5G NFV網路技術基礎課程

工研院資通所人工智慧運算平台組

李育緯 技術經理

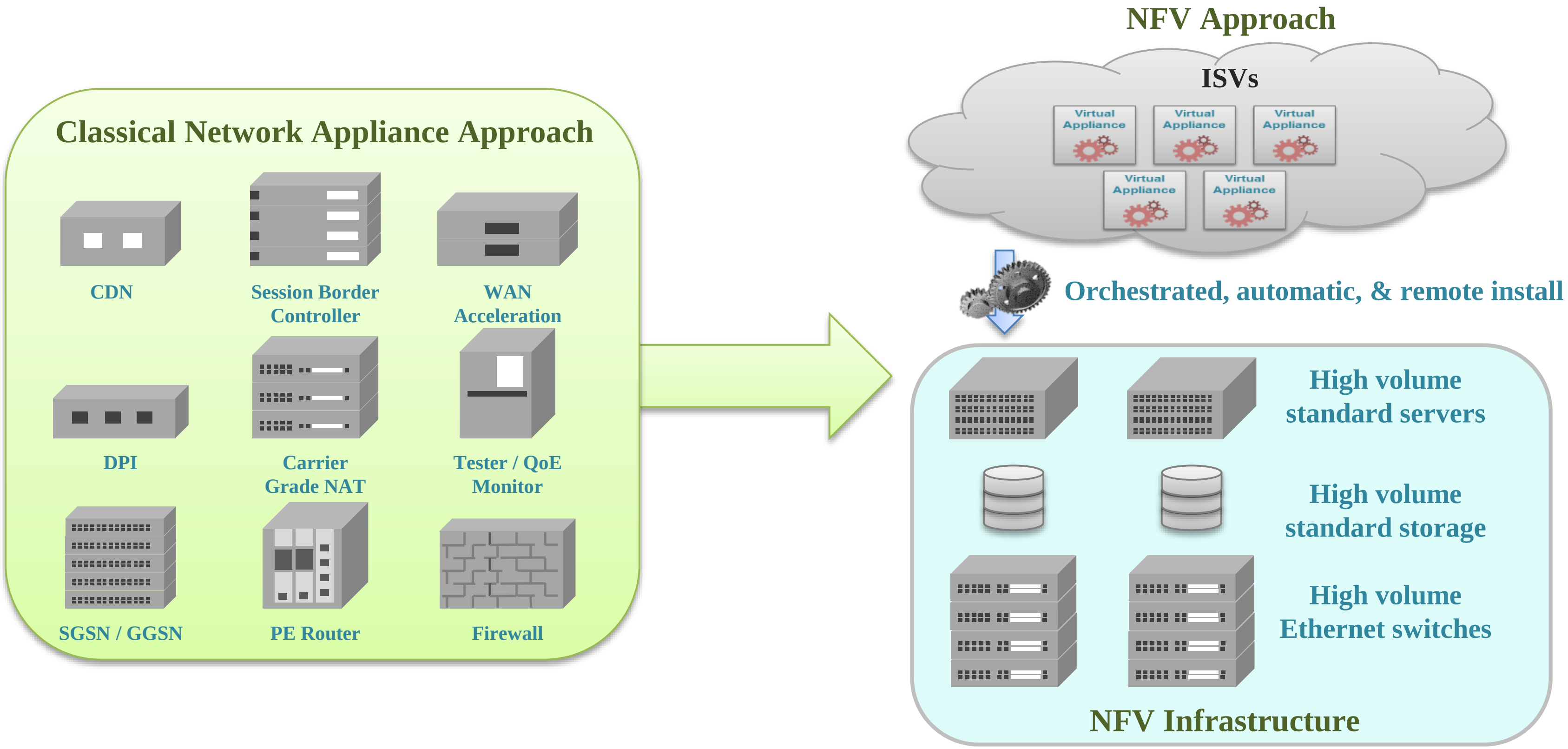
Email: rayinlee@itri.org.tw

Outline

- NFV發展現況
- NFV面臨問題與挑戰
- NFV相關解決方案

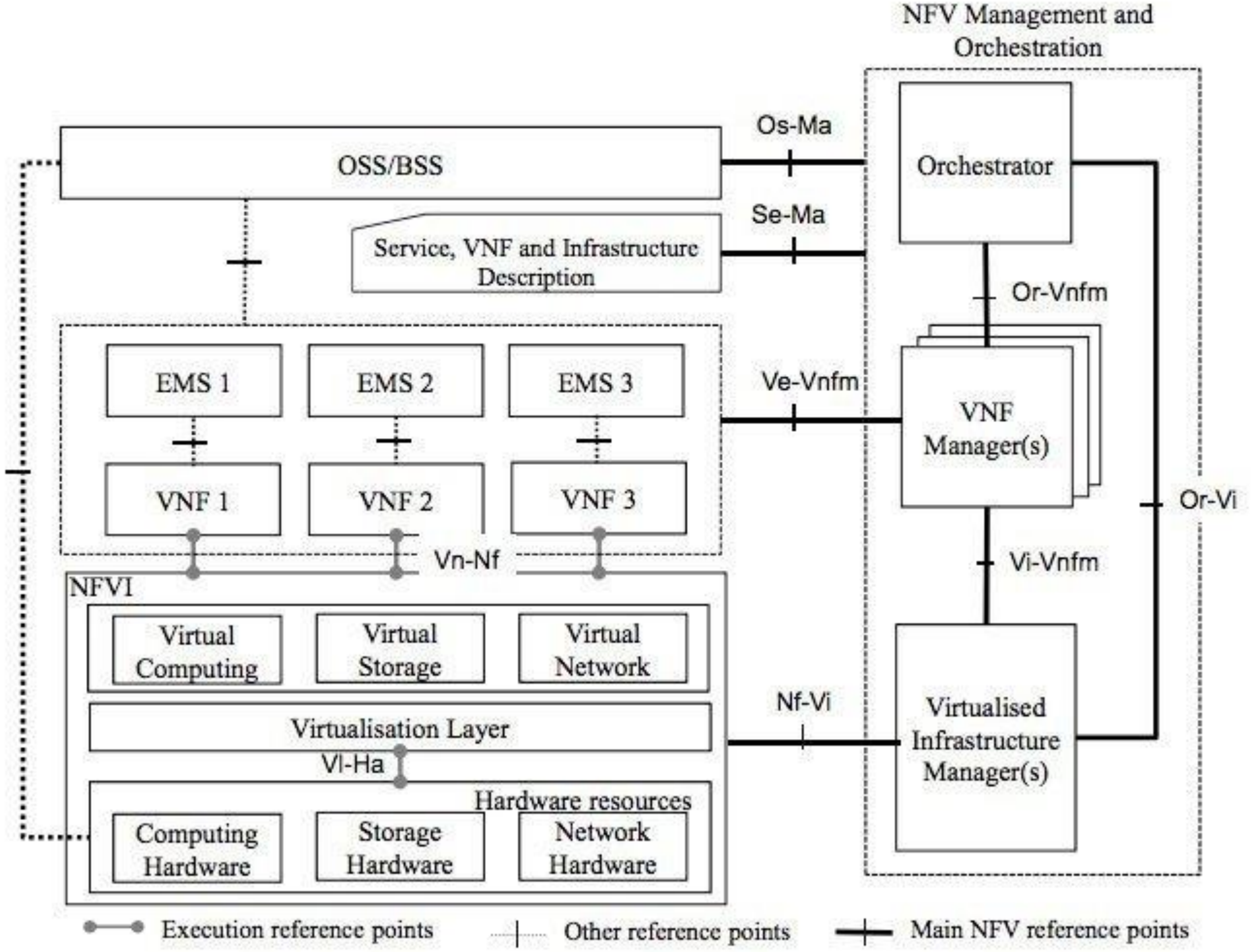
Network Function Virtualization

- 網路功能虛擬化(NFV)是將電信網路功能從實體設備抽離出來，以虛擬機(VM)或容器(Container)的形式運行在虛擬化的設備(NFVI)之上。
  - 將軟體功能與硬體設備解耦，能降低建置與管理成本。



ETSI NFV Architecture

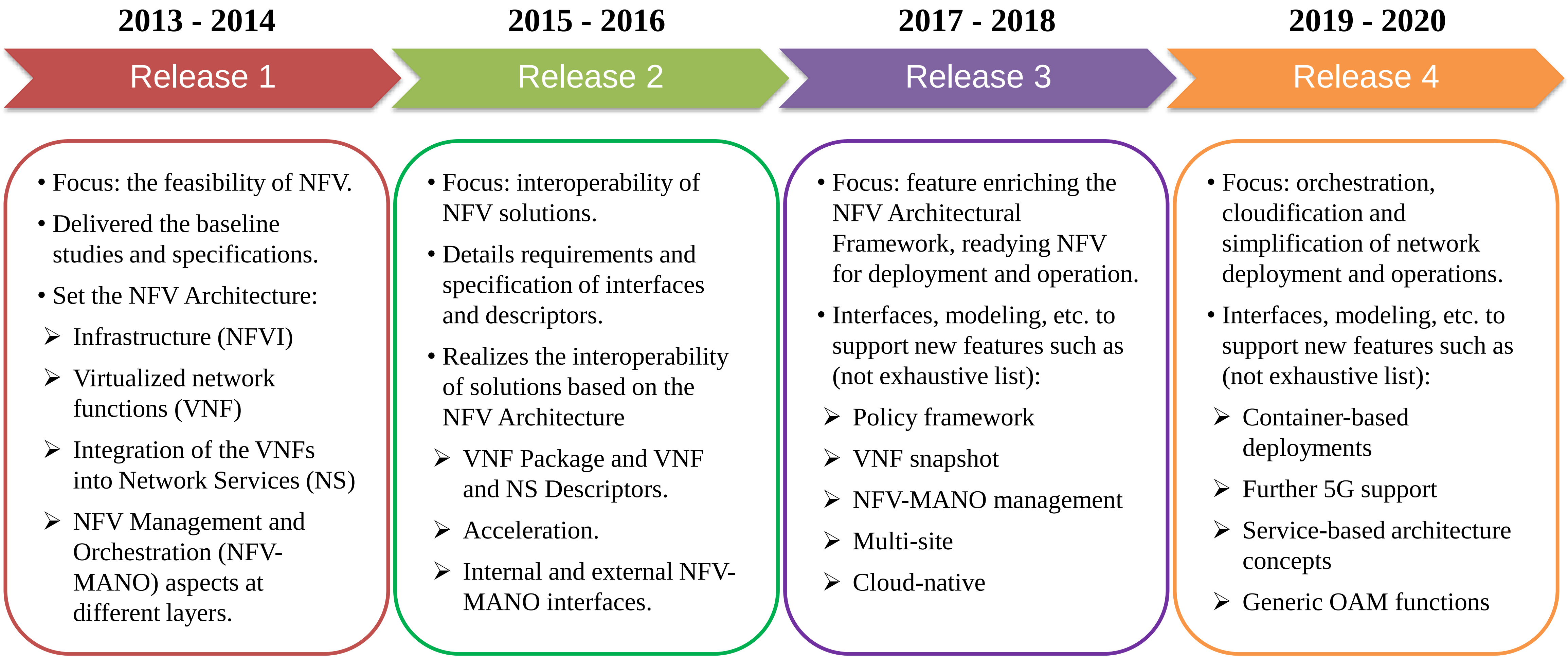
- **NFV功能模塊：**
  - Virtualized Network Function (VNF)
  - Element Management System (EMS)
  - NFV Infrastructure, including:
    - ☞ Hardware and virtualized resources
    - ☞ Virtualization Layer
  - Virtualized Infrastructure Manager(s)
  - VNF Manager(s)
  - Orchestrator
  - Service, VNF and Infrastructure
  - Operations and Business Support Systems (OSS/BSS)



NFV reference architecture framework

Source: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

ETSI NFV Release Overview



2013 - 2014

Release 1

- Focus: the feasibility of NFV.
- Delivered the baseline studies and specifications.
- Set the NFV Architecture:
  - ➢ Infrastructure (NFVI)
  - ➢ Virtualized network functions (VNF)
  - ➢ Integration of the VNFs into Network Services (NS)
  - ➢ NFV Management and Orchestration (NFV-MANO) aspects at different layers.

2015 - 2016

Release 2

- Focus: interoperability of NFV solutions.
- Details requirements and specification of interfaces and descriptors.
- Realizes the interoperability of solutions based on the NFV Architecture
  - ➢ VNF Package and VNF and NS Descriptors.
  - ➢ Acceleration.
  - ➢ Internal and external NFV-MANO interfaces.

