Dynamic Adaptation of Cooldown Period for Auto Scaling of VNFs

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Mohit Kumar Singh, Gaurav Garg, Tulja Vamshi Kiran Buyakar, Venkatarami Reddy, Antony Franklin A, and Bheemarjuna Reddy Tamma

Department of Computer Science and Engineering, IIT Hyderabad, India

Introduction

- Recently, network operators and data centers have moved towards Virtualization of Network Functions (VNFs).
- An Auto Scaling Entity (ASE) monitors the load on the VNF and scale up services depending on the usage of the VNF instance.
- ASE checks the load at each time interval called monitoring period and the VNF needs to wait for cooldown period to ensure that scaling action takes place.
- Existing algorithms, used by the operators to scale the VNFs, use a static cooldown period.
- We propose an algorithm with the dynamic adaptation of cooldown period for scaling of the VNF.

Simulation

- We generate number of HTTP requests based on poisson distribution model to a docker acting as an HTTP server and then, monitoring the CPU utilization of the docker.
- We compare the performances of both the algorithms in terms of number of instances instantiated and number of monitoring requests made over time.

Conclusions

- We conclude that the proposed algorithm is better than the existing approach, where unnecessary scalings are avoided.
- This is because the proposed algorithm checks the utilization at short and regular intervals. This enables the ASE to take the decision at the time when scaling is actually needed.
- However, the proposed algorithm induces an overhead in terms of monitoring requests to CPU.
- As future work, we will try to apply machine learning models to predict the behaviour of traffic to take effective scaling decisions.

