Technology Guide

Intel® Scalable I/O Virtualization – Decrease Startup Time of Cloud-native Network Function on Pass-through Devices

Authors

Yahui Cao

Tim O'Driscoll

Portia Seater

Madhusudan Chittim

Jingjing Wu

1 Introduction

Network Functions Virtualization (NFV) is a way to virtualize network services that have traditionally been run on proprietary hardware. These services are typically packaged as virtual machines (VMs) deployed on standards servers, which is called Virtualized Network Function (VNF). As performance is a critical requirement for VNF, device pass-through is used to dedicate an entire network device to a guest operating system (OS) running inside VM to achieve near-native performance with the security of isolation.

In cloud-native ages, microservices and containers have gained widespread acceptance in production. In containers, instead of virtualizing the underlying hardware like VM, only the OS is virtualized. Comparing to VM, container gives more flexibility, finer granularity, and less overhead. As a result, Cloud-native Network Function (CNF), which puts network function running in container, becomes a successor to VNF.

Nowadays, container is emerging as a typical way of deploying services at the edge. However, container startup latency can greatly limit efficiency of short tasks, while a growing number of workloads at edge are classified as short-time task, which needs quick response[1]. For those workloads, decreasing container startup time is becoming increasingly important. Though industry has explored large number of approaches on decreasing the container launching latency[2], there is still some device level overhead impacts on startup time that needs to be addressed.

This document describes how Intel® Scalable I/O Virtualization (Intel® Scalable IOV) enabled platform and network device are used to decrease startup time of Cloud-native Network Function on pass-through modes. Test setup and results are given to prove that Intel Scalable IOV can significantly decrease the startup time.

This document is intended for communication service providers, or anyone looking to improve their Cloud-native Network Function. Even though the goal of this document is to showcase startup time reduction in Cloud-native Network Function environment, the technologies enabled here can be used as a reference point for startup time reduce in any container or networking deployment.

This document is part of the Network Transformation Experience Kits.

[1] Fu, S., Mittal, R., Zhang, L. and Ratnasamy, S., 2020. Fast and efficient container startup at the edge via dependency scheduling. In 3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20).

[2] The Application of Kata Containers in Baidu AI Cloud White Paper.

Table of Contents

Figures

Tables

Document Revision History

Revision	Date	Description
001	August 2022	Initial release.
002	February 2023	Added System Setup table. Revised for public distribution on Intel Network Builders.

1.1 Terminology

Table 1. Terminology

Abbreviation	Description
ADI	Assignable Device Interface
CNF	Cloud-native Network Function
DMA	Direct Memory Access
DPDK	Data Plane Development Kit
EAL	Environment Abstraction Layer
FLR	Function Level Reset
IAVF	Intel Adaptive Virtual Function
Intel Scalable IOV	Intel Scalable I/O Virtualization
IOMMU	I/O Memory Management Unit
MBUF	Message Buffer
MDEV	Mediated Device
NFV	Network Function Virtualization
PASID	Process Address Space Identifier
PF	Physical Function
PMD	Poll Mode Driver
RFC	Request for Comment
SR-IOV	Single Root I/O Virtualization
UUID	Universally Unique Identifier
VDCM	Virtual Device Composition Module
VDEV	Virtual Device
VF	Virtual Function
VFIO	Virtual Function I/O
VM	Virtual Machine
VNF	Virtual Network Function

1.2 Reference Documentation

Table 2. Reference Documents

Reference	Source
DPDK WIKI	https://en.wikipedia.org/wiki/Data_Plane_Development_Kit
Intel® Ethernet 800 Series Network Adapter(s)	https://www.intel.com/content/www/us/en/products/details/ethernet/800-network-adapters.html
Intel® Scalable I/O Virtualization Technical Specification	https://www.intel.com/content/www/us/en/develop/download/intel-scalable-io-virtualization-technical-specification.html
Intel® Virtualization Technology for Directed I/O Architecture Specification	https://www.intel.com/content/www/us/en/develop/download/intel-virtualization-technology-for-directed-io-architecture-specification.html
PCI Express® Base Specification Revision 4.0 Version 1.0	https://members.pcisig.com/wg/PCI-SIG/document/10912?downloadRevision=active
Single Root I/O Virtualization and Sharing Specification	https://members.pcisig.com/wg/PCI-SIG/document/download/8238
The Application of Kata Containers in Baidu AI Cloud White Paper	https://katacontainers.io/collateral/ApplicationOfKataContainersInBaiduAICloud.pdf

2 Overview

This chapter gives a brief introduction to widely used Single Root I/O Virtualization(SR-IOV) solution for NFV usage and how it introduces latency to network function startup time. Then, a detailed description on how Intel Scalable IOV is enabled, is given from each component's perspective. In addition, a detailed analysis on how Intel Scalable IOV can decrease network function startup time is given. Finally, test setup and results are given to show that the network function startup time is significantly decreased in a container runtime environment.

