Group 31

P1 - Report

Inês Garcia 99083
João Costa 99088
Miguel Mendo 99111

Current implementation

Architecture

The current system consists of the provided webserver, the Javassist application with a new tool called InstructionCounter, which counts the number of instructions executed by a given task, and finally an auto scaler group and load balancer hosted on AWS.

Metrics

The load metrics are currently being stored in files dedicated to each javassist tool, thus, for now, we're creating a file InstructionCounter.txt in the webserver directory inside each worker.

Auto Scaler Group

The AWS implemented auto scaler group is set up with alarms from CloudWatch. One activates the trigger to add one VM whenever the overall CPU utilization from the last minute exceeds 70%. The alarm to reverse this operation, removes one VM whenever the CPU utilization is lower than 30% on average for the past minute.

Load Balancer

We're using the AWS Classic Load Balancer (CLB), set with the default distribution algorithm, for now, i.e. just a simple Round Robin.

Future implementation

We plan on implementing a custom Load Balancer and Auto Scaler Group, as requested, so that we are able to use the metrics from Javassist to aid the decision-making on both components. It will also include a custom workload heuristic scoring (henceforth WHS) algorithm applied to incoming tasks.

Load Balancer

It'll distribute incoming requests based on the tasks' WHS and the resources available in each working VM. If they are low on load, it'll be preferrable to make them do the work. Once their resources become exhausted, new, low WHS tasks can be redirected to Lambda workers to prevent, e.g., needless worker creations in case of spikes. Prolonged exhaustion of resources will be handled by the auto scaler, by creating more workers.

Auto Scaler

We are considering creating a custom Auto Scaler cluster too, in case we can infer useful knowledge from the metrics or from the WHS for this component of the system too. It will at least adjust the number of workers based on sustained demand, akin to the checkpoint turn in. If this ends up being the only behavior of the component, we will likely leave it as an AWS AS group.