Open vSwitch

November 16-17, 2017 | San Jose, CA

OVS DPDK Installation & Gotchas

Shivaram Mysore, Founder, Service Fractal Inc

Motherboard Spec & NIC

Open hardware spec of motherboard and inspect

1. # of CPUs
2. PCI slot associations with CPU socket
3. Decide which Ethernet card and slot will go where.
4. Note the number of cores on the CPU
5. DPDK supported NICs - http://dpdk.org/doc/nics

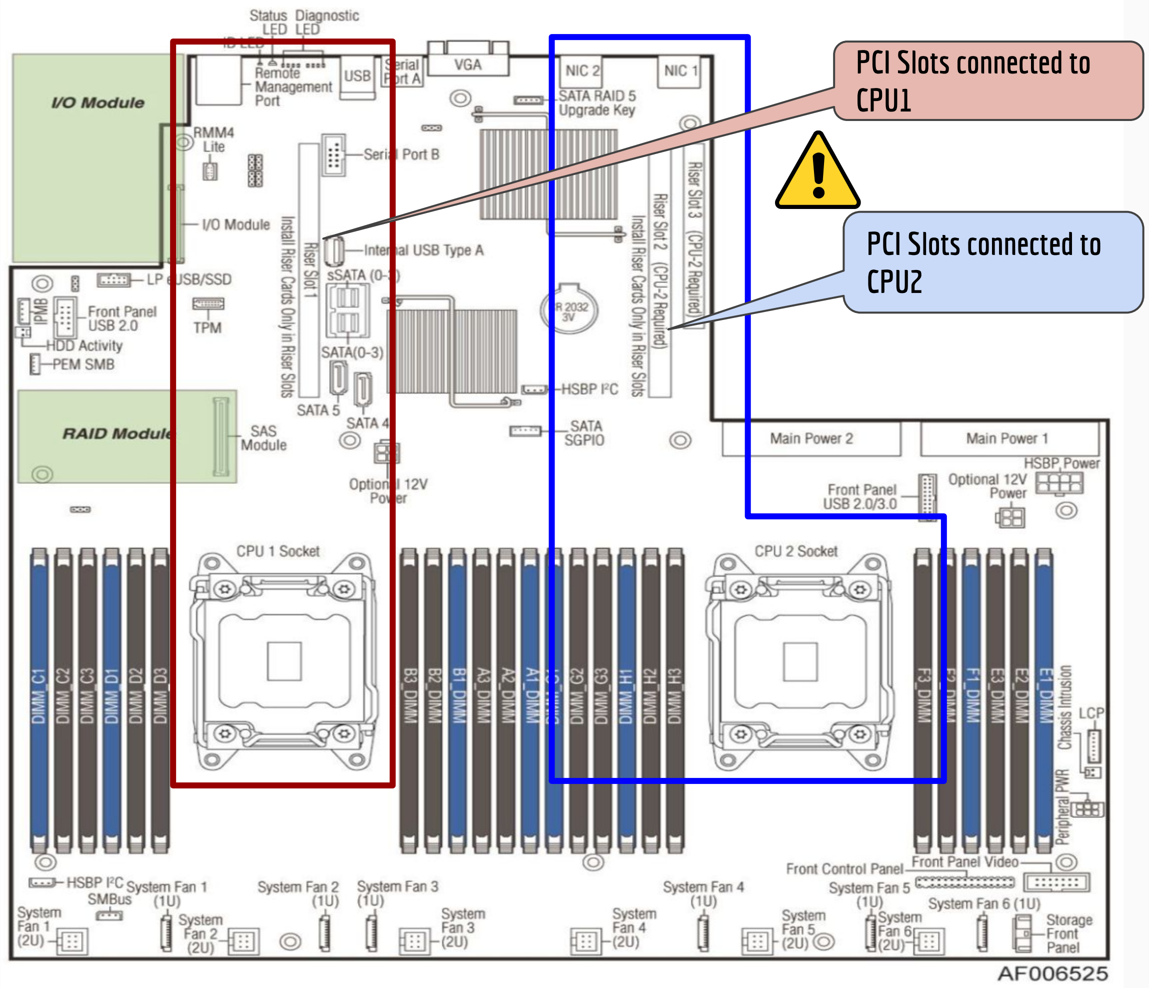
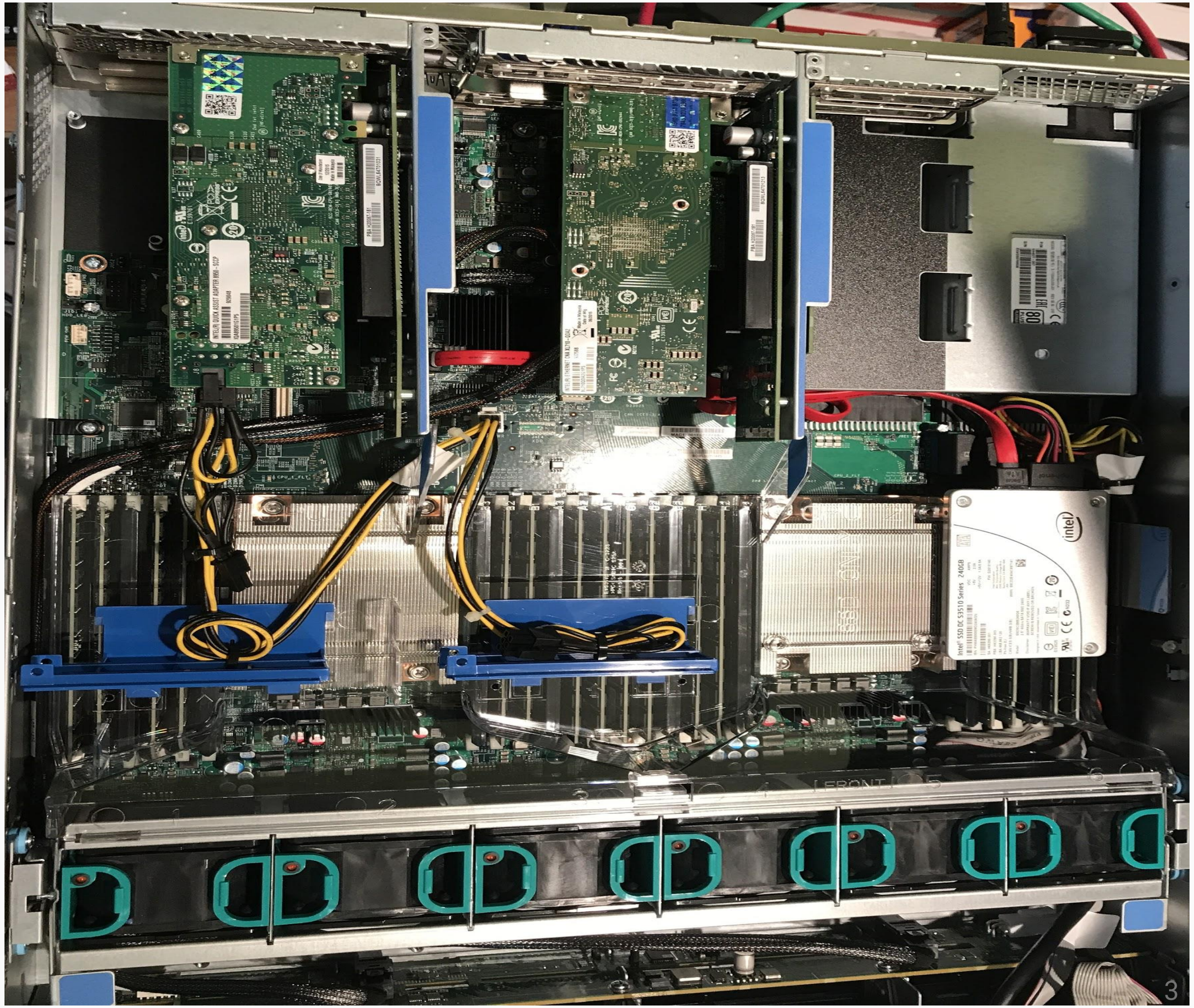


