Dynamic Auto-scaling of VNFs based on Task Execution Patterns

Asif Mehmood
Computer Engineering
Jeju National University
Jeju, South Korea
asif@jejunu.ac.kr

Talha Ahmed Khan
Computer Engineering
Jeju National University
Jeju, South Korea
talhajadun@gmail.com

Javier Jose Diaz Rivera
Computer Engineering
Jeju National University
Jeju, South Korea
shaifvier@gmail.com

Wang-Cheol Song
Computer Engineering
Jeju National University
Jeju, South Korea
philo@jejunu.ac.kr

Abstract—Investigation and collection of real-time data plays a very crucial part in the orchestration of network resources. Selection of the correct data is very important as it decides to auto-scale the resources. In cloud & SDN environments such as NFV, auto-scaling becomes more critical in terms of precision and accuracy. In our case, we propose a solution for auto-scaling the network resources based on the calculations made for every action's execution-time [1] of respective instances of a VNF. The instances for each VNF are auto-scaled on the basis of execution-times per time slot, and the number of cores that are assigned by the usage of weight factor [2] used for virtual/physical cores. Hence by using the proposed solution, we are able to enhance the proper resource provisioning to fulfill the dynamic demands [3] of future mobile networks.

Keywords—autoscaling, datacenter, sdn, nfv, vnf, execution-time, self-management, networks

I. INRODUCTION

With the advent of technology, SDN revolutionized the field of networks. The price of network devices got cheaper as the brain of those devices were extracted and put into a software. With this simple and robust approach of SDN, there were a lot of complications related to performance as well. Centralizing the control plane of network functions led to the idea of developing virtual network functions which provided a clean platform for network developers to evolve the networking systems.

Autoscaling in networks is relatively more important. Its importance is in itself an ocean, due to the nfv architecture. Previously, when the network operators had to scale up/down the resources it was a quite hectic job which was also prone to errors. Another disadvantage of traditional networks was that there was no room to improve networks. With the possibility of autoscaling network resources, there is still a lot of effort to do because of different factors involved such as clouds machines having different capabilities. So, this point drew our attention to develop applications that take decision to autoscale VNFs.

Our proposal uses clocks per cycle unit to decide how much cores to be assigned and we use monitored [4] execution-times [1] of respective VNFs by our proposed microservice. We use weight factor for hyperthreading overhead caused by v-cores.

So, in order to fulfil the criteria of our proposal, we use different projects and terminologies in the paper which are listed as M-CORD [5], OpenStack and ONOS. M-CORD platform [6] refers to mobile central office re-architected as a datacenter. It is owned by ONF and it provides a platform to develop and deploy network functions and a limited control of updating the network resources at runtime. This project makes the use of two widely used open source projects OpenStack and ONOS integrated with their own built XOS [6]. It is on the top layer of these projects allowing us to build deploy and run everything as a service.

We developed the NSSF [7] compliant to 3GPP standards and integrated it with the OAI provided network functions. Then the autoscaling algorithm was implemented and deployed to get the benefits mentioned in the coming sections.

Section II provides a brief overview of the necessary literature to clarify all of the project use-cases and terminologies used throughout this paper. Section III contains the proposed system in full details. It starts with the figure of our architecture and then proceeds with the detailed mechanism of this architecture. Section III puts light on evaluation of our system with the experimental setup procedure, comparisons made. Section IV wraps up document by making concluding remarks.

II. LITERATURE REVIEW

As it is obvious that the networks will be dependent on systems which are automatic and intelligent. One aspect which we propose and evaluate in this paper is related to autoscaling the network resources. A crucial factor that needs to be considered is clocks per cycle.

A. Related to resource-allocation

Kapil Kumar from IIIT proposed a system to dynamically allocate memory and cores for VMs. Author uses a concept of using feedback control techniques [3] based on monitored data for each of process and execution-times. Proposal gives us confidence to apply dynamicity in resource allocation process.

In paper "Autonomic Workload and Resource Management of Cloud Computing Services", author proposes a solution to reduce power consumption of machines by allocating cores and memory up to optimum range [2] by to the factors chosen.

"Auto scaling of data plane VNFs in 5G networks" [8] refers to a system design proposed to autoscale the data plane VNFs in 5G networks. This works relates to our work in terms of network slicing as both of the systems which are termed as LTE Advanced and 5G aim to provide it.

