Update tutorial
Robert Schmidt authored 5 months ago

e368776b

ORAN_FHI7.2_Tutorial.md 10.99 KiB

OPEN AIR INTERFACE	**OAI 7.2 Fronthaul Interface 5G SA Tutorial**

Table of Contents

1. Prerequisites

The hardware on which we have tried this tutorial:

Hardware (CPU,RAM)	Operating System	NIC (Vendor,Driver,Firmware)
Intel(R) Xeon(R) Gold 6154 (2x18 Core), 64GB	RHEL 8.6 (4.18.0-372.26.1.rt7.183)	QLogic FastLinQ QL41000,qede,mbi 15.35.1
Intel(R) Xeon(R) Gold 6354 18-Core, 128GB	RHEL 8.7 (4.18.0-425.10.1.rt7.220)	Intel XXV710 for 25GbE SFP28,i40e,6.02 0x80003888
AMD EPYC 7513 32-Core Processor, 256GB	Ubuntu 20.04 (5.4.143-rt64)	Intel X710 for 10GbE SFP+,i40e,5.04 0x80002530
Intel(R) Xeon(R) Gold 6354 18-Core, 128GB	RHEL 9.2 (5.14.0-284.18.1.rt14.303.el9_2.x86_64)	Intel E810-XXV for SFP,ice,1.3236.0

NOTE: These are not minimum hardware requirements. This is the configuration of our servers.

For PTP grandmaster we have used Fibrolan Falcon-RX. The O-RU which we have used for this tutorial is LITEON FlexFi.

1.1 Setup

We mention the information for our setup below, but if you want more information, you can refer to below links.

1. O-RAN-SC O-DU Setup Configuration
2. O-RAN Cloud Platform Reference Designs 2.0,O-RAN.WG6.CLOUD-REF-v02.00,February 2021

1.2 Boot command parameters for DPDK

Update the Linux boot arguments (in `/etc/default/grub`) as follows. Afterwards, rewrite the boot loader.

The Linux kernel has documentation about these parameters, e.g., for kernel 5.14, here.

Isolated CPU 0-2 are used for DPDK/ORAN and CPU 8 for `ru_thread` in our example configs further below. If you have a server with 2 NUMA

`0,2,4,6,8,10,12,14,16`.

1.2.1 Intel

~These arguments were tried on both RHEL 8.6 (4.18.0-372.26.1.rt7.183.el8_6.x86_64) and 8.7 (4.18.0-425.10.1.rt7.220.el8_7.x86_64) ~

The boot commands were tried on RHEL 9.2 (5.14.0-284.18.1.rt14.303.el9_2.x86_64).

```
mitigations=off usbcore.autosuspend=-1 intel_iommu=on intel_iommu=pt selinux=0 enforcing=0 nmi_watchdog=0 softlocku
```

1.2.2 AMD

Install real time kernel followed by updating boot arguments

```
isolcpus=0-2,8-17 nohz_full=0-2,8-17 rcu_nocbs=0-2,8-17 rcu_nocb_poll nosoftlockup default_hugepagesz=1GB hugepages
```

1.2.3 Additional performance settings

We always set our servers to maximum performance mode. This means

- disabling hyper threading
- disabling sleep states in the BIOS and/or in the OS: `sudo cpupower idle-set -D 0`
- setting real-time tuning profile: `tuned-adm profile realtime`

Make sure `tuned` is running; it might be you have to run this command after every reboot.

1.3 PTP configuration

You can refer to the following O-RAN link for PTP information.

First, disable `ntp` and/or `chrony`; they cause problems with `phc2sys`.

In our setup we used Fibrolan Falcon-RX for PTP grandmaster. Unlike the O-RAN tutorial, we install `ptp4l` (v3.1.1) using the package manager, and use a simple configuration file as shown below, where the PTP grandmaster is reachable via interafce `ens7f1`.

```
$ cat /tmp/ptp4l.conf
[global]
domainNumber            24
slaveOnly               1
verbose                 1
network_transport       L2
time_stamping           hardware
tx_timestamp_timeout    100
[ens7f1]
$ sudo ptp4l -i ens7f1 -m -H -2 -s -f /etc/ptp4l.conf   # note: ptp4l.service should not run yet
$ sudo phc2sys -w -m -s ens7f1 -R 8 -f /etc/ptp4l.conf  # same as above
```

Note to the above: this tutorial assumes a machine with a Intel E-810 NICE (using `ice` driver), so we needed to set `tx_timestamp_timeout` to a high value. In other cases, `1` is enough.

If the offset is high, make sure you are using `skew_tick=1` in your kernel commandline.

If everything works, enable the ptp4l system daemon (note: daemon options are in `/etc/sysconfig/ptp4l`) and reboot:

```
$ sudo systemctl start ptp4l.service
$ sudo systemctl status ptp4l # should be fine explain more what
$ cat /etc/sysconfig/phc2sys
OPTIONS="-a -r -r -n 24" # man!
$ sudo systemctl start phc2sys.service
```

@sagar: Pay attention that the `freq` parameters are low

2. Build OAI-FHI gNB

2.1 DPDK (Data Plane Development Kit)

Download DPDK-stable version 20.11.8, compile and install it:

```
wget http://fast.dpdk.org/rel/dpdk-20.11.8.tar.xz
tar -xJf dpdk-20.11.8.tar.xz
cd dpdk-stable-20.11.8
meson build
cd build
ninja
sudo ninja install
sudo ldconfig
```

Note! As the Quickstart guide mentions, you might need to create a file `/etc/ld.so.conf.d/dpdk.conf` with content `/usr/local/lib64` before running `ldconfig` to properly discover all DPDK libraries.