Figure 1. Server Board Component/Features Identification

My Installation

1. 2 CPUs - Xeon E5-2699 @ 2.2 Ghz
2. CPU-1
   a. On-board Intel X540-AT2 (1/10G)
   b. PCI - 1 x I350 (1G)
   c. PCI - 1 x I350
   d. PCI - 1 x 8950 Accelerator
3. CPU-2
   a. 1x Intel XL710 (40G)
4. Software:
   a. **Ubuntu 17.04**
   b. **OVS 2.8.1**



Scripts

1. Scripts available @ https://github.com/shivarammysore/faucetsdn-intel/tree/master/src/ubuntu/zesty/ovs_281
2. Ignore other variations and scripts in the above repo. This is the current incarnation of my experiments and also evolution of technology
3. Assumption: Fresh install of Ubuntu 17.04 server with just base utils and SSH
4. Scripts:
    4.1. **Setup OS** - `step1_system_setup.sh`
        4.1.1. Install additional system packages needed for OVS and DPDK
        4.1.2. Enable network modules to be loaded - **`/etc/modules`**
        4.1.3. Grub settings - **`/etc/default/grub`**
    4.2. **Configure OVS 2.8.x with DPDK** for use on Ubuntu 17.04 - `step2_ovs_dpdk_setup.sh`

Grub Settings

1. Modify file `/etc/default/grub` to include hugepages settings
   1.1. Reserve 1G **hugepages** via grub configurations. For example: to reserve 4 huge pages of 1G size - add parameters: `default_hugepagesz=1G hugepagesz=1G hugepages=4`
   1.2. For 2 **CPU cores**, Isolate CPU cores which will be used for DPDK - add parameters: `isolcpus=2`
   1.3. To use VFIO - add parameters: `iommu=pt intel_iommu=on`
   1.4. Note: If you are not sure about something, leave it as-is!!
   1.5. Your default and simple setting could be:

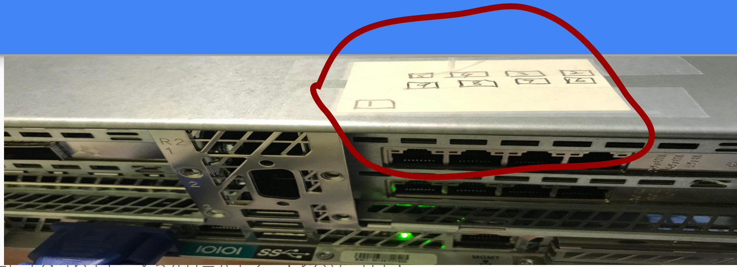
```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on iommu=pt \
default_hugepagesz=1G hugepagesz=1G hugepages=4"
```

2. After changing `/etc/default/grub`, run command: **`update-grub`**
3. **`reboot`** system for changes to take effect

Note: *To learn more about hugepages, grub, etc - please read OVS, DPDK, Linux and OS specific documentation*

Identify Ports & Modules

1. Identify ports and note them (**ip a** & **ethtool**)
2. Identify NIC cards - **lspci**



```
#  lspci
03:00.0 Ethernet controller: Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01)
03:00.1 Ethernet controller: Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01)
05:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
05:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
05:00.2 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
05:00.3 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
83:00.0 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 01)
83:00.1 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 01)
```

3. Run **dpdk-devbind** utility:

```
# dpdk-devbind --status-dev net
Network devices using DPDK-compatible driver
============================================
0000:03:00.1 'Ethernet Controller 10-Gigabit X540-AT2 1528' drv=igb_uio unused=vfio-pci
0000:05:00.0 'I350 Gigabit Network Connection 1521' drv=igb_uio unused=vfio-pci
0000:05:00.1 'I350 Gigabit Network Connection 1521' drv=igb_uio unused=vfio-pci

Network devices using kernel driver
===================================
0000:03:00.0 'Ethernet Controller 10-Gigabit X540-AT2 1528' if=eno1 drv=ixgbe unused=igb_uio,vfio-pci *Active*
0000:83:00.0 'Ethernet Controller XL710 for 40GbE QSFP+ 1583' if=ens802f0 drv=i40e unused=igb_uio,vfio-pci
0000:83:00.1 'Ethernet Controller XL710 for 40GbE QSFP+ 1583' if=ens802f1 drv=i40e unused=igb_uio,vfio-pci
```

