WHITE PAPER

COMMUNICATION SERVICE PROVIDERS

Intel-Lenovo Verified Reference Configuration for vRAN on Lenovo ThinkSystem with Intel 4th Gen Intel Xeon® Scalable Processor with Intel® vRAN Boost

Introduction

For communication service provider's, staying current with reliable, open network infrastructure is table stakes. It's important for CSPs to implement large network upgrades and adopt new technology with tried and tested equipment that is efficient, reliable, and interoperable. Intel FlexRAN™ reference software technologies for Open Radio Access Networks (RAN) is supportive of the next wave of telecom technology, deployment, and services delivery.

Lenovo's ThinkSystem SR630v3 powered by 4th Gen Intel® Xeon® Scalable processors with Intel® vRAN Boost acceleration for software-based RAN workloads and provides a significant reduction in power consumption compared to the previous generation that used discrete PCIe card-based acceleration.

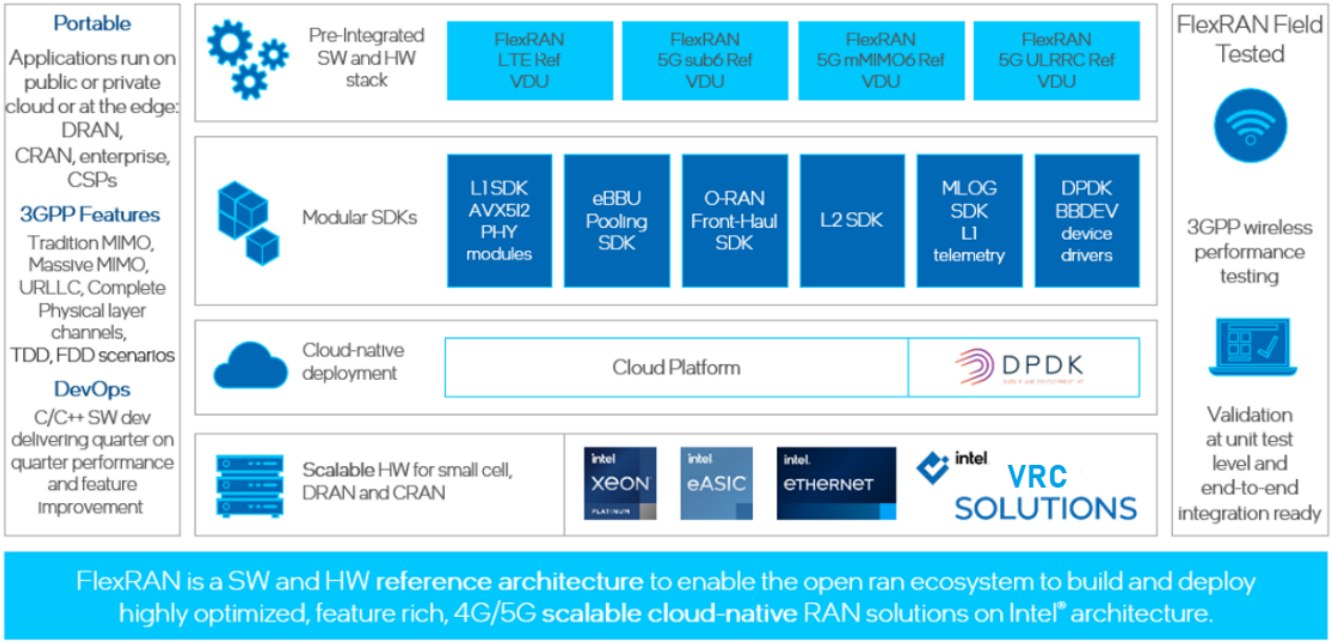
Project Scope

This verified configuration is based on Intel's cutting-edge integrated Intel® vRAN Boost, providing acceleration for L1 processing in the 4th Gen Intel Xeon® Scalable processor with Intel® vRAN Boost technology acceleration supported by Lenovo's ThinkSystem SR630v3. This verification enables Communication Services Providers (CoSPs) the ability to provide energy efficient verified for 5G vRAN solutions on Lenovo's trusted SR630v3 platform.

