CASE STUDY

vBNG Reference Application

# PATH TO A CLOUD-NATIVE BROADBAND NETWORK GATEWAY (BNG) WITH KUBERNETES*


**Authors**

**Nemanja Marjanovic**
Network Software Engineer

**Louise Daly**
Cloud Software Engineer


## Executive Summary


The latency and performance of containerized virtual network functions (VNFs) can be significantly improved when VNFs can better utilize the underlying infrastructure in a network functions virtualization (NFV) environment. Advancing this objective, Intel has demonstrated how an open-source, container orchestration system, called Kubernetes*, can be used to ensure VNFs are able to take advantage of specialized hardware platform features, such as single root input/output virtualization (SR-IOV) and CPU pinning. This case study details this effort based on a virtual broadband network gateway (vBNG) use case that employs Kubernetes on an Intel® hardware platform to deploy high-performant VNFs that benefit from advanced infrastructure management and optimized resource allocation. This vBNG reference application enables various capabilities, including the scaling of capacity (up/down) based on a time-of-day power profile, which can help lower operating expenditures (Op-Ex).


## Migration to NFV-Based Network Infrastructure

Modern telecom networks contain an ever-increasing variety of proprietary hardware, making the launch of new services more complex. New services often require network reconfiguration and on-site installation of new equipment, which in turn requires additional floor space, power, and trained maintenance staff. Service providers wishing to offer dynamic services will also be hindered by legacy networks built using fixed-function appliances, which can be difficult to maintain and evolve, thus impeding innovation.

NFV-based solutions are proven to help telco operators deliver core network services (e.g., BNG, IMS, DPI, IDS, CDNs, and EPC)[1] with greater speed and agility, and at a lower cost point while supporting traditional telco services (e.g., x-Play bundles), including broadband Internet access, voice, and TV services. Operators will be able to monetise existing services as they gradually deploy brownfield networks that provision new value-added services, foster innovation, and help drive higher average revenue per user (ARPU).

The adoption of NFV will drive a gradual phase-out of hardware-based appliances, facilitating the use of a software orchestrator to “hot-swap” a service (i.e., card vs. VNF) on-the-fly, thus minimizing the impact new services launch on the network. In an NFV environment, orchestrators, such as Kubernetes, provide an easy deployment model for efficiently allocating network applications and meeting stringent high-availability, carrier-grade network requirements.

## BNG Use Case

A BNG, formerly referred to as broadband remote access server (BRAS or BBRAS), is the access point for broadband subscribers, through which they connect to their service provider broadband network. However, traditional network-edge, hardware-based BNGs are difficult to scale elastically and cannot efficiently support the rapidly increasing global IP traffic network stemming from exponentially growing subscriber demand.

The challenge for telco independent software vendors (ISVs) is to build a performant, deterministic, and scalable BNG reference application in a containerized environment (virtual BNG or vBNG) that makes efficient use of disparate hardware infrastructure. As hardware evolves and network traffic increases, vBNGs must be capable of supporting even higher throughput rates. Higher levels of performance and resource utilization can be gained through specialized features provided by Kubernetes (K8s) resource management, such as CPU pinning, non-uniform memory access (NUMA) awareness, and device allocations.

BNGs are typically provisioned for average peak performance or dimensioned for busy periods. Scalable BNGs present an opportunity to scale down processing resources to achieve an energy efficient state during non-busy periods.

### BNG as a Service

When a connection is established between the BNG and customer premises equipment (CPE), subscribers can access the broadband services provided by communications service providers (CoSPs) or Internet service providers (ISPs), as shown in Figure 1. The BNG/edge router establishes and manages subscriber sessions, aggregates traffic from the various sessions of an access network, and routes it to the CoSP core network. Subscribers connect to the BNG, which manages their access by performing subscriber authentication, authorization, accounting (AAA), and other management functions, such as:

- IP address assignment
- Security (access control lists (ACLs))
- Policy management
- Quality of service (QoS) and traffic management

