

Elastic Auto-Scaling Architecture in Telco Cloud

Dang Sao Cao^{*,**}, Dinh Tam Nguyen^{*,**}, Xuan Chinh Nguyen^{*}, Van Thuyet Tran^{*}, Hai Binh Nguyen^{*},
Khac Thuan Lang^{*}, Van Tuan Nguyen^{*}, Ngoc Lam Dao^{*}, Thanh Tu Pham^{*},

Ngoc Son Cao^{*}, Dinh Hung Chu^{*}, Phi Hung Nguyen^{*}, Cong Dan Pham^{*}, Duc Hai Nguyen^{*},

^{*}OCS Research Center, Viettel High Technology, Viettel Group, Hanoi, Vietnam

^{**}Hanoi University of Science and Technology, Hanoi, Vietnam

(saocd, tamnd17, chinhnx4, thuyettv1)[@viettel.com.vn](mailto:saocd@viettel.com.vn)

(binhnh17, thuanlk, tuannv48, lamdn10, tupt18)[@viettel.com.vn](mailto:binhnh17@viettel.com.vn)

(soncn1, hungcd4, hungnp22, danpc, haind13)[@viettel.com.vn](mailto:soncn1@viettel.com.vn)

Abstract—Cloud Computing has been the development trend of the world because of many benefits such as cost, scalability, performance, security, speed, productivity, and reliability. Then it is applied in many different fields. In particular, in the field of Telco cloud, the movement has developed strongly in recent times of fifth-generation (5G) telecommunication applications technology, meeting the ability to create quick and reliable telecommunications services with the support of automation, calculating elastic through Auto-Scaling mechanism. Now the cloud development in two directions is Virtualized Network Functions (VNFs) and Containerized Network Functions (CNFs), so Auto Scaling must adapt to their presence. So here we offer Auto Scaling architecture with two algorithms suitable for Telco Cloud infrastructure to meet both VNF and CNF with a single module.

Keywords—Auto-scaling, Scalability, Telecom, Telco Cloud, Cloud computing, Cloud-native

I. INTRODUCTION

Cloud Computing [1] is the provision of complete computing services via the Internet. Or rather, the supply of resources is suitable for the user needs completely via the Internet. The services here may include servers, storage, software, etc. There are four main types of cloud computing: private clouds, public clouds, hybrid clouds, and multiclouds. And based on a unique set of business requirements, The Cloud computing is offered in three different service models. These three models of Cloud Computing are known as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS) [2].

In a Cloud Computing environment, virtualized technologies [3] play an important role to provide physical resources such as processors, disk storage, and broadband networks. Virtualization mainly refers to the virtualization of the foundation or abstraction of physical resources for users. In the cloud, these physical resources are considered a group of resources, so these resources can be allocated as required. Computing on the scale of the cloud system allows users to access huge and elastic resources as required. However, the demand for users in resources may vary at different times, and maintaining enough resources to meet the highest resource requirements may be expensive. Conversely, if the user only maintains the minimum

computing resources, the resources are not enough to handle the peak requirements.

Therefore, dynamic resizing is a feature that allows a server to resize the unit (virtual machine with VNF technology or pod in Kubernetes with CNF technology) to fill the new requirement of resources [4]. When a virtual machine or a Pod is under-provisioning or overprovisioning, dynamic resizing can be utilized to overcome these problems. Dynamic resizing cannot overcome load balancing since it only grows resources on a specific virtual machine or a pod. Trieu C. Chieu et al. [5] proposed a useful architecture for dynamic scaling of the web application in a virtualized cloud computing environment. Their architecture includes a load-balancer, virtual cluster orchestrator system, and auto-provisioning system, and works efficiently in virtual machines. However, a user can rent many virtual machines to perform his applications which are not only web applications. Meanwhile, for security, virtual machines for a specific user can be regarded as a group, and a user cannot access the resources from other groups. Then Che-Lun Hung et al. [6] use a dynamic-scaling scenario with a novel design of applications deployed in virtual machines in a virtual cluster. But the architecture just applies to applications running on a virtual machine. The use of containers as a base technology for large-scale systems opens many challenges in the area of resource management at run-time. Based on container technology, E. Casalicchio and V. Perciballi [7] address the problem of selecting the more appropriate performance metrics to activate auto-scaling actions.