While this document explains how to decrease CNF startup time by Intel Scalable IOV, the system setup and technology enablement demonstrated in this guide are intended as a reference guide for anyone trying to improve their container startup time for pass-through device.

2.1 Challenges Addressed

SR-IOV, widely used in NFV, is a specification that allows a PCIe device to appear to be multiple separate physical PCIe devices. SR-IOV works by introducing the idea of physical functions (PFs) and virtual functions (VFs). PFs are full-featured PCIe functions with complete control over the whole PCIe device, while VFs have limited control over the PCIe device. In NFV, PFs are normally assigned to NFV infrastructure for control plane administration, while VFs are assigned to VNF as PCI pass-through devices inside VMs to deliver near-native device performance with hardware isolation.

Typically, device startup consists of generic device initialization stage and vendor specific initialization stage. For vendor specific initialization in SR-IOV VF, most of startup latency is caused by the communication between PF and VF. There is already some optimization for PF-VF communication. For generic PCIe device initialization SR-IOV VF, device must strictly follow initialization timing procedure as defined by PCIe specification. This results in a significant startup latency of CNF on SR-IOV VF pass-through.

In this document, we present our solution to decrease startup time of CNF on pass-through devices by eliminating strict initialization timing (like device reset) in SR-IOV. Since the device probed by the driver is a virtual device (VDEV), which is emulated by software in Intel Scalable IOV. It no longer needs to follow timing procedure required by PCIe specification and allows more flexible initialization optimization from the software's perspective.

2.2 Technology Description

Intel Scalable IOV is a scalable and flexible approach to hardware assisted I/O virtualization, building on existing PCI Express capabilities, enabling it to be easily supported by compliant PCI Express endpoint device designs and the software ecosystem[3]. Intel Scalable IOV defines an approach to assign large number of multiplexed device interfaces to isolated domains at a fine granularity. The architecture defines the granularity of sharing of a device as an Assignable Device Interface (ADI). All ADIs on a device function use the same PCIe Requester ID (Bus/Device/Function number) corresponding to the device's PCIe Function. Process Address Space Identifier (PASID) is used to distinguish upstream memory transactions performed for different ADIs and to convey the address space targeted by the transaction.

To enable the Intel Scalable IOV for network device in CNF, the following hardware and software components are needed:

- 4th Gen Intel® Xeon® Scalable Processor
- System Software and Stack
- Intel® Ethernet 800 Series Ethernet Network Adapter and Device Driver
- Data Plane Development Kit (DPDK) user space libraries and network drivers

In Intel Scalable IOV framework, accesses between application and VDEV are defined as either 'data path' or 'control path' (Figure 1):

1. Data-path operations on the VDEV are mapped directly to the underlying ADI hardware for performance. In this case, memory allocated by DPDK Intel Adaptive Virtual Function (IAVF) Poll Mode Driver (PMD) can be read/write directly by the Intel® Ethernet 800 Series Network Adapter(s). Data-path is typically PCIe upstream memory transactions with PASID prefix sent from the Intel® Ethernet 800 Series Network Adapter(s), going through IOMMU with Scalable Mode support and directly into user space DPDK process memory.
2. Control-path operations are emulated by the Virtual Device Composition Module (VDCM) for greater flexibility. With the assistance of VDCM, ADI is composed as VDEV and assigned to user space. Whenever user space application accesses some VDEV's region that falls within control path (for example., PCIe CSR/BAR), access is falling or trapped into VDCM by routing through VFIO Driver Framework and VFIO Mediated Device (VFIO MDEV) according to current implementation.

[3] Intel® Scalable I/O Virtualization Technical Specification

Since Intel Scalable IOV leverages HW extension to scale out for data path and compose device for control path, it delivers excellent scalability for high density instances requirement while remaining great flexibility for control path.

