Sprocket: a serverless video processing platform.

Imagine that you're a Marvel Studio producer tasked to edit Captain America scenes in the latest Avengers trailer. To do so, you'd have go to through the drudgery of watching the entire 2 minutes video and manually identify scenes where Captain America appeared—a time-consuming and error-prone process. Imagine if you could ctrl+f Captain America, as you could for a text document, and only scenes where Captain America appeared are displayed—all in less than a fraction of a second. With the advancement of Computer Vision in recent years, this technology is, in theory, possible. Yet, the current video processing frameworks today are extremely slow and expensive. None of them are specialized in processing videos, which is a shame, considering that videos make up 70% of consumer internet traffic [1]. That is where Sprocket comes in. Sprocket is a pipeline, made from a host of cloud technologies, that practitioners can build their own machine learning models on top of. By capitalizing on the cloud's elasticity—being able to horizontally scale on demand—and using innovative techniques like straggler mitigation, Sprocket could process video, a severely under-utilized data form, cheaply and within a fraction of a second.

Sprocket leverages capabilities of the cloud to quickly process videos. Today, the cloud has become an integral part of all non-trivial projects. This is for good reason as the cloud is widely accessible and quite cheap too. Yet, the cloud has only become popular in recent years. New applications of the cloud are constantly being discovered: its potential hasn't been fully tapped into yet. We believe Sprocket has, in fact, tapped into some of this latent potential of the cloud. For one, Sprocket made use of the cloud's serverless compute, like AWS Lambda, and machine learning functionalities. By using AWS Lambda, Sprocket framework doesn't need a clunky node.js server and users could intuitively connect the right endpoints together to use Sprocket. Using the cloud's machine learning, the framework just needs a few lines of code to call the pre-trained models on the cloud. Both of these capabilities make Sprocket a very lightweight framework where users can attach their huge machine learning models onto. Another capability of the cloud used is virtualization. Through virtualization, Sprocket can easily increase more cloud instances based on its load and parallely process workloads across, sometimes, thousands of instances. This made it possible for Sprocket to process videos incredibly fast. Thus, by using the cloud, Sprocket could efficiently and cheaply process videos.

Despite rapid advancement in image recognition software over the past few years, it is puzzling that video processing software is lagging so much behind. This is odd considering that

Figure 1a: Uncompressed

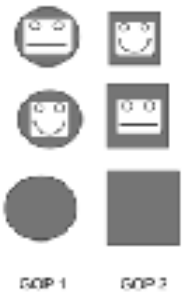

Figure 1b. GOP formed & reference frame chosen

Figure 1c. Other frames stored as differences to reference frame.

Figure 1: The Compression Process

videos are, technically, just a bunch of photos in sequence. Yet, this lag in advancement can be explained by a few reasons. First, videos that are streamed online are compressed. During compression, frames that are similar temporally and spatially are grouped together into independent Group of Photos (GOP). To illustrate this concept, look at Figure 1. The squares and circles represent different frames. In an uncompressed format, all the squares and circles are jumbled up (Figure 1a). When they are compressed, similar shapes are grouped together: circles to GOP1 and squares GOP2 (Figure 1b). In each GOP, a reference frame is chosen, so for GOP 1 it's the blank circle at the bottom and for GOP 2 it's the blank square. Frames other than the reference frame would be stored in the GOP as differences to that reference frames, so the squares and circles are shaded out, leaving behind just the emojis (Figure 1c). Thus, after compression, frames are distorted and are no longer of the same size. They need to be decoded before they can be processed by a machine learning model and for a 2-hour movie, for example, this process could take hours[2, pp.2]. Second, videos streamed online are becoming bigger, especially as 4K and VR videos, for instance, are becoming common. This further compounds the time taken to decode videos. With such a computational bottleneck in place, it's no wonder that video processing technologies aren't as advanced as they should be.

