Performance Profiling of Cloud Radio Access Networks using OpenAirInterface

Po-Chiang Lin[1] and Sheng-Lun Huang[2]
[1]Department of Electrical Engineering, Yuan Ze University, Taoyuan, Taiwan, R.O.C.
[2]Department of Communications Engineering, Yuan Ze University, Taoyuan, Taiwan, R.O.C.
Email: pclin@saturn.yzu.edu.tw Tel/Fax: +886-3-4638800 Ext. 7333

***Abstract*—NGFI, the Next Generation Fronthaul Interface, is a promising fronthaul interface for the C-RAN (Cloud Radio Access Network). NGFI is used to connect RCC (Radio Cloud Center) and RRS (Radio Remote System) in order to avoid the drawbacks of traditional CPRI (Common Public Radio Interface). In this paper we investigate the NGFI-based C-RAN. We use the OpenAirInterface (OAI) open source 4G/5G mobile communication software and GPP (general purpose processor) based servers and personal computers to build an OAI C-RAN testbed. We also use the source codes of OAI to run the performance profiling on this OAI C-RAN testbed to understand the behavior of this testbed. The purpose of this paper is to build the comprehensive performance profiling methods and results on the OAI C-RAN system, and to use these results to help designing and optimizing the OAI C-RAN system. Based on the results, we could decide which part of the system software to optimize to improve the system speed and the efficiency of memory usage.**

***Index Terms*—5G, LTE, OpenAirInterface.**

I. Introduction

Cloud Radio Access Network (C-RAN) is a novel mobile communication network architecture. The main concept of C-RAN is to split the functionalities of the base stations into two parts, including the baseband unit (BBU) which is in charge of the baseband signal processing, and the remote radio unit (RRU) which is in charge of the radio-frequency signal processing. The capability of baseband signal processing are aggregated in centralized servers, also known as the BBU pool, instead of distributed base stations. The RRU at the remote site handles the RF signal processing such as analog / digital conversion, power amplifying, and filtering. As the software defined radio technologies evolves, the BBU functionalities could be implemented by software. Therefore, current trend of C-RAN integrates the cloud computing and virtualization technologies in order to dynamically allocate BBU computing resources.

The operation of C-RAN relies on the high-speed transmission of the in-phase and quadrature signals between BBU and RRU. This transmission link is referred to as the *fronthaul*, in order to be distinguished with the *backhaul* which locates between the traditional base stations and the core networks [1]. Currently, there exist several standards for the fronthaul, including the Common Public Radio Interface (CPRI), the Open Radio equipment Interface (ORI) by the European Telecommunications Standards Institute (ETSI), and the Open Base Station Architecture Initiative (OBSAI). Among those fronthaul standards, CPRI is the most commonly used one. However, CPRI uses fibers as the point-to-point transmission media, which are not flexible, not efficient, difficult to extend, and with a huge amount of cost. Those disadvantages affect the progress and acceptance of CPRI-based C-RAN in countries/regions where the fiber infrastructure is not popular [2].

In order to overcome the disadvantages mentioned above, several telecom vendors and operators cooperated to propose the Next Generation Fronthaul Interface (NGFI) [2], [3]. NGFI is a promising fronthaul interface for the C-RAN with two important properties. First, NGFI redefines the functionalities of BBU and RRU. Some portions of the BBU functionalities could be optionally moved to RRU. Therefore, BBU and RRU are redefined as the Radio Cloud Center (RCC) and Radio Remote System (RRS), respectively. Moreover, NGFI replaces the point-to-point fiber transmission by the more popular point-to-multipoint Ethernet transmission.

In this paper we investigate theperformance profiling of NGFI-based C-RAN. We use the OpenAirInterface (OAI) open source 4G/5G mobile communication software and GPP (general purpose processor) based servers and personal computers to build an OAI C-RAN testbed. The OAI project focuses on open source software/hardware development for both the radio access networks and core networks of 3GPP mobile communication networks [4]. In the OAI project, off-the-shelf radio-frequency front-ends and personal computers are adopted, and comprehensive 3GPP protocols are implemented. Combining the affordable hardware and the open-source software, OAI provides a sound foundation on which we could improve our teaching and learning on mobile communications.

We also use the source codes of OAI to run the performance profiling on this OAI C-RAN testbed to understand the behavior of this testbed. The purpose of this paper is to build the comprehensive performance profiling methods and results on the OAI C-RAN system, and to use these results to help designing and optimizing the OAI C-RAN system. Based on the results, we could decide which part of the system software to optimize to improve the system speed and the efficiency of memory usage.

