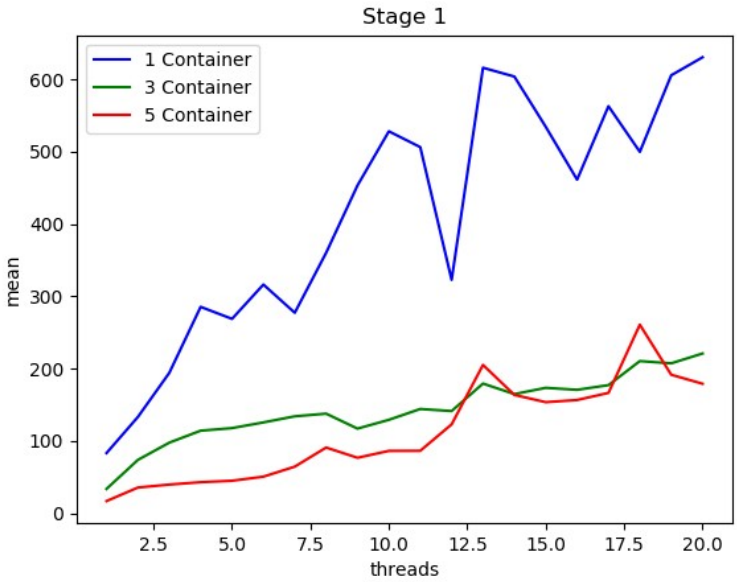
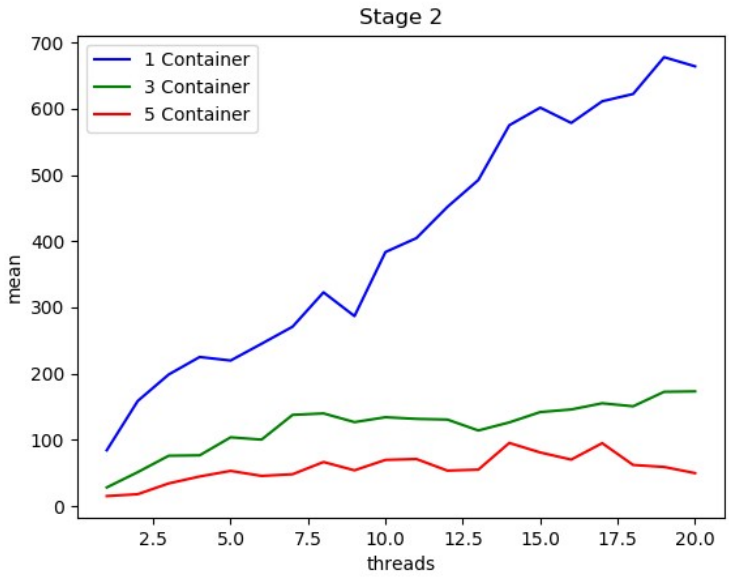
Documentation

Benchmarking:

Outcome:





Discussion:

The first thing we can immediately observe when we look at both graphs, that stage 2 appears a lot more stable. There are fewer break-ins or peaks. Also, stage 2 profits from a lower mean in most cases. Especially the 3 and 5 container lines of stage 2 are either on the same level as their opponents of stage 1 or have a lower mean.

Which makes sense, because stage 2 was containerized with docker-compose. Docker-compose is a tool made for the efficient running of multi-container applications, while docker (stage 1) is not. Docker is the predecessor of docker-compose and focuses more on single container applications.

Another thing is that the multiple containers of stage 1 and 2 are faster than the single containers of stage 1 and 2. This has to do with the fact that there are multiple threads involved. The more threads interact, the more a single container is busy, while multiple containers can share the load.

Furthermore, the lines with just 1 and 3 container suffer from a slight overhead due to the load balancer, which assumes that there are always 5 containers. If one container is missing or has failed, the balancer tries the next one. The load balancer could change the number of containers easily in the router.js file. Although, tests have shown that the differences, in this case, are quite small.

Execution:

Plot graphs:

Execute: *pyhton3 plot.py*

Execute stage 1&2:

The archive doesn't contain the redis database. The code assumes that the database *redis.rdb* can be found under: */var/cec/*

However, to start stage 1 or 2 just execute: *./execute_stage1.sh* or *./execute_stage2.sh*

Both stages get executed with 5 containers. To change that, you have to adapt the shell files. In execute_stage1.sh just remove the unneccessary containers under section: task c

In execute_stage2.sh just edit the variable at the beginning.

It is no problem to execute multiple times stage1 or stage2. Also stage 1 and then stage 2 is no problem. Although, if you want to execute stage 1 after you already executed stage 2, please execute this is in stage 2 before: *docker-compose –compatibility down*

Just to make sure that there is no problem with the naming