2017 - 2018

Release 3

- Focus: feature enriching the NFV Architectural Framework, readying NFV for deployment and operation.
- Interfaces, modeling, etc. to support new features such as (not exhaustive list):
  - ➢ Policy framework
  - ➢ VNF snapshot
  - ➢ NFV-MANO management
  - ➢ Multi-site
  - ➢ Cloud-native

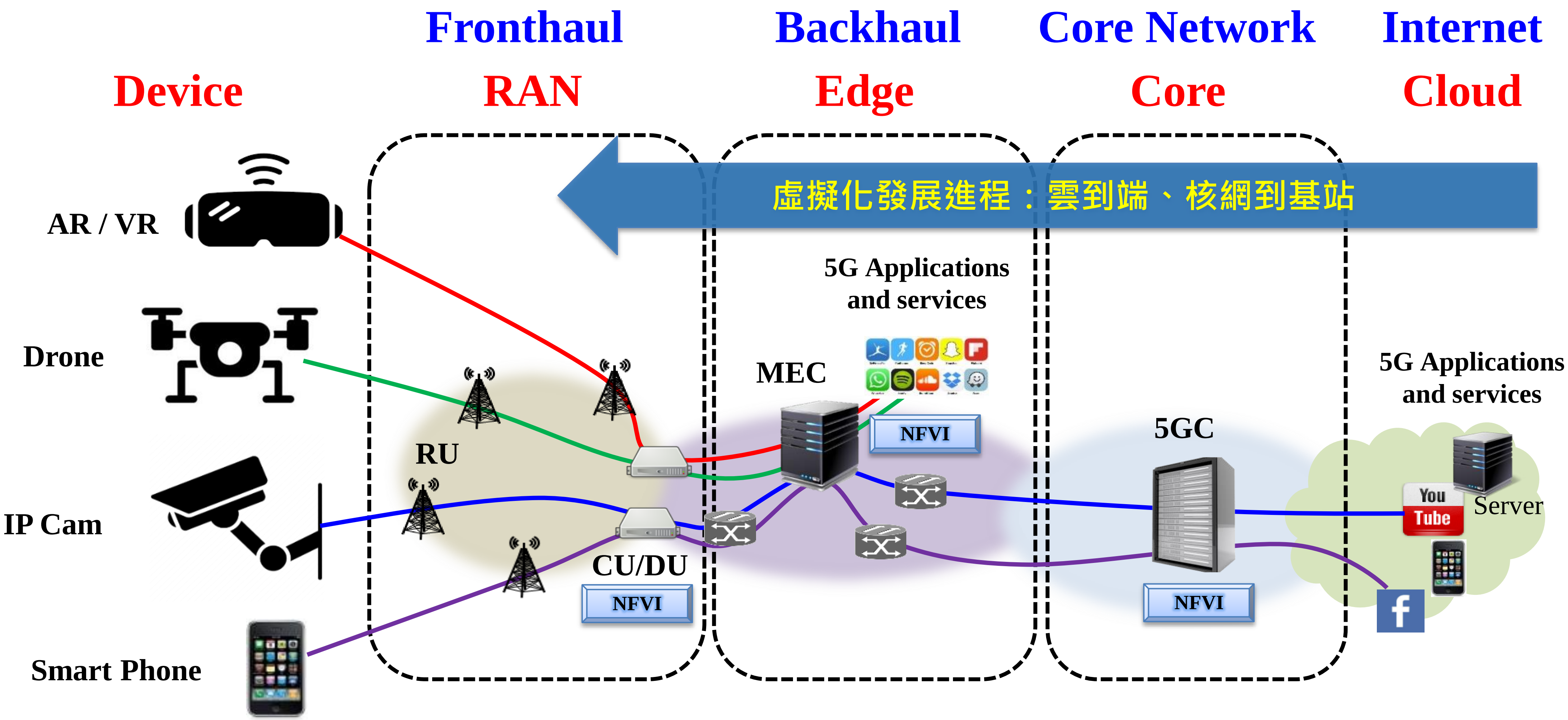
2019 - 2020

Release 4

- Focus: orchestration, cloudification and simplification of network deployment and operations.
- Interfaces, modeling, etc. to support new features such as (not exhaustive list):
  - ➢ Container-based deployments
  - ➢ Further 5G support
  - ➢ Service-based architecture concepts
  - ➢ Generic OAM functions

Source: https://docbox.etsi.org/ISG/NFV/Open/Other/ReleaseDocumentation/NFVTSC(21)000004r1_Summary_of_NFV_Releases_by_Jan_2021.pdf

5G NFV Overview

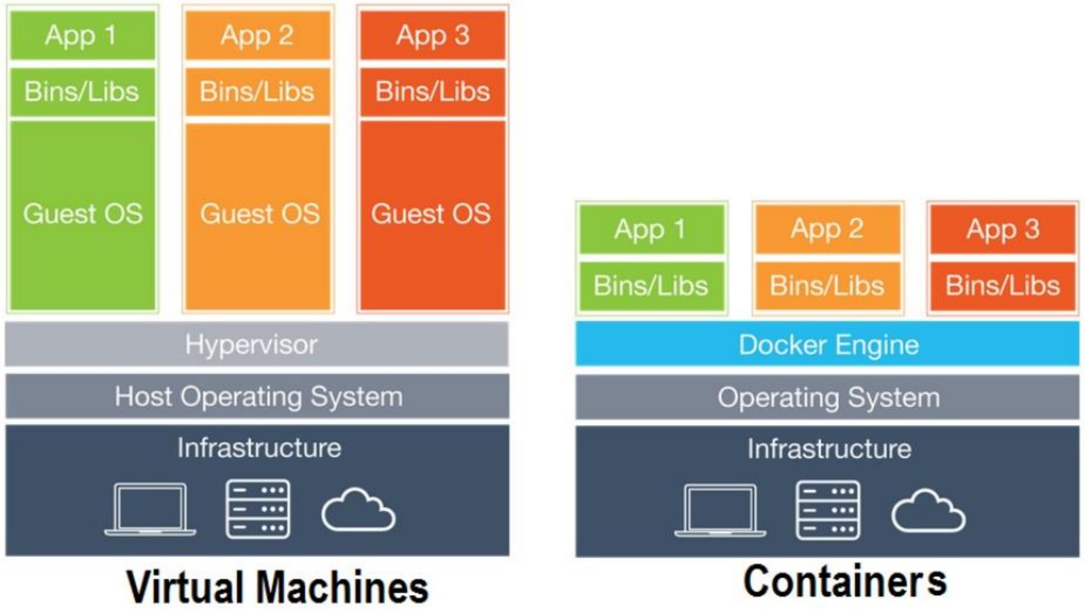


NFV面臨的問題與挑戰

- ■ 從雲到端、從核網到基站，需要更輕量化的虛擬化技術
  - ● 虛擬機(VM) => 容器(Container)
  - ● OpenStack(虛擬機管理平台) => Kubernetes(容器管理平台)
  - ● 雲原生 (Cloud Native) 的開發概念
- ■ 使用x86架構取代專用網路晶片，效能是否能滿足需求
  - ● NFV效能優化技術
    - ☞CPU與NUMA Node
    - ☞Hugepages
    - ☞網路加速技術(DPDK, SRIOV, SmartNIC)
    - ☞加速卡資源管理
  - ● Case Study: O-RAN虛擬化技術需求
- ■ NFV架構下，需要更好的效能監控、分析與管理技術
  - ● eBPF技術解說

虛擬機(VM)與容器(Container)

- 傳統的虛擬主機技術利用硬體的功能來模擬實現也可以通過作業系統軟體來實現，常見的傳統虛擬化技術，例如 **vSphere** 或 **Hyper-V** 是以作業系統為中心，而 **Container** 技術則是直接將一個應用程式所需的相關程式碼、函式庫、環境配置檔都打包起來建立沙箱執行環境。
- 其中最明顯的差異是，虛擬機器需要安裝作業系統（安裝 **Guest OS**）才能執行應用程式，而 **Container** 技術不是在 **OS** 外來建立虛擬環境，是透過作業系統內的核心系統層來打造虛擬執行環境，透過共用 **Host OS** 的作法，取代一個一個 **Guest OS** 的功用。



Source: https://www.wpgdadatong.com/tw/blog/detail?BID=B0722

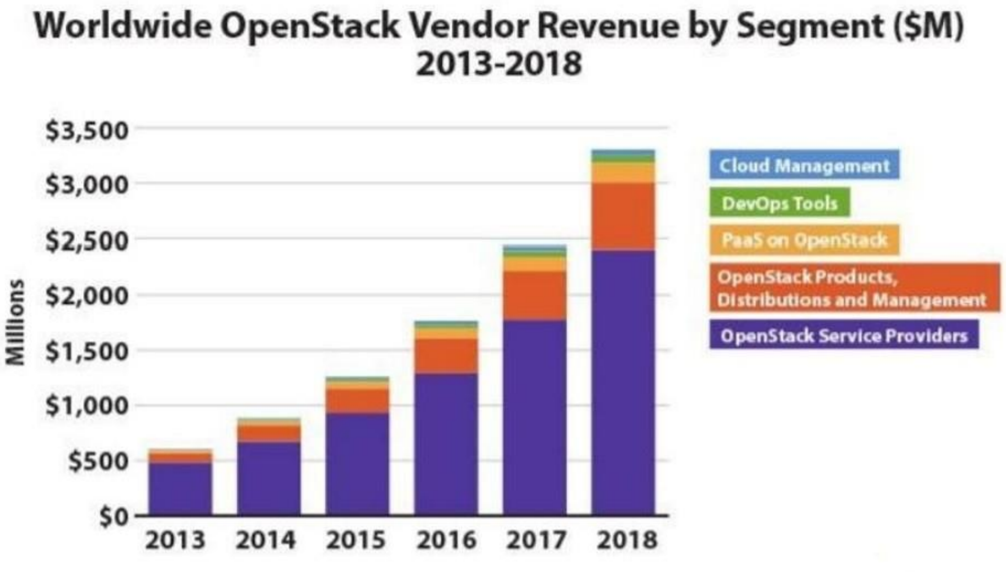
Image credits: docker.com

VM與Container比較

	VM	Container
虛擬化方式	作業系統層級	應用程式層級
OS	每個VM有自己的OS	所有Container共享Host OS
Image大小	數GB – 數百GB	數MB – 數百MB
佈署時間	數分鐘	數秒鐘
系統支持量	最多數十個	最多到數千個
隔離性	完全隔離	共用Kernel

開源雲端系統 - OpenStack

- OpenStack是開源的雲端系統，創立於2010年。
  - 由Rackspace與NASA共同發布。
  - 提供IaaS服務，為客戶提供虛擬服務器與其他資源。
  - 從第12版Liberty開始，改變Release方式，從一次下載所有模組變成下載核心模組搭配周邊套件。



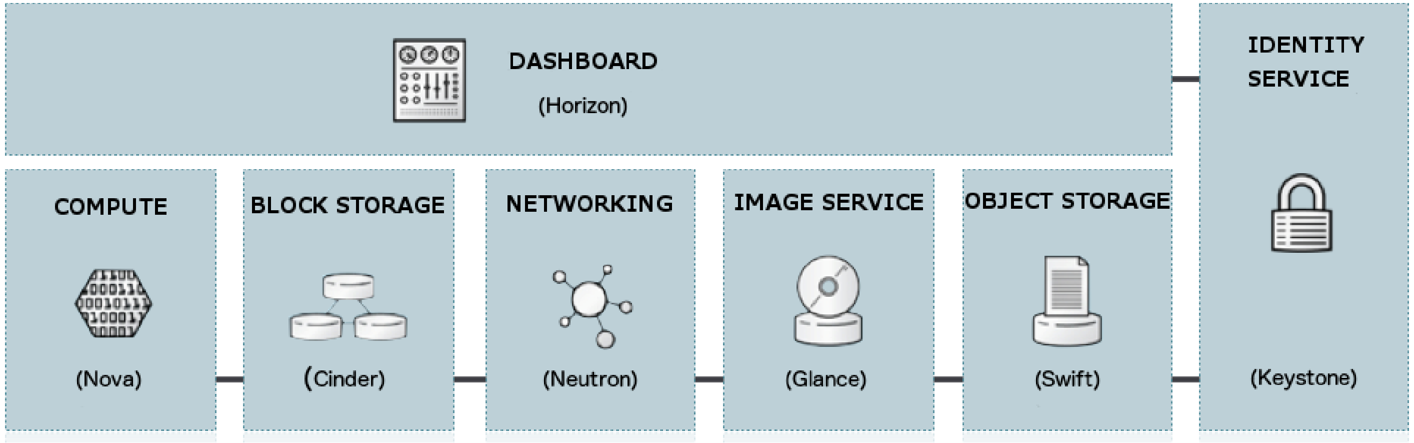
Source: 451 Research's Market Monitor & Forecast, OpenStack September 2014

版本	模組	狀態
Austin	2	EOL
Bexar	3	EOL
Cactus	3	EOL
Diablo	3	EOL
Essex	5	EOL
Folsom	7	EOL
Grizzly	7	EOL
Havana	9	EOL
Icehouse	10	EOL
Juno	11	EOL
Kilo	13	EOL
Liberty	21	EOL
Mitaka	29	EOL
Newton	32	EOL
Ocata	32	擴展維護
Pike	32	擴展維護
Queens	39	擴展維護
Rocky	40	擴展維護

T,U,V版維護中，W版開發中

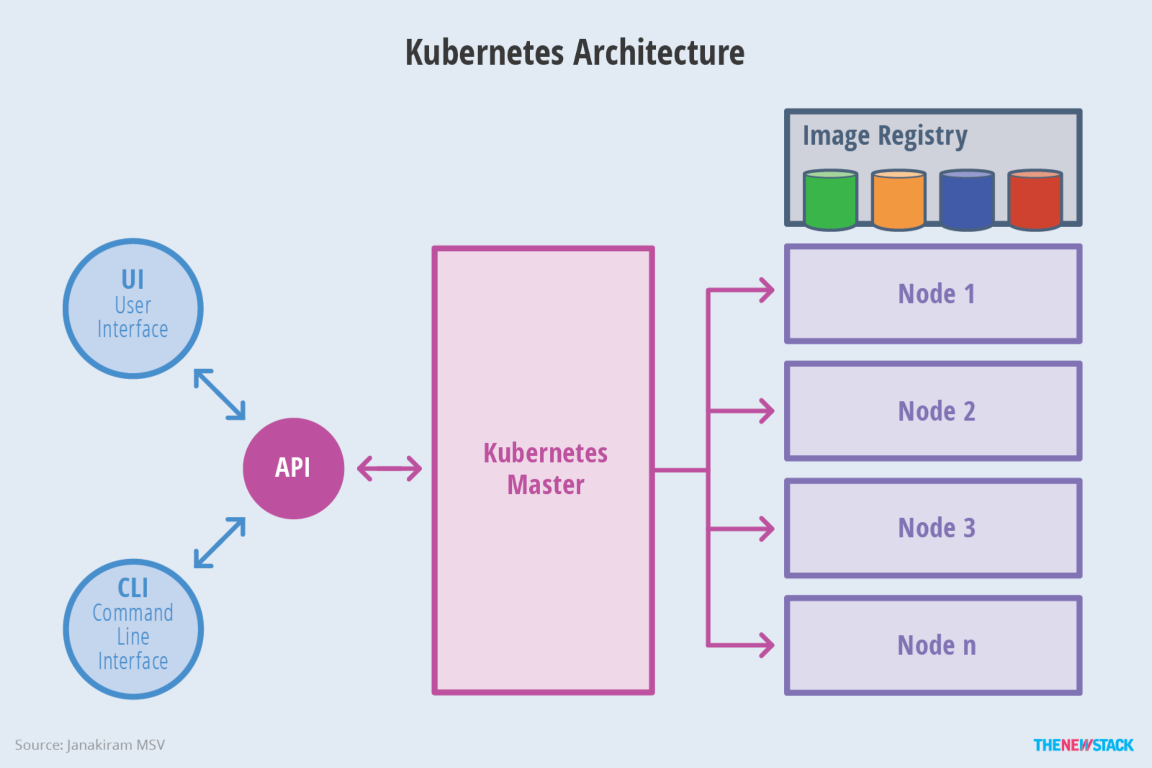
OpenStack模組介紹

- **Horizon**
  - OpenStack Dashboard，為管理員和使用者提供圖形化介面，可以管理、佈署、監控雲端系統資源(虛擬機、虛擬網路、虛擬儲存空間)。
- **Keystone**
  - OpenStack Identity，管理OpenStack用戶與可存取的服務的對應關係。
- **Nova**
  - OpenStack Compute，計算資源虛擬化，提供計算機資源(虛擬機或裸機)的管理與配置。
- **Cinder**
  - OpenStack Block Storage，儲存資源虛擬化，提供Block-level儲存資源的管理與配置。
- **Neutron**
  - OpenStack Networking，網路虛擬化，提供用戶管理各自的虛擬網路，如Router、Firewall、VPN、IP Subnet等。



