background images in these pages, which aim at testing the loading speed of the website with and without CDN. There are four subpages under our homepage. The video is under 'work' subpage, which is the main test page we need for testing speed. The 'work' subpage has a 1.5MB sample video with resolution 480x270.

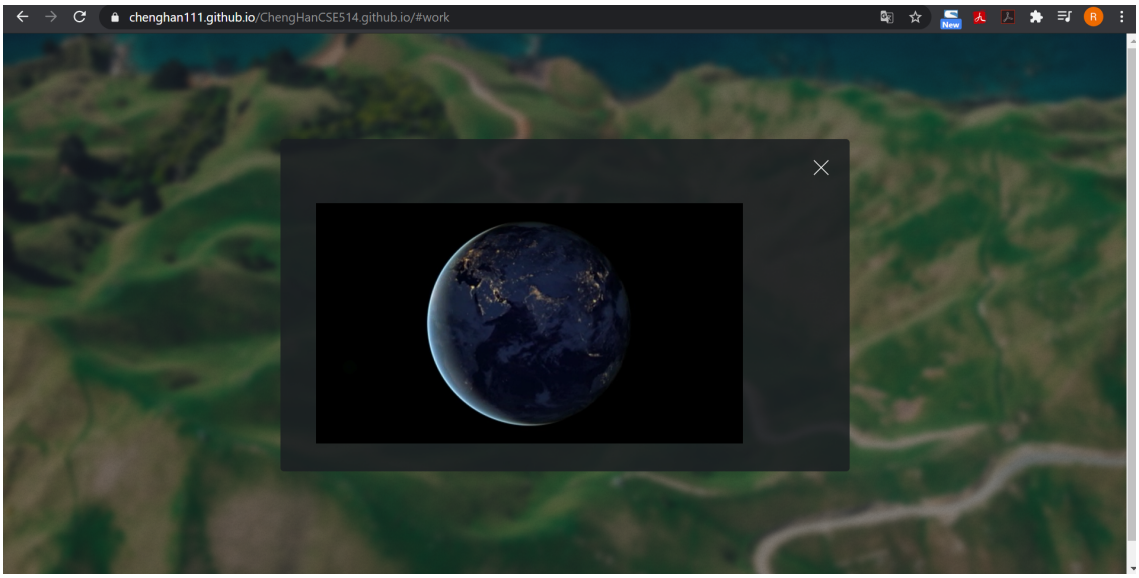

Fig. 6: Subpage 'work'

4.2 Put jQuery on CDN

jQuery is a fast, small, and feature-rich JavaScript library. It can make things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. Since it is a universal tool for browsers, if we can host it through CDN, when a user requests a file from the website, a response will be returned from the server closest to the user, which can increase the loading speed. One thing should be mentioned is that during the experiment, we disable cache since cache can influence the loading speed of website. We need to get rid of this for a fair experiment.
From Fig. 3, the jquery.min.js file is replaced by a jQuery file provided by jsdelivr, which is a free CDN for Open Source. jQuery file is available through this CDN. By replacing jQuery file, a significant increase in performance takes place in some specific testing cities. Before applying jQuery on CDN, Beijing, for example, has a over 1500ms delay. By contrast, the result after applying jQuery file through CDN has a lower than 500ms delay,

which is a huge decrease and can definitely provide a better experience when visiting this page.

Fig. 7: Put jQuery file on CDN

4.3 Put images and videos on CDN

However, replacing jQuery file is not enough since jQuery file is small in size and cannot have a big advance in many tested cities. This is not the full potential energy CDN can provide. CDN can deliver large files faster due to a close geographical distance to users. And the advance of CDN show up when delivering large files such as videos and images.
AWS is used in order to realize the designed solution. First, a Amazon S3 bucket should be build for storage. We build the bucket in US-EAST, the content will them be deployed worldwide through CloudFront. From Fig. 5, we can see that these are the website content, instead of pushing it through Github. We are now storing them through S3 bucket.

