TÉCNICO LISBOA

Cloud Computing & Virtualization

2017/2018

58543 Dário José Da Silva Ornelas

78895 Luís Miguel Paiva Sampaio Santos

91062 Tiago Alexandre Serafim Monteiro

22 April 2018

First Checkpoint

This report addresses all the functionality that was implemented as well as the future functionality to be implemented.

We are still using most of the features provided by AWS (load balancer, auto-scaler) although we already have a working sketch of the load balancer, the auto-scaler and a proxy to handle and forward requests to the worker instances.

Implemented Functionality

Webserver

The Web Server was refactored to operate in a multithreaded way and already accepts request on the endpoint mzrun.html with the following parameters:

- ?m=<maze-filename>&
- x0=<x start point>&
- y0=<y start point>&
- x1=<x end point>&
- y1=<y end point>&
- v=<velocity>&
- s=<strategy>

The output is a page with the resulting maze.

Load Balancer

The load balancer we are using at this checkpoint is the one provided by the Amazon Web Services, a Class Load Balancer, that provides routing either through transport layer or application layer. The only configuration needed was the listeners, routing http port (80) to the web servers ports (8000) and applying a security group to allow remote connections from outside and health checks.

Auto-Scaling

The auto-scaling solution is also based in AWS, we created a “Launch Configuration” with the following config AMI:

ami-7e5ef401 (created by us)

TYPE: t2.micro

SEC_GROUP: TCP 8000; TCP 22

We then created an Auto Scaling Group within the same network and subnet as our web servers and with the option to receive traffic from one or more load balancers and choose our load balancer created earlier.

Our scaling policies are between 1 and 2 machines (for now), when the number of machines is below the upper threshold (2) and the CPU Use > 60% for more than 3 minutes we create a new instance, and if we are above the lower threshold (1) and

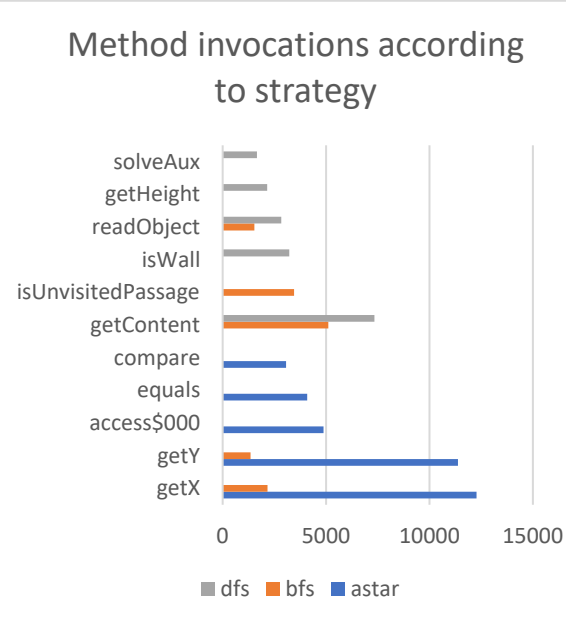
the CPU Use < 40% for more than 3 minutes we terminate an instance.

Mazerunner

No changes were made to the Mazerunner. (only instrumentation – see next point)

Instrumentation

The information in the following image was crucial to decide which methods and classes to instrument.



By analyzing the invocations of methods according to the strategy selected to solve the maze, we got the five most invocated methods. With this information we proceeded to instrument only the most relevant methods for each strategy. At this point, our metrics are being saved as a csv file with all the information regarding the request, like thread id, request parameters and metrics.

DynamoDB

We only have the basic configuration of Dynamo DB working, we still need to decide which metrics will be stored. Auto Scaler needs to be updated to take metrics into account, instead of using Amazon's standard one.