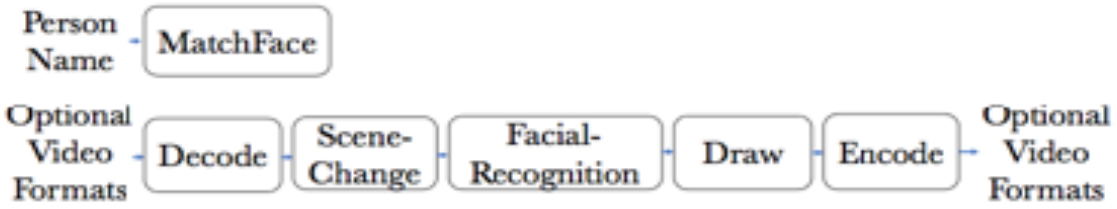

Figure 2: The Sprocket Framework [2, pp. 3]

Figure 2 shows the Sprocket framework. Sprocket have 3 broad steps: Decoding, Machine Learning, and finally encoding. All of the processes are done parallely on the cloud and use a technique called straggler mitigation, which will be discussed later. This makes Sprocket able to process videos faster than any of its competitors.

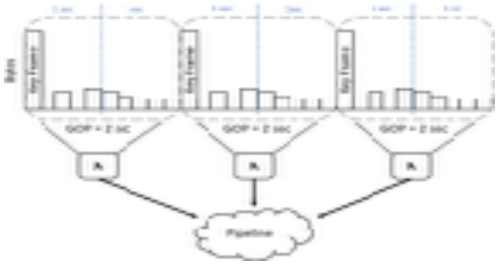

Figure 3:Each GOP has their own dedicated worker[2,pp.2]

Sprocket's first step is decoding. As mentioned previously, videos need to be decoded before they can be processed, meaning that for the square and circle illustration, the squares and circles are shaded back in. Under Sprocket, this is done parallely. The framework would first spawn as many cloud instances (or workers) as needed where each worker has their own lightweight serverless computing engine (AWS Lambda). The Group of Photos (GOP) from the compression will be divided amongst those workers, such that each GOP will have their own dedicated worker (Figure 3). In effect, all the GOPs are decoded parallely. The whole process is done within a fraction of a second. Furthermore, the parallel processing is done more efficiently using a technique called straggler mitigation.

Imagine being a worker on an assembly line. If another worker before you is working slowly, you, as well as every worker after you, will be slowed down. Similarly, in the Sprocket pipeline, one slow cloud instance, also known as a straggler, can greatly slow down the entire process, since each task in the pipeline is dependant on the previous task. Sprocket has proactive measures in place to lessen the effect of these stragglers on the overall performance through a process called straggler mitigation, shown to the left in Figure 4.

Figure 4: Straggler Mitigation

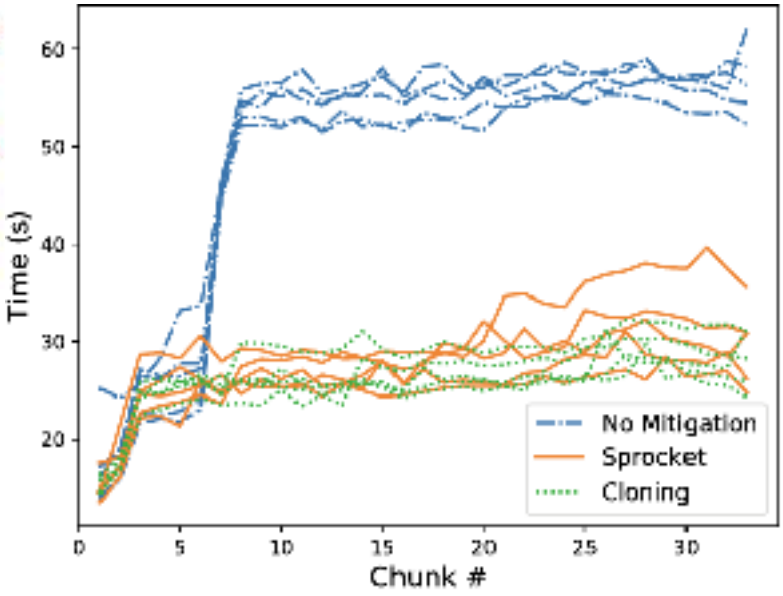

Figure 5: Performance Gains of Straggler Mitigation [2, pp. 10]

The first step of straggler mitigation is to double the length of GOPs. Then send each GOP to a pair of workers. Each worker processes one half of the GOP. Whichever worker finishes their half first proceeds to help its partner finish the other half as speculative execution. Since each GOP is double the length, there are half as many different GOPs. However, there are two copies of each GOP, so there are the same number of GOPs as before. Since most of the data in a GOP is stored in its key frame and straggler mitigation stores the same number of key frames as before, straggler mitigation ends up using less than 25% more data than before. This is a worthy trade off considering the performance gains from straggler mitigation, as seen in Figure 5.