The rest of this paper is organized as follows. In Section II we provide the related work in the literature. The system architecture and the profiling tools that we adopt in this work are presented in Section III. The numerical results and discussions are provided in Section IV. Finally, conclusions

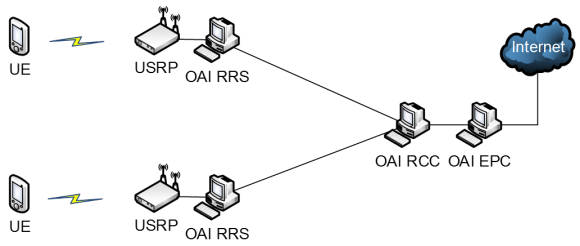

Fig. 1. OAI C-RAN architecture.

are presented in Section V.

II. Related Work

Nikaein et al. proposed the Radio Access Network as a Service (RANaaS) [5]. They described using OAI and off-the-shelf computer hardware to build cloud-based LTE platform, and they discussed the feasibility and performance of this platform. They adopted the OAI simulator to perform all experiments, so that the performance profiling results in their work could not represent the real operational performance of OAI C-RAN.

Virdis et al. provided their OAI performance profiling results in [6]. First they described their emulator environment and the scheduling architecture, which was used to design and to implement two medium access control (MAC) scheduling algorithms. Moreover, they also verified these two MAC scheduling algorithms by running OAI performance profiling, including memory requirement and execution time. Their results showed that OAI could effectively build and verify scheduling prototype in LTE emulator environments. Because their work use the OAI simulation environment called the oaisim, their performance profiling results, like the work in [5], could not represent the real operational performance of OAI C-RAN.

III. System Architecture and Profiling Tools

Figure 1 shows the OAI C-RAN architecture, including the OAI EPC (evolved packet core), RCC, RRS, and off-the-shelf UE (user equipment). Four personal computers are installed with Ubuntu Linux 14.04 and OAI software to run as the OAI EPC, RCC, and two RRS's, respectively. The universal soft radio peripheral (USRP) B210 is adopted as a radio-frequency front-end. We also adopt two smartphones with Rohde & Schwarz SwissQual QualiPoc software as the UE.

The performance measurement includes the CPU usage, memory usage, network flow for different user behavior. The profiling tools that we adopt in this work are presented in the following paragraphs.

A. perf

perf is one of the most commonly used performance counter profiling tool for Linux systems [7]. It provides a simple command-line interface for Linux performance measurements. It supports hardware performance counters, software performance counters, tracepoints, and dynamic probes.

B. iperf

iperf is a tool to actively measure the network performance of IP networks [8]. It supports various parameters, including protocols (TCP, UDP, and so on) and timing. It reports the throughput, loss, and other parameters in each measurement test.

C. Dstat

Dstat is a versatile tool for generating system resource statistics [9]. It allows users to view all of the system resources instantly. It provides the detailed information in columns, and indicates the magnitude and unit that the output is displayed with. It can also export data to CSV files for further processing.

D. top

The top program provides a real-time view of a running Linux system [10]. It displays the system summary information and the list of tasks currently running in the Linux system.

IV. Numerical Results and Discussions

A. CPU Usage for Different User Behavior

We use the perf software to measure the CPU usage for different user behavior. Figure 2 to 5 show the CPU usage for idle, YouTube, Skype, and live mode, respectively. Here the live mode means that the traffic is generated from the UE and transmitted uplink to the Internet, while in the YouTube mode the traffic is downlink. These figures show that the CPU usage for idle, YouTube, and Skype mode are of no significant difference. The main system functions that consume the majority of CPU usage are taus, generate_dci_top, send_IF4p5, recv_IF4p5, phy_procedures_eNB_TX, and rx_pucch. For the live mode, some more functions for uplink also occupy the majority of CPU usage, including ulsch_decoding, phy_threegpplte_turbo_decoder16, compute_alpha16, compute_beta16, and compute_ext16.

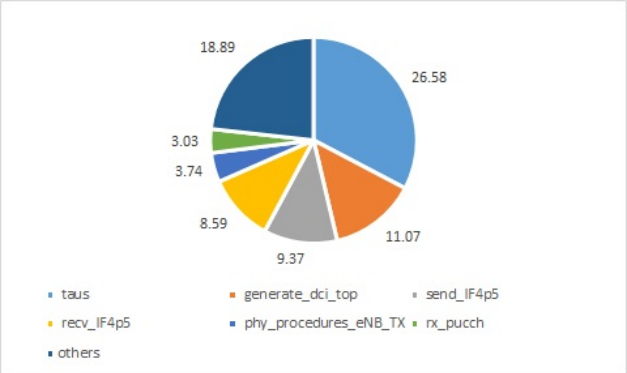

Fig. 2. CPU usage for idle mode.