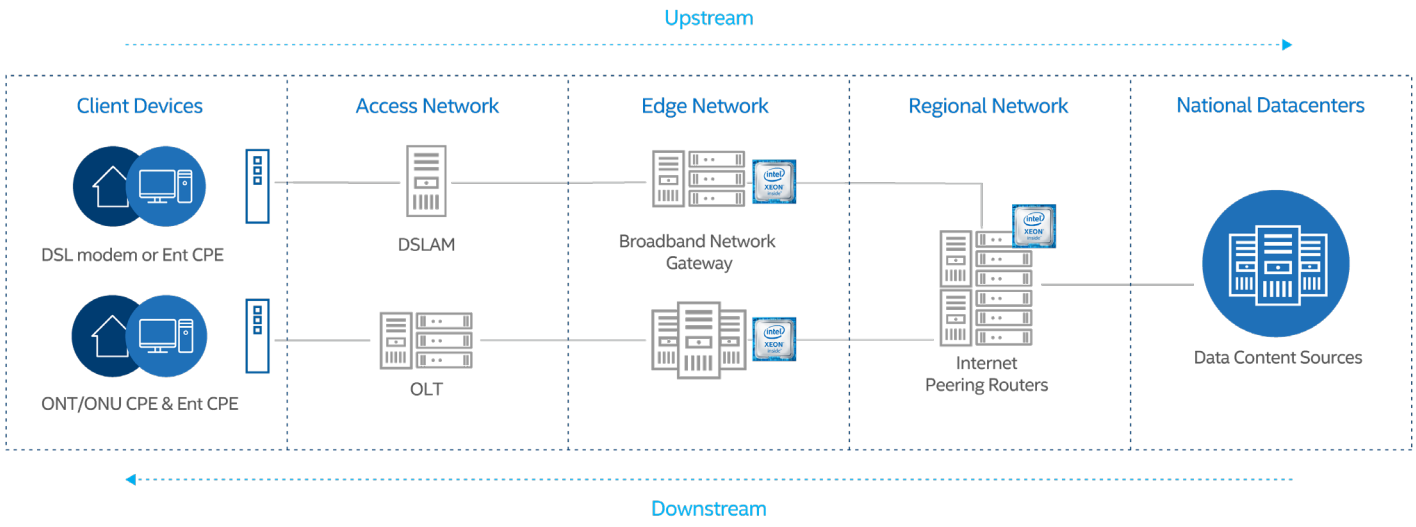


Figure 1. Example of a Network Connecting Clients to a Data Center

For more information on the BNG, please refer to the Re-Architecting the Broadband Network Gateway (BNG) in a NFV and Cloud Native World paper.[2]

Containerization and container orchestration, deployed on a BNG at the edge of a telco operator's network, can provide the scalability and efficiency required for portability and fast deployment times, as well as deliver flexibility when scaling the number of BNGs with the number of attached subscribers. BNGS can offer an ideal software-defined networking (SDN) and NFV deployment model for CoSPs/ISPs and cloud service providers (CSPs).

In order for a containerised and orchestratable vBNG to be efficiently deployed on a general-purpose server, the platform's available I/O and compute resources must be discoverable and fully utilised, and this can be achieved with Kubernetes (K8s). A good instantiation is Intel's containerized vBNG reference application, which uses Intel's CPU Manager for Kubernetes (CMK)[3] to ensure maximal performance.
The design provides basic support for CPU pinning and isolation allowing the platform to execute latency intensive workloads on dedicated CPU cores without interference from other running processes. Physical platform resource isolation is also employed to fully utilize the platform's I/O, thereby ensuring network I/O performance and determinism through the use of a SR-IOV network device plugin.[4] The plugin enables the discovery of SR-IOV network virtual functions (VFs) in a K8s host, and helps increase network I/O performance by allowing direct hardware access from vBNG pods (aka. group of containers that are deployed together on the same host).

### Kubernetes and the vBNG

Kubernetes (K8s) is an open-source container manager/orchestrator that automates the deployment, scaling, and operational functions associated with application containers across a cluster of physical hosts. It is a widely-used orchestration platform option for building and running modern cloud infrastructure with a large number of interworking containers. The nature of containers allows for the deployment of microservices, whereby each part of a service is decoupled into a separate container to provide modular development, easy deployment, and scaling models. With that being said, the Intel vBNG reference application is deployed with the decoupled approach with a single vBNG pod, consisting of uplink and downlink containers that manage the traffic loads in each direction.

K8s is becoming a de facto standard for container orchestration, offering resource management capabilities such as Intel's CMK and SR-IOV device plugin. In addition to all the benefits that come with containers, K8s presents an ideal system to deploy a performant, deterministic, and scalable vBNG that makes efficient use of disparate hardware infrastructure.

## vBNG and Kubernetes Architecture

The following sections describe the K8s architecture, supporting high-throughput, latency-sensitive telecommunications gateway functions with the vBNG as a reference application.

