An Analysis of Container-based Platforms for NFV

draft-natarajan-nfvr-g-containers-for-nfv-01

Abstract

With the technology advancements in the field of containers, they are considered a potential alternative to virtual machine based implementations. In the area of cloud applications, there are comprehensive studies and early implementations of container based platforms. This draft describes some of the challenges of using virtual machines for NFV workloads and how containers can potentially address these challenges.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.
The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire in March 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Table of Contents

1. Introduction...................................................3
2. Challenges in Virtual Machine Implementations................3
   2.1. Performance (SLA)........................................3
       2.1.1. Challenges........................................3
   2.2. Continuity/ Elasticity/ Portability....................4
       2.2.1. Challenges........................................4
   2.3. Security................................................5
       2.3.1. Challenges........................................5
   2.4. Management.............................................6
       2.4.1. Challenges........................................7
3. Benefits of Containers........................................7
4. Challenges with Containers and potential solutions..........8
5. Conclusion...................................................9
6. Future Work..................................................9
7. IANA Considerations..........................................9
8. Security Considerations......................................9
9. Contributors................................................10
10. Acknowledgements...........................................10
11. References................................................10
   11.1. Normative References................................10
   11.2. Informative References...............................10
Authors’ Addresses.............................................11
1. Introduction

This draft describes some of the challenges of using virtual machines for NFV workloads and how container-based platforms can potentially address these challenges. It also suggests future work in the area of containers.

2. Challenges in Virtual Machine Implementations

In this section, we provide our assessment of using virtual machines to host VNFs. We enlist the advantages and limitations of VMs and then discuss some open issues that can potentially be addressed by containers.

2.1. Performance (SLA)

Performance requirements vary with each VNF type and configuration. The platform should support the specification, realization and runtime adaptation of different performance metrics. Achievable performance can vary depending on several factors such as the workload type, the size of the workload, the set of virtual machines sharing the underlying infrastructure, etc. Here we highlight some of the challenges based on potential deployment considerations.

2.1.1. Challenges

- VNF provisioning time (including up/down/update) constitutes the time it takes to spin-up the VNF process, its application-specific dependencies, and additional system dependencies. The resource choices such as the hypervisor type, the guest and host OS flavor and the need for hardware and software accelerators, etc., constitute a significant portion of this processing time (instantiation or down time) when compared to just bringing up the actual VNF process. As a result, the provisioning latency is heavily dependent on the optimal choice of infrastructure resources.

- The runtime performance (achievable throughput, line rate speed, maximum concurrent sessions that can be maintained, number of new sessions that can be added per second) for each VNF is directly dependent on the amount of resources (e.g., virtual CPUs, RAM) allocated to individual VMs. Choosing the right resource setting is a tricky task. If VM resources are over-provisioned, we end up under-utilizing the physical resources. On the contrary if we under-provision the VM resources, then upgrading the resource to an advanced system setting might require scaling out or scaling up of the resources and re-directing traffic to the new VM; scaling
up/down operations consume time and add to the latency. This overhead stems from the need to account resources of components other than the actual VNF process (e.g., guest OS requirements).

If each network function is hosted in individual VMs, then an efficient inter-VM networking solution is required for performance.

Deploying VNF’s inside a virtual machine can impose several challenges in meeting Service Level Agreements (SLA). As an example, SLAs demand dynamic fine-tuning (e.g., changing base memory, allocating additional vCPUs) and instantiation of additive features (e.g., integration with hardware and software accelerators) during runtime. In most cases, achieving this task with VMs require snapshotting the current VM state, halting the VM, upgrading the VM with improved features, and re-spinning the VM, all of which have performance implications.

2.2. Continuity/ Elasticity/ Portability

VNF service continuity can be interrupted due to several factors: undesired state of the VNF (e.g. VNF upgrade progress), underlying hardware failure, and unavailability of virtualized resources, VNF SW failure, etc. Some of the requirements that need consideration are:

2.2.1. Challenges:

- VM-based VNF’s are not completely decoupled from the underlying infrastructure. As discussed in the previous section, most VNFs have a dependency on the guest OS, hypervisor type, accelerator used, and the host OS. Therefore porting VNFs to a new platform might require identifying equivalent resources (e.g., hypervisor support, new hardware model, understanding resource capabilities) and repeating the provisioning steps to bring back the VNF to a working state.