To achieve dynamic scaling, we must know about Auto-scaling and some key features of it. Auto-scaling is often referred to in the context of resource provisioning, scalability, and elasticity. These terms are often used interchangeably, but they are slightly different concepts. Understanding the differences between these concepts can help us to identify the unique issues of auto-scaling and focus on the solutions. In [8], the key features are:

- The ability to scale out (i.e., the automatic addition of extra resources during increased demand) and scale in (i.e., the automatic termination of extra unused resources when demand decreases, to minimize cost).

- The capability of setting rules for scaling out and in.
- The facility to automatically detect and replace unhealthy or unreachable instances.

With Telco cloud in [9], which is a software-defined, highly resilient cloud infrastructure that allows telcos to add services more quickly, respond faster to changes in demand, and centrally manage their resources more efficiently. It is one of the key foundational components for transforming a telecommunications company into a digital service provider. The dynamic scaling of the Telco service is indispensable to meet the requirements of Telco Cloud. Denesh Kumar et al. [10] provide a novel approach to capacity planning for telecom core networks in Public Cloud. They establish a packet processing capacity model for scaling up and scaling out VNF performance on a per VNF unit basis. This approach just focuses on Telco service run on virtual machines. But with the development of CNF technology, Telco services have been deployed on it.

In this paper, we will present an elastic auto-scaling architecture for applications deployed in virtual machines and on the pod in a Telco cloud. Moreover, we provide two auto-scaling algorithms used in our Telco cloud. The first, section 2 illustrates the elastic auto-scaling architecture in a Telco cloud computing environment. The rest of the paper is organized as follows. Section 3 is about the Scaling management plugin we use to apply auto-scaling. Then in Section 4 describes our auto-scaling algorithm. Finally, section 5 concludes the paper.

II. ARCHITECTURE DESIGN

In our Telco cloud architecture, we use network functions virtualization (NFV) architecture by European Telecommunications Standards Institute (ETSI) [11], VNFs are built on top of NFV infrastructure (NFVI), including a virtual infrastructure manager (VIM) like OpenStack to allocate resources like compute, storage, and networking efficiently among the VNFs [12]. The framework for managing NFVI and provisioning new VNFs occurs in the management, automation, and network orchestration (MANO) elements defined by NFV. And as an evolution from VNFs, cloud-native network functions (CNFs) are designed and implemented to run inside containers supported by Kubernetes ([13], [14]). This containerization of network architecture components makes it possible to run a variety of services on the same cluster and more easily onboard already decomposed applications, while dynamically directing network traffic to the correct pods. An open, consistent foundation gives us confidence that the services we deliver will run reliably, regardless of location or footprint. The building that foundation on NFV (with VNFs) and especially cloud-native architectures (with CNFs) results in improved flexibility and agility. Automation plays a critical role in keeping the ecosystem operating efficiently and more easily at scale and allows digital service providers to more rapidly change and add services and features to better respond to customer needs and demands.

With this infrastructure, we need to consider two scenarios about Telco service run on VMs (as Broadband Remote Access Server - BRAS, IP Multimedia Subsystem - IMS, firewalls, etc.) or run on Pods Kubernetes (as Online Charging System - OCS, Management and Orchestrator - MANO, Network Function Virtualized Infrastructure, etc.) in the Cloud Computing environment. A Telco service should be available at any time and should provide the fastest response time regardless of the number of users served. Therefore, a Cloud service system should scale the service dynamically; extending and shrinking the number of servers and Telco service components for large requirements and small requirements. Cloud Computing also allows users to access super-computer-level computing power by distributing tasks into large amounts of computing units (virtual machines or Pod in Kubernetes). Users may not know the exact number of computing units that should be utilized to efficiently perform their jobs. Thus, under-provisioning and over-provisioning happen often. A Cloud computing system should scale the computing units according to the workload. A scalable architecture that effectively handles these two scenarios is illustrated in Figure 1. This architecture includes three main components: Load balancer, Management, and orchestrator (MANO) system and Auto-Scaling system with two auto-scaling algorithms. The load balancer can balance the requests between different virtual machines or pods that perform the same application in a virtual cluster. Management and orchestrator system used for management (VMs on Openstack or Pods on Kubernetes) collects resource usages or other attributes of all VMs, and Pods for each virtual cluster that is running on the Cloud. Auto-Scaling system with three modules in it includes Load balancer controller, Address collector, and Scaling management plugin. It can horizontally expand/shrink the number of VMs, the number of pods according to the workload of a virtual cluster over MANO where VMs, Pods belong and the new one can serve services. If the application running on the virtual cluster consumes most of the resources or is over a defined threshold, the autoscaling system can create a new VM or new Pod that performs the same application and load balancer and then balances the requests among VMs or Pods in the same virtual cluster.