### Kubernetes Master

The K8s master node is responsible for maintaining a cluster's desired state. The Kubernetes cluster itself is made up of a master node and a set of worker nodes. When interacting with K8s, such as using the kubectl command-line interface, the user is in fact communicating with the cluster's master. The master refers to a collection of processes managing the cluster state. Typically, these processes are all run on a single node in the cluster, and this node is referred to as the master. The master can also be replicated for availability and redundancy. In Figure 2, all K8s pods are deployed from the master via the kubectl create -f command. The K8s master controls each node. For an in-depth overview of each K8s component in Figure 2, please refer to Kubernetes components website.[5]

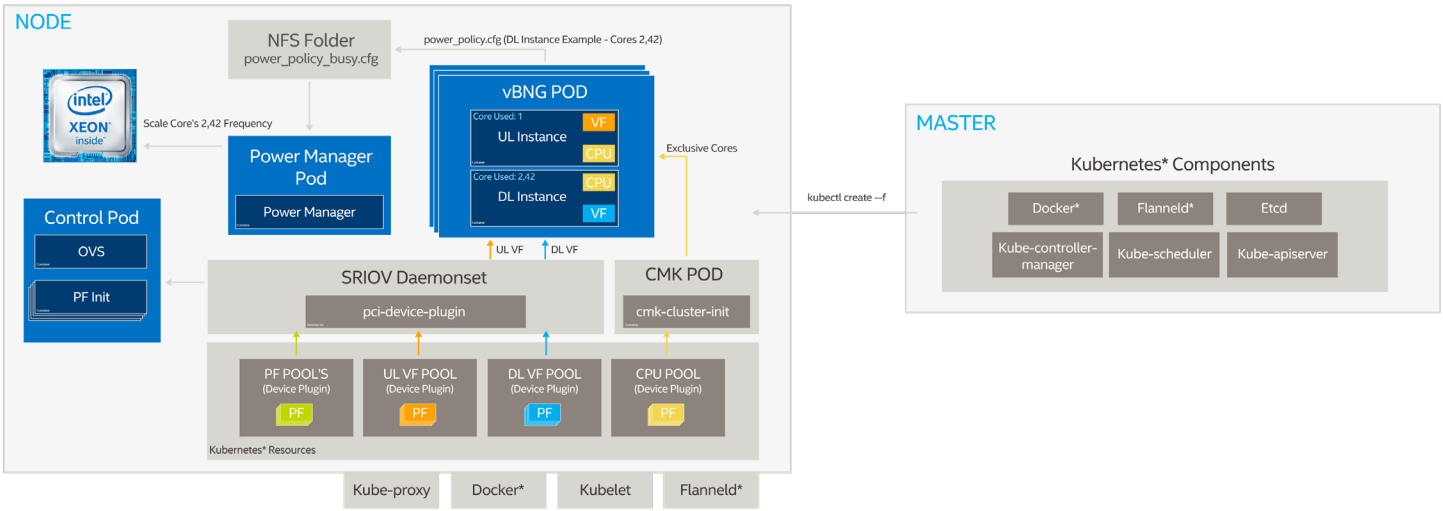


Figure 2. Master and Node Architecture

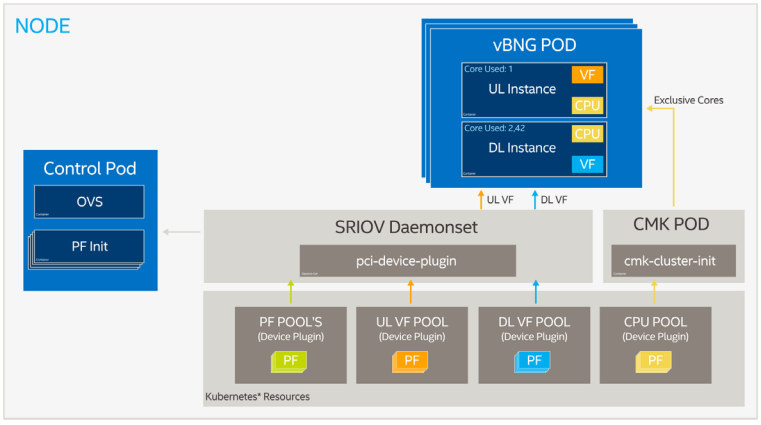


Figure 3. vBNG Node Architecture

### Kubernetes vBNG Node

The nodes in a cluster are machines (e.g., virtual machines (VMs) and physical servers) that run applications and cloud workloads, in this case the vBNG data plane reference application, as shown in Figure 3.

### CPU Manager for Kubernetes (CMK)

The CMK delivers predictable network performance and workload prioritization, a key challenge of a virtualized platform environment. For example, performance can suffer when a Linux* scheduler runs an application across multiple cores to accommodate other applications running on the system. The CMK avoids such scenarios through CPU pinning, which maximizes cache utilization, eliminates operating system thread scheduling overhead, and coordinates network I/O and their assigned platform resources. CPU pinning creates an affinity between an application and a designated CPU core.

A related feature is CPU isolation, which blocks other applications from running on the same core. This is effective in isolating VNFs from "noisy neighbours", referring to other applications running on the same core and consuming significant CPU cycles, hence impacting the performance of the priority workload. To address these issues from a K8s perspective, Intel developed CMK, an open-source project that introduces additional CPU utilization optimization capabilities to K8s.