Another document termed as “A Context Based Scheduling Approach for Adaptive Business Process in the Cloud” takes execution time into account to reduce cloud resources usage by considering a mechanism based on execution times of a task. We take maximum execution-time instead of minimum execution-time [1] to reduce the chance of service-disruption and delay.

We also refer the work done by In-Yong Jung in his research paper “Selective Task Scheduling for Time-targeted Workflow Execution on Cloud” [9]. It uses to algorithms to estimate the processing-time for applications or programs running on cloud. As capability of both algorithms are different in calculating different type of tasks. We also consider this work for the future to improve the system further for better results.

B. Related to platforms

A paper “An intent-based mechanism to create a network slice using contracts” written by Asif Mehmood, developed a solution to dynamically create network slices with help of contracts [7]. This work has been helpful in the way that this approach defines the proper flow of execution needed to allocate resources and then to provision them accordingly.

In the paper “Implementation of VNFC monitoring driver in the NFV architecture”, author proposes a monitoring [4] driver which is closer to the other management services in order to achieve the direct communication. This approach enhances the process of fetching the information from the resources.

The paper “Introducing network slice management inside M-CORD-based-5G framework” puts some light on framework provided by ONF “M-CORD”. The author proposes a slice management technique for 5G network functions. This literature is useful in the sense to get a know how to setup environment [5] and to deploy VNFs inside this framework.

III. SYSTEM DESIGN

Our system contains two contributions. An autoscaling application as depicted in “Fig. 1” at the application layer. It includes modules such as Autoscale-controller, Information-handler and a Configuration-invoker.

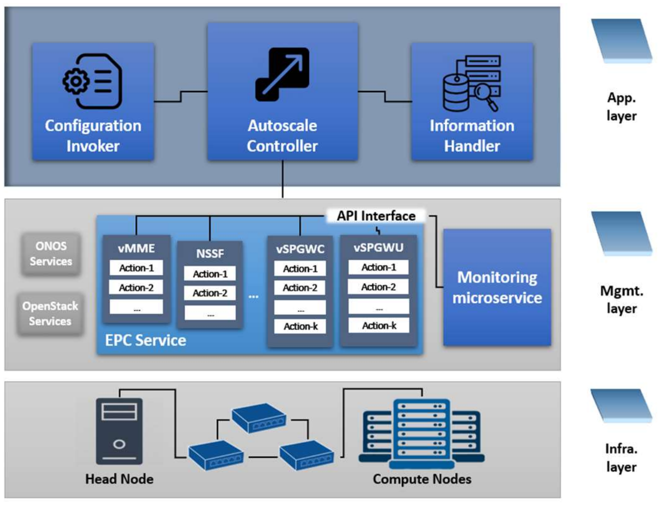


Fig. 1. Overall System Architecture for Autoscaling VNFs.

The second part of the contribution is a microservice based monitoring service shown at the management layer of the NFV architecture. It interacts with the core-network elementary management services which are termed as synchronizers [6].

A. Autoscaling application

The application’s sole purpose is to autoscale the VNFs. It consists of 3 modules termed as Autoscale Controller, Information Handler and Configuration Invoker.

The steps to complete the mechanism by our proposed system can be seen in “Fig. 2”. It is shown to avoid ambiguity. These steps work in parallel to provide a full fledge functionality. The upper part of “Fig. 1” is purely developed by us and remaining part of system is modified to fit in proposed criteria.

To clarify how the functionality achieves its goals, we start from monitoring microservice. Service fetches information from each of VNFs. We have modified synchronizers to improve efficiency. For this purpose, the API interfaces were used in order to attain platform independency. During this interval, each of the VNF’s information is fetched and then passed over to the Information Handler which stores the information in its store.

Autoscaling application comes into action to scale up/down VNFs respectively. The flow chart and algorithm are described in the coming subsections. The final decisions made by autoscaling includes conversion of execution times to required cps. The process moves further by requesting Configuration Invoker to generate configuration for VNFs. Configurations are specified in TOSCA [6] acceptable by XOS [10] and passed to Autoscale Controller, through an API interface. This way, we push system towards a state where resources are not over-utilized neither under-utilized.