III. SCALING MANAGEMENT PLUGIN

Scaling Management Plugin is a support system for auto-scaling applications for the virtual cluster. With the architect in Figure 2, it has three components:

- API is a RESTful API for a third-party component to work with this plugin.
- Rules Management is a rule engine for MANO rules use.
- Provisioning modules with roles, based on rules can detect and replace the instance with status unhealthy or unreachable.

IV. LOAD BALANCING

In our architecture, the load balancer is utilized to balance the Telco application load. Depending on the way we deploy

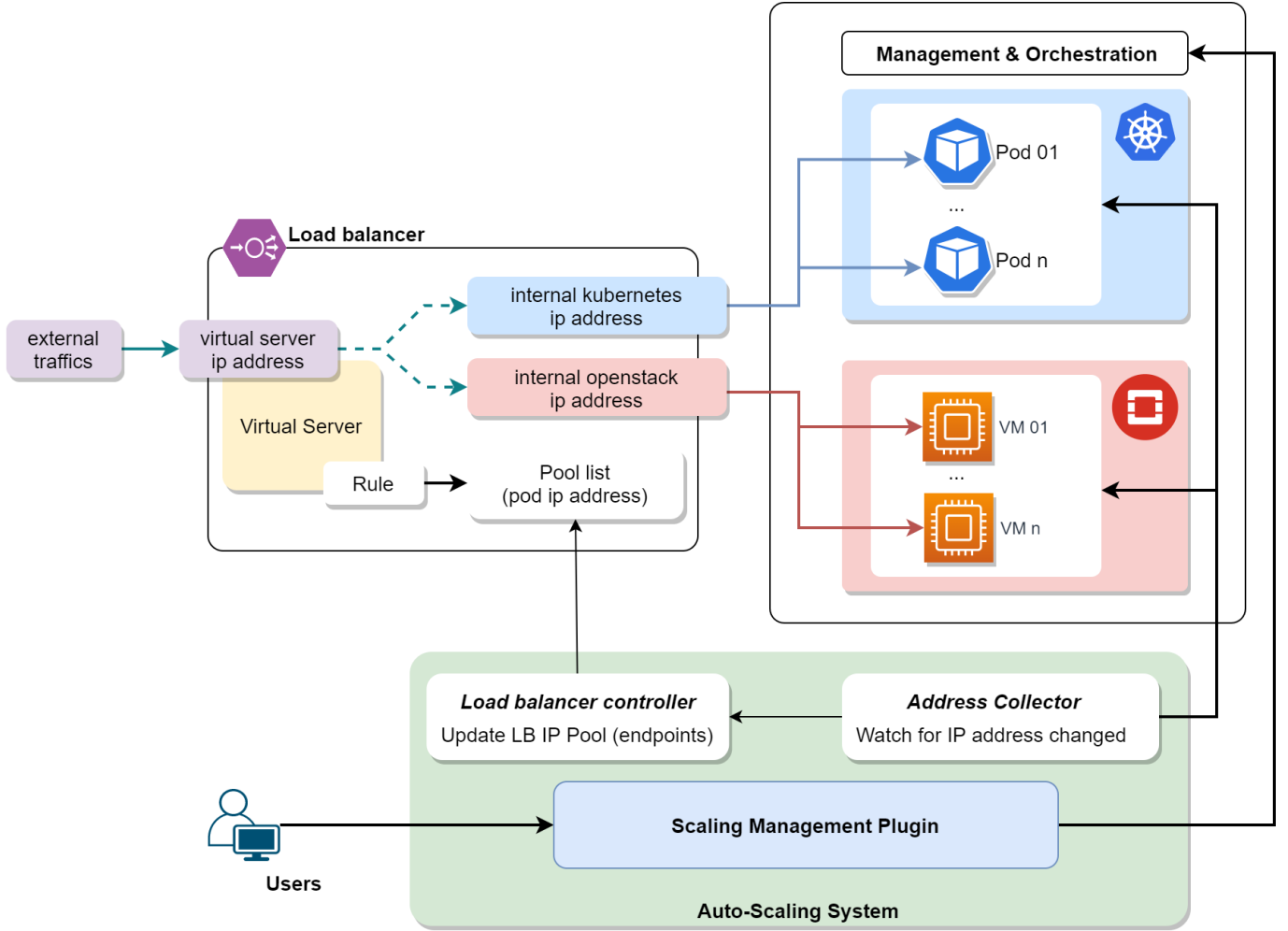


Figure 1. Auto Scaling system architecture

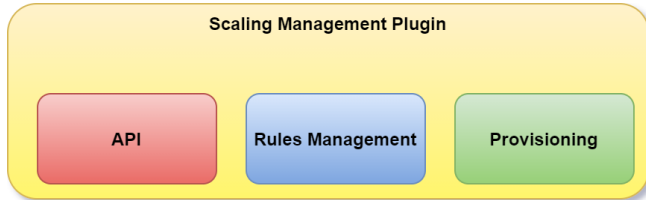


Figure 2. Scaling management plugin components

applications on virtual machines or Pod Kubernetes, we choose the appropriate load balancer.

When deploying an application on a virtual machine, Apache HTTP Load-Balancer is utilized as a load balancer to allow incoming HTTP requests to be routed into servers that perform the Telco application. Since the Apache HTTP Load-Balancer configuration can be updated dynamically, this allows a Cloud system to automatically and dynamically add new servers for a virtual cluster. The additional web servers enable the virtual cluster to scale and thus provide better response time for incoming HTTP requests.

Else deploy an application on Pod Kubernetes, we use Kubernetes Load Balancer used as a load balancer to allow incoming HTTP requests to be routed into Pods that perform the Telco application. With the configuration of Kubernetes Load Balancer can be updated dynamically, and a Cloud system to automatically and dynamically add a new pod for a virtual cluster. The additional pod enables the virtual cluster to scale and thus provide better response time for incoming HTTP requests to the system.

V. AUTO-SCALING ALGORITHM