CMK is a tool for managing core pinning and core isolation in K8s. Without the use of CMK, the kernel task scheduler will treat all CPUs as available for scheduling process threads and regularly preempts executing process threads to give CPU cycles to other applications. This non-deterministic behavior makes it unsuitable for latency-sensitive workloads and meeting service level agreements (SLAs).

The exclusive pool within the CMK assigns entire physical cores exclusively to the requesting container, meaning that no other container/process is able to run on the cores in this pool. Along with the exclusive pool, CMK allows for the creation of shared and infra pools, which can run complimentary workloads. From a vBNG perspective, vBNG pods utilize the capability to request cores from the exclusive pool, thus ensuring no other processes interfere with the vBNG's workloads.

### SR-IOV Device Plugin

SR-IOV is a technology for isolating PCI Express* resources, allowing multiple virtual environments to share a single PCI Express hardware device and physical functions (PFs) by offering multiple VFs that appear as separate PCI Express devices, a capability not natively available in K8s.

The SR-IOV network device plugin enables the discovery and advertising of SR-IOV VFs in a K8s host. SR-IOV VFs in turn allow direct hardware access from multiple K8s pods, increasing network I/O performance and making it possible to run fast user space packet processing workloads, such as applications using the Data Plane Development Kit (DPDK).

In K8s, the orchestration of SR-IOV network resources is handled by two components:

- The SR-IOV device plugin manages SR-IOV VFs as an extended resource, which allows containers to request SR-IOV VFs as a resource and enables the K8s scheduler to accurately place pods based on their requests.
- The SR-IOV container network interface (CNI) plugin then enables the pod to attach directly to the allocated VF. For higher network performance, the DPDK mode in the SR-IOV CNI plugin allows the container to bind the VF to a DPDK driver, which the containerized DPDK-based vBNG reference application fully utilizes to ensure maximum network I/O performance.

### Control Pod

The control pod is responsible for configuring the infrastructure and hardware to prepare the system for application deployment. As illustrated in Figure 4, the control pod consists of one or more instances of the DPDK sample application pf_init, which brings up PFs and sets specific filtering rules in the Intel® Ethernet Network Adapter XXV710 for the vBNG data and control plane application instances.

The Intel® Ethernet Network Adapter XXV710 is a network interface card (NIC) that supports Dynamic Device Personalization (DDP),[6] which allows dynamic reconfiguration of the packet processing pipeline within the NIC to meet specific use case protocol stack needs on demand. This can be done by adding new packet processing pipeline configuration profiles to the network adapter at runtime without the need to reset or reboot the server.

For the vBNG application, the DDP profile enables the NIC to route packets to specific data plane VFs and queues based on the unique Point-to-Point Protocol over Ethernet PPPoE header-fields, which enabled the differentiation between control plane and data plane traffic types.

The connection between the CPE and the BNG is via a PPPoE or Internet Protocol over an Ethernet (IPoE) link with each subscriber having a unique IP or PPPoE session ID.

For control plane traffic, like PPPoE session setup or PPPoE link control packets, the BNG data plane must identify and forward these packets to the control plane for processing. In a traditional BNG, the control plane and the data plane are co-located, and a local software queue is used to move packets between them. With the advent of Control and User Plane Separation (CUPS), the control plane and the corresponding data plane are most likely located in different physical locations in the network. In this case, the BNG data plane needs to forward all control packets to the control plane by generating a physical link to forward them. With the Intel® Ethernet Network Adapter XXV710, it is possible to identify these control packets and forward them to separate VFs and queues, relieving the data plane of this task. The DDP PPPoE profile enables the Intel® NIC to recognize these control plane packets and forward them separately to the control plane.