開源容器管理平台 - Kubernetes

- Kubernetes（常簡稱為K8s）是用於自動部署、擴展和管理容器化（containerized）應用程式的開源系統，提供「跨主機集群的自動部署、擴展以及運行應用程式容器的平台」。

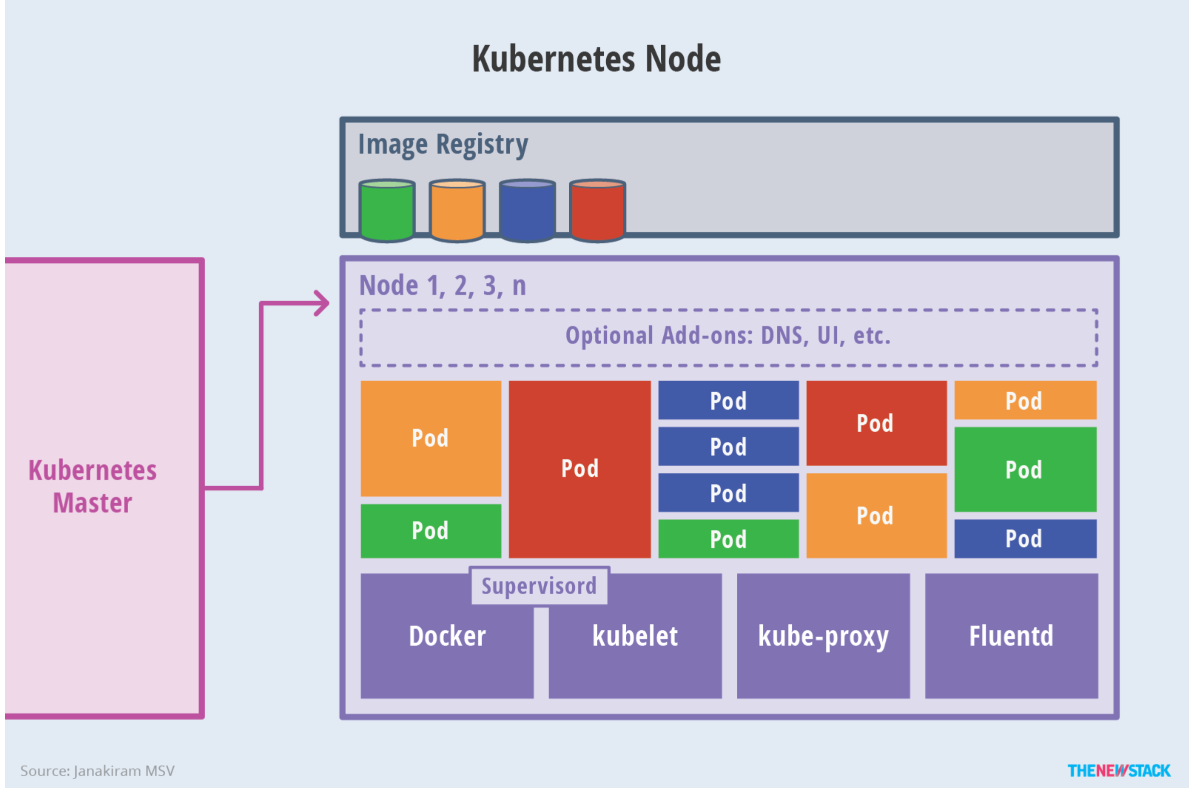


Source: https://thenewstack.io/kubernetes-an-overview/

Kubernetes Pod介紹

■ Pods

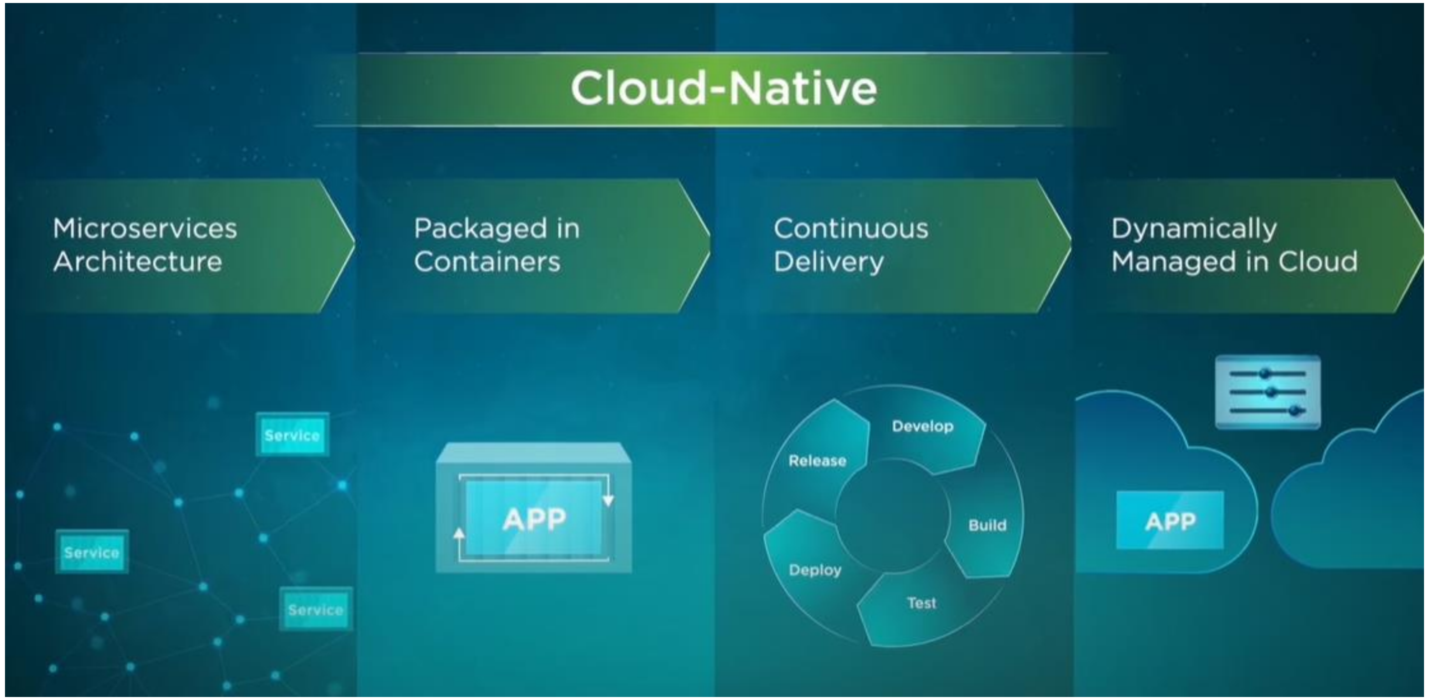
- Kubernetes的基本調度單元，可以包含一個或多個Container，同一個Pod的Container會佈署在同一台機器並共享資源。Kubernetes會幫每個Pod配置一個IP位址。



Source: https://thenewstack.io/kubernetes-an-overview/

雲原生(Cloud Native)觀念解說

- Cloud Native不是一個技術名詞，而是一種開發與運行應用服務的方法。
- 大致涵蓋下列四個特點
  - Microservices Architecture
  - Packaged in Containers
  - Continuous Delivery
  - Dynamically Managed in Cloud
- 以Cloud Native的方式進行開發，可以讓整個開發到後續維運的流程更加快速、有效率且容易更新調整。



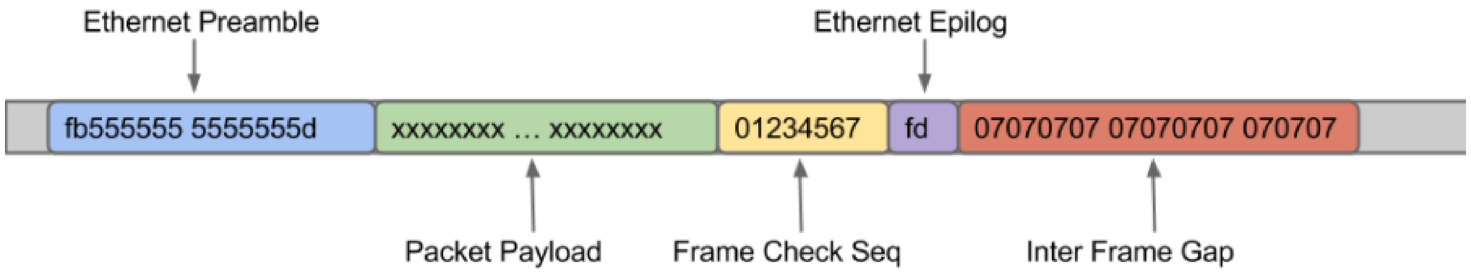
Source: https://www.youtube.com/watch?v=9Ik96SBaIvs

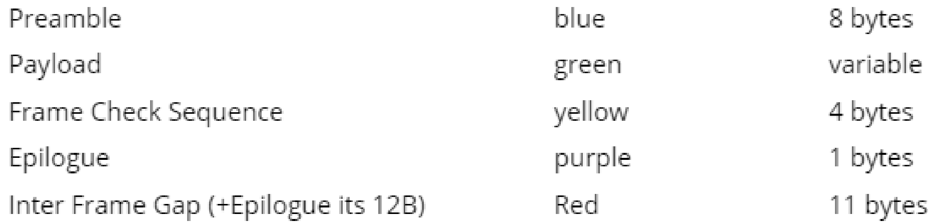
NFV面臨的問題與挑戰

- 從雲到端、從核網到基站，需要更輕量化的虛擬化技術
  - 虛擬機(VM) => 容器(Container)
  - OpenStack(虛擬機管理平台) => Kubernetes(容器管理平台)
  - 雲原生 (Cloud Native) 的開發概念
- 使用x86架構取代專用網路晶片，效能是否能滿足需求
  - NFV效能優化技術
    - ☞CPU與NUMA Node
    - ☞Hugepages
    - ☞網路加速技術(DPDK, SRIOV, SmartNIC)
    - ☞加速卡資源管理
  - Case Study: O-RAN虛擬化技術需求
- NFV架構下，需要更好的效能監控、分析與管理技術
  - eBPF技術解說

何謂10G Bit Line Rate

- 在NFV的領域，因為使用CPU來執行虛擬化網路功能(Router、Switch、Firewall等)，在討論NFV的效能時，我們要了解所謂的10G Bit究竟有多少封包要處理。
- 通常我們以64 Bytes的封包大小來估算應該要處理的封包數量
  - 64 Bytes封包包含60 Bytes的Payload與4 Bytes的Frame Check Sequence，再加上L1的20 Bytes的Overhead，封包大小總共是84 Bytes。
    - ☞ $10 \times 10^9 / 84 \times 8 = 14.88 \times 10^6$ (14.88百萬個封包)，每秒要處理14.88百萬個封包，相當於1個封包只有67ns的時間可以處理
    - ☞ DDR4的Memory Access Latency約15ns
- 若封包大小是1500 Bytes的話，又會如何呢?
  - 1500 Bytes封包，加上L1 20Bytes的Overhead，封包大小總共是1520 Bytes
    - ☞ $10 \times 10^9 / 1520 \times 8 = 822,368$ (82萬個封包)，每秒要處理82萬個封包，相當於1個封包有1216ns的時間可以處理。



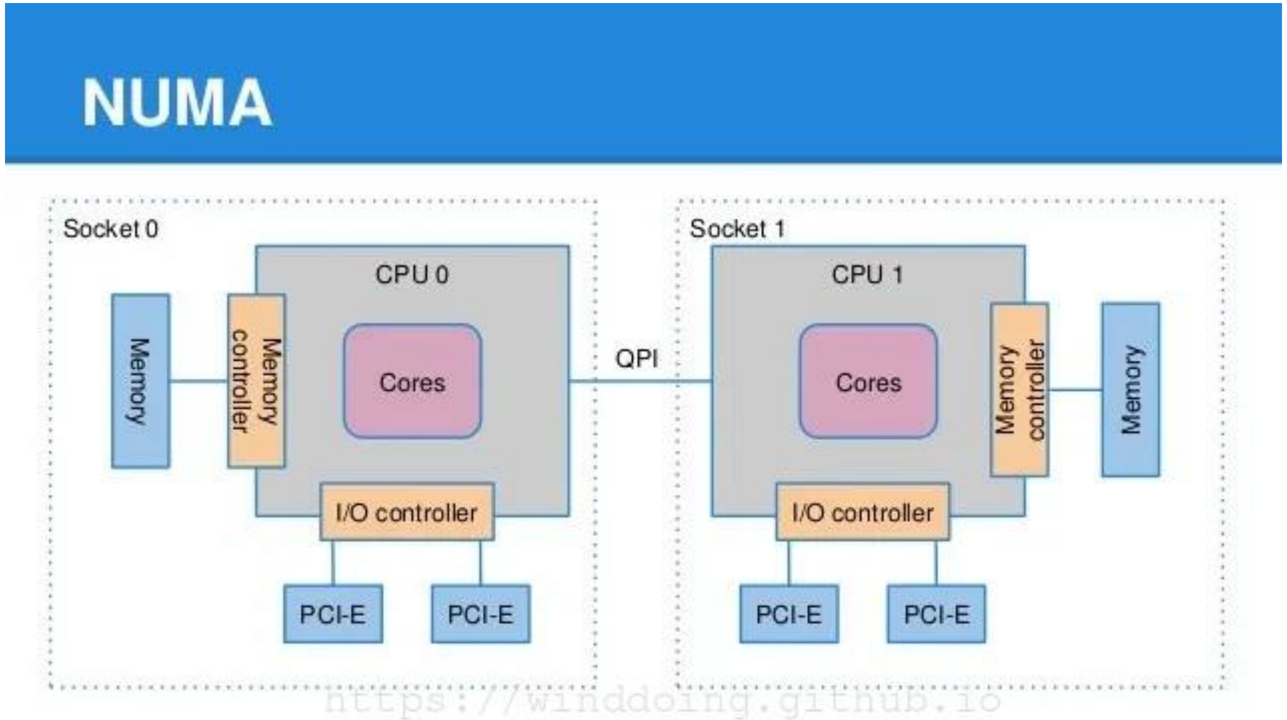


Preamble	blue	8 bytes
Payload	green	variable
Frame Check Sequence	yellow	4 bytes
Epilogue	purple	1 bytes
Inter Frame Gap (+Epilogue its 12B)	Red	11 bytes

