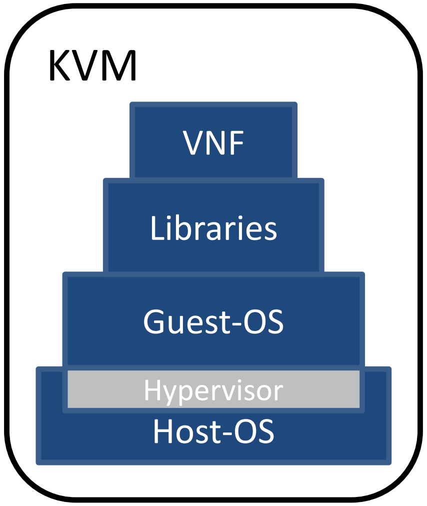
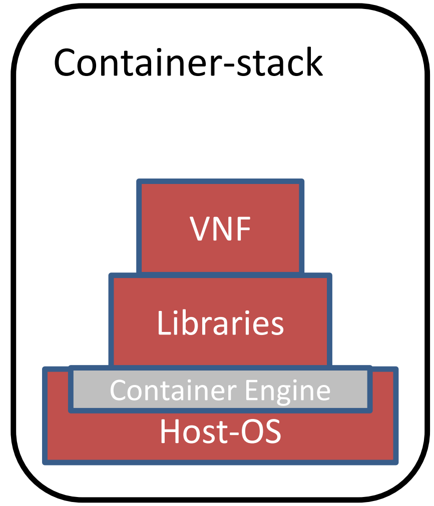
An Analysis of Container-based Platforms for NFV

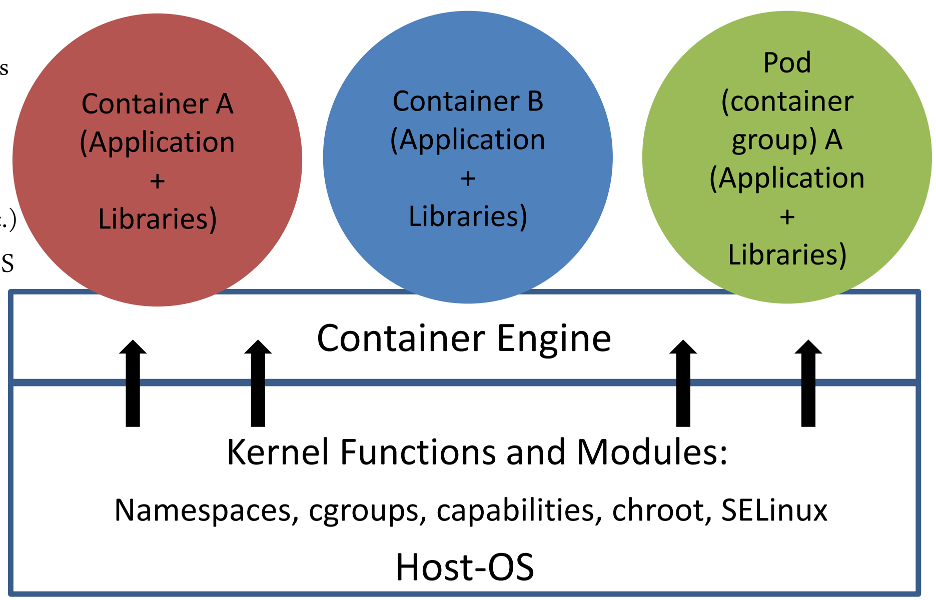
Sriram Natarajan, Deutsche Telekom Inc.
Ramki Krishnan, Dell Inc.
Anoop Ghanwani, Dell Inc.
Dilip Krishnaswamy, IBM Research
Peter Willis, BT Plc
Ashay Chaudhary, Verizon

Virtual Machine vs. Container Stack





- **Lightweight footprint**: Very small images with API-based control to automate the management of services
- **Resource Overhead**: Lower use of system resources (CPU, memory, etc.) by eliminating hypervisor & guest OS overhead