For Telco service, the Auto-Scaling system can detect whether the rules for user-defined are over the threshold in a virtual cluster. For distributed computing tasks, the Auto-Scaling system can detect whether the number of the virtual machine or the number of Pods is over the threshold of use of physical resources or defined attributes in a virtual cluster. As shown in Figure 1, the auto-scaling algorithm is implemented in the Auto-Scaling system and used to control and trigger the scale-up and scale-down on the number of virtual machine instances or Pod instances based on the statistics of the scaling

indicator. Based on the way of installing applications, on a virtual machine or Pods, we have the ways differently. With Telco applications on VM, the unit is called VNF Components (VNFCs). Another run on pod, the unit is Virtual Deployment Units (VDU).

Installation of a software appliance on a virtual machine creates a virtual appliance. A virtual appliance image is a set of virtual appliances on a virtual machine. A virtual appliance image includes guest OS and applications, and it can eliminate the installation, configuration, and maintenance costs associated with running complex stacks of software. Therefore, to simplify the provisioning process, a virtual machine appliance image template that includes the Telco application and its corresponding server is stored in the image repository of the Cloud system. The new virtual machines of servers can be rapidly created and provisioned to the virtual cluster using the corresponding appliance image template.

Algorithm 1 A threshold autoscaling algorithm

Input: Step: The number of step will be scale (Each step will contain number of the instance (pod) will be scale)

MaxStep: The maximum number of step will be scaled

CurrentStep: The current step was scaled

TargetVDUs: List of the VDUs will be scaled.