Source: https://fmad.io/blog-what-is-10g-line-rate.html

NUMA概念解說

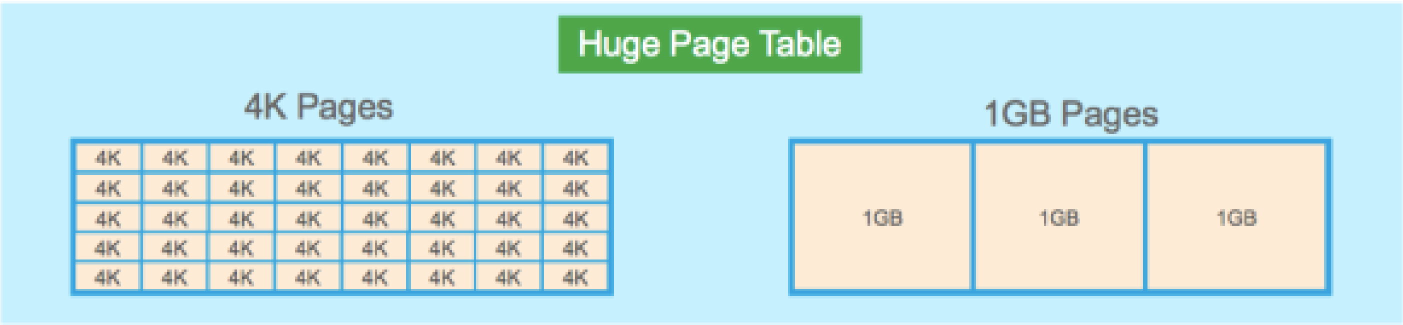
- **Non-uniform memory access (NUMA)系統是指一個主機板上有多個CPU，各CPU有各自的記憶體與IO控制器，CPU之間透過QPI介面可存取遠端的記憶體與IO設備，但CPU存取本地記憶體與IO設備的延遲與頻寬會優於存取遠端的記憶體與IO設備。**

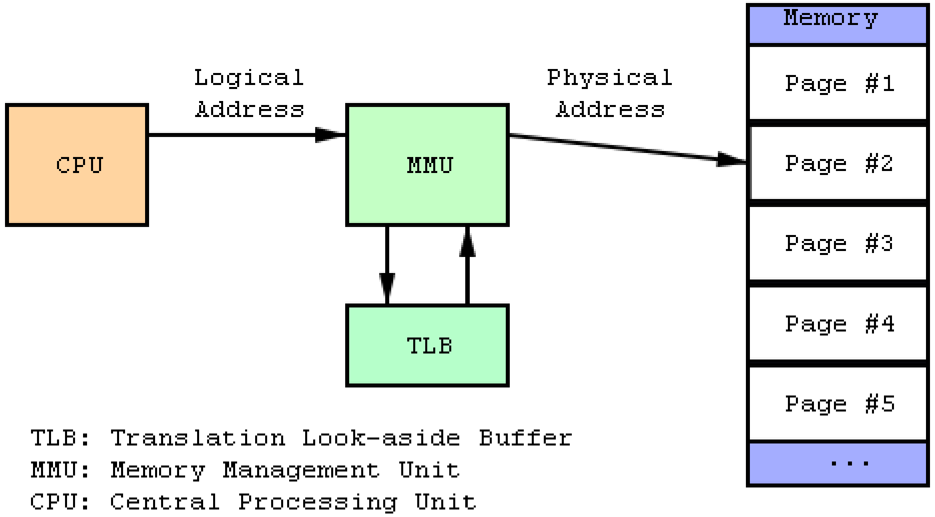


Source: https://winddoing.github.io/post/13d4e2a6.html

Hugepages

- 現行CPU可以支援不同的Page Size，如4KB, 2MB, 1GB，除了4KB以外的Page Size就稱為Hugepages。(最初Linux僅支援4KB Page Size)
- 使用Hugepage能減少虛擬記憶體與實體記憶體的轉換次數，也能減少Page Fault的發生機率，進而提升系統效能。



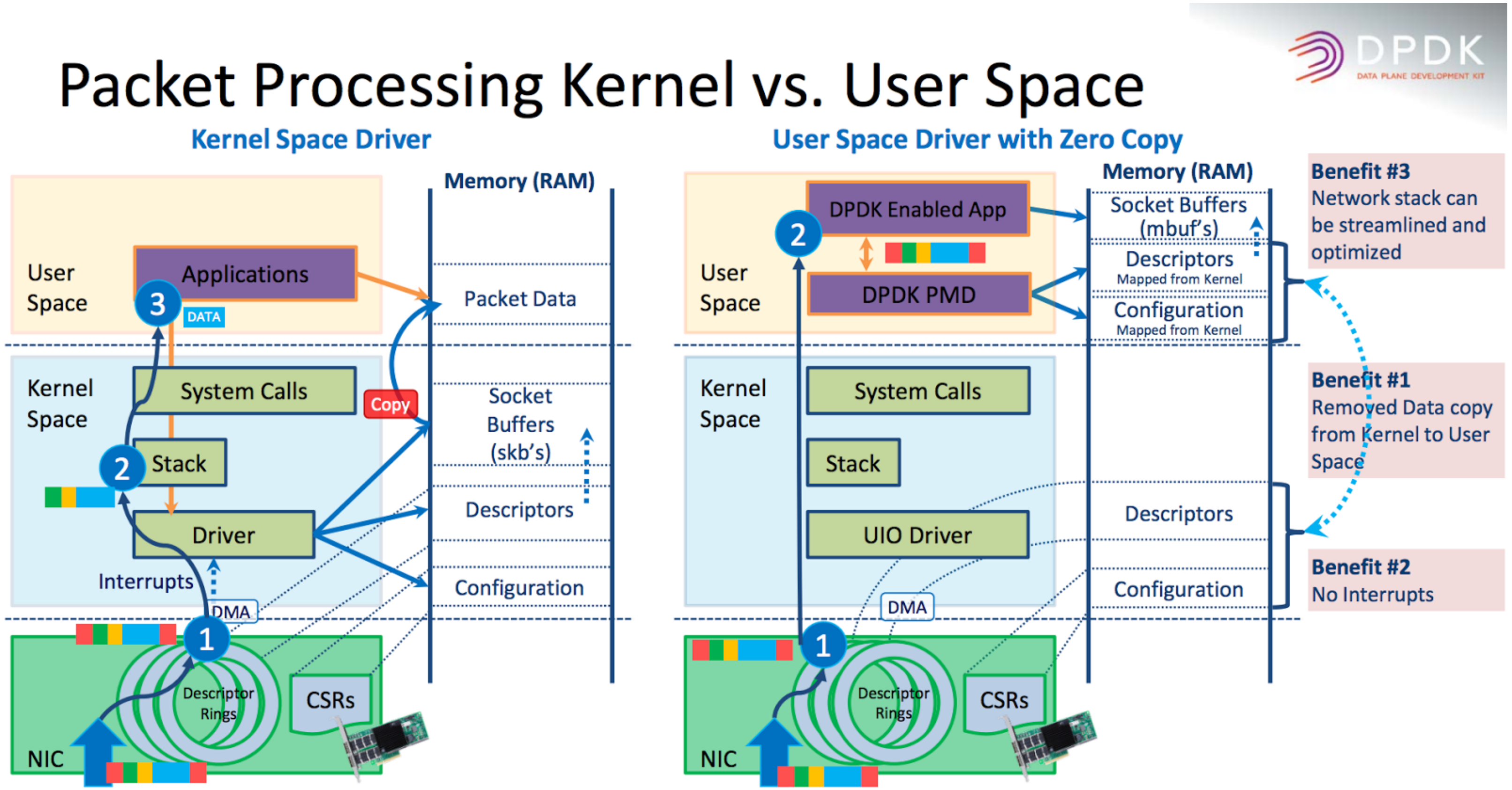


TLB: Translation Look-aside Buffer
MMU: Memory Management Unit
CPU: Central Processing Unit

Source: https://haryachyy.wordpress.com/2019/04/17/learning-dpdk-huge-pages/

DPDK介紹

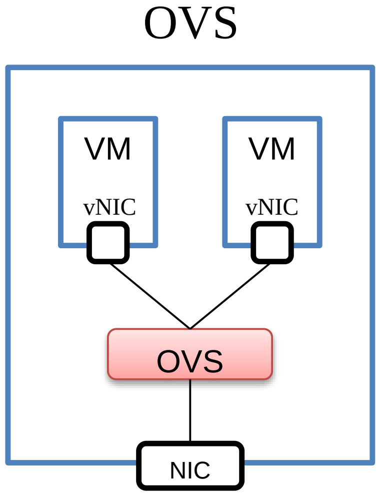
- DPDK (Data Plane Development Kit)是由Intel提供的開發工具集，不同於Linux系統以通用性設計為目的，而是專注於網路應用中數據封包的高性能處理。

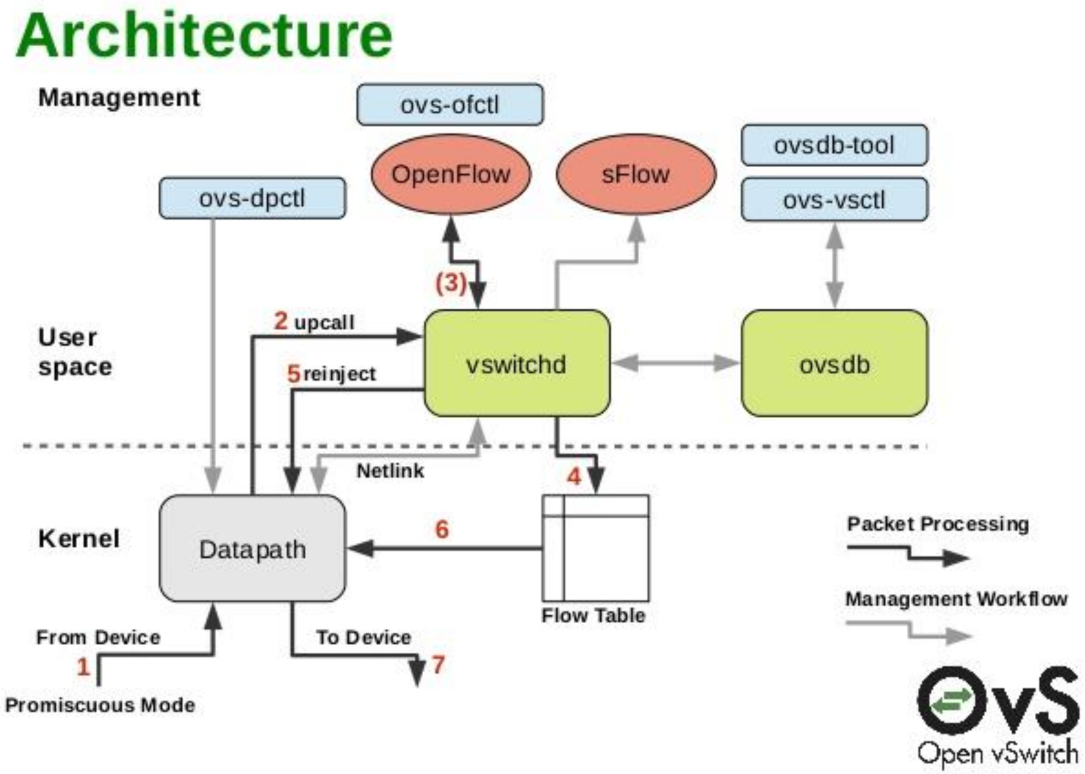


Source: https://www.slideshare.net/MichelleHolley1/dpdk-1805-inflection-point

OVS介紹

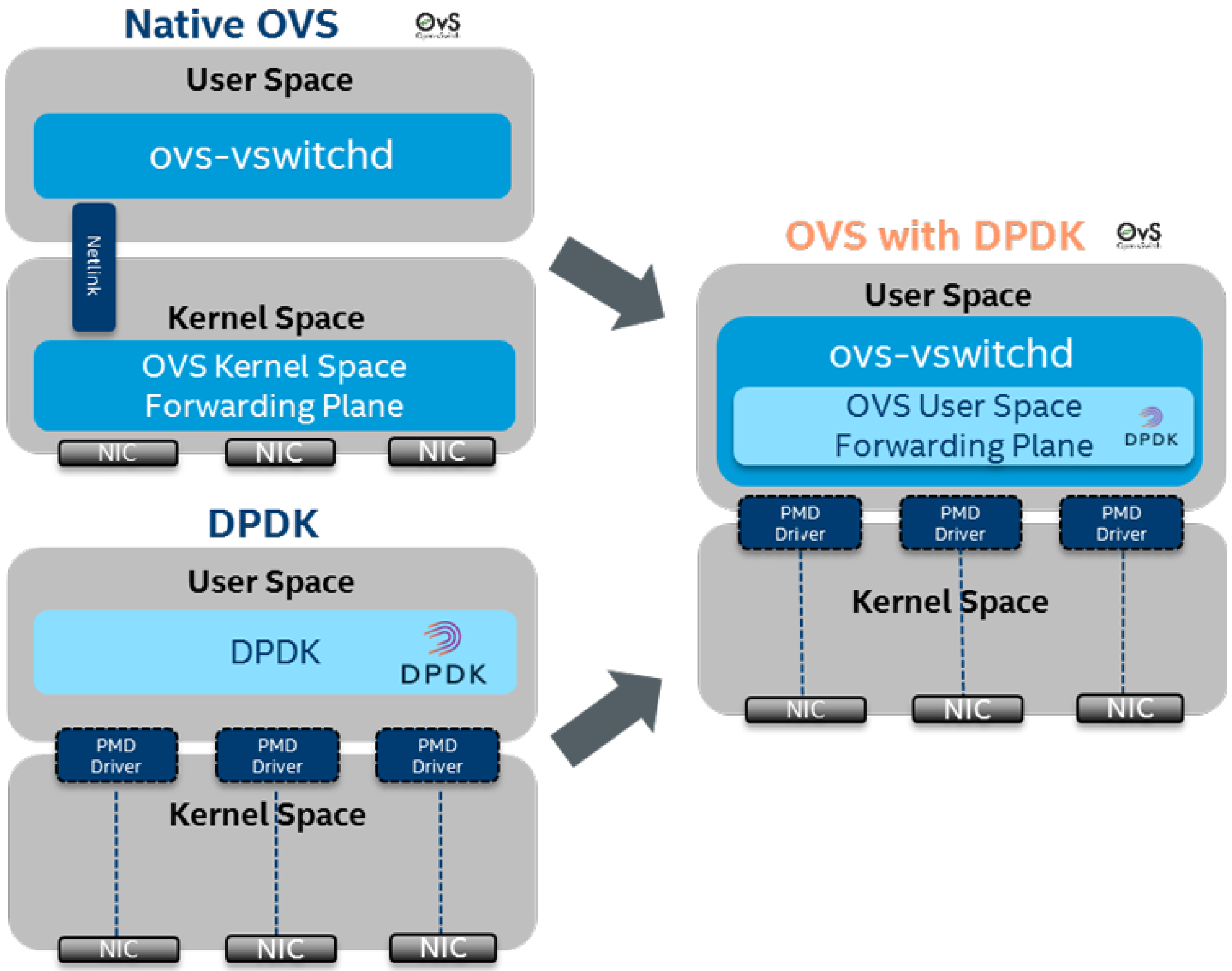
- Open vSwitch(OVS)是開源的虛擬交換器，支援VLAN/VxLAN/NVGRE等網路隔離功能，也支援QoS、sFLOW與OpenFlow協定。