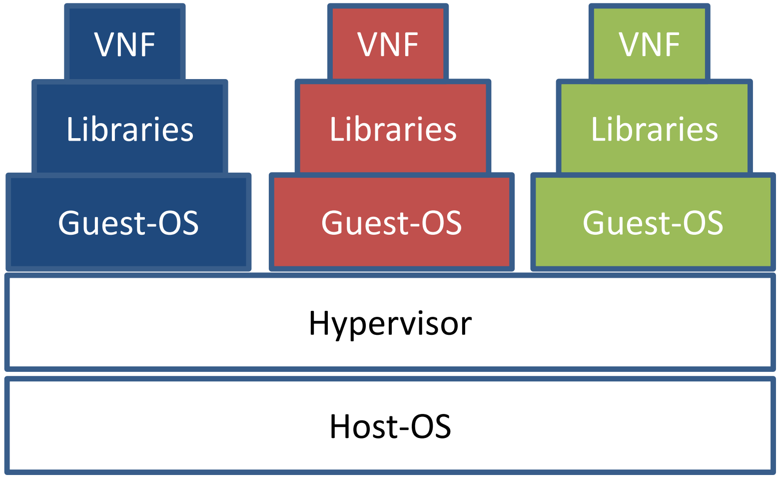
- **Deployment time**: Rapidly deploy applications with minimal run-time requirements
- **Updates**: Depending on requirements, updates, failures or scaling apps can be achieved by scaling containers up/down

VM based Network Functions

Key Challenges

Service Agility/Performance

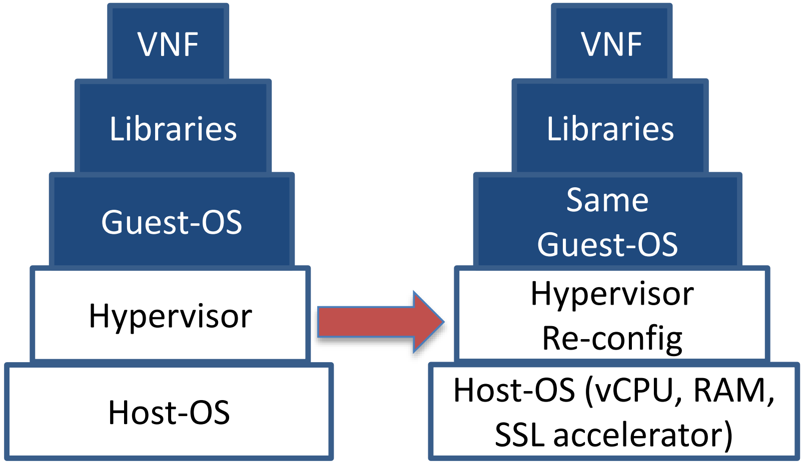
- Provisioning time:
  - Hypervisor configuration
  - Spin-up guest OS
  - Align dependencies between Guest-OS & VNFs



- Runtime performance overhead:
  - Performance proportional to resource allocated to individual VMs (throughput, line rate, concurrent sessions, etc.)
  - Overhead stems from components other than VNF process (e.g. guest OS)
  - Need for inter-VM networking solution
  - Meeting SLAs requires dynamic fine tuning or instantiation of additive features, which is complex in a VM environment