ScaleType: SCALE_OUT or SCALE_IN type

Metrics: List of the metrics to monitor.

Output: VDU of TargetVDUs will be scaled.

```

1: if (metric > ThresholdUP & CurrentStep + Step ≤
   MaxStep & ScaleType = SCALE_OUT) then
2:   for each VDU ∈ TargetVDUs do
3:     Create an Alarm for users
4:     Create new instances (VM, Pod) of VDU
5:     CurrentStep = CurrentStep + Step
6:   end for
7: end if
8: if (metric < ThresholdUP & CurrentStep + Step ≥ 0 &
   ScaleType = SCALE_IN) then
9:   for each VDU ∈ TargetVDUs do
10:    Create Alarm for users
11:    Create new instances (VM, Pod) of VDU
12:    CurrentStep = CurrentStep - Step
13:   end for
14: end if

```

With Telco service running on pod Kubernetes, the service has been packed in container images. Then it is stored in the images repository of the Cloud system. POD is the most basic ingredient to deploy and run an application, created and managed by Kubernetes. The new pod is created and provisioned to the virtual cluster based on one or some container in it with a specific Telco application.

With a defined threshold, we illustrate our auto-scaling algorithm on Algorithm 1. When Telco service runs on VM, the algorithm first determines the current VMs with the resource defined in the rules above or below given threshold numbers, respectively. If all VMs have resources defined in

the rules above the given upper threshold, a new VM will be provisioned, started, and then added to the load balancer, respectively. If there are VMs with resources defined in the rules below a given lower threshold and with at least one VM that has no resources defined in the rules, the idle VM will be removed from the load balancer and be terminated from the system. For Telco services running on Pod Kubernetes, with the rule defined (maybe about physical computing resources such as the uses of CPU and memory) will be indices for the evolution of the workload of a virtual cluster. When Telco applications run on Pod Kubernetes, the algorithm first determines the current Pods with the use of physical resources above or below given threshold numbers. If the uses of resources of all Pods are above the given upper threshold, a new Pod will be created, provisioned, started, and then perform the same computing tasks in the virtual cluster. If the uses of resources of some Pods are below a given lower threshold and with at least one Pod that has no computing job, the idle Pod will be terminated from the virtual cluster.

With the requirements of Telco service, the customers will use more resources with specific time based on area. Then the Telco service should scale in, and scale-out applications served for that time. We provide an auto-scaling algorithm that allows schedule scale application at a specific time that, it illustrates on Algorithm 2.

Algorithm 2 A scheduler autoscaling algorithm

Input: Step: The number of steps will be scaled (Each step will contain a number of the instance (pod) that will be scaled)

MaxStep: The maximum number of steps will be scaled

CurrentStep: The current step was scaled

TargetVDUs: List of the VDUs will be scaled

ScaleType: SCALE_OUT or SCALE_IN type

Timers: List of the timers to schedule.

Output: VDU of TargetVDUs will be scaled.

```

1: if (currentTime = schedulingTime & CurrentStep + Step
   ≤ MaxStep & ScaleType = SCALE_OUT) then
2:   for each VDU ∈ TargetVDUs do
3:     Create an Alarm for users
4:     Create new instances (VM, Pod) of VDU
5:     CurrentStep = CurrentStep + Step
6:   end for
7: end if
8: if (currentTime < schedulingTime & CurrentStep - Step
   ≥ 0 & ScaleType = SCALE_IN) then
9:   for each VDU ∈ TargetVDUs do
10:    Create Alarm for users
11:    Create new instances (VM, Pod) of VDU
12:    CurrentStep = CurrentStep - Step
13:   end for
14: end if