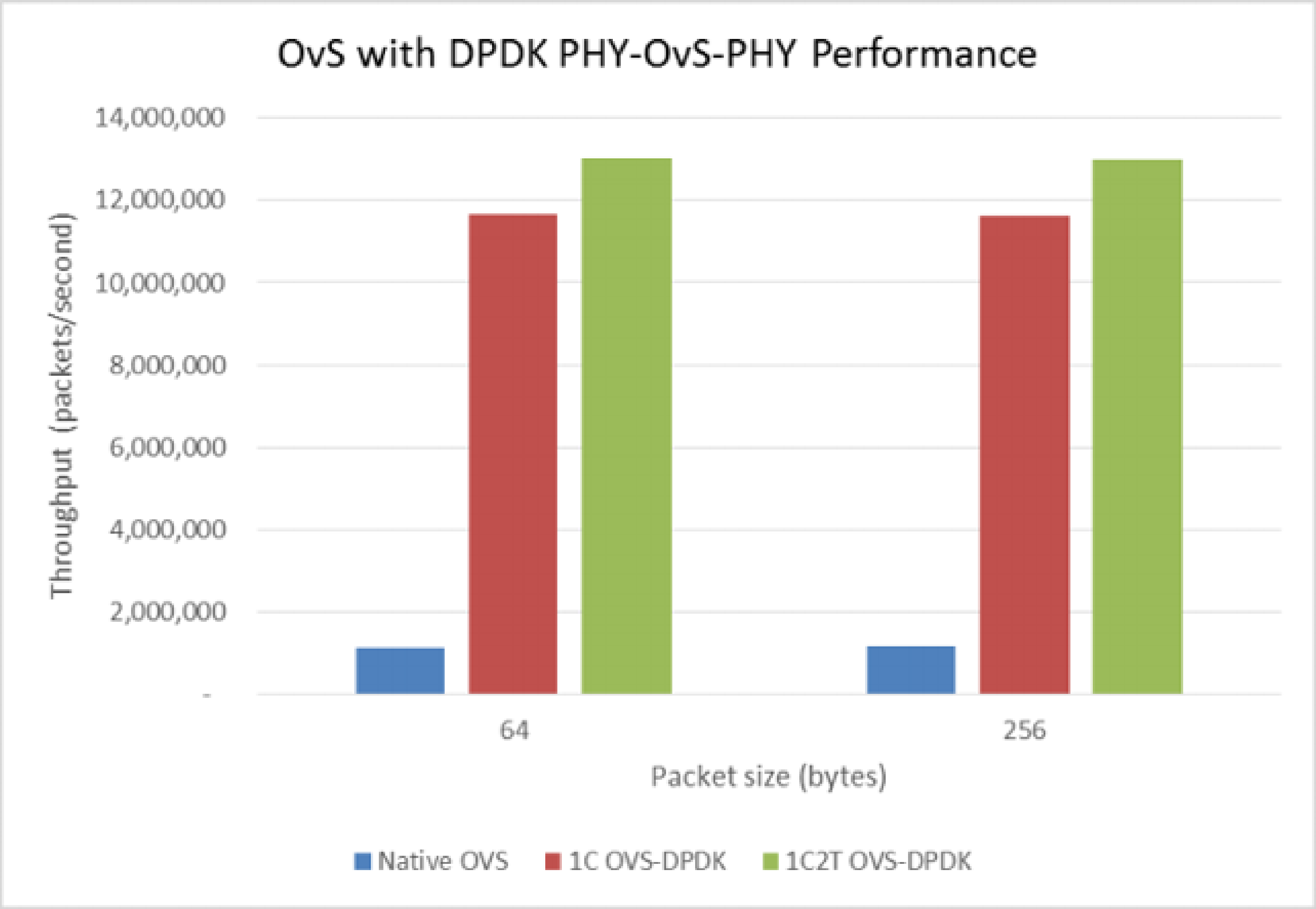
Source: https://hustcat.github.io/an-introduction-to-ovs-architecture/

OVS + DPDK



Source: https://sites.google.com/a/cnsrl.cycu.edu.tw/da-shu-bi-ji/openvswitch/dpdk-ovs

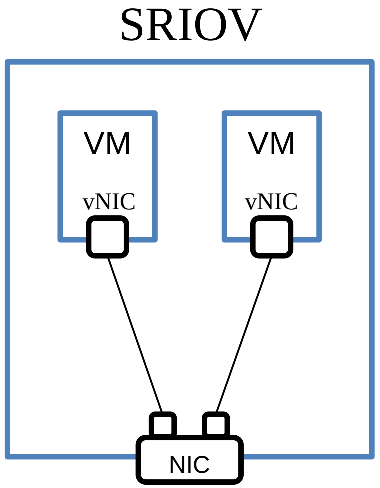
OVS+DPDK效能差異

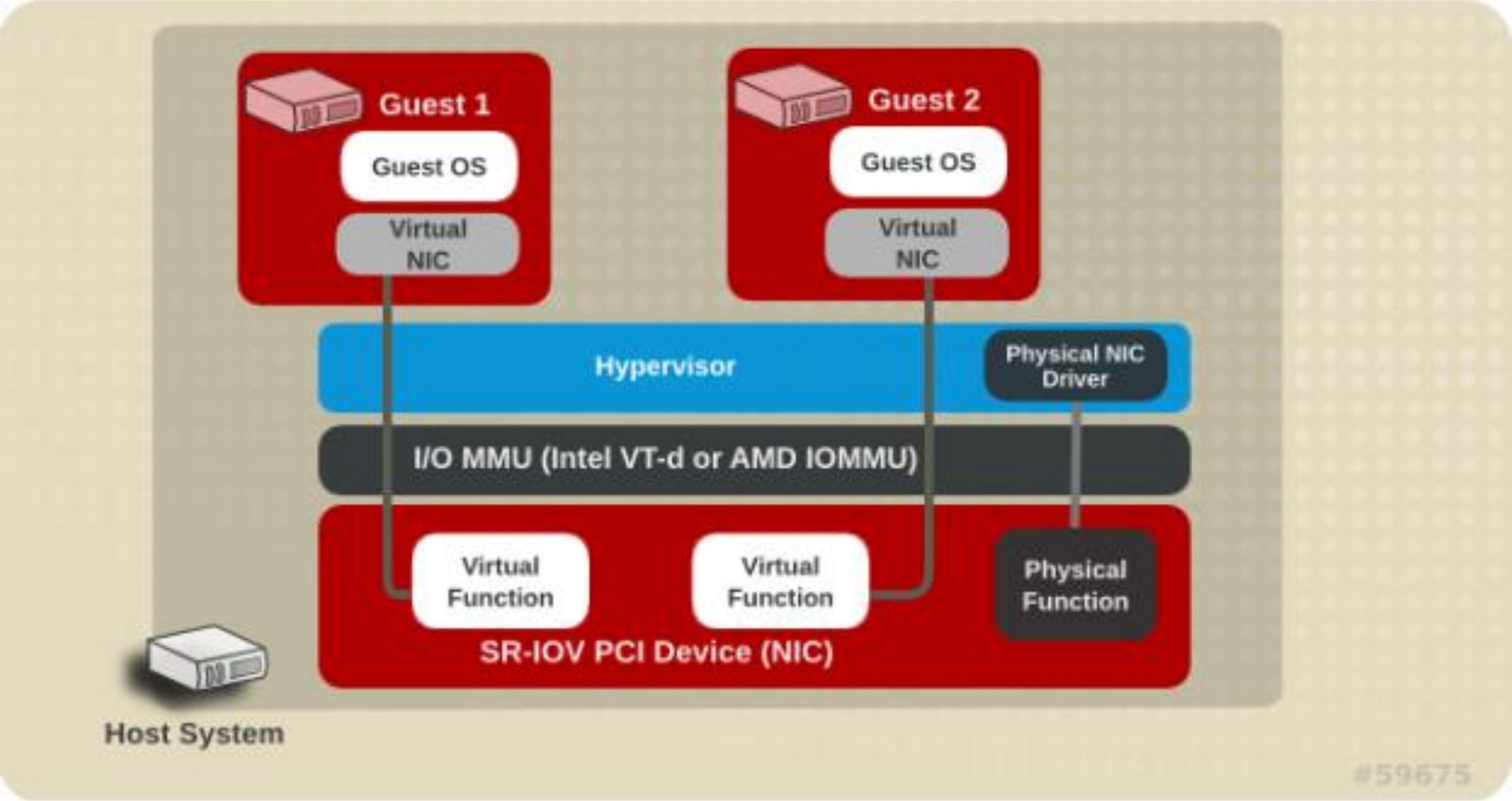


Source: https://sites.google.com/a/cnsrl.cycu.edu.tw/da-shu-bi-ji/openvswitch/dpdk-ovs

SRIOV網卡

- Single Root I/O Virtualization (SR-IOV)。SR-IOV為PCI-SIG標準，允許PCIe的I/O裝置以多個實體與虛擬裝置呈現。

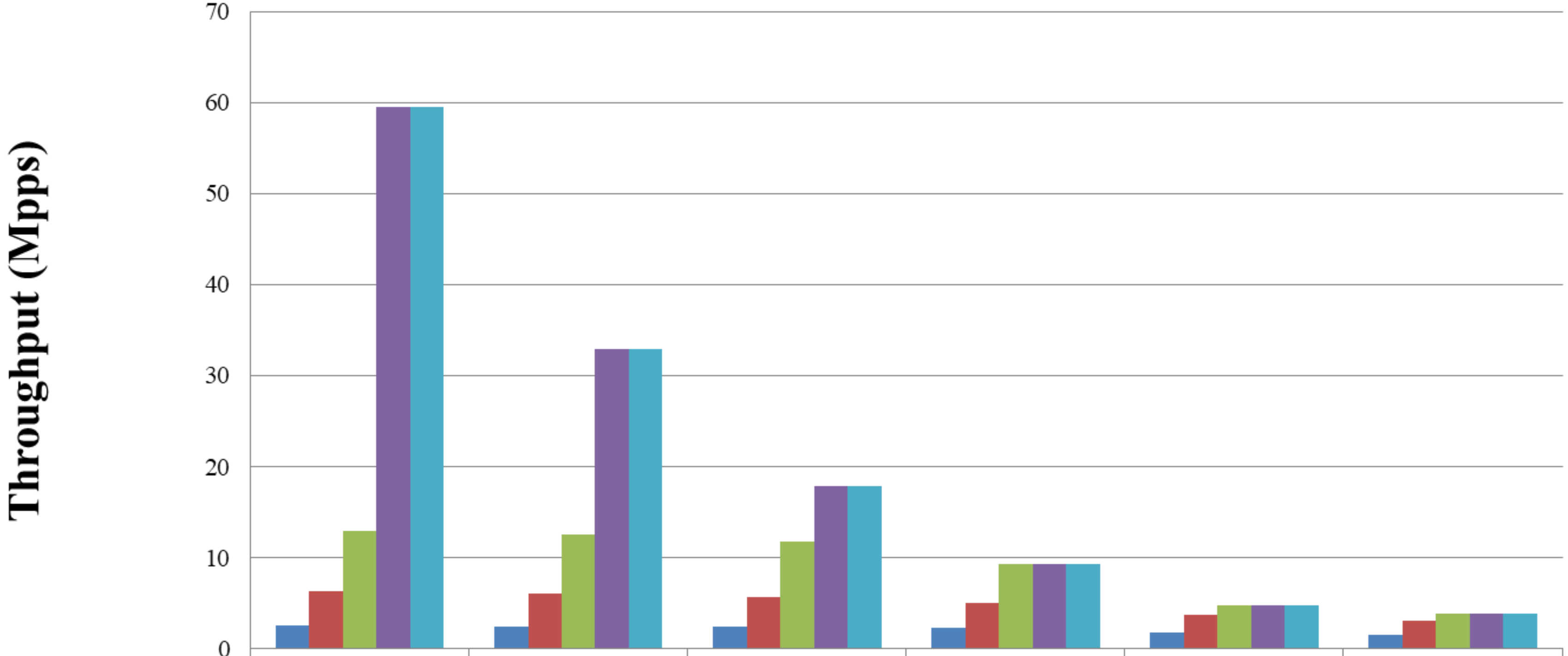




Source: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/virtualization_host_configuration_and_guest_installation_guide/chap-virtualization_host_configuration_and_guest_installation_guide-sr_iov

OVS-DPDK與SRIOV效能差異

OVS-DPDK vs SRIOV vs Baremetal 4 Port - Throughput (L2-Fwd)

