WASP SECC Course: Cloud Module Assignment Architecture

Per Boström, Kristoffer Bergman

May, 2017

1 Overview

Load balancer (both frontend and backend?), frontend, message queue, video conversion

"Nodes" related to the service:

- Load balancer
- Frontend Web API
- Message queue
- Backend Video conversion
- Video Storage
- Controller
- Monitor
- Workload generator

We will use a workload generator to simulate a number of users that submit video conversion requests

2 Workload generator

The workload generator is the test engine that simulates external users. It takes two parameters: the number of client converters and the average time between conversion requests. It will start the specified number of client threads of which each one represents a user. Each client randomly selects a video, sends a conversion request, waits for the completion, and then sleeps for a while (on average the specified time) before repeating the process.

The workload generator will also measure values such as response time and keep track of how many requests there are in the queue.

3 Controller

In order to make the service scalable, we must be able to scale up or down depending on its load. For this, we need a controller that controls the number of virtual machines used in the service.

We will use the simplifying assumption that all videos are of approximately the same size. This makes it possible to use the time from that a video conversion request is submitted until the video is converted as the control objective. Assume the average time it takes for one core to convert one video is given (including potential delays in the system), and denote it T_{conv}. With n_{queue} video conversion requests in queue and n_{vm} active VMs, the time T needed to convert all videos in the queue is given by

$$T = \frac{T_{\text{conv}} n_{\text{queue}}}{n_{\text{vm}}} \tag{1}$$

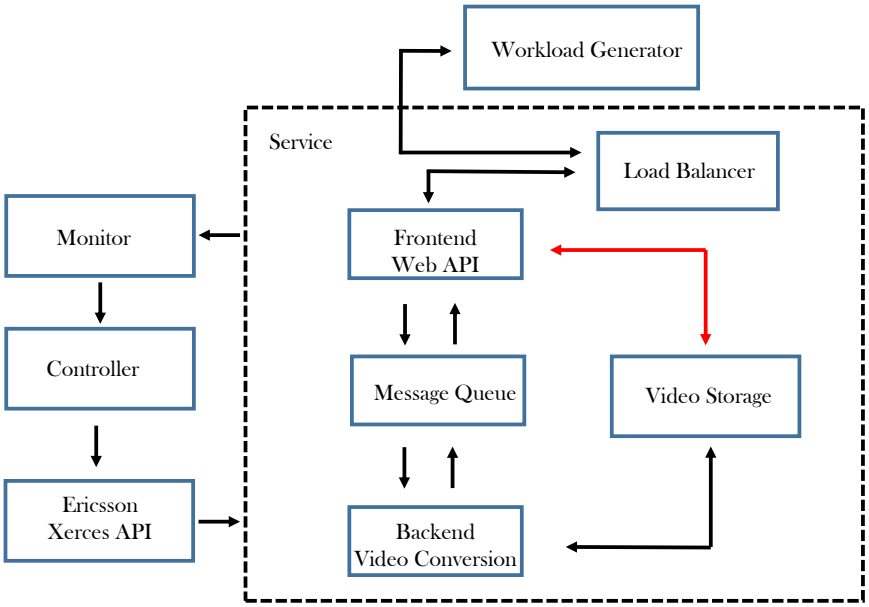

Figure 1: Overview of the architecture.

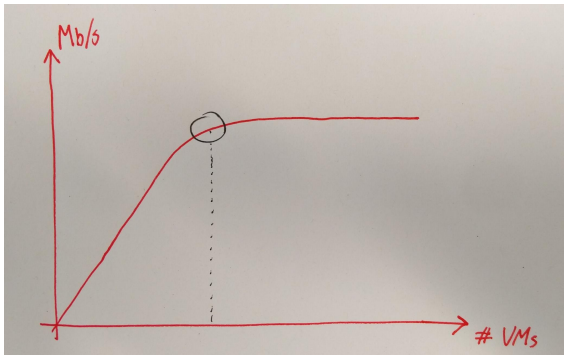

Figure 2: Assumed behavior of the video conversion rate as a function of active VMs. There is no gain in conversion rate if more VMs are added passed the circle.

Thus, if the control objective is to keep the maximum time from a request is submitted until the video is converted below $T_{\max}$, the number of VMs needed is given by

$$n_{\text{vm}} > \frac{T_{\text{conv}} n_{\text{queue}}}{T_{\max}} \tag{2}$$

This is of course a simple approach which, if needed, could be extended to also consider videos of varying sizes and possibly reordering the queue such that for example small sized videos are prioritized over larger ones.

3.1 Oversubscription

Due to oversubscription, the provided cloud environment does not guarantee that adding one VM will lead to an overall increase in performance, nor that all VMs will have the same performance over time. Assuming conversion rate in Mb/s, *i.e.* an entity inverse proportional to the time needed to process a video, depends on the number of active VMs in a way similar to Figure 3.1, it should be possible to optimize the number of VMs, to avoid wasting resources.