doc(oran): fixed a few typos and better explanation of the CPU allocation scheme
Raphael Defosseux authored 1 month ago

5a8379cd

ORAN_FHI7.2_Tutorial.md 23.32 KiB

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 (kernel)	NIC (Vendor,Driver,Firmware)
Intel(R) Xeon(R) Gold 6354 36-Core, 128GB	RHEL 9.2 (5.14.0-284.18.1.rt14.303.el9_2.x86_64)	Intel X710, i40e, 9.20 0x8000d95e 22.0.9
Intel(R) Xeon(R) Gold 6354 36-Core, 128GB	Ubuntu 22.04.3 LTS (5.15.0-1033-realtime)	Intel X710, i40e, 9.00 0x8000cfeb 21.5.9
AMD EPYC 9374F 32-Core Processor, 128GB	Ubuntu 22.04.2 LTS (5.15.0-1038-realtime)	Intel E810 ,ice, 4.00 0x8001184e 1.3236.0

NOTE: These are not minimum hardware requirements. This is the configuration of our servers. The NIC card should support hardware PTP time stamping.

NICs we have tested so far:

Vendor	Firmware Version
Intel X710	9.20 0x8000d95e 22.0.9
Intel E810	4.00 0x8001184e 1.3236.0

Vendor	Firmware Version

PTP enabled switches we have in are lab:

Vendor	Software Version
CISCO C93180YC-FX3	10.2(4)
Fibrolan Falcon-RX/812/G	8.0.25.4
Qulsar Qg2 (Grandmaster)	12.1.27

Radio units we are testing/integrating:

Vendor	Firmware
VVDN LPRU	06-v1.0.9
LiteON RU	01.00.08
Benetel 550	v0.8.1
Benetel 650	v0.8.1

Tested libxran releases:

Vendor
oran_release_bronze_v1.1
oran_e_maintenance_release_v1.0

1.1 Configure your server

1. Disable Hyperthreading (HT) in your BIOS. In all our servers HT is always disabled.
2. We recommend you to start with a fresh installation of OS (either RHEL or Ubuntu). You have to install realtime kernel on your OS (Operating System). Based on your OS you can search how to install realtime kernel.
3. Once the realtime kernel is installed then you have to change the boot arguments. You can use `tuned` command for this or you can do it manually via re-building the grub.

1.1.0 CPU allocation

This section is important to read, regardless of the operating system you are using.

Your server could be:

- 1-socket CPU (See the Ubuntu example): all the processors are sharing a single memory system.
- 2-socket CPU (see the RHEL example): processors are grouped in 2 memory systems.
  - Usually the even (ie `0,2,4,...`) CPUs are on the 1st socket
  - And the odd (ie (`1,3,5,...`) CPUs are on the 2nd socket

DPDK, OAI and kernel threads require to be properly allocated to extract maximum real-time performance for your use case.

1. **NOTE**: Currently the default OAI 7.2 configuration file requires isolated **CPUs 0,2,4** for DPDK/libXRAN, **CPU 6** for `ru_thread` and **CPU 8** for `L1_rx_thread`. It is preferrable to have all these threads on the same socket.
2. Allocating CPUs to the OAI nr-softmodem is done using the `--thread-pool` option. Allocating 4 CPUs is the minimal configuration but we recommend to allocate at least **8** CPUs. And they can be on a different socket as the DPDK threads.
3. And to avoid kernel preempting these allocated CPUs, it is better to force the kernel to use un-allocated CPUs.

Let summarize for example on a `32-CPU` system, regardless of the number of sockets:

Applicative Threads	Allocated CPUs
XRAN DPDK usage	0,2,4
OAI `ru_thread`	6

Applicative Threads	Allocated CPUs
OAI `nr-softmodem`	1,3,5,7,9,11,13,15
kernel	16-31

In below example we have shown the output of `/proc/cmdline` for two different servers, each of them have different number of numa nodes. Be careful in isolating the CPUs in your environment.