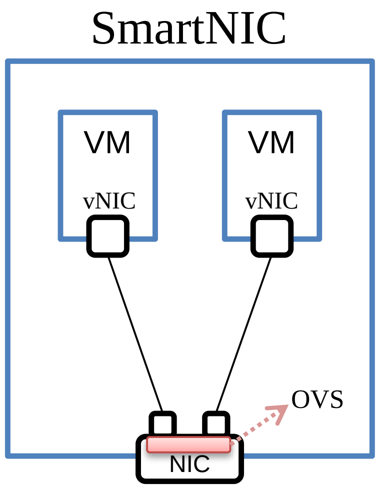


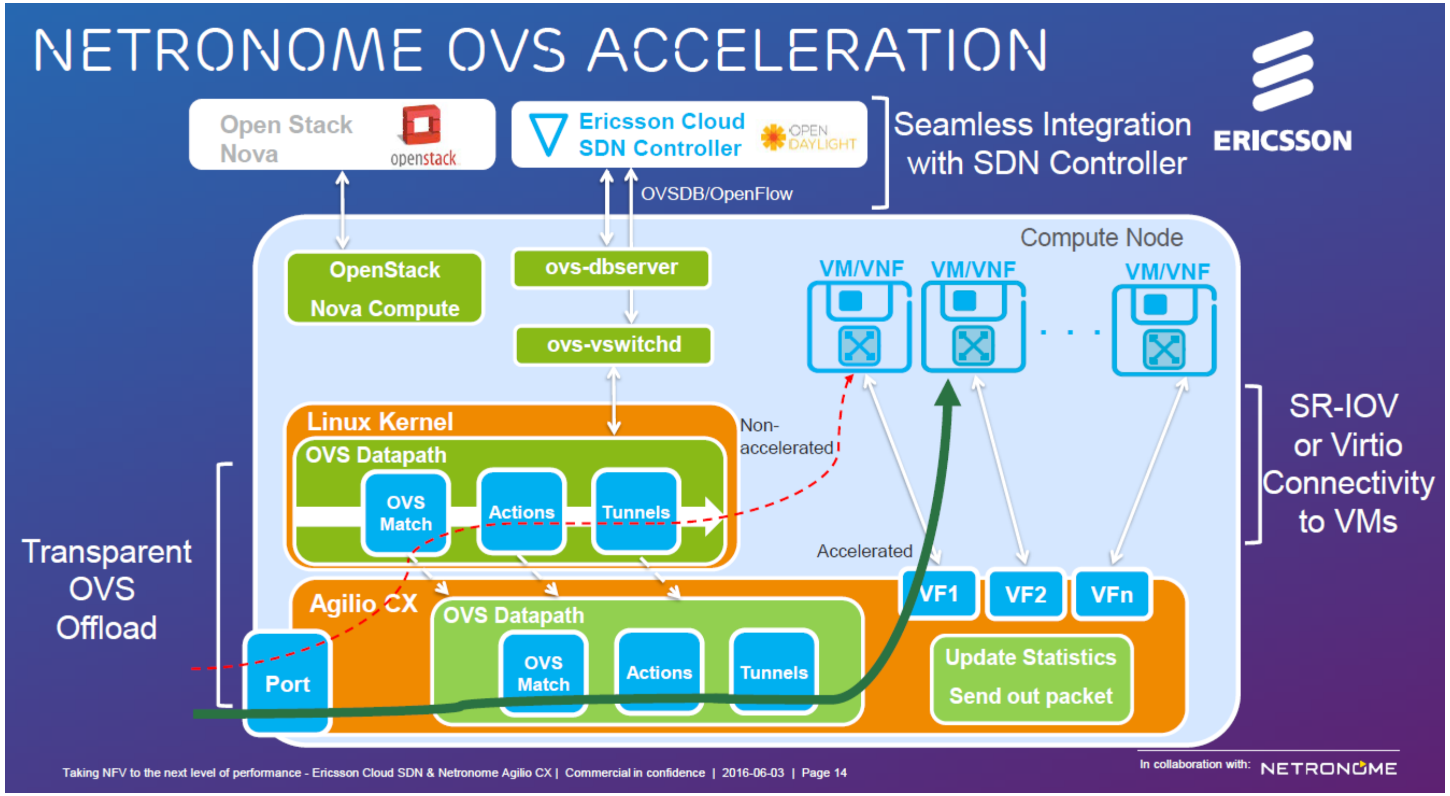
	64 Bytes	128 Bytes	256 Bytes	512 Bytes	1024 Bytes	1280 Bytes
🟦 OVS-DPDK-1-Core	2.51	2.46	2.38	2.26	1.74	1.47
🟥 OVS-DPDK-2-Cores	6.31	6.1	5.64	4.98	3.71	3.04
🟩 OVS-DPDK-4-Cores	12.96	12.5	11.82	9.32	4.77	3.83
🟪 SRIOV	59.52	32.89	17.85	9.32	4.77	3.83
■ BM	59.52	32.89	17.85	9.32	4.77	3.83

Source: ITRI NFV Performance Lab

SmartNIC

- **Netronome SmartNIC將OVS的功能做到硬體網卡中，可增加網路傳輸效能，並減少CPU資源的損耗。**



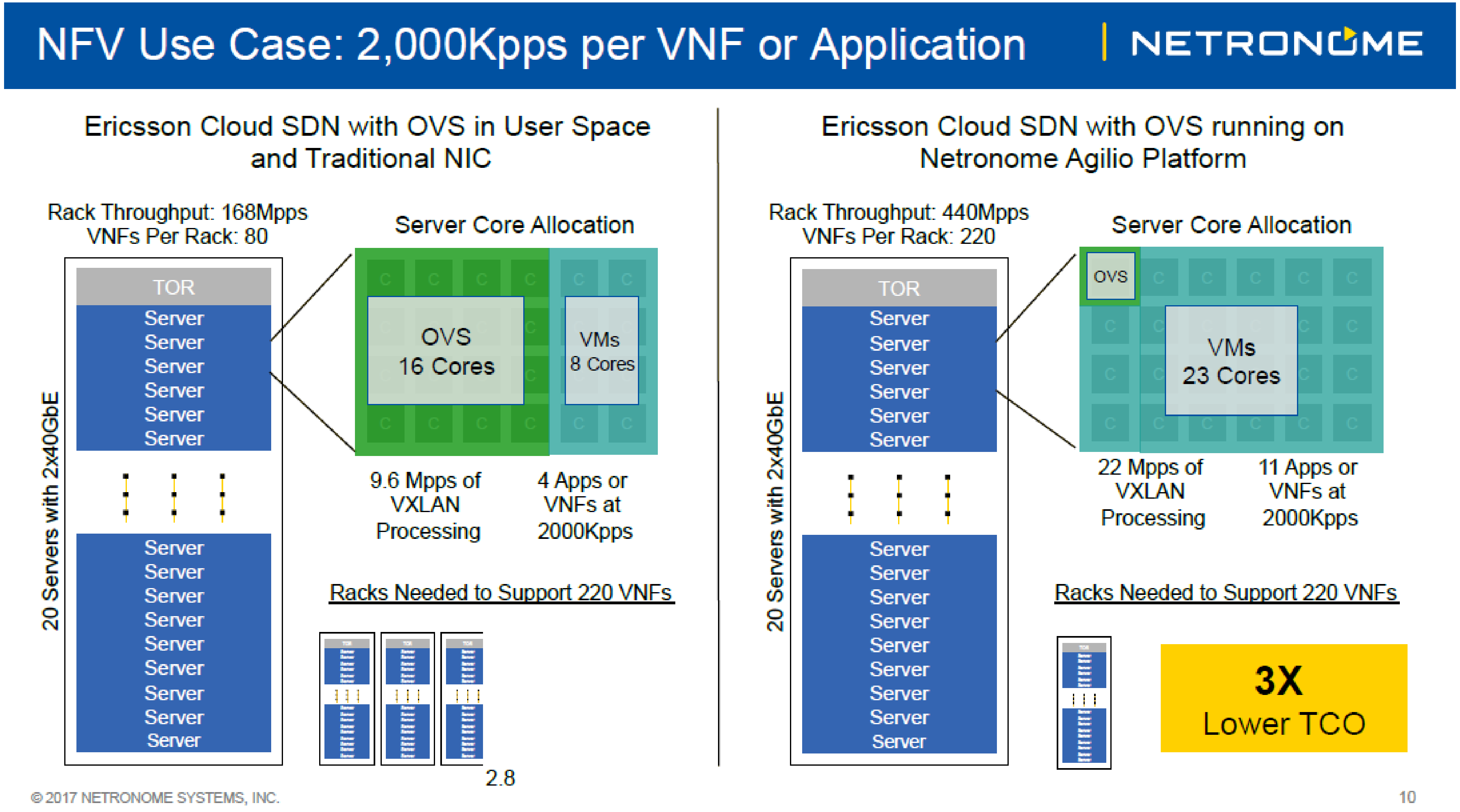

NETRONOME OVS ACCELERATION

Taking NFV to the next level of performance - Ericsson Cloud SDN & Netronome Agilio CX | Commercial in confidence | 2016-06-03 | Page 14

In collaboration with: NETRONOME

Source: https://www.slideshare.net/Netronome/ericsson-cloud-sdn-netronome-agilio-cx-taking-nfv-to-the-next-level-of-performance

Ericsson Cloud SDN & Netronome Agilio CX



NFV Use Case: 2,000Kpps per VNF or Application

NETRONOME

Ericsson Cloud SDN with OVS in User Space and Traditional NIC

Rack Throughput: 168Mpps
VNFs Per Rack: 80

Server Core Allocation

9.6 Mpps of VXLAN Processing

4 Apps or VNFs at 2000Kpps

Racks Needed to Support 220 VNFs

2.8

Ericsson Cloud SDN with OVS running on Netronome Agilio Platform

Rack Throughput: 440Mpps
VNFs Per Rack: 220

Server Core Allocation

22 Mpps of VXLAN Processing

11 Apps or VNFs at 2000Kpps

Racks Needed to Support 220 VNFs

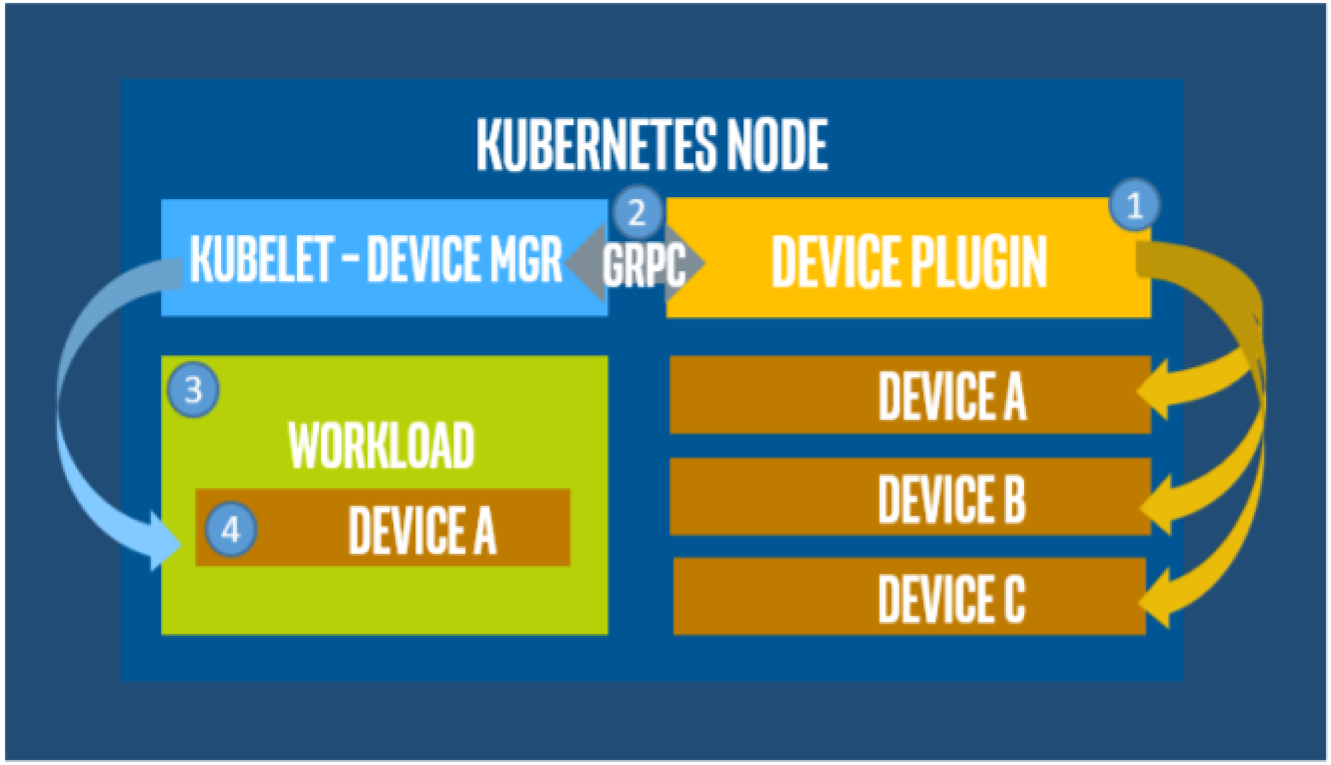
3X
Lower TCO

10

Source: https://www.slideshare.net/Netronome/ericsson-cloud-sdn-netronome-agilio-cx-taking-nfv-to-the-next-level-of-performance

加速卡資源管理

- 硬體供應商(Intel, Nvidia)已經提供許多硬體加速方案(GPU, FPGA, QAT)，這些硬體能針對特定任務提供加速運算功能，大幅節省CPU運算資源並提升效能。
- Kubernetes Device Plugin開發框架讓供應商能開發各自產品的Plugin並整合到Kubernetes的生態鏈中。



Source: https://builders.intel.com/docs/networkbuilders/intel-device-plugins-for-kubernetes-appnote.pdf

ITRI
Industrial Technology
Research Institute

Case Study:
O-RAN Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN

O-RAN.WG6.CAD-v02.01

Cloud Considerations –
Networking requirements

1. Support for Multiple Networking Interfaces

- Near-RT RIC, vO-CU, and vO-DU all depend on multiple network interfaces

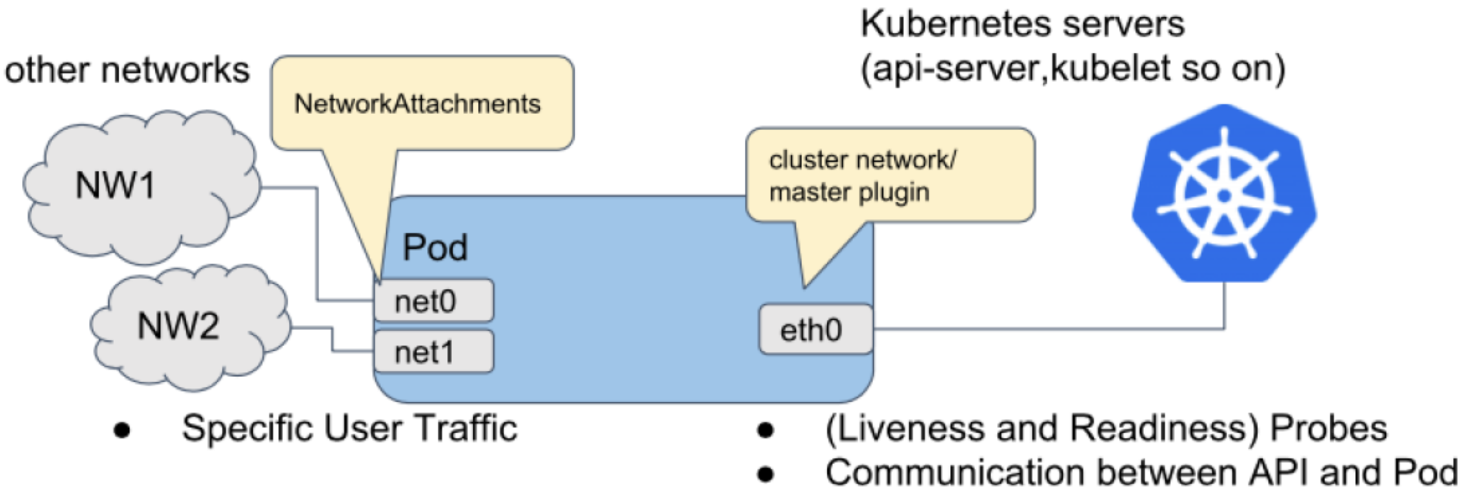


Figure 18: Illustration of the Network Interfaces Attached to a Pod, as Provisioned by Multus CNI