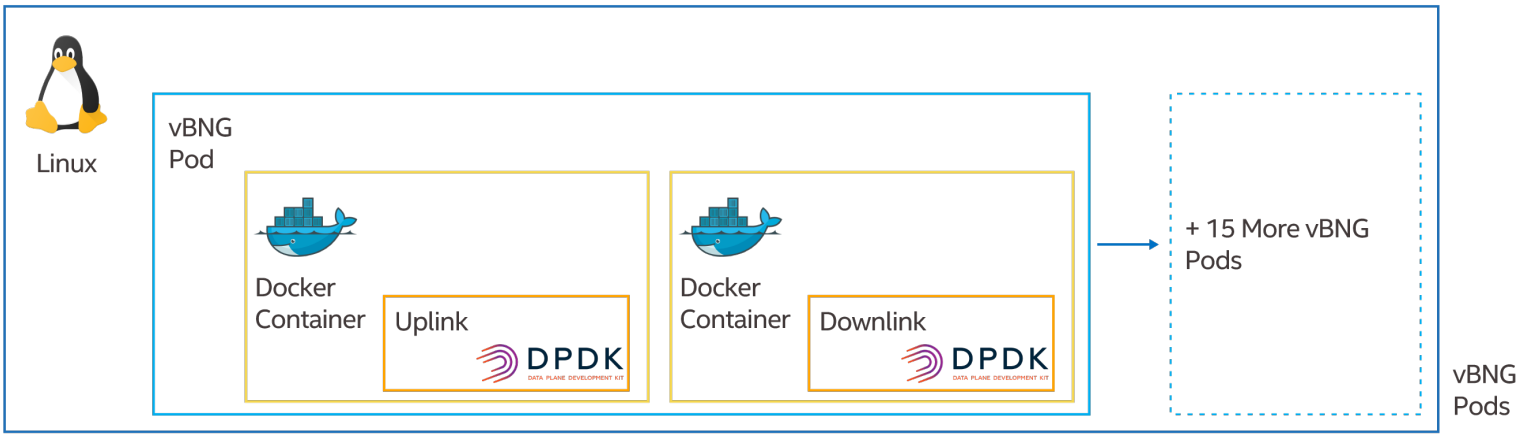


Figure 4. vBNG Node Pods

The upstream vBNG container processing blocks in Figure 5 represent the flow of traffic from the subscriber home router/CPE to the CoSP/ISP network, which is the configuration incorporated in the uplink container of the vBNG pod. For more information on Figure 5 and Figure 6 reference pipelines architecture, refer to the white paper, "Re-Architecting the Broadband Network Gateway (BNG) in a NFV and Cloud Native World."[7]

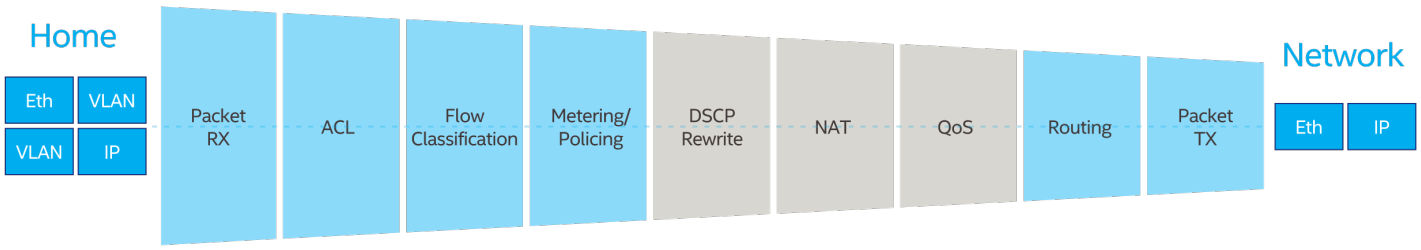


Figure 5. Uplink Processing Blocks vBNG

The downstream vBNG container processing blocks in Figure 6 represent the flow of data traffic from the ISP network to subscribers' homes. It is responsible for managing and scheduling the traffic of all the users attached to the gateway. The downstream function ensures the best use of the bandwidth and resources available to maximize users' quality of experience (QoE) based on user tariff class and traffic priorities. The average packet size of the downstream link is increasing; due in part due to IP video traffic comprising 82 percent of all global IP traffic by 2022, up from 75 percent in 2017.[8] Overall, the goal of the service provider is to ensure all subscribers are receiving the services to the highest standard while maximizing the utilization of the infrastructure in place.

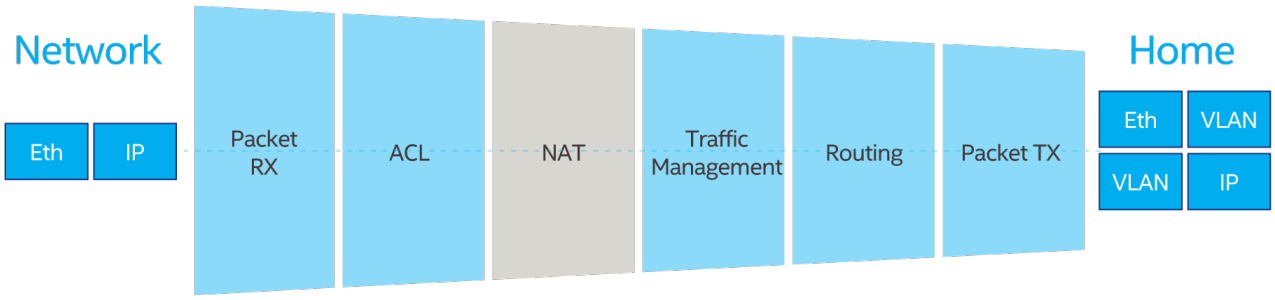


Figure 6. Downlink Processing Blocks vBNG

### Deployment Flow

Before any pods are deployed, the required plugins for resource assignment must be initialized. PCI® device addresses must be populated according to the NUMA node of deployment in /etc/pcidp/config.json with the downlink and uplink VFs, and PF pools. Figure 7 shows how users can create a resource config for each resource pool by specifying the resource name, a list of device PCI addresses, and resource type: PFs or VFs ("sriovMode": true).