In this white paper we spotlight a verified reference configuration (VRC) solution that uses Intel's FlexRAN™ reference software as a workload that comes in a ready-to-use container (CNF) from

Intel's Docker Github that can assist CoSPs in quickly verifying their own implementations and in their preparation for vRAN network rollouts.

Intel's FlexRAN™ Reference Architecture



Lenovo SR630v3 1U Server

Lenovo's ThinkSystem SR630 V3

[Lenovo ThinkSystem SR630 V3 | Rack Servers | Lenovo US]

For the SR630v3 verification, Lenovo used the following:

Hardware

Category	Details	Comment
Server	Lenovo SR630 V3	Machine Type = 7D73CTO1WW
CPU	1x 6443N	Intel® 6443N
NIC	1x Intel E810-XVVDA4	4x Port 25Gb NIC with PTP support**
Memory	8x Samsung DDR5	32GB TruDDR5 4800MHz (2Rx8) RDIMM
Storage	2x M.2	480GB SATA SSD
BMC	Version 3.11	Build ESX314P
UEFI	Version 2.21	Build ESE188E
Baseband	Integrated	Intel® vRAN Boost integrated in CPU

** Support for O-RAN LLS C1 mode

Software

Category	Details	Comment
BIOS	Configured Per Intel Doc 640685, Rev.: 1.6	See Lenovo FME for details
OS	RHEL 9.3 Realtime Kernel 5.14.0-362.8.1	
NIC	Intel E810 Firmware 4.22	
Container Runtime	containerd.io 1.6.26-3.1	
Cloudnative	Kubernetes 1.28.4	
Baseband	Intel FlexRAN™ ref. software v23.11	See Intel FME for details
DPDK	DPDK Source Release 22.11 + FlexRAN™ Patch	See Intel FME for details

Installation

Using the Lenovo XCC check the firmware version of the E810 NIC(s) and update firmware if necessary. Follow the link below to locate the correct firmware images for the SR630 V3 server from Lenovo's support center.

https://datacentersupport.lenovo.com/ie/en/products/servers/thinksystem/sr630v3/7d73

For simplicity and reduction in server count a single test server implementation was used with Red Hat's RHEL 9.3 Realtime OS, however where possible, Red Hat recommends that Ansible Playbooks on a RHOCP cluster be used to deploy containers.

Lenovo performed the steps below to achieve a Verification Reference Configuration (VRC)

BIOS Configuration

The BIOS needs to be configured very precisely for correct FlexRAN™ operation using the parameters set out in according to Intel Doc 640685, Rev.: 1.6 (see Lenovo FME for detailed steps).

Verify the Intel Hardware P_State driver is active in the BIOS and loaded during OS boot by examining dmesg

```
[root@spree-du test-results]# dmesg | grep -i hwp
[    4.021605] intel_pstate: HWP enabled by BIOS
[    4.043323] intel_pstate: HWP enabled
```

Realtime Kernel cmdline

The following kernel boot args were used with the RHEL 9.3 Realtime Kernel.

BOOT_IMAGE=(hd1,gpt2)/vmlinuz-5.14.0-362.8.1.el9_3.x86_64+rt root=/dev/mapper/rhel-root resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap intel_iommu=on iommu=pt usbcore.autosuspend=-1 selinux=0 enforcing=0 nmi_watchdog=0 softlockup_panic=0 audit=0 mce=off hugepagesz=1G hugepages=40 hugepagesz=2M hugepages=0 default_hugepagesz=1G kthread_cpus=0,31,32,63 irqaffinity=0,31,32,63 skew_tick=1 isolcpus=managed_irq,domain,1-30,33-62 nohz_full=1-30,33-62 rcu_nocbs=1-30,33-62 nosoftlockup crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M vfio_pci.enable_sriov=1 vfio_pci.disable_idle_d3=1 vfio-pci.ids=8086:57c0 systemd.unified_cgroup_hierarchy=1 intel_idle.max_cstate=0

Turbostat

Configure the processors settings via userspace:

Set FlexRAN™ cpupower options via userspace

cpupower frequency-set -g performance
cpupower frequency-set -u 2500000

Then run turbostat to verify the CPU's frequencies and C-States in use under a well-tuned system for FlexRAN™ L1 application.

Next, verify the system is suitably tuned for FlexRAN™ L1 processing. This is done with cyclictest to test scheduling latency and hwlatdetect to detect any system latencies induced by hardware or firmware.

