\*

Container management systems have appeared recently as an alternative to virtual

machines, offering relative isolation without the overhead of running virtual machines (VMs).

Containers are attractive for its use in distributed systems thanks to its horizontal scaling

properties. In this project, the performance of virtual machines and the Kubernetes container

management system is evaluated, in order to compare the **computing performance** and

**economic cost** of both alternatives.

\*

Virtualization is the key to manage all the aggregate resources that form cloud

networks. Over the bare metal hardware, instances of virtual machines and storage devices

can be created and destroyed, enabling developers to scale their services to the demand and

avoiding problems such as stated earlier.

The popularization of public cloud providers and virtualization has opened new

possibilities for adjusting the expenses to the minimum. When the user load is low, services

can scale to the minimum number of computing resources needed, if the load increases, the

service can instantiate new virtual machines, scaling the capacity of the service to

accommodate the increased number of clients.

--

Even though services backed by virtual machines do not take a lot of time to scale

(usually in several minutes), there are situations where the increase in load is fast enough to

not let the service process all the incoming requests before it scales to the new capacity.

One of the solutions to this problem that is recently gaining momentum recently is

deploying services on container based systems

--

its **benefits** over virtual machines is the capacity of

scaling in seconds instead of minutes, which should in theory let systems scale in time and

minimize the impact on the user level experience during a load increase

\*

the main objective of this project is creating an

environment to test the computing performance and horizontal scaling capabilities of systems

backed by traditional virtual machines and by container management systems, using defined

metrics to make a comparison.

\*

In conclusion, the technologies that will be used to conduct the tests needed for this

project are Amazon Web Services for the virtual machine tests and Kubernetes for the

container based tests

The following figure represents the testing environment using AWS.

F2.2

\*\*

In order to deploy the cluster into AWS, Kubernetes distribution comes with tools

to automatically create and destroy the required EC2 instances and other computing

resources on the cloud.

Environment setup

\*

The test has been designed to have a total length of 10 minutes, having a warmup

and a cooldown of 10 seconds at the start and end of the test. Requests are sent with a rate

of 50 request per second.

F3.2

\*

Each cycle takes 3 minutes with a total of 5 cycles. The request rate

is set to 50 requests per second when load is high and to 20 requests per second when load

is low.

F3.4

\*

The Kubernetes cluster was formed by 5 instances, 5 minions and 1

master instance. The traditional virtual machine configuration started with 1 instance and

could scale up to 5 instances.

The horizontal axis represents the time stamp of each sample in minutes, while in

the vertical axis the mean response time is represented through the orange color and, in the

case of the container based test, the service average CPU usage is represented with the blue

color. Green horizontal bars are used to mark when the service scales horizontally, changing

the number of containers or virtual machines depending on the test. Samples are spaced

each 30 seconds approximately, taking 10 minutes the first two tests and 15 minutes the

third one.

The graphs on the left side represent the results obtained in the container based test, while

the graphs on the right side represent the results obtained in the virtual machine based test.