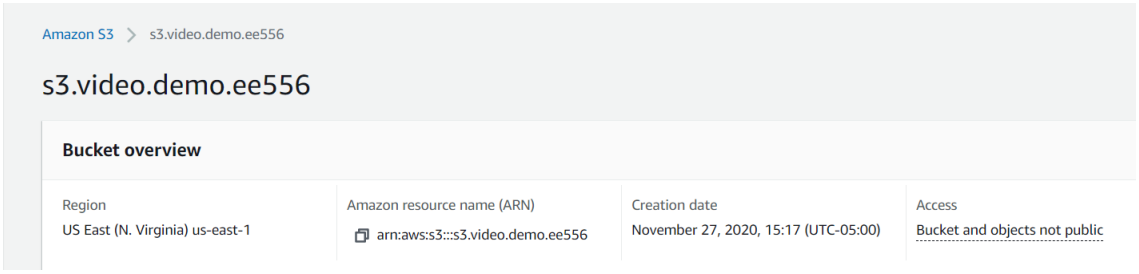

Fig. 8: S3 bucket overview

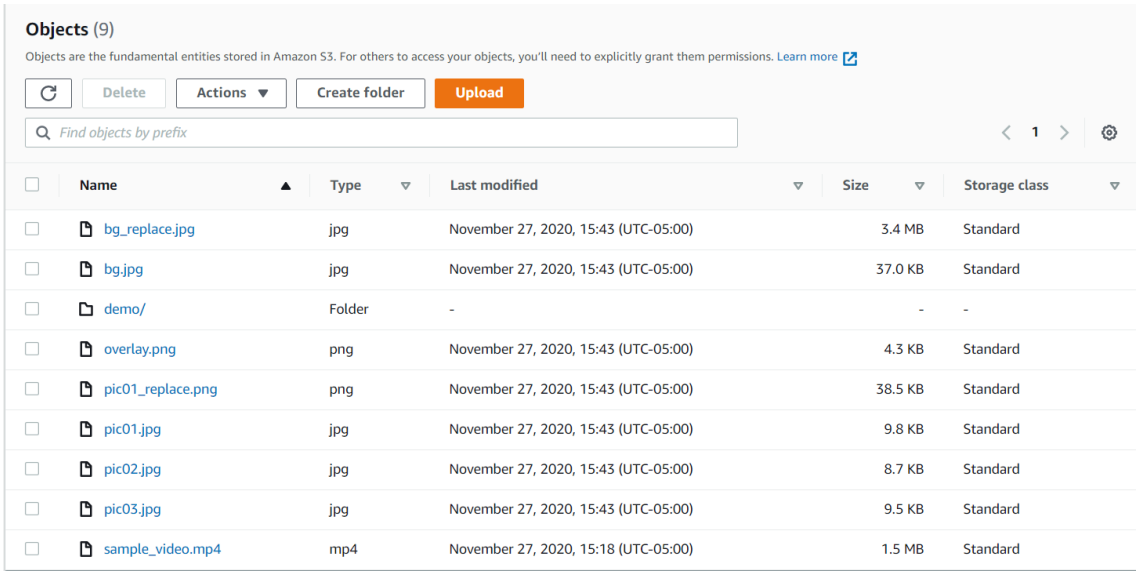

Fig. 9: Objects store in S3 bucket

CloudFront can work together with S3 bucket by choosing the right S3 bucket in CloudFront setting and deployed it to the region we want. During the experiment, we deployed it worldwide, meaning that we can reach an ideal speed in every corner of the world.

