#STREAMING JUL 19, 2022

How We Updated the Thumbnail Generator

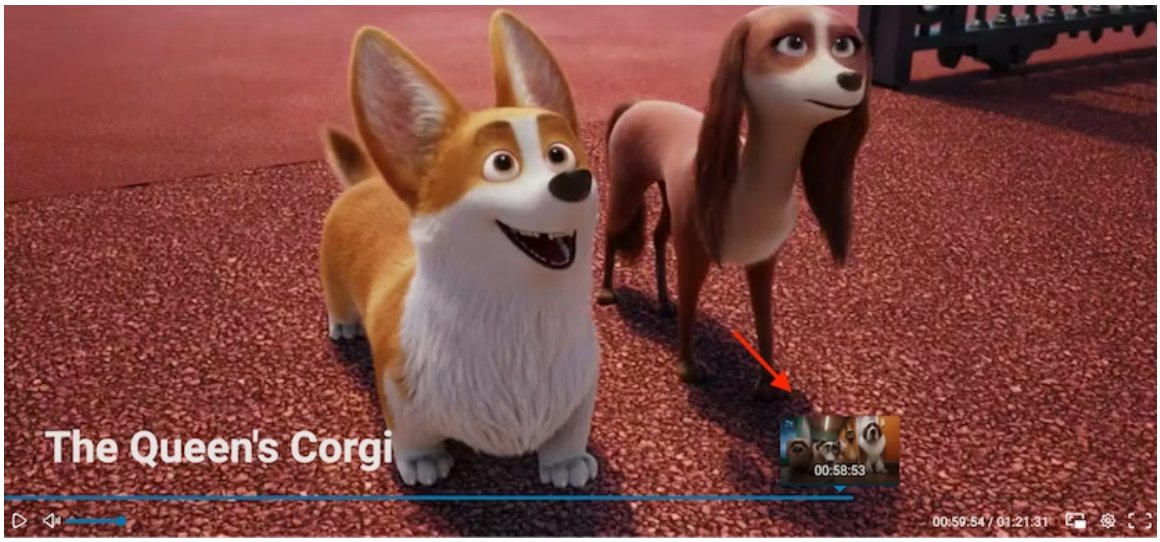
PAVEL ŠVEJDA

THUMBNAIL PREVIEW

Mel Poole

Sometimes you find a perfect film, watch it from start to finish, and the next day you want to show some hilarious moments to your friend. To find the right moments, you usually use the thumbnail preview. We've enabled this feature on the DStv platform and done the same for all the assets on Showmax too *(Showmax and DStv Now are two streaming services within the MultiChoice Connected video division)*.

You can imagine the thumbnail preview as a feature where we show small screenshots of what is going on in the part of the video you are seeking through. This feature is very common for streaming services but the process of achieving it can be different.



Until Now on DStv

We had a solution that was reliable most of the time and was working fine but the requirements for running a service are growing so we had to rewrite it and make it more “observable”.

The old solution required docker to run, which meant we already had one heavyweight dependency to work with. Another problem was the fact that we were running it on Python 2 (which is, since 2020, no longer supported, with no bug or security fixes). This was not a good approach but the component was running fine without any critical issues so we pointed our attention elsewhere.

When it comes to monitoring, we didn’t export any metrics or logs so we were quite limited in terms of debugging and dealing with issues like file rights, disk space, or just some rare random bugs.

Output Files

Our thumbnail generator creates two files from the source video: a sprite file and a VTT file.

A **sprite file** is a large image made from all of the small thumbnails that are generated from a video every N seconds. We put all of those into one file.

image in a given time will always be the same.