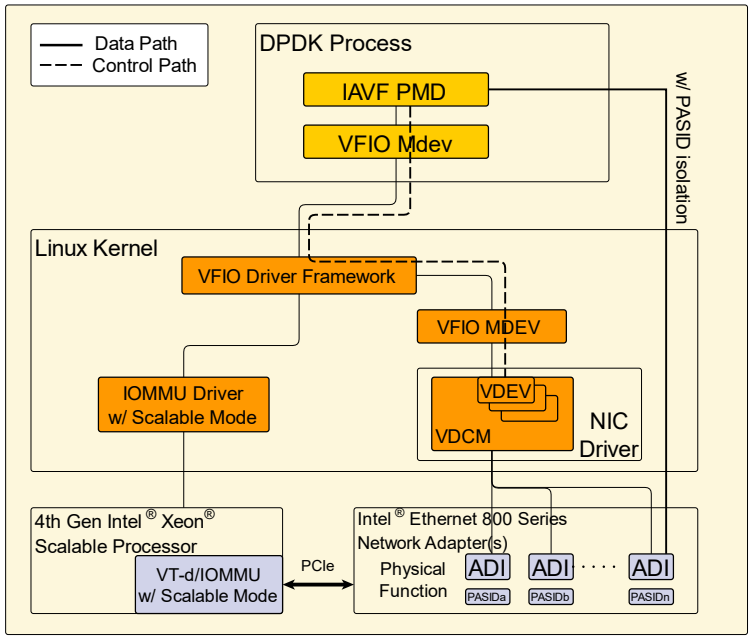

Figure 1. Intel® Scalable I/O Virtualization Architecture Diagram

2.2.1 4th Gen Intel® Xeon® Scalable Processor

Intel Scalable IOV depends on Scalable Mode Address Translation, which is introduced in Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d) since revision 3.0. 4th Gen Intel Xeon Scalable processor is among the first to implement Intel VT-d 3.0, which adds support for Scalable Mode Translation for DMA Remapping and enables PASID granular translation functions.

2.2.2 System Software and Stack

To enable Scalable Mode on the 4th Gen Intel Xeon Scalable processor, Intel IOMMU driver support for Intel VT-d 3.0/Scalable Mode is mandatory. Intel has upstreamed Scalable Mode IOMMU driver support on Linux kernel 5.2.

To pass-through ADI into user space and compose it as VDEV, VFIO[4]/MDEV[5] framework is leveraged to dedicate a collection of resources into VMs or containers. Intel has upstreamed VFIO/MDEV support for ADI on Linux kernel 5.11.

All of software mentioned in this document is assumed to be running in Linux operating system by default.

2.2.3 Intel® Ethernet 800 Series Network Adapter and Device Driver

Intel® Ethernet 800 Series Network Adapter(s) is a 100 Gbps port network adapter designed to optimize networking workloads including NFV. To support Intel Scalable IOV on the data path, the Intel® Ethernet 800 Series Network Adapter(s) supports PASID PCIe capability so that each ADI's PCIe upstream memory transactions is differentiated by a specified PASID value. The 20-bit PASID associated with a transaction is conveyed in a PCI Express PASID TLP prefix on the Intel® Ethernet 800 Series Network Adapter(s), implementing PASID granular address translation.

To support Intel Scalable IOV on the control-path, VDCM is required for composing ADI as VDEV. To implement VDCM based on VFIO mediated framework, MDEV parent operation is registered with callback to do the device emulation and management. Current VDCM implementation is packaged as part of the device driver.

2.2.4 Data Plane Development Kit

Data Plane Development Kit (DPDK) is an open-source software project that consists of libraries to accelerate packet processing workloads running on a wide variety of CPU architectures.

DPDK provides mature support for SR-IOV VF pass-through device by using VFIO framework to deliver near native performance and hardware level isolation. On initialization stage, DPDK will scan the Linux PCI bus and probe the device if it has

[4] https://www.kernel.org/doc/html/latest/driver-api/vfio.html
[5] Until then, MDEV will be deprecated and VFIO is under refactoring in Linux community. Refer to the link for more details.

been binded with vfio-pci driver, which means device is working on pass-through mode. The probing consists of two stages: generic PCIe device initialization and vendor specific initialization as described above.

