Application Note

eNodeB CPU Benchmarking

Version: 2025-12-12

Table of Contents

1 Introduction

This application note explains how to evaluate a hardware platform to host Amarisoft eNodeB. A common question frequently asked is whether a hardware is efficient enough to run Amarisoft eNodeB and how many active UEs with ongoing traffic could be served within a cell. This application note explains the eNodeB test mode functionality and how it can be used to benchmark a specific processor and hardware platform.

1.1 Background

In order to check the performance of a processor, 2 parameters need to be checked.

- eNodeB CPU load: This is the total CPU load of the processor when running eNodeB. As the eNodeB runs on multiple cores, it is effectively the sum of each CPU core load running the eNodeB software
- Baseband latency: It is defined to be the processing latency at baseband side.

In LTE network, there are 4 msec between RX and TX which means that the system has 4 msec to:

1. Convert the received analog signal to digital.
2. Transmit the IQ samples to baseband depending on used RF. For a PCIe SDR card, it would be over a PCIe bus while for a N210, it would be over Ethernet.
3. Process the DL data at physical layer (demodulation, decoding, etc) before sending it to protocol stack for further analysis.
4. Generate UL date and process it at protocol stack side as well as at physical layer side (coding, modulation, etc).
5. Transmit UL IQ samples toward RF.
6. Convert digital signal to analog.

Steps 3 and 4 correspond to pure baseband processing while steps 1, 2, 5 and 6 correspond to the IQ data transfer and radio frontend processing. The most important factor when evaluating a hardware performance is to measure the LTE baseband processing latency. It allows to assess the interaction between Amarisoft software architecture and the CPU.

Amarisoft software monitors the baseband latency (steps 3 and 4) and displays the remaining time left for radio front end (steps 1, 2, 5 and 6) via `t cpu` command.

This command can be run in eNodeB screen. Its output is as follows:

```
-Proc-  ---RX--------   ---TX--------   --- TX/RX diff (ms)
   CPU    MS/s    CPU     MS/s    CPU   min/avg/max sigma
 22.6%  11.519   9.4%   11.518   1.8%   2.8/3.3/3.8 0.3
```

`Proc/CPU` indicates the total CPU load of the eNodeB process. `RX/CPU` corresponds to the time passed when reading IQ samples from RF driver and `TX/CPU` is the time spent writing IQ samples to RF driver.

The interesting part is the `TX/RX diff` and particularly the `min` value that corresponds to the minimum available time for radio front end processing. When benchmarking a CPU, first thing to do is to check that this value is higher than a threshold depending on the radio front end.

The following screenshot depicts the output of `t cpu` command taken during a UDP DL transfer at 150 Mbps (20MHz cell 2x2 MIMO) with a Samsung Galaxy S5 using Amarisoft eNodeB coupled with a PCIe SDR card.