If "sriovMode": true is given for a resource config, the plugin will search for available VFs for the PFs listed in "rootDevices" and export the discovered VFs as a single, allocatable, extended resource pool. Otherwise, the plugin will export the root devices themselves as the allocatable extended resources. Due to setup requirements, the Intel vBNG reference application sets sriovMode to false, allowing the VF devices themselves to be allocated to specific uplink and downlink pools consumed by the vBNG instance. Similarly, PFs have dedicated pools used by the control pods.


| Field | Required | Description | Type - Accepted values | Example |
|---|---|---|---|---|
| "resourceName" | Yes | Endpoint resource name | `string` - must be unique and should not contain special characters | `"sriov_net_A"` |
| "rootDevices" | Yes | List of PCI address for a resource pool | A list of `string` - in sysfs pci address format | `["02:00.0", "02:00.2"]` |
| "sriovMode" | No | Whether the root devices are SRIOV capable or not | `bool` - true OR false[default] | `true` |
| "deviceType" | No | Device driver type | `string` - "netdevice"\|"uio"\|"vfio" | `"netdevice"` |

Figure 7. SR-IOV Device Plugin Configuration

The SR-IOV device plugin will then export the configured device allocations to the container's environment variables to be consumed by the control and vBNG pods. Each of the vBNG instances/pods requests a single PCI device from the matching uplink or downlink configuration pool, depending on whether the container implements an uplink or downlink processing pipeline. Control pods request a PCI device corresponding to its NUMA node allocated from the PF pools.

Assuming the GRUB isolcpus boot option has been set, the CMK pod can be created, which in turn creates three pools:

- Exclusive pool
- Shared pool
- Infra pool

The exclusive pool does not allow other processes to run on the cores assigned to this pool, whereas the shared and infra pools are shared pools that can run complimentary workloads. The vBNG reference application pods request cores from the exclusive pool, as they need to ensure no other processes interfere with the vBNG workloads. Once the necessary plugins have been initialized and the resources are available for consumption, the control pod can be deployed.

Multiple containers running the pf_init DPDK-based application request ports from the SR-IOV device plugin, used to initialize PF's (one pf_init instance per PF). The containers apply a specific filtering rule to the Intel® Ethernet Network Adapter XXV710 for their assigned port, enabling the NIC to route packets to specific VFs. Once the infrastructure and the resource assignment capabilities are fully functional, the vBNG pods can be deployed to run their respective workloads. On initialization, each vBNG container requests VFs from the SR-IOV device plugin, and CPU cores from the CMK enable each of the containers in the pod to run their respective workload.

### vBNG Power Management

Once all the required K8s resources are assigned to the vBNG pods and are fully operational, each vBNG container on startup writes its own power policy to the shared network file system (NFS) directory on the node (power_policy.cfg), which is based on time of day profiles containing the name of the vBNG instance, vBNG policy, and the cores the vBNG is running on.