What each line means:

> **WEBVTT** is the file format
> **Img 1** is the number of the screenshot being used. They are mostly a row of numbers from 1 - N
> **00:00:00.000 → 00:00:05.000** is the time in which we will display this preview while scrubbing through video
> **100392602_21BJGPT1_800_SUN_sprite.jpg** is the name of the sprite file we are using for previews
> **#xywh=0,0,150,84** is the position of the preview in the sprite file

```
WEBVTT

Img 1
00:00:00.000 -> 00:00:05.000
100392602 21BJGPT1_800 SUN sprite.jpg#xywh=0,0,150,84

Img 2
00:00:05.000 -> 00:00:15.000
100392602_21BJGPT1_800_SUN_sprite.jpg#xywh=150,0,150,84

Img 3
00:00:15.000--> 00:00:25.000
100392602_21BJGPT1_800_SUN_sprite.jpg#xywh=300,0,150,84

Img 4
00:00:25.000 -> 00:00:35.000
100392602_21BJGPT1_800_SUN_sprite.jpg#xywh=450,0,150,84

Img 5
00:00:35.000 -> 00:00:45.000
100392602_21BJGPT1_800_SUN_sprite.jpg#xywh=600,0,150,84

Img 6
00:00:45.000 -> 00:00:55.000
100392602_21BJGPT1_800_SUN_sprite.jpg#xywh=750,0,150,84

Img 7
00:00:55.000 -> 00:01:05.000
100392602_21BJGPT1_800_SUN_sprite.jpg#xywh=900,0,150,84

Img 8
00:01:05.000 -> 00:01:15.000
100392602_21BJGPT1_800_SUN_sprite.jpg#xywh=1050,0,150,84
```

The Generating Process on DSTv

PICK THE HIGHEST BITRATE FROM THE OPTIONS WE HAVE

We are using the highest bitrate to assure the quality of the previews. Even though we are scaling the final image down, it is good not to scale low bitrates. The final image could end up terrible and ruin the user experience.

on our storages – which is requested by studios, owners of the content.

GENERATING SCREENSHOTS

We take our decrypted video, generate “screenshots” every 10 seconds, and save them for later use in the sprite file.

GENERATING THE SPRITE

We have generated a lot of screenshots and now is the time to take them and create a large image. We are putting them into one image to reduce the number of requests and time needed before the user can see the preview while scrolling. This large image is then also reduced in size because the preview video on scrolling is small and we don’t need all of the extra quality.

GENERATING VTT FILE

We now have an image of all the screenshots and we need a way to tell the player where each preview is placed for a given time in the video.

CLEANUP

Now we have done all the work and we only need to remove all the leftovers like decoded video, generated screenshots, and all the folders we have created leaving us with only the image and VTT file.

These few steps assure that the output is usable for our players in DSTv.

What we have done better in the second try

Since the DSTv video gallery is growing larger every day, we needed to optimize the speed of the process of generating thumbnails for videos and make it more transparent for possible debugging and monitoring.

It might look like generating two files is simple but there are a lot of problems that can appear down the road. You could run out of free space on your storage disk, encounter unexpected output for the commands you are running, and other stuff. All of this forces us to have metrics and log messages on all possible points of failure. This was a big motivation to redo the generator.

The first thing we did was we ditched Python and went with **Go** as our language of choice due to the speed, goroutines, and the fact that we already had some components and pieces of code in this language (for example **sockrus** and **fqdn**. We are using goroutines for the workers processing the queue.

The second thing we did was ditch Docker as an environment to run the generator in and instead went with running it directly on the host machine. This added performance and removed the dependency on one big package that needed to be installed.

The third thing was to introduce more logic to the retry mechanism. The old solution was only iterating through available bitrates of source video when some error occurred. We added a limit to the retries to not block the queue for eternity and added a check on duplicate requests for generating. Combined with metrics and better logging we have a retry mechanism that will let us know if there are problems and retry only for a certain number of times.

The last thing we added compared to the old version are metrics and better logging (as the previous section alluded to). Since we are running our thumbnail generator as a service, we are exporting metrics on port and endpoint to be scraped into our prometheus instance. We can then use those metrics for dashboards and alerts. Speaking of dashboards, you definitely need a panel showing how many generating requests you've received and for how long the last generating request ran.

Examples of the metrics we are collecting / exporting include:

- Time it took to generate output for player
- Number of received generate requests
- Number of processed generate requests
- Number of failed generate requests
- Errors due to storage actions
- Errors due to running commands
- And more...

Planned future improvements

We have rewritten our old solution in a new language and are using new technologies but the process remains the same due to the fact that we need the same output as before. In time we plan to reduce the number of commands that need to be run and use only ffmpeg for all the images generated from the video.

Other changes that will definitely be done are new metrics and alerts based on problems that could and will appear.

SHARE ARTICLE VIA:

You might be **interested**