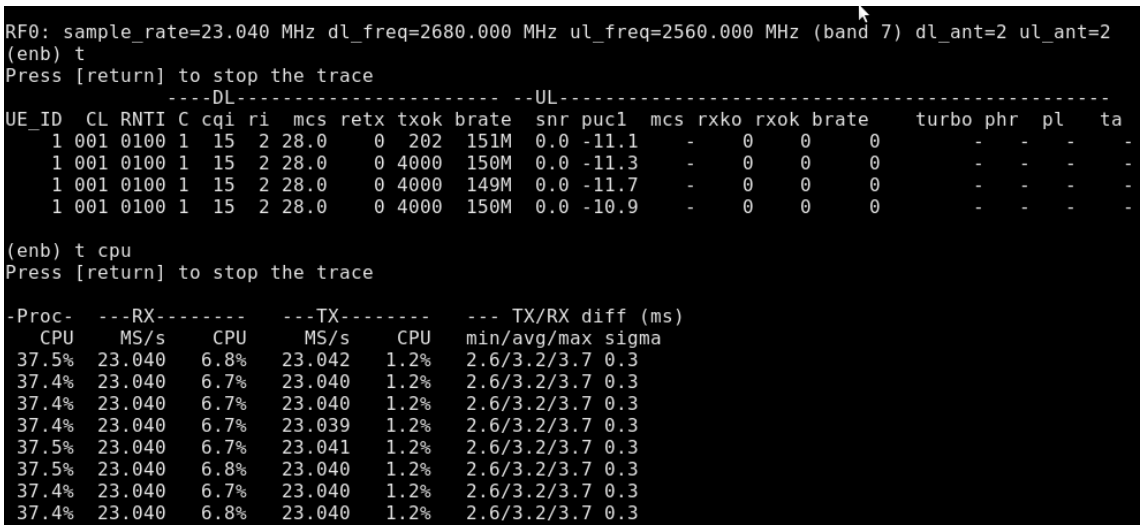
```
RF0: sample_rate=23.040 MHz dl_freq=2680.000 MHz ul_freq=2560.000 MHz (band 7) dl_ant=2 ul_ant=2
(enb) t
Press [return] to stop the trace
PRACH: cell=01 seq=12 ta=2 snr=26.5 dB
             ----DL----------------------- --UL-----------------------------------------------
UE_ID  CL RNTI C cqi ri  mcs retx txok brate  snr puc1  mcs rxko rxok brate    turbo phr  pl   ta
    1 001 003d 1   3  1  2.0    0    1   128 16.7  1.9    -    0    1    44  1/1.0/1   -   -  0.0
    1 001 003d 1  15  2 26.7    0   39 13.6k 17.6 14.9 19.9    1   19 9.25k  1/2.3/6  40  59  0.0
    1 001 003d 1  15  2 28.0    1  921 33.7M 14.0  9.9 22.7    1   21 46.1k  1/1.8/6  40  58  0.0
    1 001 003d 1  15  2 28.0    4 3996  149M 16.7 10.4 20.9    3   58 64.4k  1/1.6/6  40  57  0.0
    1 001 003d 1  15  2 27.3   23 3977  142M 15.5 13.1 19.5    5   57 62.4k  1/1.8/6  40  56  0.0
    1 001 003d 1  15  2 28.0    0 4000  149M 15.1 13.6 18.5    0   57 66.8k  1/1.0/1  40  56  0.0

(enb) t cpu
Press [return] to stop the trace

-Proc-  ---RX--------   ---TX--------   --- TX/RX diff (ms)
   CPU    MS/s    CPU     MS/s    CPU   min/avg/max sigma
 37.0%  23.040   5.8%   23.041   1.2%   2.6/3.2/3.7 0.3
 37.1%  23.040   5.8%   23.039   1.2%   2.6/3.2/3.7 0.3
 37.0%  23.040   5.8%   23.040   1.2%   2.6/3.2/3.7 0.3
 37.1%  23.040   5.8%   23.040   1.2%   2.6/3.2/3.7 0.3
 37.1%  23.040   5.8%   23.041   1.2%   2.6/3.2/3.7 0.3
 37.0%  23.040   5.8%   23.040   1.2%   2.5/3.2/3.7 0.3
 36.9%  23.040   5.7%   23.039   1.2%   2.6/3.2/3.7 0.3
 37.1%  23.040   5.8%   23.041   1.2%   2.6/3.2/3.7 0.3
```

We can see that there is a large margin for RF processing as the `min` value is around 2.6 msec which means that the baseband took at worst case 1.4 msec to finish its processing leaving 2.6 msec at the RF side.

2 eNodeB Test Mode

In order to facilitate the load and latency measurements without a complicated test setup, we can use eNodeB test mode. This mode enables specific tests where UE contexts are automatically created. The goal is to define an environment close enough to what we are targeting in terms of number of UEs and average throughput. For more details about test mode, please refer to eNodeB documentation.

The following screenshot depicts the output of `t cpu` command taken with eNodeB in PDSCH test mode using PCIe SDR card simulating one single UE with continuous PDSCH reception. In this mode, the cell properties `pdcch_format, pdsch_mcs, forced_ri, forced_cqi, transmission_mode, dl_256qam` can be used to force specific PDSCH parameters.



In our example, the simulation is done by adding the following lines to configuration file `config/mimo-2x2-20mhz.cfg`. By setting parameters `forced_ri: 2, forced_cqi: 15`, the PDSCH is transmitted using DL MCS of 28 in MIMO mode.

```
forced_cqi: 15,
forced_ri: 2,

test_mode: {
  type: "pdsch", /* PDSCH continuous reception */
  pdsch_retx: false, /* if false, don't force the UE to retransmit in case of erro
  rnti: 0x100, /* RNTI for PUSCH */
},
```

It is not always possible to measure the performance of a certain hardware with a radio front end as the radio might not be available. In this case, the eNodeB software could be run with a dummy RF allowing to make measurement without any radio front end.

The following screenshot depicts the output of `t cpu` command taken with eNodeB in test mode using a dummy RF driver and simulating one single UE with continuous PDSCH reception at MCS 28. The same configuration file as the previous test is used but this time the eNodeB has been launched with `-n` option which forces the use of dummy RF driver.

```
RF0: sample_rate=23.040 MHz dl_freq=2680.000 MHz ul_freq=2560.000 MHz (band 7) dl_ant=2 ul_ant=2
(enb) t
Press [return] to stop the trace
             ----DL---------------------- --UL-----------------------------------------------
UE_ID  CL RNTI C cqi ri  mcs retx txok brate  snr puc1  mcs rxko rxok brate    turbo phr  pl   ta
    1 001 0100 1  15  2 28.0    0  202  151M  0.0   -     -    0    0     0        -   -    -    -
    1 001 0100 1  15  2 28.0    0 4000  149M  0.0   -     -    0    0     0        -   -    -    -
    1 001 0100 1  15  2 28.0    0 4000  150M  0.0   -     -    0    0     0        -   -    -    -
    1 001 0100 1  15  2 28.0    0 4000  149M  0.0   -     -    0    0     0        -   -    -    -
    1 001 0100 1  15  2 28.0    0 4000  150M  0.0   -     -    0    0     0        -   -    -    -

(enb) t cpu
Press [return] to stop the trace

-Proc-  ---RX--------   ---TX--------   --- TX/RX diff (ms)
   CPU    MS/s    CPU    MS/s    CPU   min/avg/max sigma
 30.1%  23.039   1.2%   23.040   0.3%   2.6/3.2/3.7 0.3
 30.1%  23.040   1.2%   23.040   0.3%   2.6/3.2/3.7 0.3
 30.0%  23.040   1.2%   23.040   0.3%   2.6/3.2/3.7 0.3
 30.1%  23.040   1.2%   23.040   0.3%   2.6/3.2/3.7 0.3
 30.1%  23.040   1.2%   23.040   0.3%   2.6/3.2/3.7 0.3
 30.1%  23.040   1.2%   23.040   0.3%   2.6/3.2/3.7 0.3
 30.1%  23.040   1.2%   23.040   0.3%   2.6/3.2/3.7 0.3
 30.1%  23.040   1.2%   23.040   0.3%   2.6/3.2/3.7 0.3
```