Driver currently used and available

PCI Address

Bind DPDK Interfaces

1. Make sure relevant modules are loaded (`modprobe, modinfo`)

```
# modprobe igb_uio
# lsmod | grep igb
```

2. Bind the interface (`dpdk-devbind`)

```
# /sbin/dpdk-devbind --bind=igb_uio eno2
# /sbin/dpdk-devbind --status
```

FYI

3. **`vfio-pci`** (Linux Kernel > v4.1 & similar to **`pci-stub`**)
   a. The VFIO driver is an IOMMU/device agnostic framework for exposing direct device access to userspace, in a secure, IOMMU protected environment. In other words, this allows safe, non-privileged, userspace drivers. VM on host can be considered as a “userspace driver”.
   b. If UEFI secure boot is enabled, the Linux kernel may disallow the use of UIO on the system. Therefore, devices for use by DPDK should be bound to the **`vfio-pci`** kernel module rather than **`igb_uio`**
   c. https://www.kernel.org/doc/Documentation/vfio.txt & http://dpdk.org/doc/guides/linux_gsg/linux_drivers.html

Setup OVS

1. Setup OVS for DPDK enabled NICs (`ovs-vsctl`)

```
# ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-mem="1024,1024"
# ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
```

2 CPUs - memory for each

⚠

2. Add OVS Bridge, **`ovs-br0`** (`ovs-vsctl`) & set bridge IP address

```
# ovs-vsctl add-br ovs-br0 -- set bridge ovs-br0 datapath_type=netdev \
    protocols=OpenFlow13 other_config:datapath-id=0x11
# ip addr add 10.10.10.10/16 dev ovs-br0
```

Original Interface name

Interface name as shown on OVS bridge (remember dpdk-p0)

3. Add ports to OVS Bridge

```
# ovs-vsctl add-port ovs-br0 eno2 -- set Interface eno2 type=dpdk \
    options:dpdk-devargs=0000:03:00.1 ofport_request=1
```

PCI Address

Request OVS to number this port as 1

OVS Bridge

1. OVS Bridge status (ovs-vsctl)

```
# ovs-vsctl show
5a5285d9-5aa2-4941-9430-fc6cc4be1f17
    Bridge "ovs-br0"
        Controller "ssl:10.10.10.20:6653"
            is_connected: true
         Port "eno2"
            Interface "eno2"
                type: dpdk
                options: {dpdk-devargs="0000:03:00.1"}
```

Interface name per set Interface

PCI Address

```
# ovs-vsctl -- --columns=name,ofport list Interface
name                : "eno2"
ofport              : 1
```

Requested OVS port number

Making DPDK Interfaces Persistent (across reboots)

Edit `/etc/dpdk/interfaces` file and add all PCI addresses and corresponding modules to be loaded:

```
# <bus>   <id>        <driver>
pci     0000:04:00.0    vfio-pci
pci  0000:04:00.1    uio_pci_generic
## Interfaces on this machine that needs to be loaded on (re)boot
pci     0000:03:00.1    igb_uio
pci     0000:05:00.0    igb_uio
```

Caution: This has not worked for me consistently

- loads wrong driver only **`igb`** (kernel mode) instead of **`igb_uio`** (user space)
- never sure that OVS has picked up the bound interface later, etc).

Hence, I just delete the bridge and run the script which binds all the devices and ports.

Thankyou!

All code is available @ https://github.com/shivarammysore/faucetsdn-intel/blob/master/src/ubuntu/zesty/ovs_281/

Questions, Suggestions, Comments:

- Shivaram.Mysore at gmail
- shivaram_mysore