Modifying the `linux` command line usually requires to edit the `/etc/default/grub`, run a `grub` command and reboot the server.

1.1.1 RHEL 9.2

Below is the output of `/proc/cmdline` of a two numa node server,

```
NUMA:
  NUMA node(s):          2
  NUMA node0 CPU(s):     0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34
  NUMA node1 CPU(s):     1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35
```

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

1.1.2 Ubuntu 22.04.3 LTS

Below is the output of `/proc/cmdline` of a single numa node server,

```
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0-31
```

```
isolcpus=0-15 nohz_full=0-15 rcu_nocbs=0-15 kthread_cpus=16-31 rcu_nocb_poll nosoftlockup default_hugepagesz=1GB hu
```

1.1.3 Common

Configure your servers to maximum performance mode either via OS or in BIOS. If you want to disable CPU sleep state then use the below command:

```
sudo cpupower idle-set -D 0
#to enable
sudo cpupower idle-set -E
```

After the OS is installed change your kernel to Realtime and install `tuned-adm` command.

```
tuned-adm profile realtime
```

The above information we have gathered either from O-RAN documents or via our own experiments. In case you would like to read the O-RAN documents then here are the 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 PTP configuration

There are two ways of installing PTP,

1. You can refer to the following o-ran link for PTP configuration.

```
git clone http://git.code.sf.net/p/linuxptp/code linuxptp
git checkout v2.0
make && make install
```

```
./ptp4l -i ens1f1 -m -H -2 -s -f configs/default.cfg
./phc2sys -w -m -s ens1f1 -R 8 -f configs/default.cfg
```

```
#RHEL
sudo dnf install linuxptp -y
#Ubuntu
sudo apt install linuxptp -y
```

Once installed you can use this configuration file for ptp4l (`/etc/ptp4l.conf`). Here the clock domain is 24 so you can adjust it according to your PTP GM clock domain

```
[global]
domainNumber            24
slaveOnly               1
time_stamping           hardware
tx_timestamp_timeout    1
logging_level           6
summary_interval        0
#priority1               127

[your_PTP_ENABLED_NIC]
network_transport       L2
hybrid_e2e              0
```

Probably you need to increase `tx_timestamp_timeout` for Intel E-810. You will see that in the logs of ptp.

Create the configuration file for phc2sys (`/etc/sysconfig/phc2sys`)

```
OPTIONS="-a -r -r -n 24"
```

The service of ptp4l (`/usr/lib/systemd/system/ptp4l.service`) should be configured as below:

```
[Unit]
Description=Precision Time Protocol (PTP) service
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
EnvironmentFile=-/etc/sysconfig/ptp4l
ExecStart=/usr/sbin/ptp4l $OPTIONS

[Install]
WantedBy=multi-user.target
```

and service of phc2sys (`/usr/lib/systemd/system/phc2sys.service`) should be configured as below:

```
[Unit]
Description=Synchronize system clock or PTP hardware clock (PHC)
After=ntpdate.service ptp4l.service

[Service]
Type=simple
EnvironmentFile=-/etc/sysconfig/phc2sys
ExecStart=/usr/sbin/phc2sys $OPTIONS

[Install]
WantedBy=multi-user.target
```

1.4 DPDK(Data Plane Development Kit)

```
# on debian
sudo apt install wget xz-utils
# on fedora
sudo dnf install wget xz
cd
wget http://fast.dpdk.org/rel/dpdk-20.11.7.tar.xz
```

Note tht you can also use a more recent version of DPDK: `wget http://fast.dpdk.org/rel/dpdk-20.11.8.tar.xz`.