Using a dummy RF ignores the impact of radio front end, specially the I/O part (Ethernet, USB, PCIe...), on the overall system performance. A precise measurement can only be performed in real conditions in the presence of a radio front end and real UEs. The test mode only gives an approximate indication of load and latency. This could easily be observed when comparing the results of `t cpu` command using the dummy RF driver versus Samsung Galaxy S5.

Finally, the most interesting test mode type to use is the `load` test mode. In this mode, several UEs are instantiated and all are transmitting and receiving at the same time. You can specify the number of UEs as well as a defined error rate in UL/DL. The cell properties `pusch_mcs, forced_ri, forced_cqi` can be used to set the simulated radio conditions. It provides a convenient way to benchmark the processor in the presence of multiple UEs with simultaneous UL and DL traffic.

The following screenshot depicts the output of `t cpu` command taken with eNodeB in load test mode using a dummy RF driver and simulating 100 UEs with continuous PDSCH and PUSCH.

```
[root@enb1-sophia enb]# ./lteenb  -n config/enb_load.cfg
LTE Base Station version 2017-11-30, Copyright (C) 2012-2017 Amarisoft
This software is licensed to Amarisoft.
Support and software update available until 2021-12-07.

RF0: sample_rate=23.040 MHz dl_freq=2662.500 MHz ul_freq=2542.500 MHz (band 7) dl_ant=2 ul_ant=2
(enb) t g
Press [return] to stop the trace
--#UE---------       --RRC------------ --DL------------- --UL-------------
conn idle disc prach reqst reest pagng  retx  txok brate  retx  rxok brate
 100    0    0     0     0     0     0     0   202 63.6M     2   222 36.5M
 100    0    0     0     0     0     0    34  3967 62.4M    51  4360 35.9M
 100    0    0     0     0     0     0    45  3956 62.1M    48  4360 35.9M
 100    0    0     0     0     0     0    36  3965 62.2M    45  4360 36.0M

(enb) t cpu
Press [return] to stop the trace

-Proc-  ---RX--------   ---TX--------   --- TX/RX diff (ms)
   CPU    MS/s    CPU    MS/s    CPU   min/avg/max sigma
 53.1%  23.037   1.1%   23.040   0.3%   2.4/3.0/3.5 0.3
 53.1%  23.040   1.1%   23.040   0.3%   2.4/3.0/3.5 0.3
 53.1%  23.040   1.1%   23.040   0.3%   2.4/3.0/3.5 0.3
 53.1%  23.040   1.1%   23.040   0.3%   2.4/3.0/3.5 0.3
```

This is generated by adding the following lines to the eNodeB configuration file.

```
pusch_mcs: 20,
forced_cqi: 15,
test_mode: {
  type: "load",
  ue_count: 100,
```

```
  pusch_fer: 0.01,
  pdsch_fer: 0.01,
},
```

3 Benchmarking Methodology

Based on what we explained above, evaluating a hardware platform requires the following steps:

- Prepare the eNodeB configuration file as per your cell configuration.
- Add `load` test mode to your eNodeB configuration file by setting the number of users within the cell and the error rate as per your nominal use case.
- Use the `pdsch_mcs, forced_cqi, forced_ri` to set a certain data rate in DL.
- Use the `pusch_mcs` to set the data rate in UL.
- Start eNodeB with `-n` option if you do not have access to the radio head. Run the `t cpu` command in eNodeB screen and check `TX/RX diff min`. This value should be higher than a threshold depending on your target radio front end.
- If you have access to the radio head, then you can run the eNodeB and check the output of `t cpu` command. You should have at least `TX/RX diff min > 0.5 msec`. Having a couple of bad values does not impact the overall functionality of the system statistically.

4 Additional Information

For any technical issue, please raise a ticket from our support site at `https://support.amarisoft.com/`.
To learn more about our technology and solutions, e-mail us at `customer@amarisoft.com` or visit `https://www.amarisoft.com`.