Cloud Resource Scheduling, Monitoring, and Configuration Management in the Software Defined Networking Era

Khaled Alwasel[1], Ayman Noor[1], Yinhao Li[1], Ellis Solaiman[1], Saurabh Kumar Garg[2], Prem Prakash Jayaraman[3], Rajiv Ranjan[1]
[1]School of Computing Science, Newcastle University, United Kingdom
[2]Information and Communication Technology, University of Tasmania
[3]Faculty of Science, Technology and Engineering, Swinburne University of Technology

In recent years, Cloud Computing has emerged as a major technology for delivering on-demand IT (Information Technology) services to consumers across the globe [1]. It provides IT resources based on a pay-as-you model and offers a rich set of services and tools. The Cloud Computing stack consists of three layers – Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a service (SaaS). IaaS offers hardware resources (computation, network, and storage) as virtualized cloud services. PaaS provides software services (appliances) and programming tools, whereas SaaS provides ready to use application stacks.

In order to ensure bespoke application performance (e.g., minimize response time, maximize throughput); each layer of the Cloud stack requires optimum resource Scheduling, Monitoring, and Configuration-management (SMC). Numerous studies have addressed SMC by proposing various frameworks, models, and algorithms. Most of these studies however have assumed *traditional, non-programmable* Cloud DataCentre (CDC) networks. In this article, we discuss how resource scheduling, monitoring, and configuration-management becomes a challenging task when considering Software Defined Networking (SDN), and Network Function Virtualisation (NFV) performance parameters as part of overall cloud application scheduling process.

The recent technology revolution of so-called **Software Defined Networking (SDN)** [2] has given full control and programmability of CDC network. **Network Function Virtualization (NFV)**, on other hand, is a recent networking concept often presented with SDN, delivering network services using software processes hosted on CDC-based virtual machines (VMs), instead of proprietary dedicated hardware.

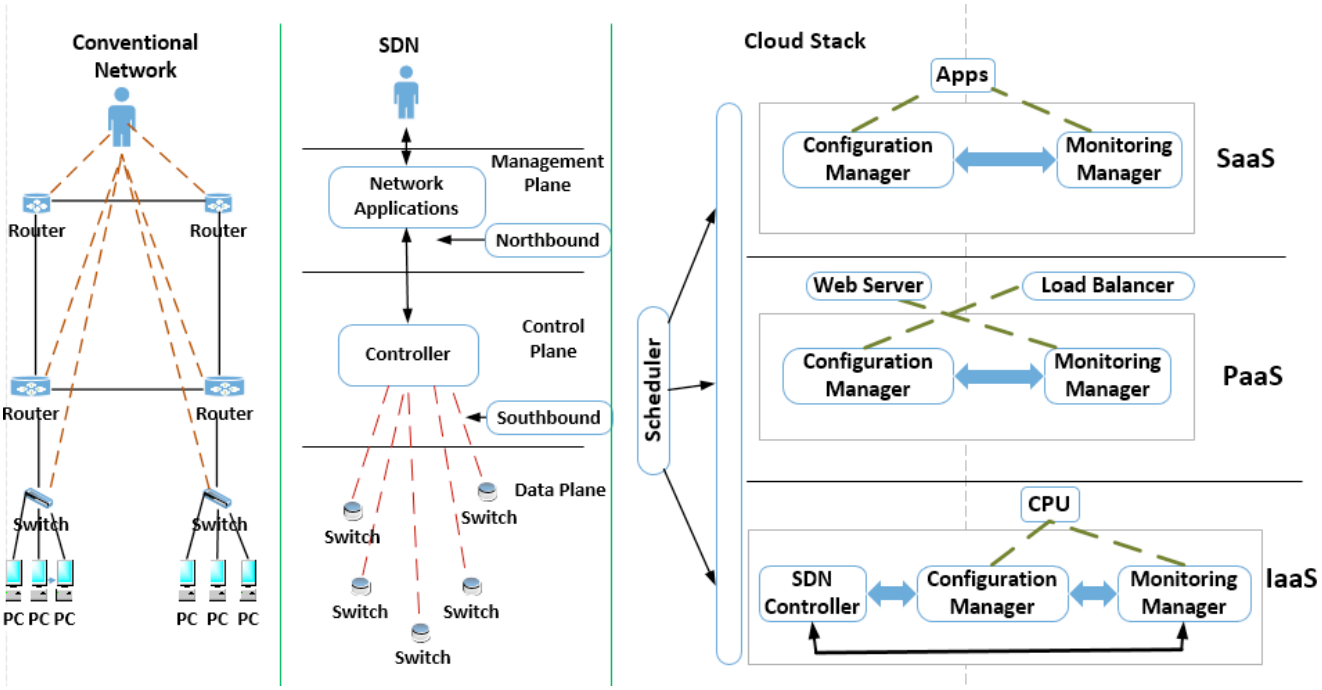

Figure 1: Conventional Network vs SDN network along with Cloud Scheduler, Monitor, Configuring Manager Architecture.

