Task 1. Scaling distributed Systems using direct and indirect communication middleware

The goal of this task is to implement two scalable applications using four communication middleware services XMLRPC, PyRO, REDIS and RabbitMQ.

First of all, you will implement an InsultService that can receive insults remotely and store them in a list only if they are not already there. The service also offers mechanisms to retrieve the entire list of insults, and an insult broadcaster that periodically publish random events to interested subscribers every 5 seconds.

Secondly, you will implement an InsultFilter service based on the Work queue pattern. This service allows clients to send texts that may or may not contain insults. The Filter service will replace insults in the text with "CENSORED" and store the filtered text in a result list. The service could also return the result list if required.

Provide unit tests or demonstration scripts that show how both services work correctly. Implement them using the four aforementioned technologies. Once developed the four versions you must now prepare a performance analysis with the following considerations:

1. Single-node performance analysis: Stress test both services using a single node (InsultService, InsultFilter) to understand how many requests can process. Compare the four systems using plots with the results of the four implementations and reason about those results.
2. Multiple-nodes static scaling performance analysis: Stress test both services using now one, two, and three nodes and calculate speedups obtained in the four systems. Modify the architectures if needed to accommodate more servers.
3. Multiple-nodes dynamic scaling performance analysis: Select one architecture or modify it combining different middleware services to improve its performance. Propose a dynamic scaling mechanism that can add or remove nodes according to the workload. Note that Python threads do not use correctly CPUs in the machine, use processes instead (Python multiprocessing library)

We propose you the following formulas:

Speedup

$$S = \frac{T_1}{T_N}$$

Where:

- S = Speedup
- T_1 = Execution time with a single worker (or single processor)
- T_N = Execution time with N workers (or processors)

Dynamic scaling using message arrival rate

$$N = \left\lceil \frac{\lambda \times T}{C} \right\rceil$$

Where:

- N = Number of workers required
- λ = Arrival rate of messages (messages per second)
- T = Average processing time per message (seconds per message)
- C = Capacity of a single worker (messages per second)

Alternatively, if backlog is considered:

Dynamic scaling based on backlog and message arrival rate

$$N = \left\lceil \frac{B + (\lambda \times T_r)}{C} \right\rceil$$

Where:

- B = Current queue backlog (messages waiting to be processed)
- T_r = Target response time (seconds)

Provide a github repository with all code and a readme file with instructions to run the different tests. Upload to moodle a pdf containing the link to the gihtub repository and documentation about the design and architecture of each solution, and an experiment section explaining all plots and results. Compare the different solutions and justify the results that you obtain.