RenderFarm - Checkpoint - CNV 2016/2017

Catarina Cepeda
Instituto Superior Técnico
catarinacepeda@tecnico.ulisboa.pt

João Peralta
Instituto Superior Técnico
joao.peralta@tecnico.ulisboa.pt

Luís Santos
Instituto Superior Técnico
luis.d.santos@tecnico.ulisboa.pt

1. Introduction

We are quite far behind in terms of what would be expected for this stage but we have thought out our metrics system and how to use these metrics to our advantage in the load balancer and in the auto scaling group.

1.1. Current Implementation

The current implementation consists of a:

- Multi-threaded web server which can receive the requests and render an image.
- Metrics collection with BIT where by the RayTracer class is instrumented for dynamic code results logged to a file as of now.

2. System Architecture

The system as we have devised it is very similar in architecture to the image proposed in the project description although our goals for the metrics storage system are somewhat different.

2.1. Metrics Storage

Our goal for metrics storage is to use a DynamoDB table to store the relevant metrics collected by each of the instances this table is directly updated by the instance running the Render Server by the instrumented code. These metrics will then be processed by a separate EC2 instance running code which transforms the the metrics and stores them in a cache in such a way that the load balancer is able to query said cache quickly for a available server on which the rendering can be done.

2.2. Metrics Collection

As to metrics collected we are still unsure of which tactic to use as for useful metrics. We are not sure whether we should collect information about the complexity of scenes and use those approximations of complexity based on the number of relevant methods called by the RayTracer etc to estimate the duration of new queries coming in. Or if we should collect information about the current execution of the Render Servers in such a way that we have information at every second of the number of method calls made during that unit of time which in turns lets us be aware of the overall load of the EC2 instance and approximate a measure of ”health” based metrics from those fixed intervals.

We would also like to take advantage of custom cloud watch metrics to set custom rules for the Auto-Scaling group and for this we would need information at fairly regular intervals.

2.3. AutoScaling

For Auto Scaling we have so far only used a basic autoscaler based off of the one exemplified in the lab classes with growth in instances when cpu load is over 80% for more than 60 seconds and a decrease when load falls below 40% for the instances in the group. For the grace period and health checks we have so far left the default values in place.

3. Work for Final Hand In

For the final hand in we have to:

- Choose a metrics collection approach that suits us best and implement the necessary code to write the metrics to a DynamoDB.
- Implement a metrics calculation service to compact the metrics collected for use in the Load Balancer so as to

keep Load Balancer work to a minimum.

- Publish custom CloudWatch metrics from the Render Servers to improve the auto scaling of the EC2 instances.