- Service continuity requirements can be classified as follows: seamless (with zero impact) or non-seamless continuity (accepts measurable impacts to offered services). Achieving seamless service continuity is harder when VNFs are hosted in VMs, since this requires an efficient high availability solution or a quick restoration mechanism that can bring back the VNF to an operational state. (Note that the need for an efficient high availability solution or quick restoration mechanism is not unique...
to VM based implementations.) For example, an anomaly caused by a hardware failure can impact all VNFs hosted on that infrastructure resource. To restore the VNF to a working state, the user should first provision the VM (process + guest OS + hypervisor info), spin-up and configure the VNF process inside the VM, setup the interconnects to forward network traffic, manage the VNF-related state, and update any dependent runtime agents.

- Addressing the service elasticity challenges require holistic view of the underlying resources. The challenges for presenting a holistic view include the following:
  - Performing Scalable Monitoring: Scalable continuous monitoring of the individual resource’s current state is needed to spin-up additional resources (auto-scale or auto-heal) when the system encounters performance degradation or spin-down idle resources to optimize resource usage.
  - Handling CPU-intensive vs I/O-intensive VNFs: For CPU-intensive VNFs the degradation can primarily depend on the VNF processing functionality. On the other hand, for I/O intense workloads, the overhead is significantly impacted by the hypervisor features, its type, the number of VMs it manages, the modules loaded in the guest OS etc.

2.3. Security

Broadly speaking, security can be classified into:

- Security features provided by the VNFs to manage the state, and
- Security of the VNFs and its resources.

Some considerations on the security of the VNF infrastructure are listed here.

2.3.1. Challenges

- The adoption of virtualization techniques (e.g., para-virtualization, OS-level) for hosting network functions and the deployment need to support multi-tenancy requires secure slicing of the infrastructure resources. In this regard, it is critical to provide a solution that can ensure the following:
  - Provision the network functions by guaranteeing complete isolation across resource entities (hardware units, hypervisor, virtual networks, etc.). This includes secure
access between VM and host interface, VM-VM communication, etc. For maximizing overall resource utilization and improving service agility/elasticity, sharing of resources across network functions must be possible.

- When a resource component is compromised, quarantine the compromised entity but ensure service continuity for other resources.
- Securely recover from runtime vulnerabilities or attacks and restore the network functions to an operational state. Achieving this with minimal or no downtime is important.

Realizing the above requirements is a complex task in any type of virtualization options (virtual machines, containers, etc.)

- Resource starvation / Availability: Applications hosted in VMs can starve the underlying physical resources such that co-hosted entities become unavailable. (Note that the resource starvation challenge is not unique to VM based implementations.) Ideally, countermeasures are required to monitor the usage patterns of individual VMs and ensure fair use of individual VM resources.

### 2.4. Management

The management and operational aspects are primarily focused on the VNF lifecycle management and its related functionalities. In addition, the solution is required to handle the management of failures, resource usage, state processing, smooth rollouts, and security as discussed in the previous sections. Some features of VM-based management solution include:

- **Centralized control and visibility**: Support for web client, multi-hypervisor management, single sign-on, inventory search, alerts & notifications.

- **Proactive Management**: Creating host profiles, resource management of VMs, dynamic resource allocation, auto-restart in HA model, audit trails, patch management.

- **Extensible platform**: Define roles, permissions and licenses across resources and use of APIs to integrate with other solutions.

Thus, the key requirements for a management solution...
o Simple to operate and deploy VNFs.

o Uses well-defined standard interfaces to integrate seamlessly with different vendor implementations.

o Creates functional automation to handle VNF lifecycle requirements.

o Provide APIs that abstracts the complex low-level information from external components.

o Is secure.

2.4.1. Challenges

The key challenge is addressing the aforementioned requirements for a management solution while dealing with the multi-dimensional complexity introduced by the hypervisor, guest OS, VNF functionality, and the state of network.

3. Benefits of Containers

Containers (when compared to VMs) can provide better service agility as it allows us to run the VNF process directly in the host environment. This eliminates the provisioning and processing delay associated with spinning up (or down/update) guest OS, kernel driver association, and hypervisor processing time. This facilitates meeting the SLA requirements of different VNFs. The placement problem for finding a container that is running on hardware of a certain type, e.g. hardware with certain offloads, remains to be addressed.

Containers share the host OS and only require resource allocation for the individual VNF process which usually results in better runtime performance when compared to VMs.

With containers, the inter-VNF communication latency depends on the inter-process communication option (when hosted in the same host) such as bridge mode, sharing the host’s network stack, sharing network namespace between containers, etc. or the networking solution (e.g., network overlays, virtualization, etc.) used between clusters of nodes (when VNFs are hosted across multiple nodes). This eliminates the overhead introduced by the guest OS’s network stack, as long as the containerization technology provides sufficient isolation between containers.
Auto-scaling VNFs or achieving service elasticity in runtime can be simplified by the use of container based VNFs due to the lightweight resource usage of containers. Using containers can simplify the allocation of additional resources to existing containers or quickly spinning up alternate containers, as it only requires booting the VNF process and handling the state transition associated with it. This can significantly reduce the downtime or upgrade time.