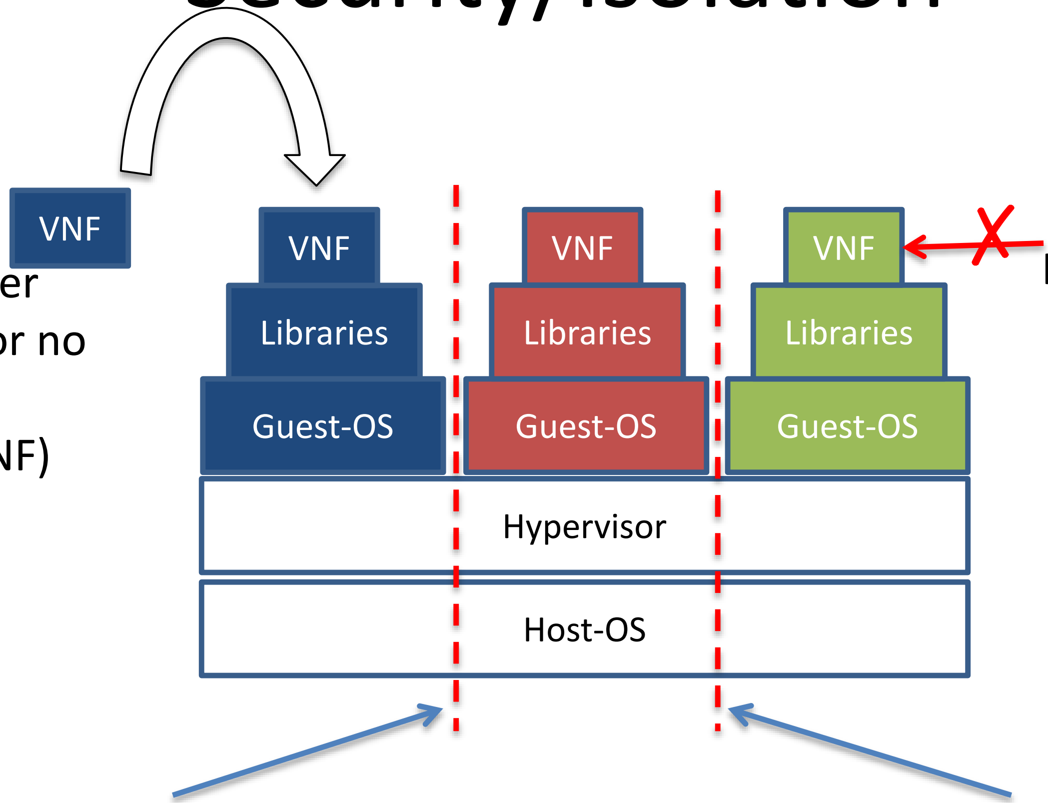
Portability/ Elasticity/Scalability

- Porting VNFs require:
  - Identifying suitable nodes for new VNF instances (or re-locating existing instances). For example, resource types, available capacity, guest OS images, hypervisor configs, HW/SW accelerators, etc.)
  - Allocating required resources for new instances
  - Provisioning configs to components in the guest OS, libraries and VNF



- Elastic scalability needs are driven by workloads on the VNF instances, and stateful VNFs increase the latency to spin up new instances to fully working state.

Security/Isolation



Securely recover with minimal or no downtime (reschedule VNF)

If VNF is compromised (misconfiguration, etc.), how to securely quarantine the VNF, but ensure continuity of other VNFs?

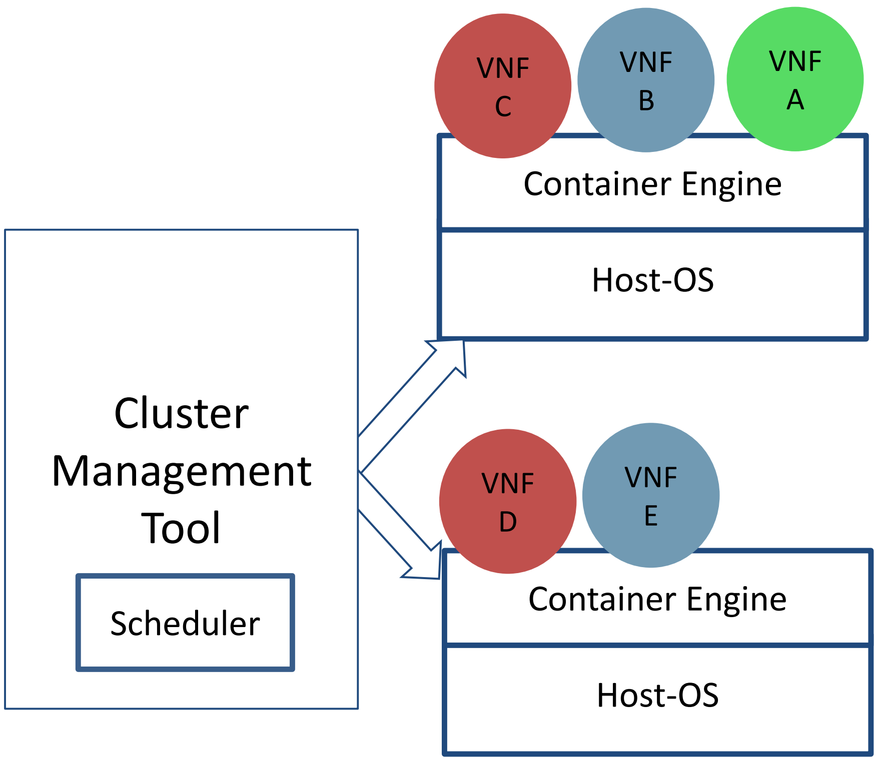
Guarantee complete isolation across resource entities (hardware units, hypervisor, protection of shared resource, isolation of virtual networks, L3 cache, QPI, etc.)

