WHITE PAPER

5G L2 SW Architecture Best Practice on IA

Intel® FlexRAN reference architecture in the 5GNR

Authors

Ziyi Li
Fan He
Peng Huang
Minjun Li
Leifeng Ruan
Yao Dong

Table of Contents

Background

5G is the next technological revolution. Transformed 5G networks are designed to support a broad range of devices and use cases, with faster speeds, less latency, and more capacity. However, 5G is confronted with many challenges, first of which is TTI (Transmission Time Interval) decreasing from 1ms to 0.125ms while data rate boosting by 10 folds. This demands sharply improved processing capacity of base station, and rigorous real-time performance of software stack. In this context, the architecture design of L2 packet processing, and its implementation and optimization on x86 server platform are vital to fulfilling the 5G throughput and real-time performance target.

Huge throughput demand of 5G brings a great challenge to optimizing locking/unlocking overhead, and translation lookaside buffer (TLB) miss when multiple tasks/threads get, split or release memory. For MAC/RLC packet processing, Intel adopts DPDK Mempool to buffer management, where the memory is mapped into hugepage. The IA-friendly BBUpooling is also adopted as the framework of FlexRAN L2+ reference library. In this regard, Intel® FlexRAN reference architecture is a solution Intel has offered for the base station.

Intel® FlexRAN reference architecture is an off-the-shelf general-purpose x86 server system, also a virtualized platform containing components of Intel processors, I/O and FPGAs. This reference architecture enables the highest level of flexibility with the programmable on board features, memory and I/O. The FlexRAN scales from small to large capacities with the same set of components running different applications or functions, ranging from the RAN to core network and data centre including edge computing and media, enabling economics of scale.

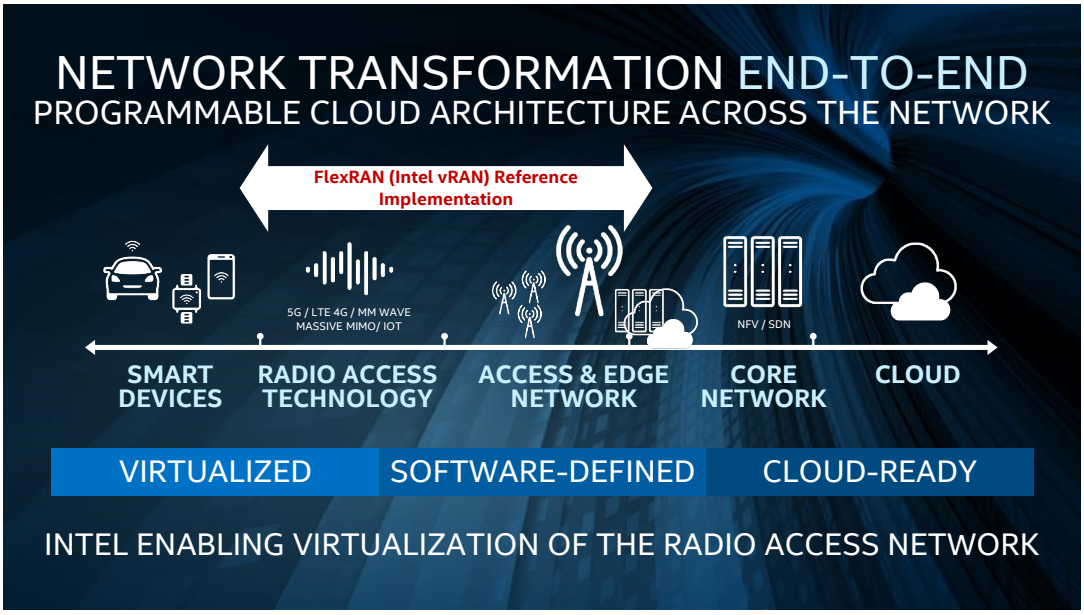

Figure 1. FlexRAN: Intel's 5G Network Reference Architecture

Buffer Management under Hugepage

High performance buffer management is the key performance indicator for packet processing, especially for NR with high throughput and packet rate. FlexRAN L2+ reference library applies Intel Hugepage to reduce high TLB miss. Enabling hugepage makes it possible for the operating system to support memory pages greater than the default (usually 4 KB). Along with the increase of program application size and memory used by application, it will greatly increase the frequency of using TLB, which leads to TLB miss. Using very large page sizes can improve system performance by reducing the amount of system resources required to access memory. Hugepage sizes vary from 2 MB to 1 GB, depending on the operating system and hardware architecture. Intel uses hugepage to reduce the operating system maintenance of page states, and increase Translation Lookaside Buffer (TLB) hit ratio.