As shown in Figure 1, SDN consists of three layers: The *Management plane*, the *Control plane*, and the *Data-plane*. The Control plane (decision maker) is abstracted away from networking devices (e.g., switches and routers) to a central logic controller, providing a global-network view, programmability, and finer-grained control. It communicates with the data and management planes via two interfaces: southbound and northbound, respectively. The Data plane receives traffic policies from the control plane (e.g., via OpenFlow protocol), and forwards traffic accordingly. The Management plane consists of software services (e.g., via APIs) used by administrators to control and configure network devices.

1 Scheduling Challenges

Scheduling (a well-known NP-complete problem) is an essential mechanism for cloud computing aiming at managing application performance driven diverse goals including minimizing response times, maximizing resource utilization, and maximizing throughput [3, 4, 5]. It plays an important role at each layer of the Cloud stack where operations are mapped appropriately to intended components based on given Quality of Service (QoS) parameters and scenarios. Numerous studies have contributed to cloud scheduling in terms of workflows and VM, but none of them consider SDN based scheduling along with the application and VM (mult-level scheduling). SDN can provide the cloud scheduler with critical factors (e.g., link availability, link utilization, and link failure at run time), resulting in smarter scheduling decisions. However, integrating inputs from the three layers is a non-trivial task making scheduling decisions a difficult task.

The key challenging issues in SDN based scheduling is that there are too many parameters from each of three layers that need to be considered and integrated. The application parameters (first layer) include response time, processing time, rate of failure, throughput, and priorities. The VMs' parameters (second layer) include total number of VMs, total number of idle VMs, total number of VM reservation requests, and cost. For each VM, the scheduler also need to retrieve CPU capacity and utilization, memory capacity and utilization, and storage capacity and utilization. The SDN's parameters (third layer) include topology (re)configuration based on changes in real time, routing mechanisms based on packets/flows in real time, and link capacity, link failure, link reservation, and bandwidth utilization. The scheduler needs to retrieve simultaneously all the parameters from the three layers of the cloud stack at a particular time instance. The second issue is how the design of decision making model that can integrate these parameters each of which might be giving a conflicting picture. Clearly, designing a scheduler while addressing these issues is undoubtedly a daunting task, requiring further investigation of new schemes, models, and algorithms.

Scheduling related research can be categorized into three areas based on above mentioned the problem: application, VM, and SDN scheduling. To the best of our knowledge, no attempt has been made for designing a scheduler that is able to operate according the parameters generated from the proposed architecture. Nevertheless, there are research work addressing the problem considering the application layer or/and VM layer parameters. For example, Sukhpal et al. [3, 4] proposed techniques to provision resources based on QoS requirements (e.g., CPU utilization) resulting in less execution time and cost. Warneke et al. [5] presented a Nephele framework with the principle of job scheduling and task execution where a job manager divides a given job's process into a number of tasks and then assigns VM(s) accordingly. Still, research on designing a multi-level scheduler that is Application-VM-SDN-Agnostic has not been solved yet.

2 Monitoring challenges

Monitoring can be defined as the process of observing or checking the quality of something over a period of time. Similarly, monitoring in Cloud computing is the process of auditing and managing the operational workflow, and the different processes within a cloud-based IT asset or infrastructure. Cloud monitoring is an important part of cloud security and management [14, 15], and it can be implemented within Cloud infrastructure through automated software providing central access and control over the cloud infrastructure. There are many applications of cloud monitoring, such as:

1. Troubleshooting: Monitoring allows analysis of network protocols and behaviour. It also helps in finding network traffic problems.

2. Security: Monitoring mechanisms help with sampling data to look for network-based attacks.

3. Performance: Monitoring helps in optimizing the performance of the CDC resources and network.

In SDN, there can be many problems that can remain undetected in the network, thus requiring process monitoring process. Monitoring in SDN also helps in checking the utilization of CPU, latency and the total request count. Additional activities can also be checked such as memory usage and data volume. Due to these features, monitoring plays an important in SDN data centres [12]. Network monitoring and visibility has become increasingly challenging since IT infrastructure must support network, server, and storage virtualization, as well as user access to cloud-based applications. Integration of service context and dynamic or real time change are among the hard challenges of the monitoring process.