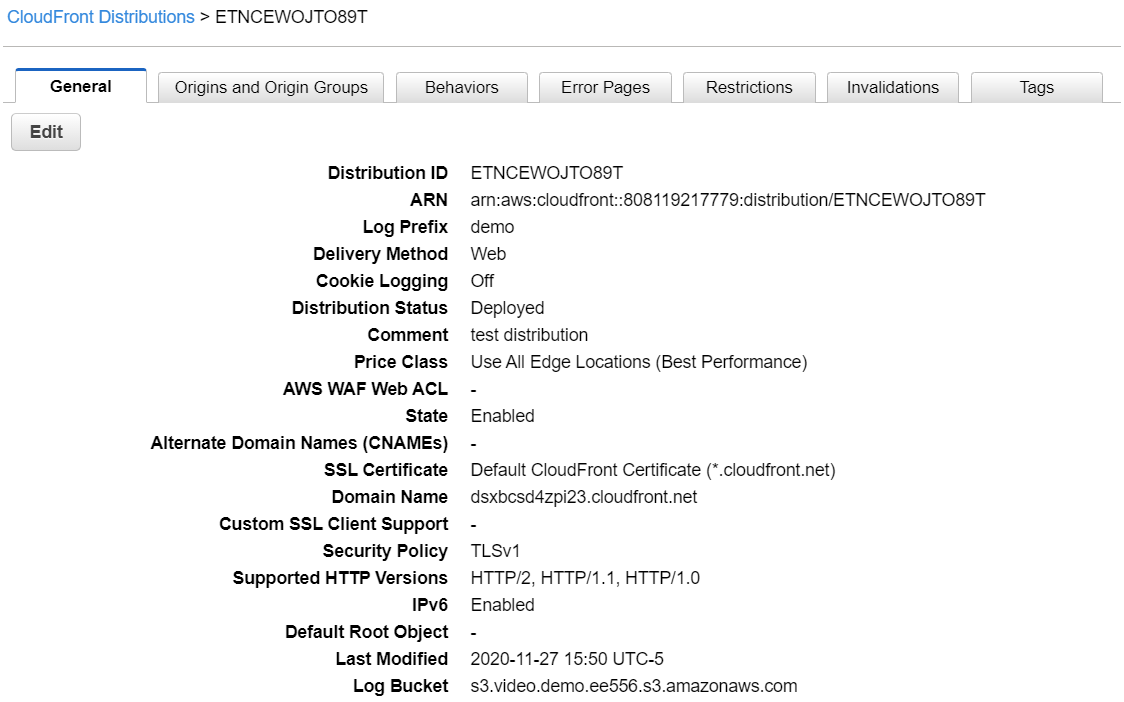

Fig. 10: CloudFront Distributions

Above we have shown some setting on the deployment of CDN, CloudFront now have the access to the content in specific S3 bucket. And CloudFront generate the unique domain name therefore we can access the content we want through the domain name. Take sample_video.mp4 as an example, we can now access it through the following link:
https://dsxbcsd4zpi23.cloudfront.net/sample_video.mp4 Then we need to replace all the source src for background images and the test video source to the domain name plus image or video. Fig. 7 provides a replacement for sample_video.mp4, the address is exactly the link provided above. The other background images are replaced similarly in this way.

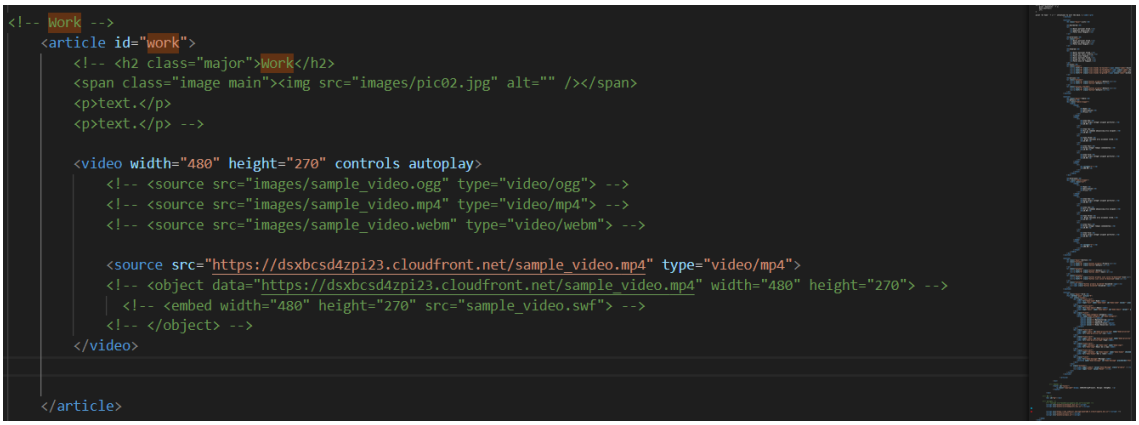

Fig. 11: Source src replacement in index.html

In order to better understand how much speed we can increase during this process. Three results are recorded. The first one is the loading speed of subpage 'work' without any CDN strategy. The second one is the loading speed of the same page with only video replaced. The third one is the result after all the backgrounds images and sample video are provided by CDN. The three results are provided by Fig. 8. Two things should be noticed. The first one is that all the test results are under the situation that the web is disable cache, meaning that every time we refresh the same page, the page need to download the same content. Another thing should be mentioned is that we can compare the 'Finish' every time we refresh the page. This indicates the time browser fully load the page.