```

VI. CONCLUSION

In this paper, we have presented two scaling scenarios to address the automatic scalability of Telco applications on VM and On Kubernetes in a virtual cluster on the virtualized Cloud Computing environment. The proposed Cloud Computing architecture is constructed with a load balancer, a Management and orchestrator system, and an Auto-Scaling system. The load balancer is utilized to route and balance user requests to Cloud services deployed in a virtual cluster. The Management and orchestrator system is used to collect the use of physical resources and define attributes of each virtual machine or each pod in a virtual cluster. The Auto-Scaling system with Scaling management plugin is utilized to dynamically provision the virtual machines, and Pods based on the rule user-defined or the use of the resources in a virtual cluster. In addition, idle virtual machines or pods are destroyed, and then the resources can be released. On the other hand, the energy cost can be reduced by removing the idle virtual machines, and pods. Our architecture has demonstrated the proposed algorithms for Telco cloud are capable of handling sudden load requirements, maintaining higher resource utilization, and reducing energy costs. The Scaling management plugin of the Auto-scaling system is the essential element in enhancing resource utilization, thus reducing the infrastructure and management costs.

ACKNOWLEDGMENT

We want to express our gratitude to Viettel High Technology Industry Group and Hanoi University of Science and Technology for providing us with computing equipment and research resources.

REFERENCES

- [1] E. Thomas, M. Zaigham, and P. Ricardo, "Cloud computing concepts, technology & architecture," 2013.
- [2] M. U. Bokhari, Q. M. Shallah, and Y. K. Tamandani, "Cloud computing service models: A comparative study," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2016, pp. 890–895.
- [3] D.-N. Le, R. Kumar, G. N. Nguyen, and J. M. Chatterjee, *Cloud computing and virtualization*. John Wiley & Sons, 2018.
- [4] R. H. Enterprise, "Kernel based virtual machine," URL <http://www.linux-kvm.org>, 2022.
- [5] T. C. Chieu, A. Mohindra, A. A. Karve, and A. Segal, "Dynamic scaling of web applications in a virtualized cloud computing environment," in *2009 IEEE International Conference on e-Business Engineering*. IEEE, 2009, pp. 281–286.
- [6] C.-L. Hung, Y.-C. Hu, and K.-C. Li, "Auto-scaling model for cloud computing system," *international journal of hybrid information technology*, vol. 5, no. 2, pp. 181–186, 2012.
- [7] E. Casalicchio and V. Perciballi, "Auto-scaling of containers: The impact of relative and absolute metrics," in *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2017, pp. 207–214.
- [8] H. Alipour, Y. Liu, and A. Hamou-Lhadj, "Analyzing auto-scaling issues in cloud environments," in *CASCON*, vol. 14, 2014, pp. 75–89.
- [9] B. Silverman and M. Solberg, *OpenStack for architects: design production-ready private cloud infrastructure*. Packt Publishing Ltd, 2018.
- [10] D. Kumar, S. Chakrabarti, A. S. Rajan, and J. Huang, "Scaling telecom core network functions in public cloud infrastructure," in *2020 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2020, pp. 9–16.

- [11] Q. Duan, N. Ansari, and M. Toy, "Software-defined network virtualization: An architectural framework for integrating sdn and nfv for service provisioning in future networks," *IEEE Network*, vol. 30, no. 5, pp. 10–16, 2016.
- [12] www.redhat.com. Vnf and cnf, what's the difference? [Online]. Available: <https://www.redhat.com/en/topics/cloud-native-apps/vnf-and-cnf-whats-the-difference>
- [13] J. Arundel and J. Domingus, *Cloud Native DevOps with Kubernetes: building, deploying, and scaling modern applications in the Cloud*. O'Reilly Media, 2019.
- [14] B. Ibryam and R. Huß, "Kubernetes patterns," 2019.



Dang Sao Cao received his B.Sc. degree in Mathematics and Informatics Engineering in August 2020 from Hanoi University of Science and Technology, Vietnam. He is currently a cloud engineer in Viettel High Technology, Viettel Group and a M.Sc. student at Hanoi University of Science and Technology in Vietnam. He has many years of experience with cloud computing and cloud native platforms, especially for telecommunications. His research focus on high performance computing, operating system, networking, cloud computing, network security.



Dinh Tam Nguyen received his B.Sc. degree in Electronics and Communication Engineering in 2020 from Hanoi University of Science and Technology, Vietnam. He is currently a cloud engineer in Viettel High Technology, Viettel Group and a M.Sc. student at Hanoi University of Science and Technology in Vietnam. He has many years of experience with cloud computing and cloud native platforms, especially for telecommunications. His research interests include networking, cloud computing, network security, the Future Internet.