Cyclictest

Run cyclictest on all isolated vCPUs for 12 hours

taskset -c 0,1-30,33-62 cyclictest -m -p95 -h 15 -a 1-30,33-62 -t 60 --mainaffinity=0 -D 12h --histfile=cyclictest_histogram.txt

Check the Max Latencies in the output file cyclictest_histogram.txt to ensure the values are under 10 uSec as this will ensure the timing precision needed for L1-PHY processing.

cat cyclictest_histogram.txt

…………

Total: 043199997 043199997 043199997 043199998 043199999 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043200000 043199999 043199999 043199999 043199999 043199999 043199999 043199999 043199999 043199999 043199999 043199999 043199999 043199999 043199999 043199999 043199999 043199998 043199998 043199998 043199998 043199998 043199998 043199998 043199998 043199998 043199998 043199998 043199998 043199998 043199998 043199998 043199998 043199997 043199997 043199997 043199997 043199997 043199997 043199997

Min Latencies: 00002

Avg Latencies: 00002

Max Latencies: 00006 00008 00006 00006 00006 00006 00006 00006 00007 00006 00006 00006 00006 00006 00006 00006 00006 00006 00007 00006 00006 00006 00006 00006 00006 00006 00006 00006 00006 00006 00006 00006 00006 00006 00006 00006

…………

hwlatdetect

Run hwlatdetect for 24 hours on the isolated vCPUs to verify there are no latency issues generated the hardware or firmware.

```
# echo 7ffffffe,7ffffffe > /sys/kernel/tracing/tracing_cpumask
# echo per-cpu > /sys/kernel/tracing/hwlat_detector/mode
# hwlatdetect --threshold 10 --hardlimit 10 --duration 12h --window 1000000us --width 500000us --report
hwlat_result.log --watch
hwlatdetect:  test duration 86400 seconds
  detector: tracer
  parameters:
      CPU list:         None
      Latency threshold: 10us
      Sample window:     1000000us
      Sample width:      500000us
    Non-sampling period:  500000us
      Output File:       /root/hwlat-dir/hwlat_result.log

Starting test
Test finished
Max Latency: Below threshold
Samples recorded: 0
Samples exceeding threshold: 0
Report saved to hwlat_result.log (0 samples)
```

Intel Memory Latency Checker

Intel's MLC tool was used to verify the memory latency of the system.

```
Intel(R) Memory Latency Checker - v3.11
Measuring idle latencies for sequential access (in ns)...
                Numa node
Numa node            0
       0         107.8

Measuring Peak Injection Memory Bandwidths for the system
Bandwidths are in MB/sec (1 MB/sec = 1,000,000 Bytes/sec)
Using all the threads from each core if Hyper-threading is enabled
Using traffic with the following read-write ratios
ALL Reads        :      221388.6
3:1 Reads-Writes :      193298.9
2:1 Reads-Writes :      189724.2
1:1 Reads-Writes :      174501.6
Stream-triad like:      186934.5

Measuring Memory Bandwidths between nodes within system
Bandwidths are in MB/sec (1 MB/sec = 1,000,000 Bytes/sec)
Using all the threads from each core if Hyper-threading is enabled
Using Read-only traffic type
                Numa node
Numa node            0
       0        221418.4

Measuring Loaded Latencies for the system
Using all the threads from each core if Hyper-threading is enabled
Using Read-only traffic type
Inject  Latency Bandwidth
Delay   (ns)    MB/sec
==========================
 00000  229.06   221317.5
 00002  229.88   221297.6
 00008  224.99   221448.2
 00015  219.44   221353.3
 00050  213.64   221951.5
 00100  136.03   159302.3
 00200  120.08    73904.4
 00300  117.10    50584.9
 00400  115.07    38231.3
 00500  114.81    30534.4
 00700  113.86    22168.5
 01000  112.95    15797.6
 01300  111.83    12308.7
 01700  111.02     9569.3
 02500  110.83     6716.5
 03500  110.47     4965.6
 05000  110.77     3688.1
 09000  110.09     2294.9
 20000  109.83     1362.6

Measuring cache-to-cache transfer latency (in ns)...
Local Socket L2->L2 HIT  latency        73.8
Local Socket L2->L2 HITM latency        75.0
```

Container Instantiation

Refer to these two Intel guides to build a container environment for the FlexRAN™ containers.
https://networkbuilders.intel.com/solutionslibrary/network-and-edge-reference-system-architectures-vran-setup-flexran-software-quick-start-guide

https://hub.docker.com/r/intel/flexran_l1_spree

Note. It is not necessary to install a FlexRAN™ build environment if using Intel's pre-built FlexRAN™ container images that can be pulled from the Docker github above. You will need to have a Docker Hub account in order to pull this Docker image.