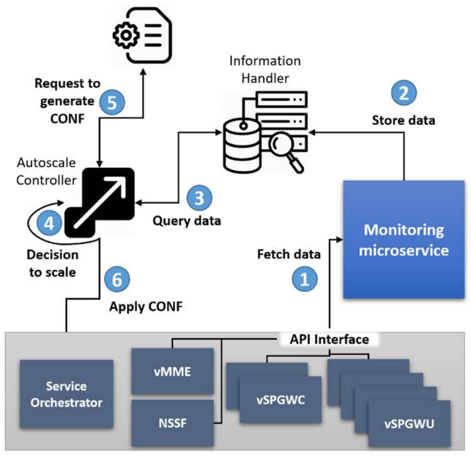


Fig. 2. Mechanism-steps for the overall system.

B. Microservice based monitoring service

The microservice shown in “Fig. 2” is deployed in parallel with the synchronizers which are often termed as the elementary management services. The purpose of deploying it in such a fashion is to enhance the performance of fetching information. As it is very close to the deployed VNF’s management services, latency is reduced. It can be seen in “Fig. 1” that synchronizers and the monitoring microservice are deployed on head-node. It is shown in “Fig. 1” that each of VNF is composed of a number of actions having its own execution-times. Requests posted to each VNF is in fact a request to perform an action at the end, which has a fixed execution-time.

C. Autoscale application internals

The internals of this application include an Information handler, Autoscale controller and a Configuration invoker. Each module playing its role to fulfil the criteria

1) Information handler

The module stores real-time information for the number of the requests received by each action of VNF. It is evident from "Fig. 3" that we can extract information of cps from execution-times for each request. This approach allows the usage of real-time data and a highly customizable approach to store data.

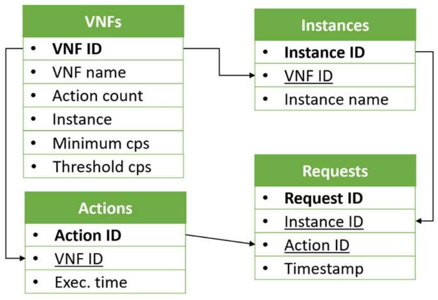
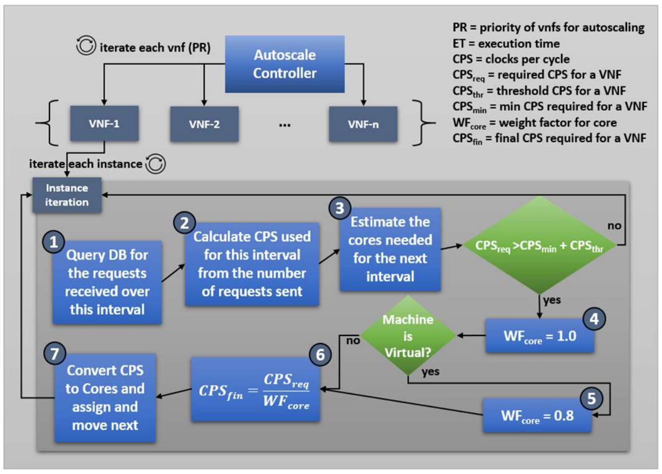


Fig. 3. Design for the Information Handler's storage.

2) Autoscaling controller

This subsection explains decision-making process of autoscaling the VNFs by iterating each of the VNF instances as shown in "Fig. 4" and "Fig. 5". Information of instance is fetched from database shown in "Fig. 3". Then with help of using number of requests and execution-times we estimate cps.

If the resources to be allocated are onto a physical machine, the weight factor for core is set to a value of 1.0 otherwise it is set as 0.8. Reason being that the virtual-core has less tendency than the physical-core.



iterate each vnf (PR)
Autoscale Controller
VNF-1
VNF-2
...
VNF-n
PR = priority of vnfs for autoscaling
ET = execution time
CPS = clocks per cycle
CPS_{req} = required CPS for a VNF
CPS_{thr} = threshold CPS for a VNF
CPS_{min} = min CPS required for a VNF
WF_{core} = weight factor for core
CPS_{fin} = final CPS required for a VNF
iterate each instance
Instance iteration
1 Query DB for the requests received over this interval
2 Calculate CPS used for this interval from the number of requests sent
3 Estimate the cores needed for the next interval
$CPS_{req} > CPS_{min} + CPS_{thr}$
no
yes
4 $WF_{core} = 1.0$
Machine is Virtual?
no
yes
5 $WF_{core} = 0.8$
6 $CPS_{fin} = \frac{CPS_{req}}{WF_{core}}$
7 Convert CPS to Cores and assign and move next

Fig. 4. Flow-chart for Autoscaling decision-making.