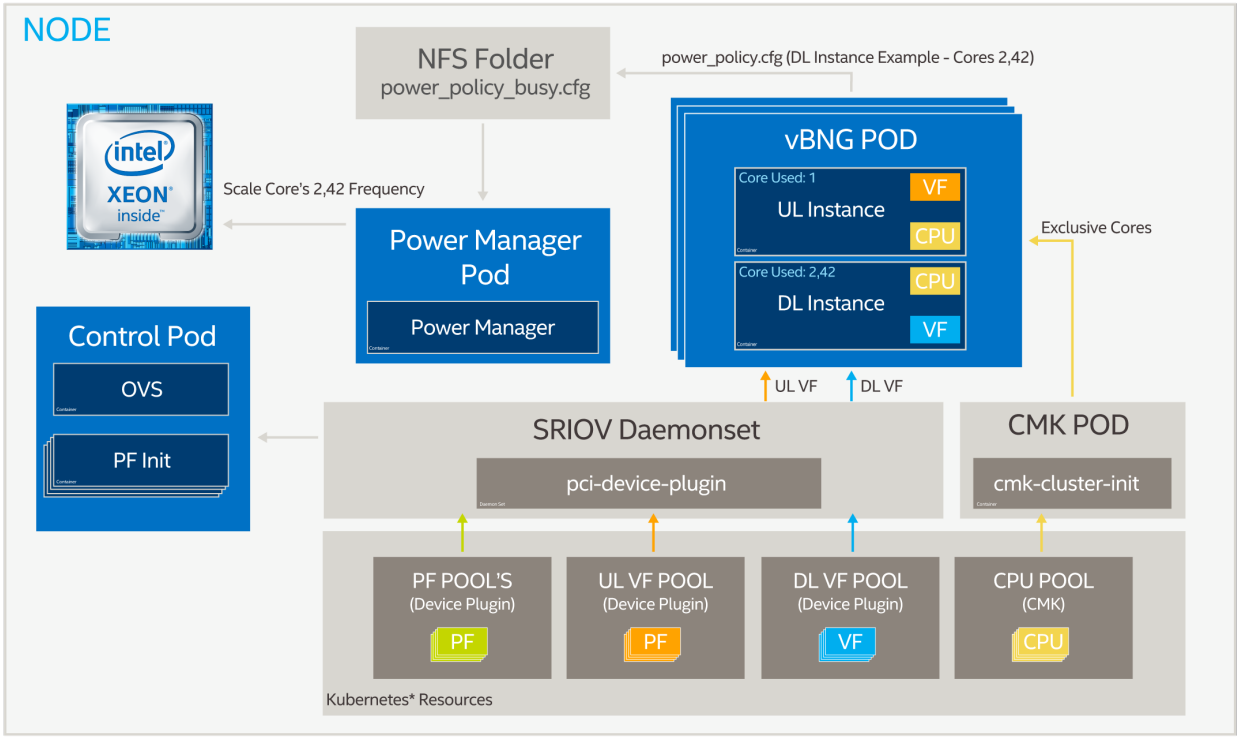


Figure 8. vBNG Node Architecture with Power Management Capabilities

A busy time of day profile example for a downlink instance is shown in Figure 8. The power manager pod reads each of the power policies specified by the vBNG pods and accordingly sets the frequency of the cores assigned to the specific vBNG instance. In the case of the power_policy_busy.cfg demonstrated in Figure 8 the frequency of the core (2,42) specified by this particular vBNG downlink container is set to max P1 for that platform (e.g., 2.5 GHz). For more information on P-States, please refer to the website, Power Management States. The capability to reduce power consumption is also an option with the use of power_policy_quiet.cfg, which will reduce the frequency of the core to the minimum P-State available on that platform, which is further investigated in the vBNG Power Management paper.

```
{"policy": {
  "name": "  vbng-dl-0-0  ",
  "command": "command_type",
  "policy_type": "TIME",
  "busy_hours":[ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 ],
  "quiet_hours":[ ],
  "core_list":[2,42]
}}
```

Figure 9. Busy Hours Power Policy for DL vBNG

## Performance

Performance measurements on the vBNG pipeline block, shown in Figure 10, were taken on a single socket server with Intel® Hyper-Threading Technology (Intel® HT Technology) enabled with all vBNG instances operating on the power_policy_busy.cfg, which as described, sets all vBNG core to max P1. The number of vBNG instances were scaled using Kubernetes, deploying from one to eight instances using the 8:1 downlink:uplink traffic ratio. The same traffic profile was applied to all vBNG instances (4,000 flows, downlink packet size=512B, uplink packet size=128B), and all the cumulative throughput (downlink+uplink) across all instances was measured. The optional grey blocks were not enabled. Figure 11 features performance testing by Intel in April 2019. This demonstrates the throughput of an Intel® Xeon® processor-based server (dual socket) running one through sixteen vBNG instances (eight per socket). The throughput scales almost linearly, and with eight instances deployed, 334 gigabits per second (Gbps) of traffic is processed. This is achieved using 24 data processing cores (1.5 cores per instance for sixteen instances).