Intel® FlexRAN L2+ reference library adopts DPDK Mempool and mbuf as the base of buffer management, where Hugepage is also applied. This chapter focus on zero-copy based 5GNR MAC/RLC packet processing application on DPDK Mempool.

The workflow at the transmission side of 5GNR MAC/RLC packet processing is described in the following six steps:

Step 1: Get blocks from Mempool for "SDU" when receiving packets from the upper layer, and use rte_th_rx_burst()/ rte_pktmbuf_allocate(); the element size is the sum of MAC header maximum length, RLC header maximum length, and packet length.

Step 2: Use rte_pktmbuf_chain() to put all packets in an array, wait for the transmission grant from MAC, and pre-add RLC headers according to packet length, etc.

Step 3: After receiving the transmission grant from MAC scheduler, use rte_pktmbuf_chain() to concatenate packets into MAC PDU.

Step 4: If segmentation is needed, use rte_pktmbuf_clone() and rte_pktmbuf_attach() to split the packet into two segments; the second segment will be stored in the original buffer, and needs to be sent out within the current slot, and allocated into the clone buffer. DPDK will maintain the reference count for each mbuf.

Step 5: If AM is applied, use rte_pktmbuf_clone() to copy packets into the transmission queue and wait for ARQ's feedback.

Step 6: Use rte_pktmbuf_free() to free allocated buffer for RLC packets, and transmit to PHY according to MAC/PHY interface.

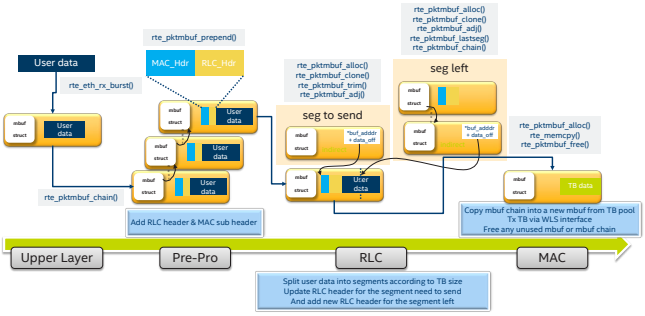

Figure 2. Transmission Workflow of 5GNR MAC/RLC Packet Processing

The workflow at the receiving side of 5GNR MAC/RLC packet processing is described in the following five steps:

Step 1: Use rte_pktmbuf_alloc() to allocate buffer for MAC PDU after receiving packets from PHY.

Step 2: Split MAC PDU into MAC sub-PDU according to 3GPP TS38.321, then deliver it to RLC by using pointer offset to identify RLC header position within the buffer.

Step 3: Use rte_pktmbuf_trim() and rte_pktmbuf_adj() to re-assemble segments into a complete RLC SDU.

Step 4: If AM is applied, and NACK has been received for certain packets, its buffer pointer in the transmission queue will be connected into a retransmission queue and wait for next transmission grant to be resent.

Step 5: Send to the upper layer with rte_eth_tx_burst() or according to interface between RLC and upper layer.

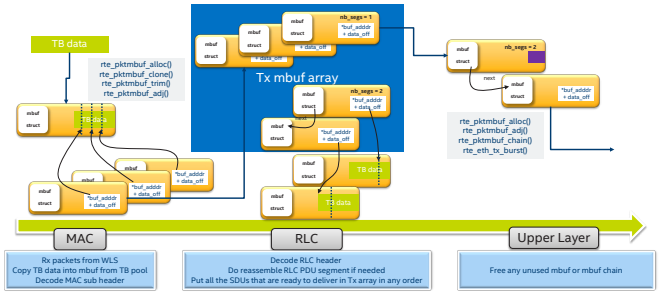

Figure 3. Receiving Process of 5GNR MAC/RLC Packet Processing

System Architecture

This section illustrates the system architecture of FlexRAN L2+, including the BBUpooling framework, multi-cell pooling mode, task split for packet processing, and L2 BKMs.