Resource hungry VNF can starve the shared resources (noisy neighbor effect) that are allocated to other VNFs; Need to monitor and cut-off hungry VNF usage

Containerized Network Functions

Key Benefits, Challenges and Potential Solutions

Service Agility/Performance/Isolation (1)

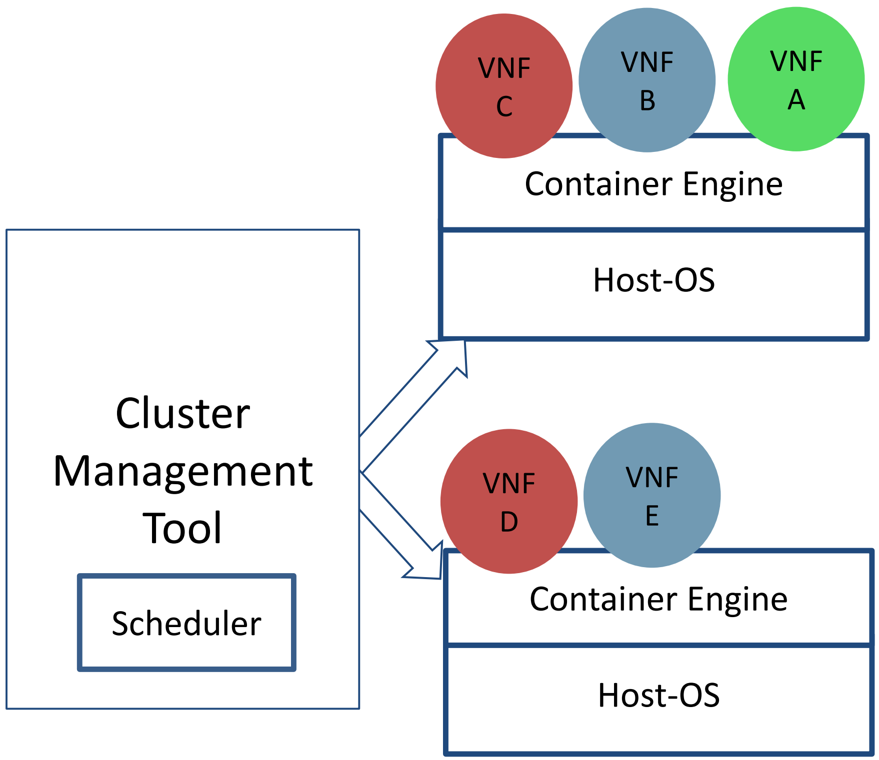


Key Benefits:

- Containers can provide better service agility (e.g. dynamically provision VNFs for offering on-demand services), and performance as it allows us to run the VNF process directly in the host environment

- Inter-VNF communication latency depends on inter-process communication option (when hosted in the same host)

Service Agility/Performance/Isolation (2)



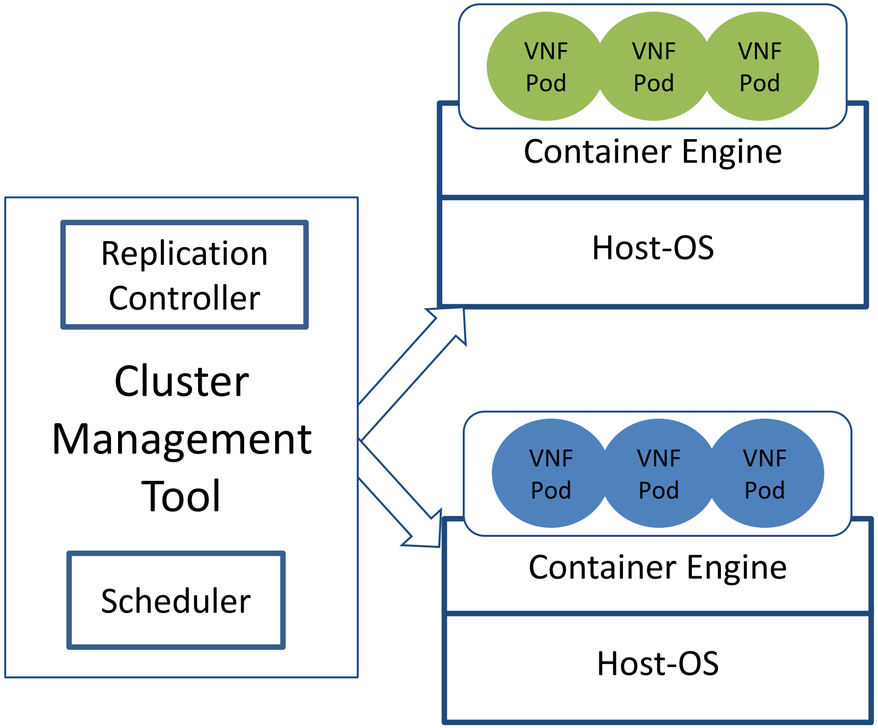
Key Challenges:

- Isolation: Containers create a slice of the underlying host using techniques like namespaces, cgroups, chroot etc.; several other kernel features that are not completely isolated.
- Resource Mgmt: Containers do not provide a mechanism to quota manage the resources and hence susceptible to the "noisy neighbor" challenge.