Algorithms

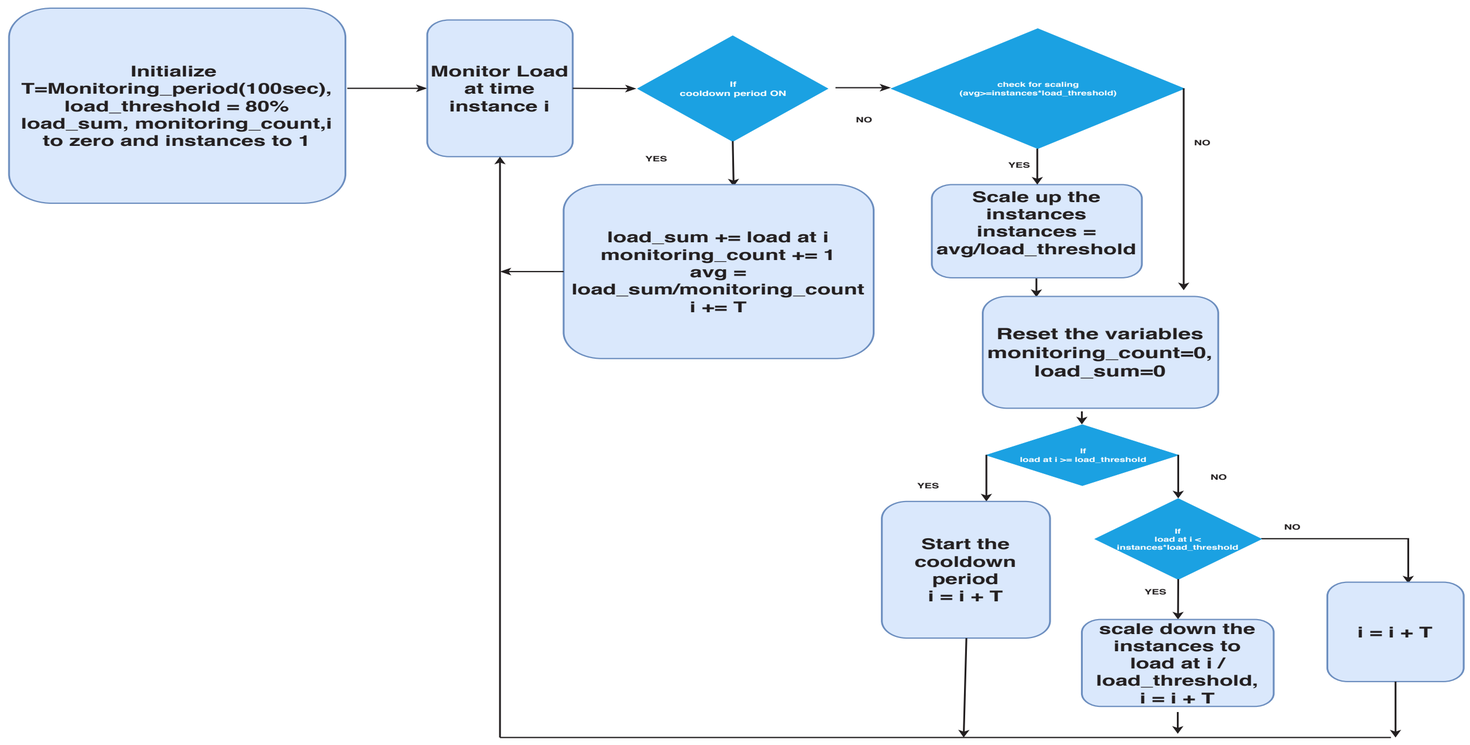


Fig. 1: **Existing Scaling Algorithm.**

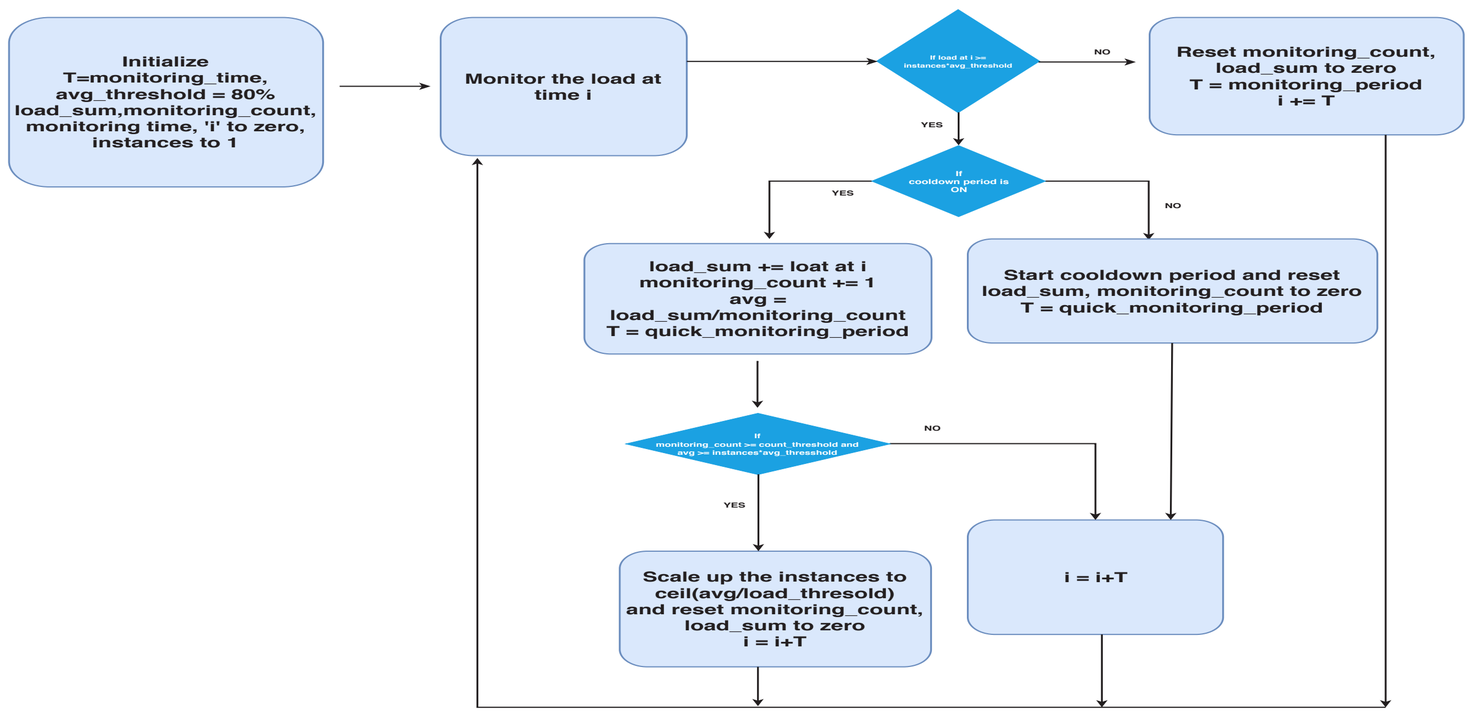
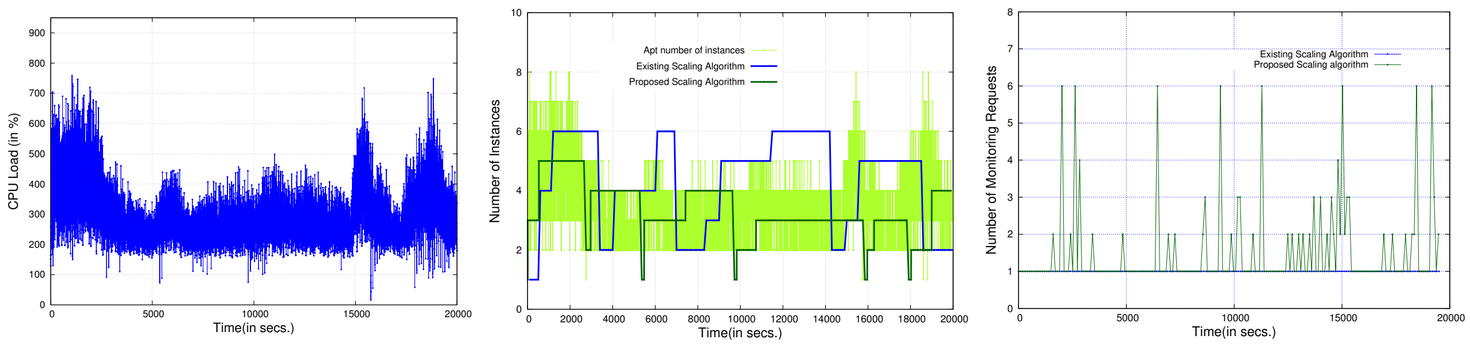


Fig. 2: **Proposed Scaling Algorithm.**

Results



(a) **CPU util. of docker.** (b) **# Active instances.** (c) **# Monitoring requests.**

Fig. 3: **Evaluation of the proposed algorithm**

- Fig. 3 (a) shows the CPU load pattern on the server, on which we test the algorithms.
- Fig. 3 (b) shows how do both algorithms react to traffic. The proposed algorithm uses less number of instances at the times when less number should have been used, and vice-versa.
- Fig. 3 (c) shows number of CPU monitoring requests made in each interval. A total of 200 and 394 monitoring requests have been made by existing and proposed algorithm respectively.