BBUpooling Framework

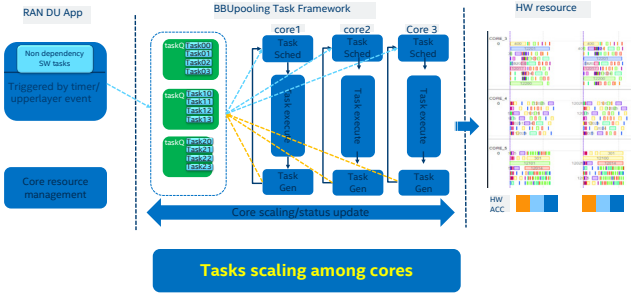

Figure 4. BBUpooling Task Framework

The BBUpooling task framework is optimized to better serve the radio access network (RAN). It helps customers leverage IA computing resource to design high-efficient software, and allows them to tap IA general-purpose processing and virtualization to design software with high flexibility.

The BBUpooling framework has been proved most efficient in helping RAN application with parallel and pooling. It has the following features:

- Breaking down the system into reasonable sizes of tasks that can be executed parallel.
- A task consists of the algorithms which are executed against the data.
- A task might depend on another task, and the whole system contains a series of chained tasks.
- As tasks are prioritized and time sensitive, the priority of tasks may or may not update dynamically.

5G L2+ Reference Design

This chart describes the multi-cell MAC/RLC design based on BBUpooling framework. Each cell is constructed with the pipeline of packet processing and runs in parallel.

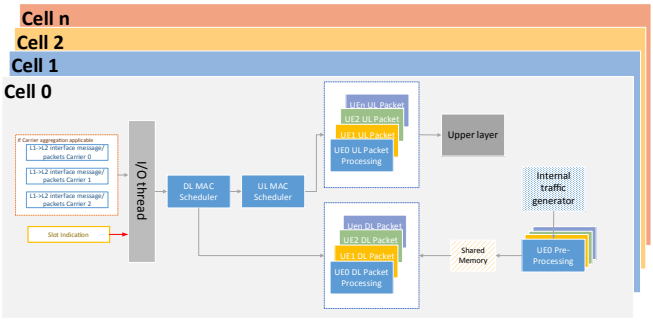

Figure 5. MAC/RLC Design Based on BBUpooling Framework

- DPDK timing transmits through IO thread, and IO thread will bind to a dedicated core.
- This architecture is based on a case using internal traffic generator, and the traffic generator function is included in DL RLC pre-processing task.
- RT DL packet processing task and RT UL packet processing task can be executed parallel on different cores between UEs at the time of high throughput and sufficient core resources. It is recommended to split the task processing time around 50k cycles to avoid task switch overhead.

The advantages of BKMs and BBUpooling have been proven through FlexRAN L2+ reference library:

- It is recommended to pre-define task dependency, rather than generate next-level task after current task.
- Cores belonging to the same socket can be grouped into one queue to avoid cache miss. Multiple cells can share the same task queue.
- Set task priority to pre-fetch tasks in the core, in order to reduce cache miss and scheduler overhead.

Optimization Method

Intel® C++ Compiler (ICC)

Intel® C++ Compiler (ICC) is a group of C and C++ compilers from Intel available for Windows, Linux, and Intel-based devices. Following BKMs are recommended during performance tuning.

1. Compilation without optimization

   First of all, the performance tuning needs to be based on a qualified application. You need to ensure the application correctness before you start the performance tuning.

2. Enable the general optimization

   Enable the commonest compiler optimization in this step. You have multiple options to select according to your application scenarios, for example, ICC option (-fast) maximizes speed across the entire program.

3. Enable processor-specific optimization

   If you have a specific target processor for your application, you can use option -x<code> (Linux*) to enable the processor-specific optimization. This option tells the compiler which processor features it may target to optimize, including which instruction sets and optimizations it may generate. It also enables optimizations in addition to Intel® feature-specific optimization.