Potential Solutions:

- Kernel Security Modules: SElinux, AppArmor
- Resource Mgmt: Kubernetes
- Platform Awareness: ClearLinux

Elasticity & Resilience

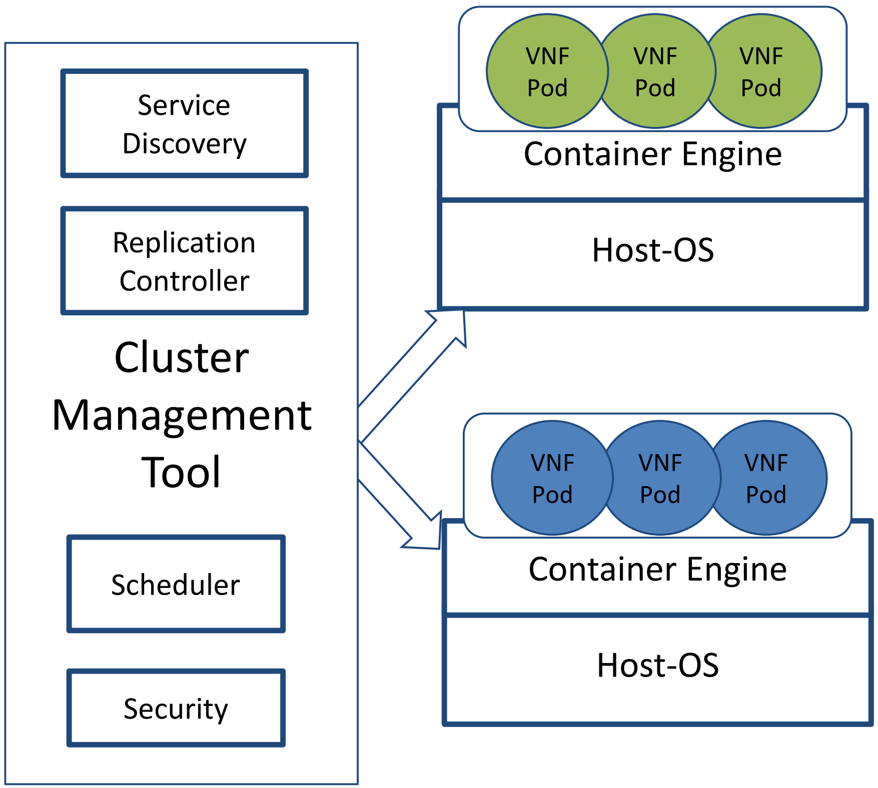


Key Benefits:

- 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 (e.g. Mesosphere/Kubernetes)

- Container management solutions (e.g. Kubernetes) provide self-healing features such as auto-placement, restart, and replacement by using service discovery and continuous monitoring

Operations & Management



Key Challenges:

- Containers are supported in selective operating systems such as Linux, Windows and Solaris
- In the current range of VNFs, many don't support Linux OS or other OSes such as Windows and Solaris

Potential Solutions:

- Hybrid deployment with VMs and containers can be envisioned, e.g. leverage ideas from Aptible technology currently used for applications

Conclusion and Future Work

Conclusion and Future Work

- Use of containers for VNFs appears to have significant advantages compared to using VMs and hypervisors, especially for efficiency and performance
  - "Virtual Customer CPE Container Performance White Paper," http://info.ixiacom.com/rs/098-FRB-840/images/Calsoft-Labs-CaseStudy2015.pdf
    - Test Setup:
      - COTS server with Intel Xeon E5-2680 v2 processor
      - Virtual CPE VNFs (Firewall etc.) fast path optimized using Intel DPDK
      - Measured L2-L3 TCP traffic throughput per core
    - VM (KVM) environment with SRIOV -- 5.8Gbps
    - Containers (LXC) environment -- 7.2Gbps
      - **~25% PERFROMANCE IMPROVEMENT OVER VMs**
- Opportunistic areas for future work
  - Distributed micro-service network functions
  - VNF Controller discovery/management/etc. standardization
  - etc.

Call for Action

- Address aforementioned challenges
- Further research to identify currently unknown challenges
- Vendors to consider developing container based solutions – especially to support proof of concepts and field trials
- Reach consensus on a common framework for use of containers for NFV
- Field trial container-based VNFs