Intel has pushed an RFC version of MDEV bus patch[6] to support Intel Scalable IOV initialization on DPDK. The main idea of this patch is to scan the Linux MDEV bus and probe the device like a normal PCI device if it is composed as a PCI device. To support passing through ADI, most of Intel® Ethernet Adaptive Virtual Function (Intel® Ethernet AVF) Poll Mode Driver (PMD) code is re-used with some minor modification.

2.2.5 Mediated Device Lifecycle

Since device is virtually emulated by VDCM in Intel Scalable IOV, the MDEV lifecycle is managed by software.

The typical lifecycle for a mediated device is shown in Figure 2.

1. Administrator loads the Intel® Ethernet 800 Series Network Adapter(s) device driver, and VDCM as part of the driver registers VFIO Mediated Device (MDEV).
2. Administrator creates MDEV by echoing some Universally Unique Identifier (UUID) into the file system entry and device driver will allocate ADI for this MDEV:
   echo "83b8f4f2-509f-382f-3c1e-e6bfe0fa1002" > /sys/class/mdev_bus/0000:XX:XX.X/mdev_supported_types/ice-vdcm/create
3. When DPDK process is launched, EAL will probe MDEV. Then, PASID value is requested by the device driver and allocated by the IOMMU driver. Once the initialization succeeds, EAL opens MDEV and performs general device initialization. After general device initialization succeeds, IAVF PMD is loaded, and the device specific initialization begins.
4. DPDK process performs control-path operation by issuing VFIO device read/write, which goes through system software and stack, and finally goes into VDCM module in the Intel® Ethernet 800 Series Network Adapter(s) device driver.
5. DPDK process performs data path operation by issuing DMA read/write request to ADI on the Intel® Ethernet 800 Series Network Adapter(s) hardware. The DMA read/write request is PCIe upstream memory transactions identified by PF's PCIe Requester ID (Bus/Device/Function number) and PASID with IOMMU address translation and protection for each ADI.
6. When you exit the DPDK process, it also closes MDEV.
7. Destroy MDEV by:
   echo 1 > /sys/bus/mdev/devices/83b8f4f2-509f-382f-3c1e-e6bfe0fa1002/remove