Cloud Considerations –
Networking requirements

2. Support for High Performance N-S Data Plane

RU-DU間要求高效能，但可能有多個vO-DUs共享一個NIC的情況，故需SRIOV或Pass-through (virtual switch不行, delay太高)

- The Fronthaul connection between the O-RU/RU and vO-DU requires high performance and low latency → so need high performance packets handling
- Multiple vO-DUs may share the same physical networking interface

➔ The following two items are needed:

1. Support for SR-IOV (assign SR-IOV NIC interfaces to the containers)
2. Support for PCI pass-through for direct access to the NIC by the container

Cloud Considerations –
Networking requirements

3. Support for High-Performance E-W Data Plane

- High-performance E-W data plane throughput is a requirement for near-RT RIC, vO-CU, and vO-DU 這些元件會有較大量東西向的流量，要高效能可使用dpdk vSwitch讓container/VM之間走userspace networking
- Commonly used option: virtual switch
- High performance option: DPDK-based virtual switch

Cloud Considerations – Networking requirements

4. Support for Service Function Chaining

- When the service requirement or flow direction needs to be changed, the SFC can easily implement it, instead of having to restart and reconfigure the services, networking configuration and Containers/VMs
- For container: e.g. https://networkservicemesh.io/

Cloud Considerations –
Assignment of Acceleration Resources

- The cloud platform needs to be able to assign the specified accelerator device to container instance or VM instance
- For example
  - some L1 protocols require an FFT algorithm (to compute the DFT) that could be implemented in an FPGA or GPU
  - vO-DU would need the PCI Pass-Through to assign the accelerator device to the vO-DU

Cloud Considerations –
Performance Feature Requirements

1. **Support for Pre-emptive Scheduling (➔Pre-emptive kernel)**

   因為有的app需要deterministic的回應時間，或有的wireless app需要priority-based的環境

   - for an application to get deterministic response times for events, interrupts and other reasons
   - to support high throughput, multiple accesses and low latency, some wireless applications need the priority-based OS environment

2. **Support for Node Feature Discovery (NFD)**

   Cloud platform要能發現node有哪些硬體功能並用labels標記，則NF description裡就可用labels來表達其硬體需求

   - NFD: the cloud platform should support to discover the hardware capabilities on each node and advertise it via labels vs. nodes, so allows O-RAN Cloudified NFs' descriptions to have hardware requirements via labels

Cloud Considerations –
Performance Feature Requirements

3. Support for CPU Affinity and Isolation

- The vO-DU, vO-CU and even the near-RT RIC are performance sensitive and require the ability to consume a large amount of CPU cycles to work correctly
- For container: CMK for Kubernetes

4. Support for Dynamic HugePages Allocation

- When an application requires high performance and performance determinism, the reduction of paging is very helpful 減少paging對效能的拖累
  - ☞vO-DU, vO-CU and even near-RT RIC
- For container: HugePages configs in pod yaml

Cloud Considerations –
Performance Feature Requirements

5. Support for Topology Manager

Managing the NUMA topology to ensure:

Container、application所使用的devices，是(連)在同一個NUMA region

- ● the placement of specified containers on cores which are on the same NUMA region
- ● the devices which the application uses are also connected to the same NUMA region

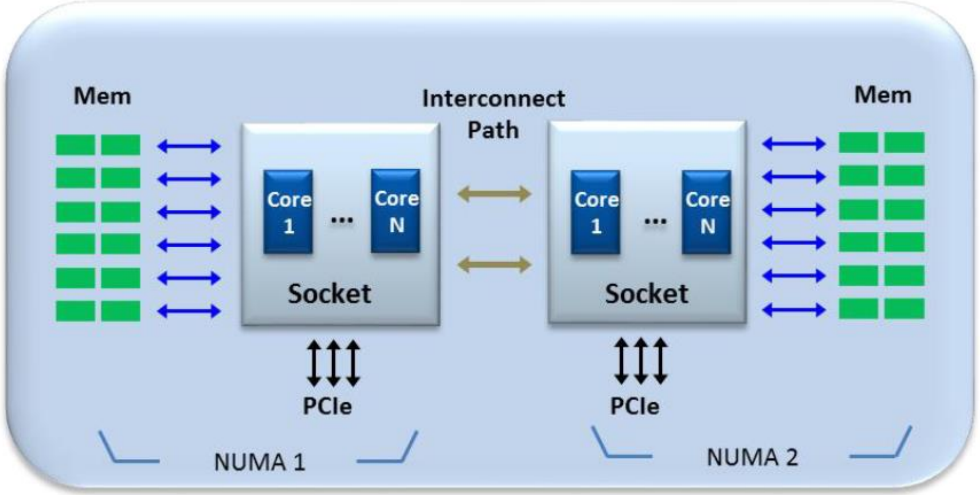


Figure 20: Example Illustration of Two NUMA Regions

Cloud Considerations –
Performance Feature Requirements

6. Support for Scale In/Out

視當時需求scale in/out，特別像是O-CU需因應網路負載做調整

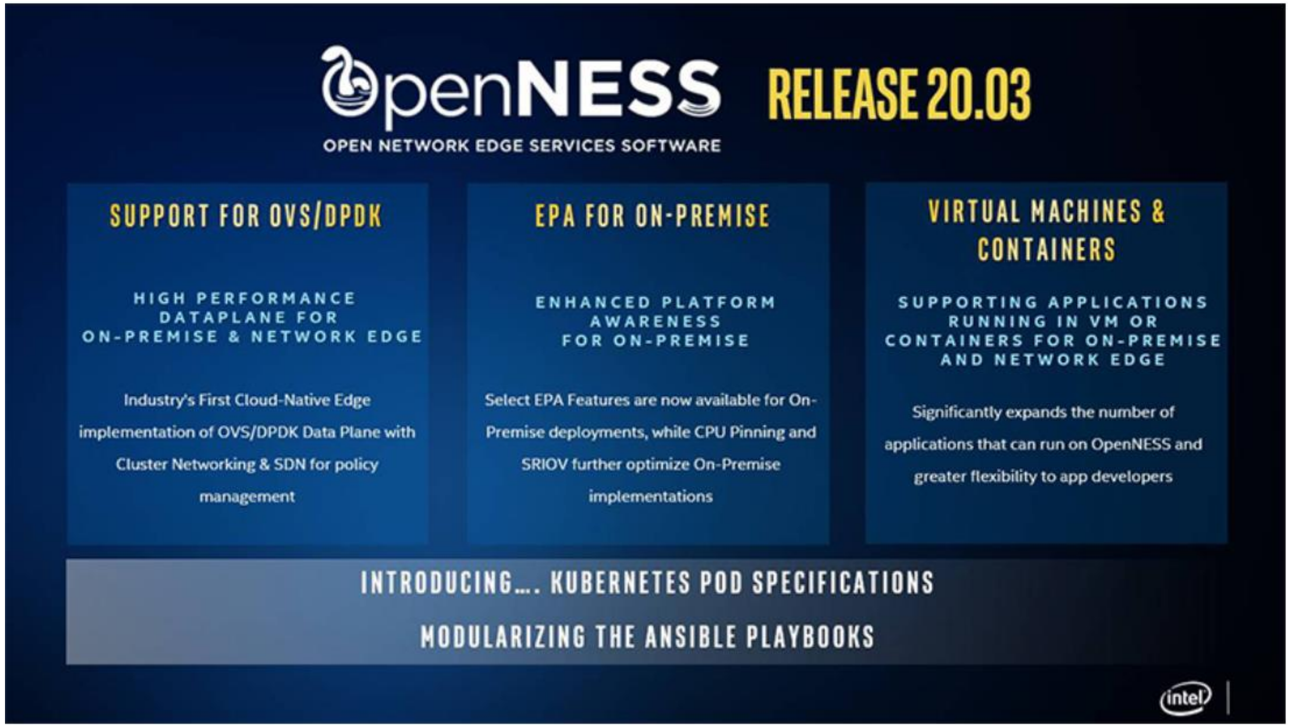
- if demand is changing dynamically, especially for the O-CU, the service needs to be scaled in/out according to service requirements such as subscriber quantity

7. Support for Device Plugin

- support the ability to discover, advertise, schedule and manage devices such as SR-IOV, GPU, and FPGA
- For container: Kubernetes provides a Device Plugin Framework that you can use to advertise system hardware resources to the Kubelet

Intel OpenNESS Edge Platform

- **OpenNESS是Intel推出的開源NFV平台，根據NFV高效能的需求，整合了軟硬體加速技術，並提供自動化部署的服務，可節省開發團隊整合眾多開源軟體的時間與精力。**



OpenNESS RELEASE 20.03

OPEN NETWORK EDGE SERVICES SOFTWARE

SUPPORT FOR OVS/DPDK

HIGH PERFORMANCE DATAPLANE FOR ON-PREMISE & NETWORK EDGE

Industry's First Cloud-Native Edge implementation of OVS/DPDK Data Plane with Cluster Networking & SDN for policy management

EPA FOR ON-PREMISE

ENHANCED PLATFORM AWARENESS FOR ON-PREMISE

Select EPA Features are now available for On-Premise deployments, while CPU Pinning and SRIOV further optimize On-Premise implementations

VIRTUAL MACHINES & CONTAINERS

SUPPORTING APPLICATIONS RUNNING IN VM OR CONTAINERS FOR ON-PREMISE AND NETWORK EDGE

Significantly expands the number of applications that can run on OpenNESS and greater flexibility to app developers

INTRODUCING…. KUBERNETES POD SPECIFICATIONS

MODULARIZING THE ANSIBLE PLAYBOOKS

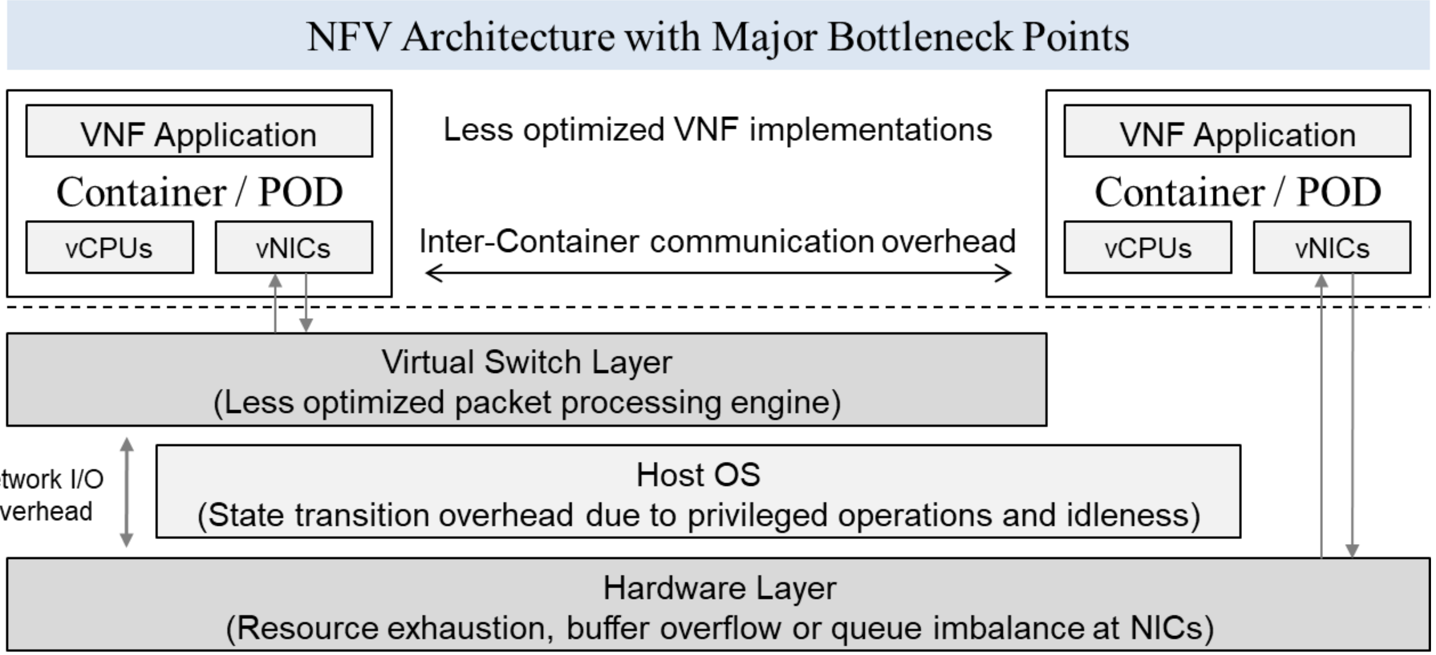
	Kubespray	OpenNESS
Open Source	V	V
Multus CNI	V	V
SRIOV CNI	X	V
DPDK Device Plugin	X	V
Accelerator Device Plugin	X	V
NFD Plugin	X	V
CPU Management Kit	X	V
Hugepage Allocation	V	V
License	Apache 2.0	Apache 2.0

NFV面臨的問題與挑戰

- 從雲到端、從核網到基站，需要更輕量化的虛擬化技術
  - 虛擬機(VM) => 容器(Container)
  - OpenStack(虛擬機管理平台) => Kubernetes(容器管理平台)
  - 雲原生 (Cloud Native) 的開發概念
- 使用x86架構取代專用網路晶片，效能是否能滿足需求
  - NFV效能優化技術
    - ☞CPU與NUMA Node
    - ☞Hugepages
    - ☞網路加速技術(DPDK, SRIOV, SmartNIC)
    - ☞加速卡資源管理
  - Case Study: O-RAN虛擬化技術需求
- NFV架構下，需要更好的效能監控、分析與管理技術
  - eBPF技術解說