Some container management solutions (e.g., Kubernetes [KUBERNETES-SELF-HEALING]) provide self-healing features such as auto-placement, restart, and replacement by using a service discovery mechanism and continuously monitoring the health of individual or group of containers. When a container process encounters a failure, the platform auto detects the issue and seamlessly recovers from failures. This can address some of the service continuity requirements needed in VNF deployments.

4. Challenges with Containers and potential solutions

Resource Management/Isolation/Security: Containers create a slice of the underlying host using techniques like namespaces, cgroups, chroot etc. However, there are several other kernel features that are not completely isolated from the processes running inside containers. This can allow a vulnerable container to compromise the host or containers belonging to other users (e.g., resource starvation).

Potential Solution: Guaranteeing complete isolation across entities requires an efficient access control mechanism and resource quota mechanism. Usage of kernel security modules like SELinux [SELINUX], AppArmor [APPARMOR] along with containers can provide the required features for a secure VNF deployment. Usage of resource quota techniques such as those in Kubernetes [KUBERNETES-RESOURCE-QUOTA] can provide the typical resource guarantees for a VNF deployment. Additionally, a hybrid deployment with VMs and containers can be envisioned depending on the degree of isolation needed between VNFs.

Cross-VNF compatibility and Operating System dependency: As of today, containers are supported in selective operating systems such as Linux, Windows and Solaris. On the other hand, in the current range of VNFs, many don’t support Linux OS or other OSes such as Windows and Solaris. Depending on the nature of the software associated with VNFs, and the libraries installed inside
a container, and the underlying OS version that a container utilizes, some VNFs may not be compatible with other VNFs.

Potential Solution: A hybrid deployment with VMs and containers can be envisioned to address this problem. The VNFs which don’t run on container supported OSes can be run in VMs. Additionally, one could envision each set of compatible VNFs running within a specific VM, with different sets of VNFs running on different VMs, where the VMs run on a hypervisor. A notable additional challenge in this solution is state transfer between containers and virtual machines, including but not limited to latency, interoperability, etc.

. Overall Performance: Unlike VMs, containers can run directly on the host OS and thus exhibit significant performance benefits. As an example, the whitepaper [VCPE-CONTAINER-PERF] demonstrates ~25% throughput improvement for TCP traffic for a Virtual Enterprise Customer Premises Equipment (vE-CPE) use case as described in [ETSI-NFV-USE-CASES]; the environments which were compared were containers using LXC and VM using KVM.

5. Conclusion

The use of containers for VNFs appears to have significant advantages compared to using VMs and hypervisors especially for efficiency and performance. With this background, the authors urge the industry to address the future work areas, especially solutions for the challenges, as described in Section 4 and consider container-based VNFs in real deployments beyond proof-of-concepts.

6. Future Work

Opportunistic areas for future work include but not limited to developing solutions to address the challenges in VNF containerization described in Section 3, distributed micro-service network functions, etc.

7. IANA Considerations

This draft does not have any IANA considerations.

8. Security Considerations

VM-based VNFs can offer a greater degree of isolation and security due to technology maturity as well as hardware support. Since container-based VNFs provide abstraction at the OS level, it can introduce potential vulnerabilities in the system when deployed
without proper OS-level security features. This is one of the key implementation/deployment challenges that needs to be further investigated.

In addition, as containerization technologies evolve to leverage the virtualization capabilities provided by hardware, they can provide isolation and security assurances similar to VMs.

9. Contributors

10. Acknowledgements

The authors would like to thank Vineed Konkoth for the Virtual Customer CPE Container Performance white paper.

11. References

11.1. Normative References

11.2. Informative References


[SELINUX] "Security Enhanced Linux (SELinux) project,"
http://selinuxproject.org/

[APPARMOR] "Mandatory Access Control Framework,"
https://wiki.debian.org/AppArmor

[VCPE-CONTAINER-PERF] "Virtual Customer CPE Container Performance
Labs-CaseStudy2015.pdf

Authors’ Addresses

Sriram Natarajan
Deutsche Telekom Inc.
sriram.natarajan@telekom.com

Ram (Ramki) Krishnan
Dell
ramki_krishnan@dell.com

Anoop Ghanwani
Dell
anoop@alumni.duke.edu

Dilip Krishnaswamy
IBM Research
dilikris@in.ibm.com

Peter Willis
BT
peter.j.willis@bt.com

Ashay Chaudhary
Verizon
ashay.chaudhary@verizon.com