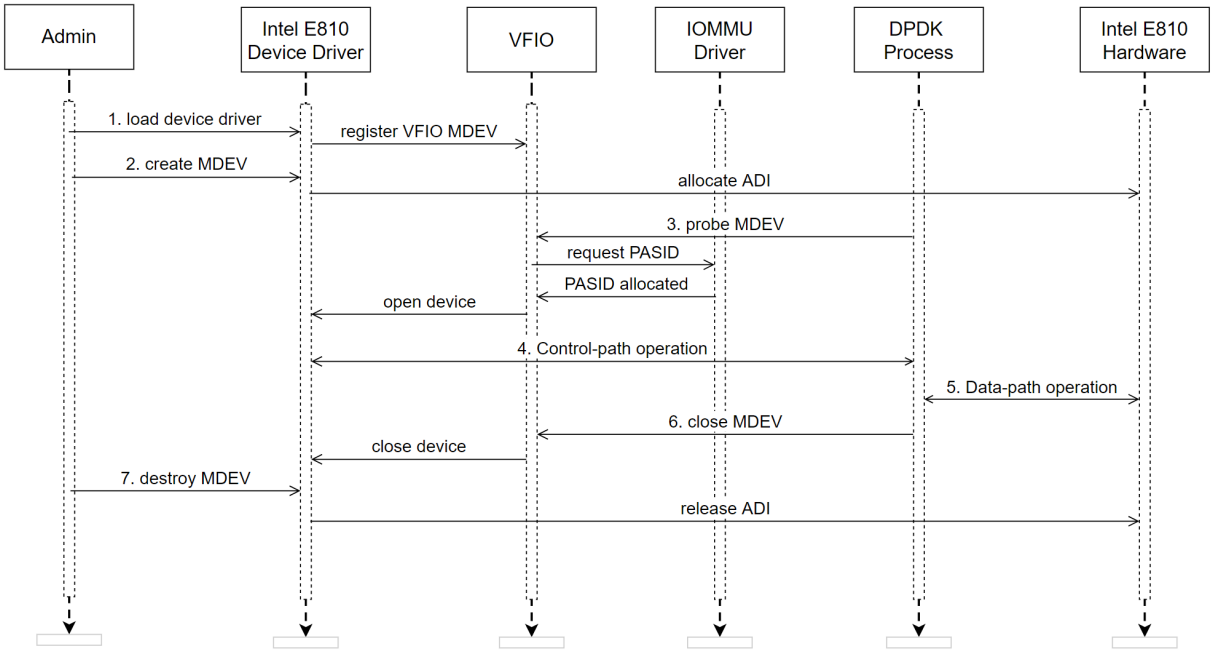

Figure 2. Mediated Device Lifecycle Sequence Diagram

[6] https://patches.dpdk.org/project/dpdk/cover/20210601030644.3318-1-chenbo.xia@intel.com/

2.2.6 Decreasing Startup Time by Intel® Scalable I/O Virtualization

Intuitively, startup time for an application can be mapped to the period of "Open MDEV" and "Control-path operation" as shown in Figure 2.

For DPDK application, the initialization procedure consists of the following stages listed:

1. The Environment Abstraction Layer (EAL) initialization: EAL is responsible for gaining access to low-level resources such as hardware and memory space. It provides a generic interface that hides the environment specifics from the applications and libraries. From a device's perspective, EAL will scan Linux PCI bus, performs generic device initialization, and performs vendor specific initialization by loading vendor driver.
2. Message buffer (MBUF) allocation: MBUF generally carries network packet buffers sent to or received from the network adapter, though it can be any data (like control data and events).
3. Device configuration: This stage is to configure the device's transmit queue, receive queue, mac address, and other setup. Once everything is set up successfully, the device is started to transmit and receive network packets.

For DPDK application running on SR-IOV VF, when VF is probed and opened for the first time during EAL initialization, PCIe Function Level Reset (FLR) is triggered implicitly (first reset) to perform device reset. Since device must complete the FLR within some period as defined by PCIe specification, software has to keep busy waiting during this period to meet the timing requirement. Even worse, then EAL will explicitly call VFIO device reset, which is causing another FLR (second reset) and latency. According to PCIe specification, the reset period may last about several hundred milliseconds.

For DPDK application running on Intel Scalable IOV VDEV, since all control-paths are emulated by VDCM, device is a virtually software representation. Software has great flexibility to eliminate the strict timing and procedure required by a real device. As a result, unnecessary first reset in SR-IOV VF cases can be removed. For the second reset, device don't need to follow PCIe FLR timing procedure either. Device reset can be implemented by VDCM in a lighter approach with less time consumed.

3 Deployment

The test environment is set up with DPDK testpmd probing MDEV backed by the Intel® Ethernet Network Adapter E810-CQDA2 on the 4th Gen Intel Xeon Scalable processor. To simulate the CNF run time environment, application is set up to run inside container. In order to show the benefits of Intel Scalable IOV, testpmd with SR-IOV VF backed by the E810-CQDA2 on 4th Gen Intel Xeon Scalable processor is benchmarked for comparison. Except for the difference between passing through Intel Scalable IOV ADI or SR-IOV VF, rest of the setup are the same. The full system setup is displayed in Table 3.

Table 3. System Setup

Item	Description
CPU	4th Gen Intel® Xeon® Scalable processor @1.8GHz 56 CPU cores x 2 NUMA nodes
Memory	256GB: 32GB x 4 DIMMs x 2 NUMA modes @ 4800MHz DDR5
Network Adapter	Intel® Ethernet Network Adapter E810-CQDA2
Network Adapter Firmware Version	4.00 0x800109c9 1.3204.0
BIOS	EGSDCRB1.SYS.0072.D10.2201171350. Turbo boosting is turned ON. Hyperthreading is turned ON.
Microcode	0x8f000250
Operating System	Fedora release 33
Linux Kernel Version	5.12
Kernel GRUB command	quiet crashkernel=512M intel_iommu=on,sm_on
DPDK Version	DPDK 22.07.0-rc0 + S-IOV patch
Test Info	Test by Intel as of 05/27/2022

3.1 Test Setup for Intel Scalable IOV VDEV (Test-VDEV)

In this test, VDEV is set up to be passed through into DPDK testpmd process. By passing a VFIO group device node into container, testpmd is allowed to perform control-path and data-path operations on VDEV through VFIO MDEV framework.