While end-to-end monitoring is important for all type of Cloud applications (multi-tier web, content delivery network, etc.), it is much more critical in context of big data analytics application where network performance determines (e.g. network throughput and latency) performance of virtual machine hosted analytics software component (e.g., mappers and reducers in context of Apache Hadoop) In big data analytics applications, SDN can be leveraged to program switches in order to provide optimal data flow paths between data analytics software components on the fly, during each stage of data analysis. SDN enables better QoS between the virtual machines by dedicating more cross-links between them depending on the type of the analysis process [7]. However, this approach brings several challenges, such as: 1) enabling all vendor technologies to implement both SDN controller integration with virtual machine schedulers. and 2) how to instrument SDN devices with monitoring agent. The implementation of Network Function Virtualization (NFV) can decrease the amount of hardware required to launch and operate network services. However, NFV also brings some challenges of its own, such as: 1) performance degradation, 2) the elasticity of service provisioning may require the consolidation and migration of VNFs based on traffic load and user demand. 3) Implementation of NFV brings a new set of security concerns, as virtual applications running within data centres might not owned by network operators.

Monitoring a network is a top concern within IT departments. This is especially the case as monitoring efforts are ubiquitously leveraged to meet network security and performance goals. Monitoring can be effected by the unavailability of useful tools and alerting capabilities. In different networks, it is necessary to capture, store and analyse the vast amount of monitoring data. In short, we can say that monitoring is quite difficult and challenging due to different problems like Integration of service context and Dynamic or real time change.

An ideal scenario would be one where the capabilities of SDN and NFV are combined to provide a holistic monitoring solution. SDN has capacity of building and managing large IP/Ethernet networks by separating the network's control, whereas NFV (with the purpose of reducing deployment costs) has capability of virtualizing network functions and migrate them to generic servers [13].

There are several commercial monitoring tools available for big data monitoring, they are: Monitis, RevealCloud, LogicMonitor, Nimsoft, Nagios, SPAE by SHALB, CloudWatch, OpenNebula, CloudHarmony, Windows Azure FC etc. These tools can provide a report in the form of a graph or chart about the load on servers and other information. Each of these tools have their own advantages and drawbacks. For example "RevealCloud" can only provide information for the last 30 days. These tools use different communication protocols for performing tasks, which may or may not work perfectly with SDN and/or NFV communication protocols.

3 Configuration Management Challenges

Configuration management is the detailed recording and updating of information that describes an enterprise's hardware and software. In a Cloud computing context, configuration management supports the management of services by providing information about how the services are being assembled or bundled together. This information is crucial to the other service management processes, especially for change management, incident management, or problem management. It is also crucial to ensure meeting all agreed-to service levels. Many tools are made for automating the

cloud applications configuration management. Among these tools, CFEngine, Puppet, Chef, Ansible are well known. Different cloud-based services (e.g., IaaS, PaaS, or SaaS), will require different levels of configuration management. In the IaaS layer, the consumer of IaaS services usually has control over the configuration aspects of the resources, such as which operating system to run on a virtual machine, or how to utilize the storage resources, or how to assign IP address to a provisioned virtual machine. In the PaaS layer, configuration management could be performed on the individual components of the platform, such as the automated integration between database and application server layer in context of multi-tier web applications. In the SaaS layer, configuration management operations include configuration end user's account and access credentials with the CDC-hosted SaaS application.

To allow information flow between the management plane and other planes (Figure 1), configuration management interfaces need to be in place. These interfaces allow settings to be enforced across network devices, and information to be sent back into the management plane, for example, for reporting to a network administrator [9]. In traditional networks, the logic of the data and control planes is confined inside each network device according to a well-defined set of standardized protocols. With the separation of planes, as SDN promotes, the need to bootstrap the communication between the data and control planes becomes a basic requirement. Configuring this communication can be particularly complex, considering that both planes can operate under protocols defined by software. Moreover, changes in any plane may affect such communication directly. Ideally, a new management interface is required than can manage configuration properly in SDN.

Given the need to deploy and setup novel data forwarding devices into a traditional network, administrators must configure these devices in order to have them operational. In the most basic and common scenario, the administrator would interact with a command line interface on the device and have it configured by performing many intricate proprietary commands. In the case of SDN data forwarding devices, there are still a few parameters to be configured, such as the communication policy with the control plane (e.g., drop every packet when the control plane is unavailable or follow the last valid set of rules). In response to these necessities, the Open Networking Foundation (ONF) proposed the OpenFlow Management and Configuration Protocol (OF-Config). This protocol allows operational staff to assign controllers to switches, set ports up/down, configure queues, assign certificates for the communication with the control plane, set up tunnels, handle versioning, and retrieve device capabilities. OF-Config is already a significant step toward tackling dataplane-related management requirements. On the other hand, this protocol is targeted specifically to OpenFlow networks; therefore, it is tied to the limited view of SDN employed by this technology (e.g., fixed dataplane and logically centralized control plane).