4. Enable IPO optimization

   Interprocedural Optimization (IPO) is an automatic, multi-step process. It allows the compiler to analyze your code to determine where you can benefit from specific optimizations. With the IPO option, you may achieve additional optimizations.

5. Enable PGO optimization

   Profile-guided Optimization (PGO) improves application performance by reorganizing code layout to reduce instruction-cache problems, shrinking code size, and reducing branch mispredictions. PGO provides information to the compiler about most frequently executed areas of an application. Knowing these areas, the compiler will be more selective and specific in optimizing the application.

6. Enable optimization report option

   This option tells the compiler to generate an optimization report, and indicates how detailed it should be. Specify values 0 through 5. If you specify zero, no report will be generated. For levels n=1 through n=5, each level contains all information of the previous level, perhaps including some other information. Level 5 generates the most detailed report.

Intel® VTune™ Amplifier

Intel® VTune™ Amplifier is a performance analysis tool for users to develop serial and multithreaded applications. It helps you analyze the algorithm choices and identify where and how your application can benefit from available hardware resources.

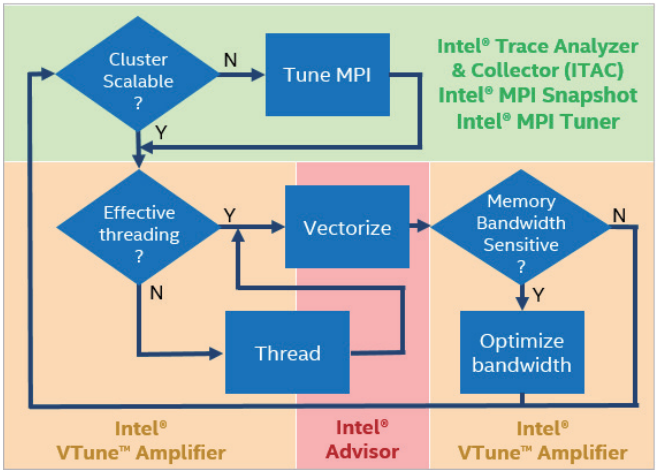

Figure 6. Typical Optimization Cycle

Intel provides these performance analysis tools that can help to go through this performance optimization workflow. Please ignore the top two elements if you are not running on a cluster. You may start from any point according to your application scenarios.

For example, Intel® Trace Analyzer & Collector (ITAC) is a graphical tool to understand MPI application behaviors, quickly identify bottlenecks, improve correctness, and achieve high performance for parallel cluster applications running on Intel® architecture. It improves weak points and strong scalability among different applications.

Supported Configuration

Based on Intel® Xeon® Gold 6148 CPU @ 2.40GHz, FlexRAN L2+ reference library supports MAC/RLC packet processing for both mmw and sub-6GHz. Supported configurations are listed as below for reference.

Feature	mmw	sub-6GHz
Slot length (us)	125	500
Specification	TS 38.321 R15.0.0; TS 38.322 R15.0.0; TS 38.214 R15.0.0	
Bandwidth per carrier	100MHz	
Cell number	6	
Active user number/cell	400	
DL scheduled user number/cell	16	
UL scheduled user number/cell	8	
DL Layer number/cell	16	
UL Layer number/cell	8	
TDD configuration	DL:UL = 4:1	
Packet Size (Byte)	1480; IMIX 340	
RLC Mode	TM, UM, AM	

Table 1. FlexRAN L2+ Reference Library Configuration

Summary

5G network demands high-level flexibility and scalability to meet the latency, coverage, capacity and various algorithms, analytics and application needs. Intel® FlexRAN reference architecture can run different workloads on different x86 server platforms due to the general purpose nature of its architecture. Except L1 signal processing of 5G, Intel® FlexRAN reference architecture also explores L2 to give customers a one-stop Intel L1+L2 reference design on x86 server platform. Working diligently and improving step by step, Intel is innovating rapidly to build 5G networks that will usher in a seamlessly connected, powerfully smart future.

1218/AM/BCS/PDF ♻ Please Recycle 338556-001US