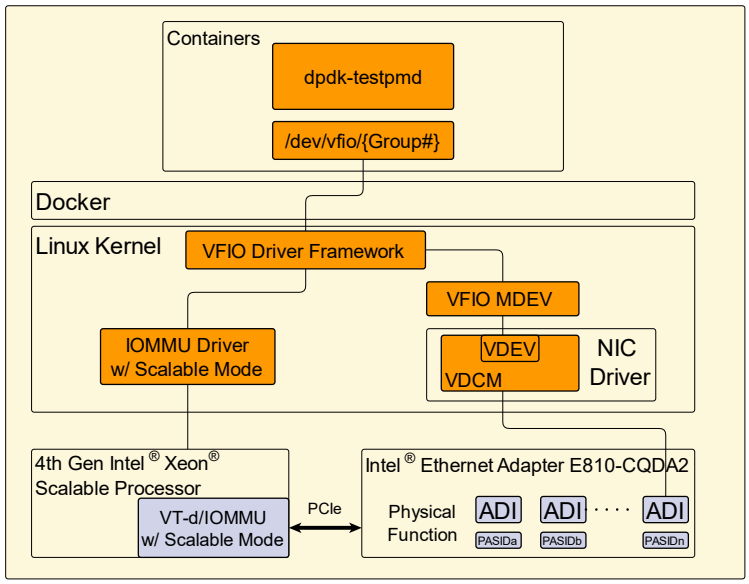

Figure 3. Intel® Scalable I/O Virtualization VDEV Test Setup Diagram

3.1.1 Setup Steps for Test-VDEV

Config Linux kernel boot arguments using the following command:

```
intel_iommu=on,sm_on
```

1. Load Intel E810 device driver and create MDEV.

```
# insmod ice.ko
# echo "83b8f4f2-509f-382f-3c1e-e6bfe0fa1001" >
/sys/class/mdev_bus/0000:16:00.0/mdev_supported_types/ice-vdcm/create
```

2. Setup 2M huge page and mount as /dev/hugepages

```
# echo 1024 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB
# echo 1024 > /sys/devices/system/node/node1/hugepages/hugepages-2048kB
# mount -t hugetlbfs -o pagesize=2M none /dev/hugepages
```

3. Launch container with pre-built docker image.

```
# docker run -it --name={NAME} --privileged --device=/dev/vfio/459 --device=/dev/vfio/vfio -
-ulimit memlock=-1:-1 -v /dev/hugepages:/dev/hugepages -v /dev:/dev {IMAGE}
```

4. Launch DPDK testpmd

```
# dpdk-testpmd -c 0x3 -a 83b8f4f2-509f-382f-3c1e-e6bfe0fa1001 --
```

3.2 Test Setup for SR-IOV VF (Test-VF)

In this test, SR-IOV VF is set up to be passed through into DPDK testpmd process. By passing a VFIO group device node into container, testpmd is allowed to have direct access on VF through VFIO PCI framework.

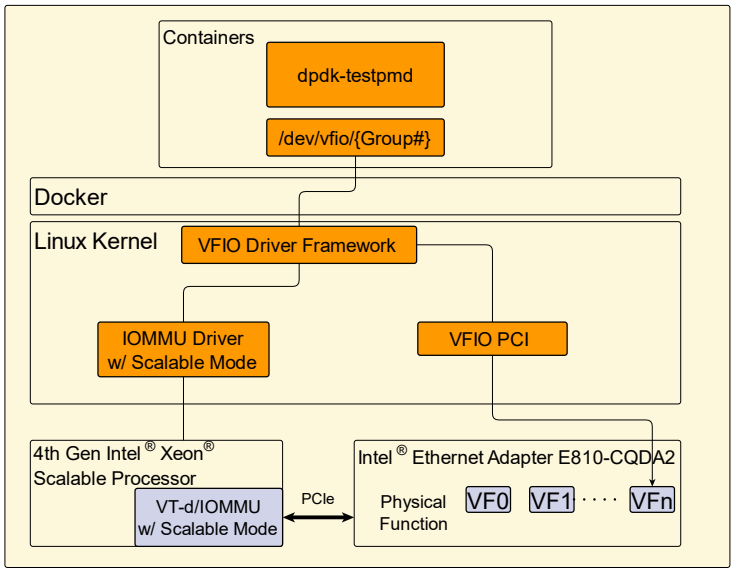

Figure 4. SR-IOV VF Test Setup Diagram

3.2.1 Setup Steps for Test-VF

Config Linux kernel boot arguments using the following command:
`intel_iommu=on,sm_on`