2.2 Build ORAN Fronthaul Interface Library

Download ORAN FHI library

```
git clone https://gerrit.o-ran-sc.org/r/o-du/phy.git
cd phy
git checkout oran_release_bronze_v1.1
```

Apply patches (available in `oai_folder/cmake_targets/tools/oran_fhi_integration_patches/`)

```
git apply oran-fhi-1-compile-libxran-using-gcc-and-disable-avx512.patch
git apply oran-fhi-2-return-correct-slot_id.patch
git apply oran-fhi-3-disable-pkt-validate-at-process_mbuf.patch
git apply oran-fhi-4-process_all_rx_ring.patch
git apply oran-fhi-5-remove-not-used-dependencies.patch
```

Set up the environment (change the path if you use different folders)

```
export XRAN_LIB_DIR=~/phy/fhi_lib/lib/build
export XRAN_DIR=~/phy/fhi_lib
export RTE_SDK=~/dpdk-stable-20.11.8 # SDK is locally
export RTE_TARGET=x86_64-native-linuxapp-gcc
```

The `RTE_SDK` and `RTE_TARGET` are for FHI compilation. The `XRAN_LIB_DIR` will be for OAI (ROBERT: check)

Compile Fronthaul Interface Library

```
cd phy/fhi_lib/
./build.sh
```

2.3 Build OAI gNB

```
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git -b develop
cd openairinterface5g/cmake_targets
./build_oai --ninja -I # if this is the first time on this machine
./build_oai --gNB --ninja -t oran_fhlib_5g
```

Make sure that all libraries are correctly linked:

```
ldd ~/openairinterface/cmake_targets/ran_build/build/liboran_fhlib_5g.so
```

should show all SO references as resolved.

3. Configure Server and OAI gNB

In the following, we will configure DPDK to use two virtual PCI devices (network functions) on one physical PCI device. The keyword here is PCI Express I/O virtualization, and the Linux kernel has documentation, for the interested reader. We will load two virtual DPDK devices on the physical

You should have:

- The network interface on which C/U traffic is exchanged, below `ens7f1`
- The PCI addresses of the virtual devices, below
- The MAC addresses of DU and RU. O-RAN suggests `00:11:22:33:44:66` for the DU, and the RU comes with a pre-configured MAC address. Refer to your O-RU documentation for more details, below it will be `e8:c7:4f:1e:c7:11`
- If you use: the VLAN tags if you use them (below: `564`)

Further notes: O-RAN documents suggest to use the same MAC addresses for C-plane/U-plane communication. The network driver of the card used here (E-810) does not support it, so we will use `00:11:22:33:44:66` (C-plane) and `00:11:22:33:44:67` (U-plane).

3.1 Create and bind virtual devices and set network device properties

Following above introduction, we configure the virtual devices and bind them as below. Further, we set the MTU as required, and set maximum ring buffers.

TODO: which one is C-plane, which one is U-plane?

```
# print executed commands
set -x

# set two devices
sudo sh -c 'echo 0 > /sys/class/net/ens7f1/device/sriov_numvfs'
sudo sh -c 'echo 2 > /sys/class/net/ens7f1/device/sriov_numvfs'

# create virtual functions on physical one with MAC addresses and VLAN
sudo ip link set ens7f1 vf 0 mac 00:11:22:33:44:66 vlan 564 spoofchk off
sudo ip link set ens7f1 vf 1 mac 00:11:22:33:44:67 vlan 564 spoofchk off
sleep 1

# These are the DPDK bindings for C/U-planes on vlan 564
sudo python3 /usr/local/bin/dpdk-devbind.py --force --unbind c3:11.0
sudo python3 /usr/local/bin/dpdk-devbind.py --force --unbind c3:11.1
sudo modprobe vfio_pci
sudo python3 /usr/local/bin/dpdk-devbind.py --bind vfio-pci c3:11.0
sudo python3 /usr/local/bin/dpdk-devbind.py --bind vfio-pci c3:11.1

# set correct MTU
sudo ifconfig ens7f1 mtu 1500

# set high ringbuffers, check with ethtool -g ens7f1
sudo ethtool -G ens7f1 rx 8160 tx 8160

ip link show ens7f1
ip address show ens7f1
```

Note: we recommend you put the above in a script for later reference, e.g., after a reboot.

O-RAN documents mention to use the `iavf` driver to setup virtual interfaces. In our case, that driver was loaded automatically.

You can check that the virtual devices have been created with

```
ip link show
```

3.2 Update fronthaul interface configuration in `oran.fhi.json`

Configure the O-RAN FHI configuration file, in JSON format. This file is required by the FHI library, called by OAI.

- DU MAC Address: Parameter `o_du_macaddr`
- RU MAC Address: Parameter `o_ru_macaddr`
- PCI address: Parameter `dpdk_dev_up` and `dpdk_dev_cp`
- C-plane/U-plane VLAN tags: `cp_vlan_tag` and `up_vlan_tag`

4. Run OAI gNB

```
ln -s ../../../targets/PROJECTS/GENERIC-NR-5GC/CONF/oran.fhi.json oran.fhi.json
sudo ./nr-softmodem -O ../../../targets/PROJECTS/GENERIC-NR-5GC/CONF/oran.fh.band78.fr1.273PRB.conf --sa --reorder-
```