The challenge of configuration management in SDN is to automate the management of configurations in each plane (application plane, control plane, data plane) by an administrator via management interfaces. SDN and its most known realization, OpenFlow, has enabled widespread and vendor-neutral programmability of the control plane of the network environment. However, despite such proposals from research and standardization bodies, the management plane still lacks a comparable interface and protocol for capability discovery, device management and monitoring [10]. Therefore, network management is still heavily human-centred with minor autonomous behaviour. The Organization for the Advancement of Structured Information Standards (OASIS) is working on a Topology and Orchestration Specification for Cloud Applications (TOSCA), to enable the creation of portable cloud applications and the automation of their deployment and management. This standard provides a concept named management plan, which makes the management of complex enterprise applications automated, repeatable, traceable, and less error prone. However, the abstraction focuses on higher-level network services such as database management systems (DBMS) and their relationship to other entities, not on details of the individual forwarding elements.

In terms of Network Function Virtualisation (NFV), the agile and automated management of virtualized network functions (VNFs) throughout their lifecycles becomes a foremost objective [11]. TOSCA operationalizes the deployment of VNFs and triggers their initial configuration. A TOSCA template is a file-based description of VNF-related things and a set of execution guidelines for deploying and operating them as a single entity on cloud infrastructure. Each TOSCA-defined node has interfaces through which it can be manipulated by an initial configuration application or script, which is referenced in the TOSCA template. However, each VNF instance needs to be further configured at runtime to fulfil the specific needs of a consumer and service. Runtime configuration is beyond the scope of a TOSCA template.

References

[1] Q. Zhang, L. Cheng, and R. Boutaba. "Cloud computing: state-of-the-art and research challenges." *Journal of internet services and applications* 1.1 (2010): 7–18.

[2] D. Kreutz, F. Ramos, P. Verissimo, C. Rothenberg, S. Azodolmolky, S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *IEEE*, pp. 14–76, January 2015.

[3] S. Sukhpal, C. Indervee. "Q-aware: Quality of service based cloud resource provisioning," *Computers & Electrical Engineering*, pp. 138–160, October 2015.

[4] S. Sukhpal, C. Indervee, "QRSF: QoS-aware resource scheduling framework in cloud computing," *The Journal of Supercomputing*, pp. 241–92, September 2014.

[5] D. Warneke, O. Kao, "Nephele: efficient parallel data processing in the cloud," *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, 2009.

[6] L. Cui, F. R. Yu and Q. Yan, "When big data meets software-defined networking: SDN for big data and big data for SDN," *IEEE Network*, vol. 30, no. 1, pp. 58–65, January–February 2016.

[7] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia and W. Kellerer, "Interfaces, attributes, and use cases: A compass for SDN," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 210–217, June 2014.

[8] D. Raumer, L. Schwaighofer and G. Carle, "MonSamp: A distributed SDN application for QoS monitoring," *Federated Conference on Computer Science and Information Systems*, Warsaw, pp. 961–968, 2014.

[9] Wickboldt, Juliano Araujo, et al. "Software-defined networking: management requirements and challenges." *IEEE Communications Magazine*, 53.1 (2015): 278-285.

[10] Sieber, Christian, et al. "Towards a programmable management plane for SDN and legacy networks." *NetSoft Conference and Workshops (NetSoft)*, 2016.

[11] Mijumbi, Rashid, et al. "Network function virtualization: State-of-the-art and research challenges." *IEEE Communications Surveys & Tutorials* 18.1 (2016): 236–262.

[12] S. R. Chowdhury, M. F. Bari, R. Ahmed and R. Boutaba, "PayLess: A low cost network monitoring framework for Software Defined Networks," *IEEE Network Operations and Management Symposium (NOMS)*, Krakow, pp. 1–9, 2014.

[13] G. Gardikis et al., "An integrating framework for efficient NFV monitoring," *IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, pp. 1–5, 2015.

[14] E. Solaiman, R. Ranjan, P. P. Jayaraman, K. Mitra. "Monitoring Internet of Things Application Ecosystems for Failure", *IT Professional*, IEEE, 2016.

[15] E. Solaiman, I. Sfyrakis, C. Molina-Jimenez. "A State Aware Model and Architecture for the Monitoring and Enforcement of Electronic Contracts". *18th Conference on Business Informatics (CBI)*, IEEE, 2016.