Algorithmic representation of mechanism is shown in "Fig. 5". For each iteration, relevant and required resources are calculated based on cps. Finally, the algorithm auto-scales the resources as an outcome. By taking a few important factors into consideration shown in both "Fig. 4" and "Fig. 5". Efficiency of system developed will be evaluated

```
PR = { vnf_3, vnf_n, ... vnf_2 }
for all vnf in PR do
    for all instance of vnf do
        initialize $cps_{fin} = null$
        query requests for instance $instance$
        calculate $cps_{req}$ for next interval
        setup the $cps_{min}$ and $cps_{thr}$ for this $instance$
        if $cps_{req} > (cps_{min} + cps_{thr})$ then
            $wf_{core} = 1.0$
            if $machine\ is\ virtual$ then
                $wf_{core} = 0.8$
            end if
            $cps_{fin} = \frac{cps_{req}}{wf_{core}}$
        end if
        $GHz_{req} = cps_{fin} * 10^9$
    end for
end for
```

Fig. 5. Algorithm for Autoscaling decision-making.

3) Configuration invoker

This section describes responsibility of module. It generates configurations and then passes onto the Autoscaling controller, in TOSCA format. These configurations are reflected on system via synchronizers which were modified for each VNF instance.

IV. EVALUATION AND RESULTS

We start by evaluating our results in cpu usage by VNFs, assigned cpu usage and total cpu available usage of whole system. "Table I" shows cpu usage in percentages. It can be observed that usage assigned is fit according to situation. At time t=0, the usage is nearly 60%. In addition, with passage of time (from t=0 to t=4), the usage percentage increases up to 90%.

TABLE I. USAGE OF ASSIGNED – INDIVIDUAL

VNF	Usage % at time "t" for VNFs				
	t = 0	*t = 1*	*t = 2*	*t = 3*	*t = 4*
eNB	64.29	90.00	90.00	80.00	88.89
vMME	64.29	80.00	100.00	90.00	90.00
NSSF	33.33	100.00	75.00	75.00	75.00
vHSS	66.67	60.00	75.00	100.00	80.00
vSPGWC	58.33	75.00	100.00	62.50	83.33
vSPGWU	83.33	63.64	100.00	100.00	90.00

The graphical form of "Table I" is given in "Fig. 6". All VNFs usage is increasing with passage of time showing that resources are being used up to almost full capacity.

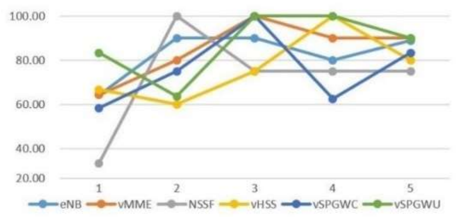


Fig. 6. Usage of Assigned CPU – individual.

Table referred as "Table II" shows cpu usage in cores for all VNFs at time interval "t" and the assigned cpu is decreases. For time interval t=0, usage of assigned resources is moving upwards and only 13% part is not in use for interval t=4.

TABLE II. USAGE OF ASSIGNED – OVERALL

Time	Overall assigned usage % at time "t"			
	Assigned	*Assign used*	*Used*	*Available*
t = 0	64	*41*	*64.06*	*35.94*
t = 1	47	*36*	*76.60*	*23.40*
t = 2	42	*39*	*92.86*	*7.14*
t = 3	45	*38*	*84.44*	*15.56*
t = 4	44	*38*	*86.36*	*13.64*

The graphical form drawn from "Table II" can be seen in "Fig. 7" below, that the resources are being fully utilized.

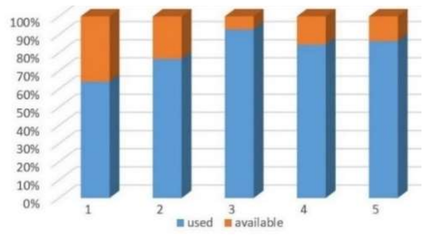


Fig. 7. Overall assigned usage % at time "t".

Table referred as "Table III" shows the readings observed for total number of cores over the time intervals "t". From time interval t=0 to t=4, the freed-up resources have moved from 0% to 31.25%.