FlexRAN™ Test Cases

Pull the containers for FlexRAN™ testing on 4th Gen Intel® Xeon Scalable with Intel vRAN Boost from Intel's Docker Github.

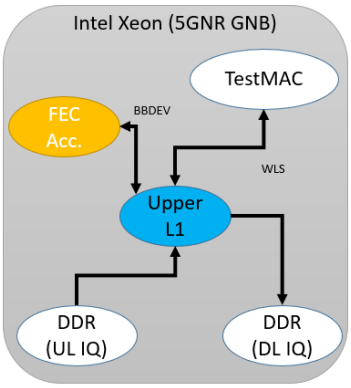
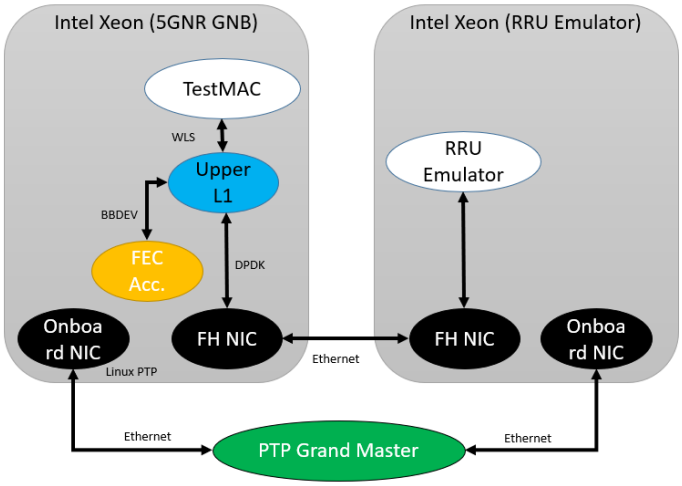
https://hub.docker.com/r/intel/flexran_l1_spree

Use docker to pull the image.

docker pull intel/flexran_l1_spree:v23.11

Intel FlexRAN™ testing supports both timer mode and xran mode. Timer mode was selected to keep the test hardware to a minimum as test run in a single server however, this does not test the PTP timing and Fronthaul NICs.

Timer mode vs Xran mode

Testing Method	Timer Mode Testing	O-RAN Mode Testing
Picture		
LDPC FEC Offload	vRAN Boost	vRAN Boost
FH NIC Card	N/A	Intel E810 Ethernet Controller

Create the flexran-dockerimage-release pod. The command-line included in the Intel L1 and L2 timer mode FlexRAN™ container creation will automatically run a series of FlexRAN™ testcases, alternatively testcases can be selected and run manually.

Manually controlling the FlexRAN™ test cases

Window 1, start the timer mode Pod and monitor the baremetal with htop
kubectl create -f flexran_timer_mode.yaml
htop