Xuan Chinh Nguyen is currently a software deployment engineer at Viettel High Technology - Viettel Group, graduated from the Hanoi University of Industry in 2019. He has many years of experience with software, cloud computing, cloud native platform. His research focus on algorithm, operation system, cloud computing.



Van Thuyet Tran received his B.Sc. degree in Information System Engineering in 2013 from Hanoi University of Science and Technology, Vietnam. He is currently working as a Project Management of Viettel Group. He experienced in developing 5G Core systems.



Hai Binh Nguyen was graduated in Information Technology in December 2013 from Posts and Telecommunications Institute of Technology, Vietnam. Currently, he is head of cloud orchestration platform department at Viettel High Technology, Viettel Group. He is very experienced with many years working with cloud computing in both infrastructure and applications domain. Specially, he focuses on cloud native, high performance systems in Telecommunication Core Network.



Khac Thuan Lang received his B.Sc. degree in Mathematics and Informatics Engineering in 2014 from Hanoi University of Science and Technology, Vietnam. He is currently a cloud engineer in Viettel High Technology, Viettel. He has industry experience in developing online charging system, database system and cloud computing in telecom.



Phi Hung Nguyen received his B.A. degree in computer science in 2018 from Franklin & Marshall College, Lancaster, Pennsylvania, USA. Since 2019, he has been working as a data analyst at Viettel High Technology Industries Corporation (VHT) of Viettel Group. His research interests revolve around distributed machine learning in telecommunications.



Van Tuan Nguyen received his B.Sc. degree in Electronics and Telecommunications Engineering in 2013 from Hanoi University of Science and Technology, Vietnam. He is currently a cloud engineer in Viettel High Technology, Viettel. He has industry experience in developing online charging system, cloud computing in telecom.



Cong Dan Pham received his PhD. Degree in Probability and Statistics from Aix-Marseille University, France in Jun 2014. He has academic experience in universities like Aix-Marseille University, Hanoi University of Education, and industry experience companies like Viettel Group. He has published various journal and conference papers in IEEE and ScienceDirect. His research interests include probability and statistics, data science and machine learning.



Ngoc Lam Dao received his B.Sc. degree in Networks and Communication Engineering in 2021 from VNU - Hanoi University of Engineering and Technology, Vietnam. He is currently a cloud engineer in Viettel High Technology Industries Corporation (VHT) and a M.Sc. student in the VNU - Hanoi University of Engineering and Technology. His research interested include networking, Internet of Things, Telecommunication and cloud computing. He has many years of experience with cloud computing platforms and is starting with in-depth



Duc Hai Nguyen received his B.Sc. in Mathematics and Informatics Engineering from Hanoi University of Science and Technology, Vietnam in 2009. He is currently director of the Research Center for OCS (Online Charging Platform) of Viettel High-Tech Industry Corporation (VHT), Viettel Group. He has many years of experience in research and development of 5G core systems such as online charging system. His research interests include telecommunications core networks, Cloud Computing.

research on cloud computing for telecommunications.



Thanh Tu Pham received his Networking Engineering Degree from Vietnam National University, Hanoi in Dec 2019. He has full-time academic experience in University of Engineering and Technology of Vietnam National University at Hanoi, and industry experience companies like Viettel Group. He was a good student of the school with many research activities. He specializes in research on virtualization infrastructure such as storage, network, compute, etc. and the security aspect of Cloud infrastructure.



Ngoc Son Cao received his B.Sc. degree in Information Security from Post and Telecommunications Institute of Technology, Vietnam. He is currently a cloud engineer in Viettel High Technology, Viettel. He has industry experience in supporting, automating, and optimizing mission critical deployments.



Dinh Hung Chu is currently a software deployment Engineer in Viettel High Technology - Viettel Group, graduating from the Posts and Telecommunications Institute of Technology in 2019. He has many years of experience with cloud computing, cloud native platform. His research focus on algorithm, operation system, cloud computing.