TABLE III. USAGE OF TOTAL CPU – OVERALL

Time	Total CPU usage % at time "t"				
	Total	*T- used*	*T-free*	*Used*	*Free*
t = 0	64	64	0	100.00	0.00
t = 1	64	47	17	73.44	26.56
t = 2	64	42	22	65.63	34.38
t = 3	64	45	19	70.31	29.69
t = 4	64	44	20	68.75	31.25

The graphical form drawn from "Table III" can be seen in "Fig. 8" below, showing that the resources are being freed up as the time is passing from t=0 to t=4.

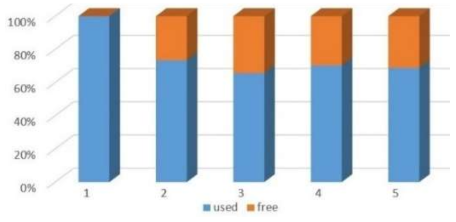


Fig. 8. Total CPU usage % at time "t" (bar-graph).

The graphical form drawn from "Table III" can be seen in "Fig. 9" below which shows that the total number of resources are freed with the passage of time. The curve line-graph can be seen in the below figure.

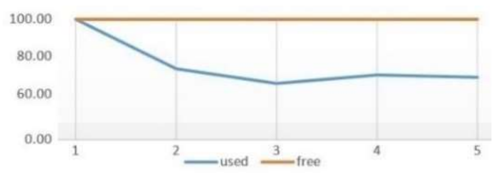


Fig. 9. Total CPU usage % at time "t" (curve-graph).

V. CONCLUSION

As autoscaling is an important aspect of management of resources and an addition is its automatic nature. We managed to propose a solution where the different factors such as execution-time, cps and weight-factor are considered during the decision-making process of autoscaling. This approach resulted in better resource management capabilities as the time interval passes on. The first reason being, it's very true nature of freeing up the unused or unneeded resources. Second reason was the usage of assigned resources were optimal reflecting that the resources were not overutilized nor underutilized. For future direction, we plan to enhance this mechanism by taking other resources into account and by introducing other factors as the different resource type usages rely on a variety of other factors.

ACKNOWLEDGEMENTS

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2019-2017-0-01633) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation).

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B01016322).

REFERENCES

[1] Molka Rekik; Khoulould Boukadi; Hanene Ben Abdallah, "A Context Based Scheduling Approach for Adaptive Business Process in the Cloud" 2014 IEEE 7th International Conference on Cloud Computing

[2] Farah Fargo; Cihan Tunc; Youssif Al-Nashif; Ali Akoglu; Salim Hariri, "Autonomic Workload and Resources Management of Cloud Computing Services" 2014 International Conference on Cloud and Autonomic Computing

[3] Kapil Kumar; Nehal J. Wani; Suresh Purini, "Dynamic Memory and Core Scaling in Virtual Machines" 2015 IEEE 8th International Conference on Cloud Computing

[4] Hyunsik Yang; Briytone Mutichiro; Younghan Kim, "Implementation of VNFC monitoring driver in the NFV architecture" 2017 International Conference on Information and Communication Technology Convergence (ICTC)

[5] Muhammad Tahir Abbas; Talha Ahmed Khan; Asif Mahmood; Javier Jose Diaz Rivera; Wang-Cheol Song, "Introducing network slice management inside M-CORD-based-5G framework" NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium

[6] https://guide.xosproject.org/dev/synchronizers.html

[7] Asif Mehmood; Talha Ahmed Khan; Javier Diaz Rivera; Wang-Cheol SONG, "An intent-based mechanism to create a network slice using contracts" Proceedings of Symposium of the Korean Institute of communications and Information Sciences

[8] Tulja Vamshi Kiran Buyakar; Anil Kumar Rangisetti; A Antony Franklin; Bheemarjuna Reddy Tamma, "Auto scaling of data plane VNFs in 5G networks" 2017 13th International Conference on Network and Service Management (CNSM)

[9] In Yong Jung; Chang Sung Jeong, "Selective Task Scheduling for Time-Targeted Workflow Execution on Cloud" 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS)

[10] Chen Zheng; Lei Wang; Sally A. McKee; Lixin Zhang; Hainan Ye; Jianfeng Zhan, "XOS: An Application-Defined Operating System for Datacenter Computing" 2018 IEEE International Conference on Big Data (Big Data)