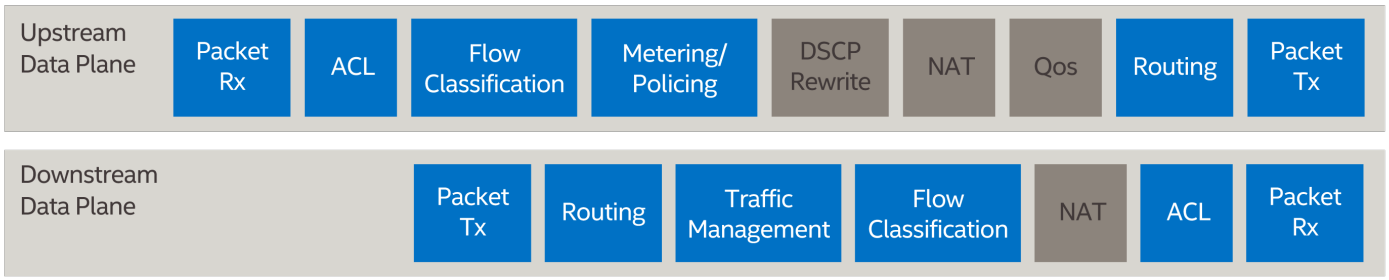


Figure 10. Performance Testing Pipeline Blocks

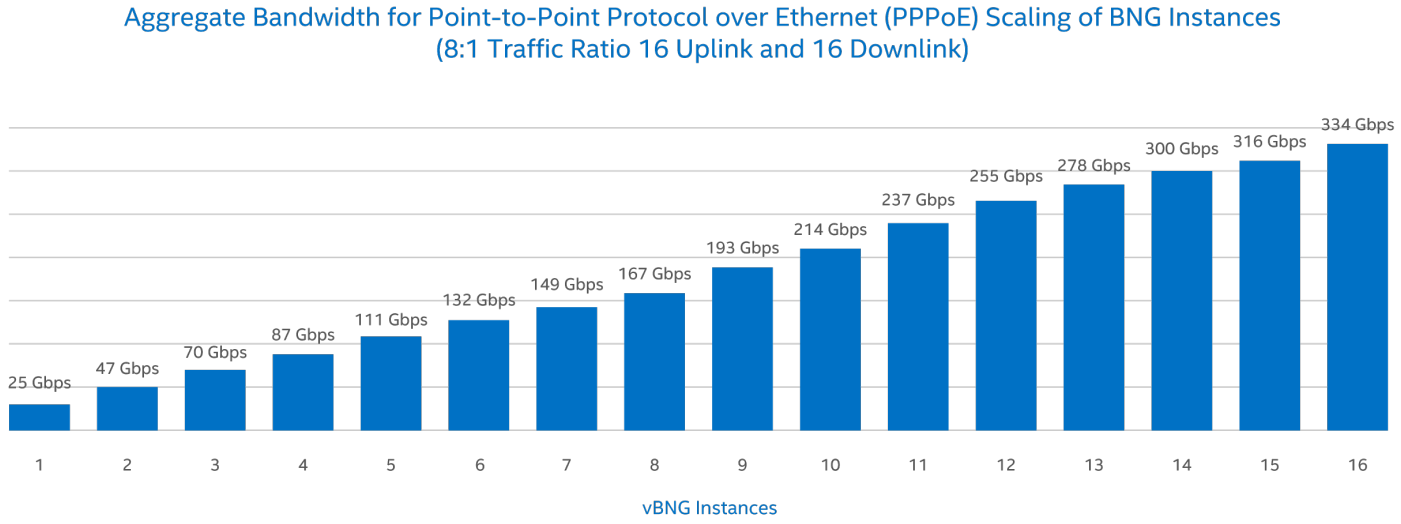


Figure 11. Intel® Xeon® Processor-Based Sever (Dual Socket) Throughput Running Multiple vBNG Instances[9]

## Summary

High performant VNFs have specific needs around infrastructure management and optimal resource utilization when being deployed in an NFV-enabled environment. Kubernetes features, such as CPU Manager for Kubernetes (CMK) and the SR-IOV device plugin, are just some contributions from Intel to facilitate a container-based NFV deployment. Intel is also helping enable other capabilities, such as Node Feature Discovery and NUMA Manager for Kubernetes, which will be compatible with other Kubernetes resource management features that support a broad array of cluster abilities for the wide range of workloads.

Intel is actively engaged with the K8s community, focused on enabling features that make it easier for telco grade gateways to be deployed with a cloud infrastructure manager. Using a vBNG reference application/use case, Intel has successfully demonstrated how Kubernetes capabilities can enable vBNG pods to attach directly to an SR-IOV VF via the SR-IOV device plugin. In addition, the CMK enabled workloads to be assigned to specific CPU cores, enabling containerized VNFs to fully utilize the underlying infrastructure for low-latency, high-performance use cases.  Network operators gain the additional ability to reduce platform power consumption during periods of low network activity in order to lower Op-Ex.


1. BNG: Broadband network gateway; IMS: IP multimedia core network subsystem; DPI: Deep packet inspection; IDS: Intrusion detection system; CDN: Content delivery networks; EPC: Evolved packet core.
2. Intel white paper, “Re-Architecting the Broadband Network Gateway (BNG)” 2019, https://www.intel.com/content/www/us/en/communications/network-transformation/broadband-network-gateway-architecture-study.html
3. https://github.com/intel/CPU-Manager-for-Kubernetes.
4. https://github.com/intel/sriov-network-device-plugin.
5. https://kubernetes.io/docs/concepts/overview/components.
6. “Dynamic Device Personalization for Intel® Ethernet 700 Series “ https://software.intel.com/en-us/articles/dynamic-device-personalization-for-intel-ethernet-700-series.
7. Intel white paper, “Re-Architecting the Broadband Network Gateway (BNG)” 2019, https://www.intel.com/content/www/us/en/communications/network-transformation/broadband-network-gateway-architecture-study.html
8. “Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper,” February 27, 2019, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html
9. Performance results are based on testing at Intel Corporation as of April 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.