1. Load Intel E810 device driver and create SR-IOV VF.
```
# insmod ice.ko
# echo 1 > /sys/bus/pci/devices/0000:16:00.0/sriov_numvfs
```
2. Bind SR-IOV VF with vfio-pci driver.
```
# {DPDK_SOURCE_DIR}/usertools/dpdk-devbind.py -b vfio-pci 16:01.0
```
3. Setup 2M huge page and mount as /dev/hugepages
```
# echo 1024 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB
# echo 1024 > /sys/devices/system/node/node1/hugepages/hugepages-2048kB
# mount -t hugetlbfs -o pagesize=2M none /dev/hugepages
```
4. Launch container with pre-built Docker image.
```
# docker run -it --name={NAME} --privileged --device=/dev/vfio/459 --device=/dev/vfio/vfio --ulimit memlock=-1:-1 -v /dev/hugepages:/dev/hugepages -v /dev:/dev {IMAGE}
```
5. Launch DPDK testpmd
```
# dpdk-testpmd -c 0x3 -a 16:01.0 --
```

4 Results

The benchmark results for both of Test-VDEV and Test-VF is shown in Figure 5 with detailed data listed in Table 4.

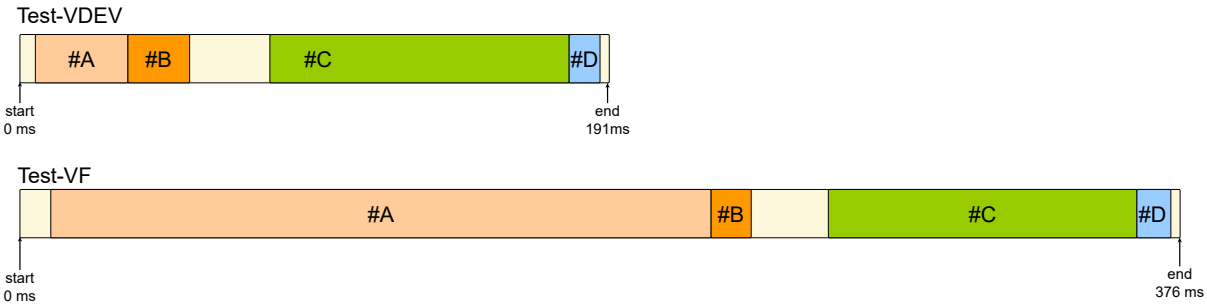

Figure 5. Test-VDEV and Test-VF Startup Time Diagram

For Test-VDEV, 191 milliseconds are consumed to finish all the initialization. For Test-VF, 376 milliseconds are consumed to finish all the initialization. According to Table 1, Test-VDEV only takes 30 milliseconds to finish the generic device initialization while Test-VF takes 214 milliseconds. Except for that, rest of the components take similar time for either Test-VDEV or Test-VF.

Table 4. Startup Time Comparison between Test-VDEV and Test-VF

	#A EAL - Generic device initialization (ms)	#B EAL – Vendor specific initialization(ms)	#C MBUF allocation (ms)	#D Device configuration(ms)	Other(ms)	Total(ms)
Test-VDEV	30	20	97	10	34	191
Test-VF	214	13	100	11	38	376

As a result, Intel Scalable IOV significantly decreases testpmd startup time in container environment due to the reason that it takes much less time to do generic device initialization.

5 Summary

This guide describes how Intel® Scalable I/O Virtualization can be leveraged on 4th Gen Intel Xeon Scalable processor and Intel® Ethernet 800 Series Network Adapter to decrease startup time of CNF on pass-through devices due to flexible device composition and lifecycle management by software.

This guide gives a comprehensive description on how Intel Scalable IOV is enabled from hardware and software components' perspective as well as qualitative analysis on why it can decrease startup time. Finally, test setup and result are given to demonstrate that Intel Scalable IOV can significantly decrease startup time in a container run time environment.

While this knowledge is specific to decrease the startup time for CNF, the technologies enabled by these hardware and software components are intended to be used as a reference for anyone looking to optimize their application startup time on pass-through device.

0223/DN/WIT/PDF 739221-002US