DPDK Compilation and Installation

```
# Installing meson : it should pull ninja-build and compiler packages
# on debian
sudo apt install meson
# on fedora
sudo dnf install meson
tar xvf dpdk-20.11.7.tar.xz && cd dpdk-stable-20.11.7

meson build
ninja -C build
sudo ninja install -C build
```

Verify the installation is complete

Check if the LD cache contains the DPDK Shared Objects after update:

```
sudo ldconfig -v | grep rte_
        librte_fib.so.0.200.2 -> librte_fib.so.0.200.2
        librte_telemetry.so.0.200.2 -> librte_telemetry.so.0.200.2
        librte_compressdev.so.0.200.2 -> librte_compressdev.so.0.200.2
        librte_gro.so.20.0 -> librte_gro.so.20.0.2
        librte_mempool_dpaa.so.20.0 -> librte_mempool_dpaa.so.20.0.2
        librte_distributor.so.20.0 -> librte_distributor.so.20.0.2
        librte_rawdev_dpaa2_cmdif.so.20.0 -> librte_rawdev_dpaa2_cmdif.so.20.0.2
        librte_mempool.so.20.0 -> librte_mempool.so.20.0.2
        librte_pmd_octeontx2_crypto.so.20.0 -> librte_pmd_octeontx2_crypto.so.20.0.2
        librte_common_cpt.so.20.0 -> librte_common_cpt.so.20.0.2
....
```

On Fedora-based OS, you may not have the `/usr/local/lib` or `/usr/local/lib64` paths in the LD_LIBRARY_PATH:

```
sudo echo "/usr/local/lib" > /etc/ld.so.conf.d/local-lib.conf
sudo echo "/usr/local/lib64" >> /etc/ld.so.conf.d/local-lib.conf
sudo ldconfig -v | grep rte_
```

Check if the PDK-CONFIG tool discovers the libraries:

```
pkg-config --libs libdpdk --static
-lrte_node -lrte_graph -lrte_bpf -lrte_flow_classify -lrte_pipeline -lrte_table -lrte_port -lrte_fib -lrte_ipsec -l
```

Once again on Fedora-based OS, you may not have the `/usr/local/lib` or `/usr/local/lib64` paths in the PKG_CONFIG_PATH:

```
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib64/pkgconfig/
pkg-config --libs libdpdk --static
```

If you want to de-install this version of DPDK

Go back to the version folder you used to build and install

```
cd ~/dpdk-stable-20.11.7
sudo ninja deinstall -C build
```

2. Build OAI-FHI gNB

Clone OAI code base in a suitable repository, here we are cloning in `~/openairinterface5g` directory,

```
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git ~/openairinterface5g
cd ~/openairinterface5g/
```

2.0 Retrieve ORAN Fronthaul Interface Library Patches

We have made patches for the different releases of the FHI library:

```
# TODO when merging: remove the 2 following lines
# At the time of writing, the `fhidriver_E_rk` was not yet merged into `develop`
git checkout fhidriver_E_rk
# If the `fhidriver_E_rk` does not exist anymore, then they are in `develop` branch
git checkout develop
# We have different versions of patches per O-RAN releases
ls cmake_targets/tools/oran_fhi_integration_patches/
E  bronze
```

2.1 Build ORAN Fronthaul Interface Library

Download ORAN FHI DU library

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

2.1.1. Using the Bronze Release

```
git checkout oran_release_bronze_v1.1
```

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

```
cp ~/openairinterface5g/cmake_targets/tools/oran_fhi_integration_patches/bronze/oran-fhi-*.patch .

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
```

2.1.2. Using the E Release

```
git checkout oran_e_maintenance_release_v1.0
```

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

```
cp ~/openairinterface5g/cmake_targets/tools/oran_fhi_integration_patches/E/*.patch .

git apply oaioran_E.patch
```

2.1.3. Build (valid for any release)

Set up the environment (change the path if you use different folders). We recommend you copy all variables in a file and then you source that file

```
export XRAN_LIB_DIR=~/phy/fhi_lib/lib/build
export XRAN_DIR=~/phy/fhi_lib
export RTE_SDK=~/dpdk-stable-20.11.7
export RTE_TARGET=x86_64-native-linuxapp-gcc
export RTE_INCLUDE=/usr/local/include
```

Compile Fronthaul Interface Library

```
cd ~/phy/fhi_lib/lib
make clean
make XRAN_LIB_SO=1
...
[AR] build/libxran.so
./build/libxran.so
```

The shared library object `~/phy/fhi_lib/lib/build` SHALL be generated.