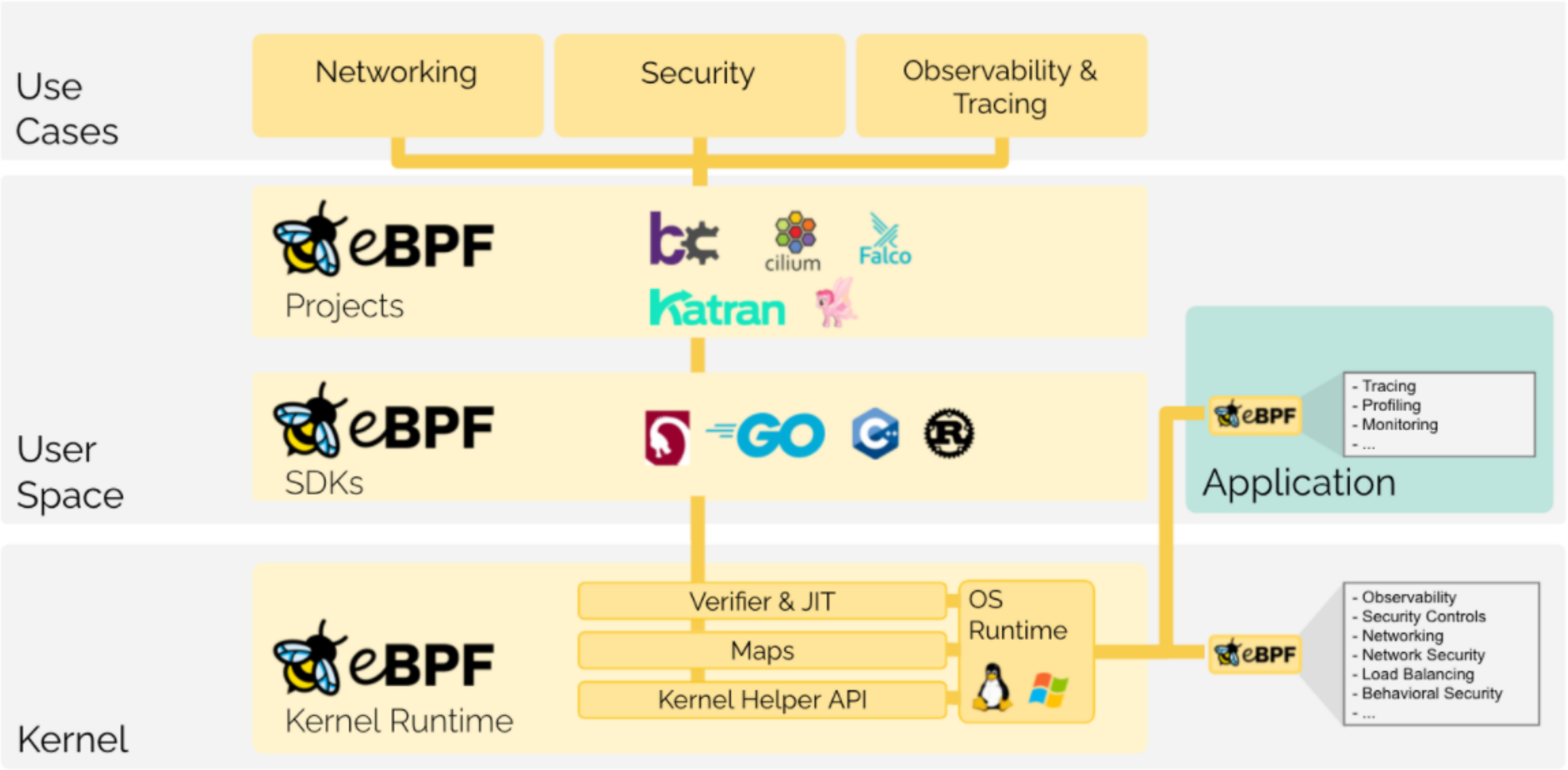
Performance Issue in NFV Environment

- **Traffic hidden from monitoring**
  - **Cisco estimates that about 73% of data-center traffic will come from within the data center by 2019—most of this traffic is virtual machine to virtual machine (VM to VM) communication**
  - **The inability to isolate production from monitoring traffic**
- **No Solution to address performance uncertainty in NFV**
  - **Where is the major bottleneck point of an NFV system**


NFV Architecture with Major Bottleneck Points
VNF Application
Container / POD
vCPUs
vNICs
Less optimized VNF implementations
Inter-Container communication overhead
VNF Application
Container / POD
vCPUs
vNICs
Virtual Switch Layer
(Less optimized packet processing engine)
Network I/O Overhead
Host OS
(State transition overhead due to privileged operations and idleness)
Hardware Layer
(Resource exhaustion, buffer overflow or queue imbalance at NICs)

eBPF介紹

- **eBPF 是一項革命性的技術，起源於 Linux 內核，可以在作業系統內核中運行沙盒程序。 它用於安全有效地擴展內核的功能，而無需更改內核源代碼或加載內核模塊。**

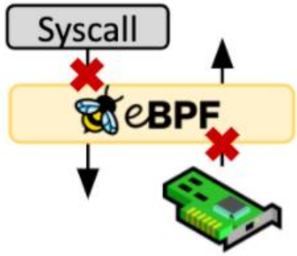


Source: https://ebpf.io/

eBPF應用情境

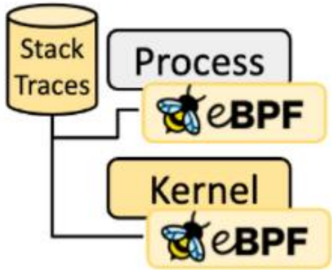
- **eBPF 被廣泛用於各種情境：預防性應用程序和容器運行時安全實施、幫助應用程序開發人員跟踪應用程序、在雲原生環境中提供高性能網路和負載平衡、以低開銷提取細粒度的安全可觀察性數據。**

1. Security



eBPF能查看所有System Call與網路操作，並結合控制功能可以打造更好的安全管理系統。

2. Tracing & Profiling

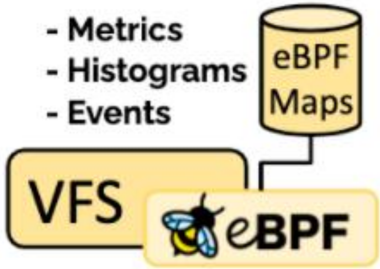


eBPF程序能掛載探測點到系統內核與應用程序，因此可以針對應用程序與系統運作行為提供前所未有的可見度，透過有效的方式提取有意義的數據，可進一步解決系統效能問題。

3. Networking

eBPF 的可編程性允許添加額外的Protocol Parser並輕鬆編程任何轉發邏輯以滿足不斷變化的需求，而無需更改Linux 內核的數據包處理邏輯。

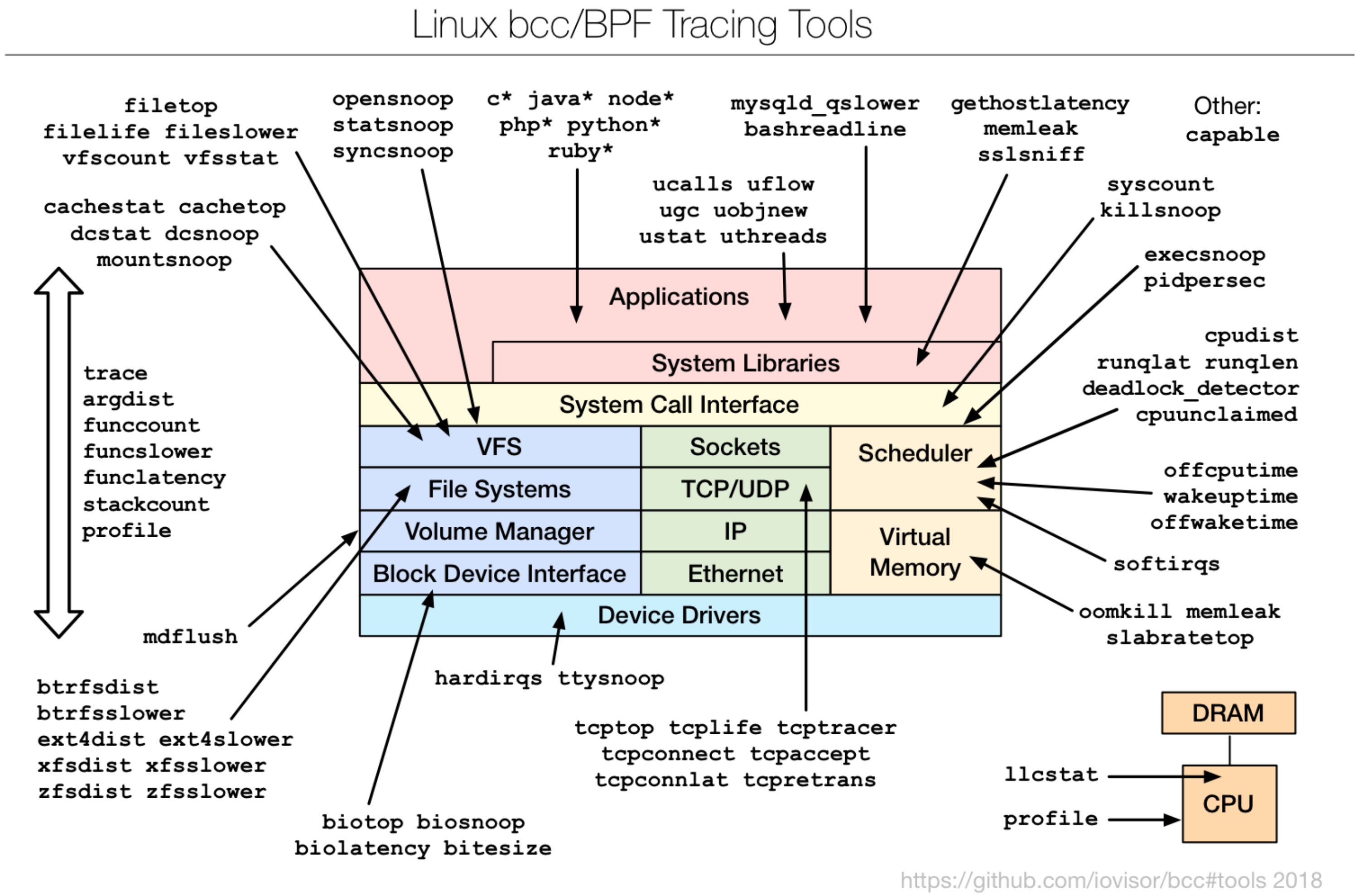
4. Observability & Monitoring



eBPF 不依賴於作業系統提供的監控數據與指令，而是支持客製化的數據收集，僅蒐集內核與應用程序因特定事件所生成的必要資訊，可顯著降低整體系統開銷

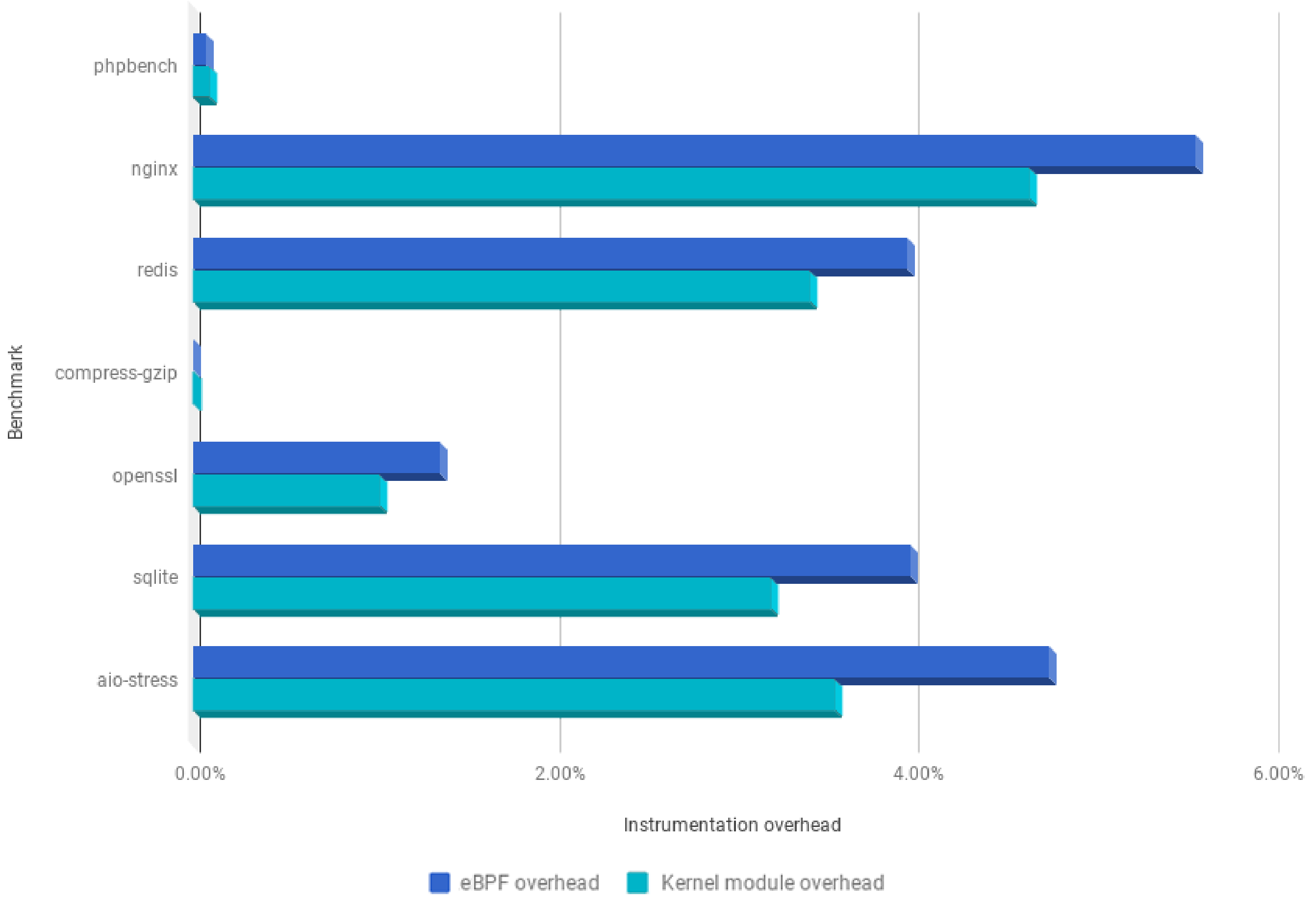
Source: https://ebpf.io/

eBPF Tracing Point



Source: https://hackmd.io/@sysprog/linux-ebpf

Low overhead tracing with eBPF



Source: https://sysdig.com/blog/sysdig-and-falco-now-powered-by-ebpf/

總結

- NFV技術從2013年演進至今已經相當成熟，並且有許多成功案例證明NFV是商用可行的技術。
- 導入NFV技術並不是一件簡單的事，需掌握效能優化的技術，以及自動化管理維運的思維，所幸目前已經有許多開源軟體可供整合，可大幅降低導入NFV的技術門檻。
- NFV系統的效能優化只是第一步，後續更重要的是如何做到自動化的管理維運，可以先從系統的監控著手，接下來從監控數據中進行分析，進而做到自動化的除錯、優化、與管理，導入AI進行系統分析與管理已經是現在進行式。
- 本課程是根據講師在工研院開發5G NFV平台所應用到的技術做個初步介紹，後續如果有任何問題需要討論或交流，可與講師聯絡，聯絡資訊如下：
  - 工研院資通所技術經理 李育緯
  - Email: rayinlee@itri.org.tw