Now that the video has been decoded, it can be processed. The first challenge would be to break the video into coherent scenes. This means that the computer has identify the beginning and end of every scene. For people, it may be easy to identify the start and end of a scene, but for a computer this is non-trivial. However, with the help of a computer vision model and algorithm, this can be accomplished. After that, the computer needs to identify the scenes that contain the target object and this is done by calling on the cloud's machine learning model. It's note-able that image detection is just an example use case chosen by the study to illustrate how the framework could be used. Users could substitute the image detection model with their own models to do a myriad of video processing works, like identifying an actor by voice or even

finding a scene by inputting movie lines. The right scenes would then be stitched together and then encoded back to its input format. It's important to note that all the tasks-- scenes detection, machine learning and encoding--are processed parallely by thousands of cloud instances: all done in a fraction of a second.

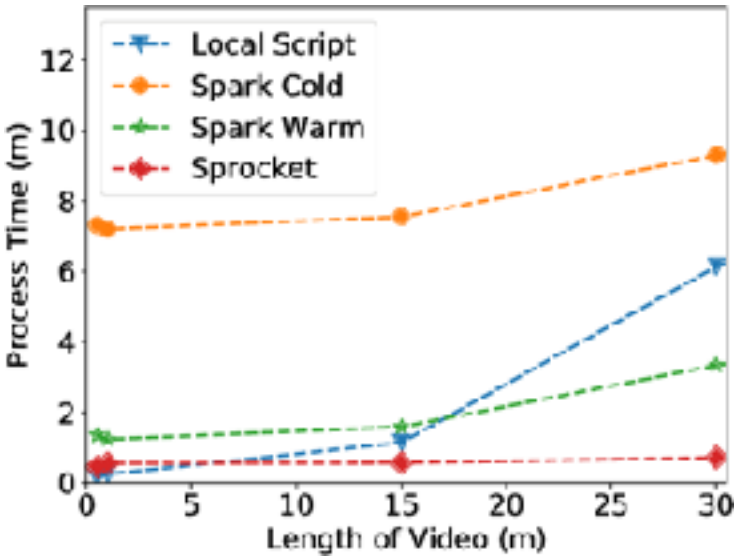

Figure 6: Performance Gains of Sprocket [2, pp. 11]

There are many cloud services other than Sprocket. However, most of them aren't built for video processing like Sprocket is. Other services take time to allocate a fixed number of cloud instances, whereas Sprocket quickly allocates a variable number of cloud instances depending on the size of the video. This gives Sprocket a significant performance boost in video processing over its competitors. Also, Sprocket is serverless, unlike many other cloud services, which makes it less dependant on a central authority. Sprocket has all of this for a better price. To process a 30 minute video, Sprocket costs only $0.63, whereas Amazon EC2 and Amazon Spark Warm cost $2.38 and $1.42, respectively [2, pp.11].

Sprocket provides a serverless cloud video processing framework that makes use of high parallelism and provides low latency, all at a low cost. While we went over one use case of Sprocket, there are many more ways to use Sprocket for video processing such as label detection, text detection, speech transcription, and much more. We believe Sprocket is truly revolutionary because it makes computationally expensive video processing more accessible.

Resources

[1] Cisco Visual Networking Index: Forecast and Methodology, 2016– 2021. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/ visual-networking-index-vni/complete-white-paper-c11-481360.html

[2] Lixiang Ao, Liz Izhikevich, Geoffrey M. Voelker and George Porter. Sprocket: A Serverless Video Processing Framework. University of California, San Diego, 2018. https://cseweb.ucsd.edu/~gmporter/papers/socc18-sprocket.pdf?fbclid=IwAR0St33FY9Nji3AV0J9b2vOiLlU-cOFK3mWqwQl_Yt9djxk3FgwbXSYBZCA