Copy it to `/usr/local/lib` :

```
sudo cp ~/phy/fhi_lib/lib/build/libxran.so /usr/local/lib
sudo ldconfig -v | grep libxran
        libxran.so -> libxran.so
```

2.2 Build OAI gNB

```
# You should have already cloned above and switched to the proper branch.
cd ~/openairinterface5g/cmake_targets
# on debian
sudo apt install -y libnuma-dev
# on RHEL
sudo dnf install -y numactl-devel
./build_oai --gNB --ninja -t oran_fhlib_5g (Add, -I if you are building for the first time on server for installing
#check if all the libraries are properly linked to liboai_transpro.so
ldd ran_build/build/liboran_fhlib_5g.so
    linux-vdso.so.1 (0x00007ffc9bdfc000)
    librte_node.so.0.200.2 => /usr/local/lib64/librte_node.so.0.200.2 (0x00007f2da93bd000)
    librte_graph.so.0.200.2 => /usr/local/lib64/librte_graph.so.0.200.2 (0x00007f2da93b2000)
    librte_bpf.so.0.200.2 => /usr/local/lib64/librte_bpf.so.0.200.2 (0x00007f2da93a2000)
    librte_flow_classify.so.0.200.2 => /usr/local/lib64/librte_flow_classify.so.0.200.2 (0x00007f2da939c000)
    librte_pipeline.so.20.0 => /usr/local/lib64/librte_pipeline.so.20.0 (0x00007f2da9376000)
    librte_table.so.20.0 => /usr/local/lib64/librte_table.so.20.0 (0x00007f2da935a000)
    librte_port.so.20.0 => /usr/local/lib64/librte_port.so.20.0 (0x00007f2da9340000)
    librte_fib.so.0.200.2 => /usr/local/lib64/librte_fib.so.0.200.2 (0x00007f2da9332000)
...
    libm.so.6 => /lib64/libm.so.6 (0x00007f2d974a4000)
    libxran.so => /usr/local/lib/libxran.so (0x00007f2d49000000)
    libnuma.so.1 => /lib64/libnuma.so.1 (0x00007f2d97494000)
    libc.so.6 => /lib64/libc.so.6 (0x00007f2d48c00000)
    /lib64/ld-linux-x86-64.so.2 (0x00007f2da93d3000)
```

In case `liboai_transpro.so` does not find `libxran.so` , then you can copy XRAN shared library object:

```
sudo cp ~/phy/fhi_lib/lib/build/libxran.so /usr/local/lib
sudo ldconfig
```

3. Configure OAI gNB

On this source branch (ie `fhidriver_E_rk`) and later on for `develop` branch, the configuration of the OAI-DU is based on 2 files:

1. the usual OAI gNB/DU configuration file: `~/openairinterface5g/targets/PROJECTS/GENERIC-NR-5GC/CONF/oran.fh.band78.fr1.273PRB.conf`

3.1. Adapt the OAI-DU configuration file to your system/workspace

CAUTION: this section is correct for the Bronze Release support.

TODO: update for E Release.

Edit the `targets/PROJECTS/GENERIC-NR-5GC/CONF/oran.fh.band78.fr1.273PRB.conf` with:

- The PLMN section shall match the one defined in the AMF
- `amf_ip_address` shall be the correct AMF IP address in your system
- `GNB_INTERFACE_NAME_FOR_NG_AMF` and `GNB_IPV4_ADDRESS_FOR_NG_AMF` shall match your DU N2 interface name and IP address
- `GNB_INTERFACE_NAME_FOR_NGU` and `GNB_IPV4_ADDRESS_FOR_NGU` shall match your DU N3 interface name and IP address
- Adjust the frequency, bandwidth and SSB position
- Set an isolated core for L1 thread `L1_rx_thread_core` in our environment we are using CPU 8
- Set an isolated core for RU thread `ru_thread_core` in our environment we are using CPU 6
- `phase_compensation` should be set to 0 to disable when it is performed in the RU and set to 1 when it should be performed on the DU side
- `sdr_addrs = "dummy -c /home/oaicicd/openairinterface5g/targets/PROJECTS/GENERIC-NR-5GC/CONF/o-ran-dat-files/config_o_du_static_vvdn.dat -p 2 0000:31:06.0 0000:31:06.1"` shall be modified to match to your workspace and the binding values (see later)
  - On our system, the `~` folder corresponds to `/home/oaicicd`
  - The fact that we are providing an absolute path to the O-RAN FHI configuration dat file makes it easier to manage.

3.2. Adapt the O-RAN fronthaul interface configuration dat file to your system

The first example we are providing (`~/openairinterface5g/targets/PROJECTS/GENERIC-NR-5GC/CONF/o-ran-dat-files/config_o_du_static_vvdn.dat`) is derived from the example within the Fronthaul Interface Library source base (`~/phy/fhi_lib/app/usecase/mu1_100mhz/config_file_o_du.dat`)

You may have to restart from this original version, based on phy release version or other usecase.

What you shall modify at the RU MAC address fields:

- `ruMac0`
- `ruMac1`

Set the VLAN tags, mtu and

- `c_plane_vlan_tag`
- `u_plane_vlan_tag`
- mtu

Set the cores for DPDK

- systemCore (absolute coreId)
- ioWorker (it is a mask: 1<<coreid) (For example if you select core 2 then this value should be 4)
- ioCore (absolute coreId)

Adjust the frequency, bandwidth and any other parameter which is relevant to your environment.

3.3. Bind Devices

Here are the commands to create Virtual Functions (VFs). We recommand to copy in a file and use it when the system is restarted. We are not creating presistant interfaces. Below are the variables you have to fill according to your environment

```
du-c-plane-mac-addr --> DU C plane mac-address
du-u-plane-mac-addr --> DU U plane mac-address
physical-interface --> Physical interface through which you can access the RU
vlan --> vlan is defined in the RU configuration
mtu --> mtu is specified by the RU vendor
lspci-address-c-plane-vf --> Bus address for c plane vf
lspci-address-u-plane-vf --> Bus address for u plane vf
```

For both the mac-addresses you can use mac-address which is pre-configured in the RUs (not valid for all the RUs) `00:11:22:33:44:66`. If your system has Intel E-810 NIC cards/ ICE driver. Then you have to choose different mac-addresses (this valid for the kernels which we have mentioned). If the RU vendor requires untagged traffic then remove the vlan and on the switch you have to configure the vlan as access vlan. In

case the mtu is different than 1500. Then you have to update the mtu and similarly on the switch interface.

```
sudo ip link set <physical-interface> vf 1 mac <du-u-plane-mac-addr> vlan <vlan> mtu <mtu> spoofchk off
## you have to search for the created virtual interfaces and add their address lscpi | grep Virtual
sudo modprobe vfio_pci
sudo /usr/local/bin/dpdk-devbind.py --bind vfio-pci <lspci-address-c-plane-vf>
sudo /usr/local/bin/dpdk-devbind.py --bind vfio-pci <lspci-address-u-plane-vf>
```

To unbind the VFs

```
sudo sh -c echo "2" > /sys/class/net/<physical-interface>/device/sriov_numvfs
sudo /usr/local/bin/dpdk-devbind.py --unbind <lspci-address-c-plane-vf>
sudo /usr/local/bin/dpdk-devbind.py --unbind <lspci-address-u-plane-vf>
```

4. Start OAI gNB

```
cd ran_build/build
```

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

For example if you have two numa nodes (for example 18 CPU per socket) in your system and odd cores are non isolated then you can put the thread-pool on `1,3,5,7,9,11,13,15`. Else if you have 1 numa node either you can use isolated cores or non isolated. Just make sure that isolated cores are not the ones defined earlier.