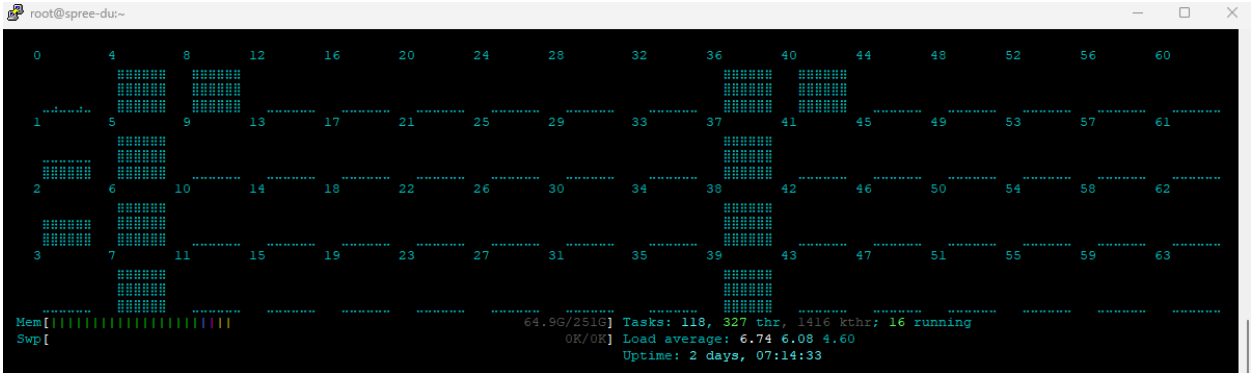
Window 2, connect to the l1app container
kubectl -c flexran-l1app exec -it flexran-dockerimage-release -- bash
root@flexran-dockerimage-release:/home/flexran# ./docker_entry.sh -m timer
root@flexran-dockerimage-release:/home/flexran/flexran/bin/nr5g/gnb/l1# ./l1.sh -e

Window 3, connect to the testmac container
kubectl -c flexran-testmac exec -it flexran-dockerimage-release -- bash
root@flexran-dockerimage-release:/home/flexran# ./docker_entry.sh -m timer
root@flexran-dockerimage-release:/home/flexran/flexran/bin/nr5g/gnb/testmac# ./l2.sh --testfile=spr-sp-mcc/sprsp.cfg

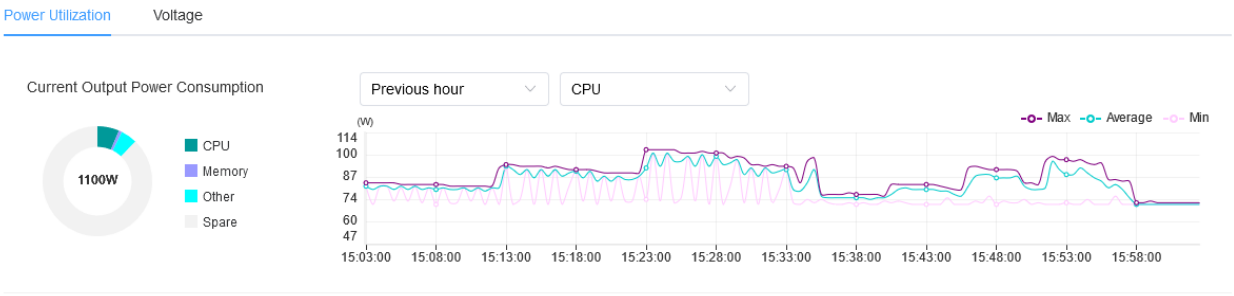
The all-in-one testcase configuration file “sprsp.cfg” runs 79 FlexRAN™ test cases using multiple cell configurations and prints the result on completion.

```
All Tests Completed, Total run 79 Tests, PASS 79 Tests, and FAIL 0 Tests
--------------------------------------------------------------------------------
mem_mgr_display_size:
        HEAP Buffers Alloc:       792 /         Size: 15,903,424,662
        DPDK Buffers Alloc:         0 /         Size:               0
        XRAN Buffers Alloc:         0 /         Size:               0
       TOTAL Buffers Alloc:       792 /         Size: 15,903,424,662
--------------------------------------------------------------------------------
```

Individual vCPUs load from L1 and testmac containers monitored from htop during test run.



Overall CPU Power consumption from baseband monitored from XCC during a run of testmac in timer mode.



In Conclusion

Lenovo and Intel have successfully developed a high-performance VRC for the FlexRAN™ reference software as a workload, utilizing the Lenovo SR630 V3 powered by the 4th Gen Intel® Xeon® Scalable processors with Intel® vRAN Boost Technology.

This collaborative effort is outlined in the paper, highlighting the various system tests conducted to verify the expected performance of VRC vRAN configurations.

As the telecommunications landscape undergoes continuous evolution driven by operator and end-user requirements, the close partnership between Intel and Lenovo will continue to deliver the most advanced technologies. The shared objectives of this collaboration include mitigating risks associated with deploying open solutions on servers like the Lenovo SR630 V3, simplifying complexities through Intel innovation, and ultimately delivering optimal performance, flexibility, and improved Total Cost of Ownership (TCO) for customers.