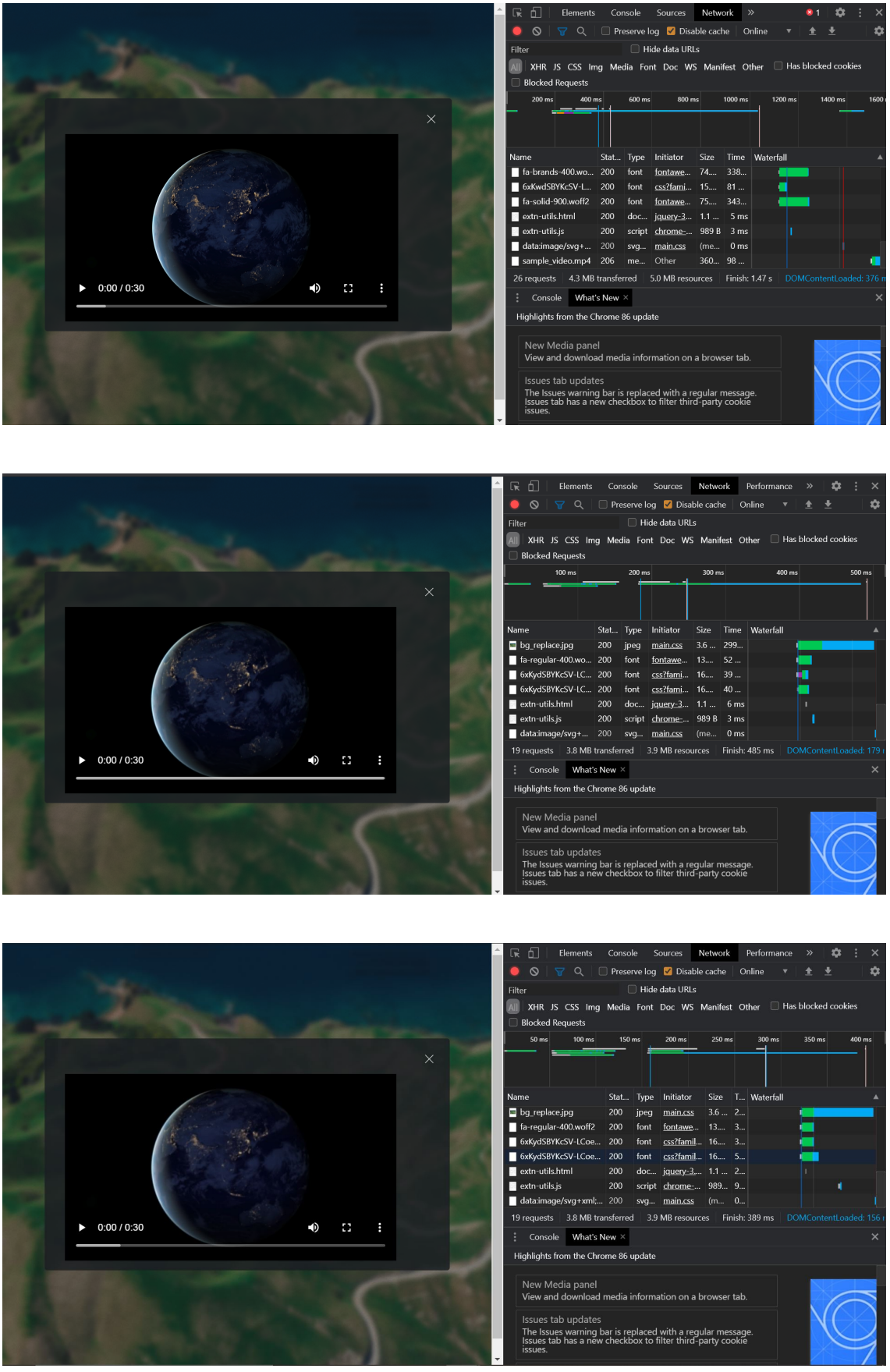

Fig. 12: Three approaches on replacement

From Fig. 8, we can observe that the finish time for without CDN, video-only CDN, and all content CDN are 1.47s, 485ms and 389ms with the same network condition after 10 tests for each of them. We can observe that a huge reduce in loading time takes place when video content is publish through CDN. And additional background images can load faster when they are deployed on CDN as well.

5 References