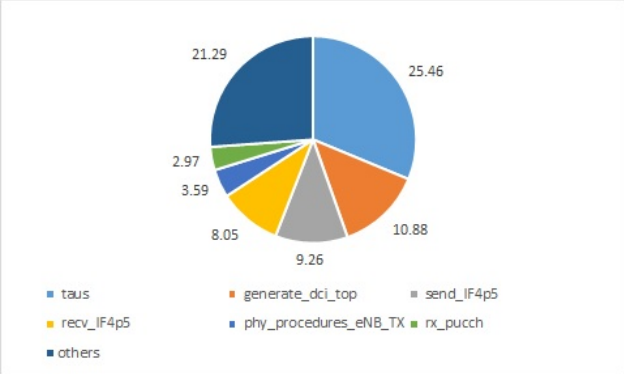

Fig. 3. CPU usage for YouTube mode.

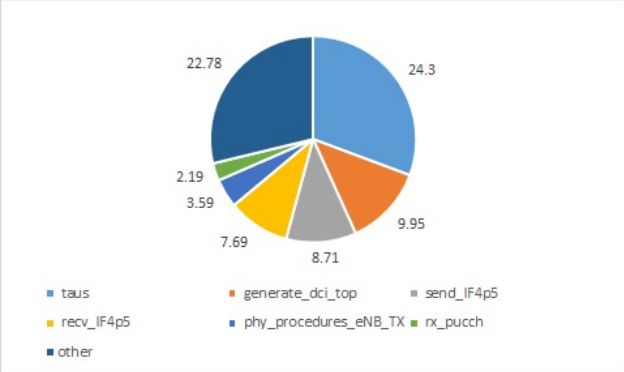

Fig. 4. CPU usage for Skype mode.

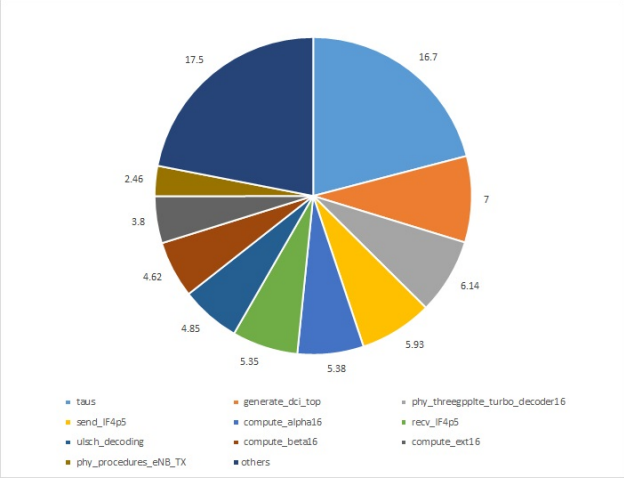

Fig. 5. CPU usage for Live mode.

B. CPU, Memory, and Network Usage in C-RAN

Figure 6 to 11 show the CPU, memory, and network usage in C-RAN. In the horizontal axis, the term "no" means that the OAI program is not running. Figure 6 and 7 show that, when the RCC runs the OAI program, the CPU usage increases from 13% to 42% as the iperf transmission bandwidth increases. Meanwhile, the memory usage is about 1 GB, regardless of the iperf transmission bandwidth. Figure 8 and 9 show that, when the RCC runs the OAI program, the CPU usage is about 35% and the memory usage is about 0.5 GB. These two values do not change with the increase of the iperf transmission bandwidth. When the EPC runs the OAI program, the CPU usage is only 0.1% to 0.3%, and the memory usage is neglectable. For the sake of simplicity, we do not show the illustrative results for EPC. Figure 10 and 11 show that the fronthaul occupies about 70 Mbps traffic load for both directions. This value is fixed, no matter there exists user traffic or not. Note that we obtain this result when the number of resource blocks is set as 25, i.e., 5 MHz bandwidth. When the number of resource blocks increases to 50, i.e., 10 MHz bandwidth, the amount of traffic load in the fronthaul increases to about 140 Mbps.

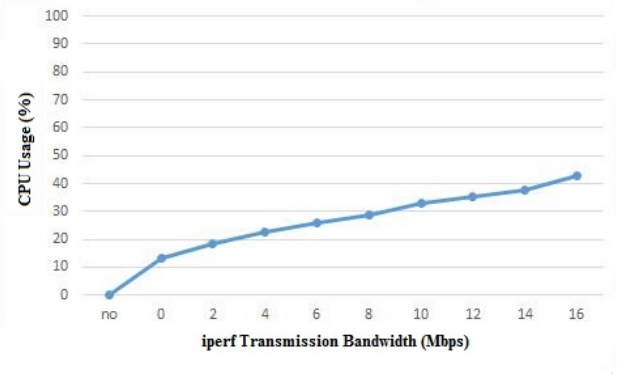

Fig. 6. CPU usage in RCC.

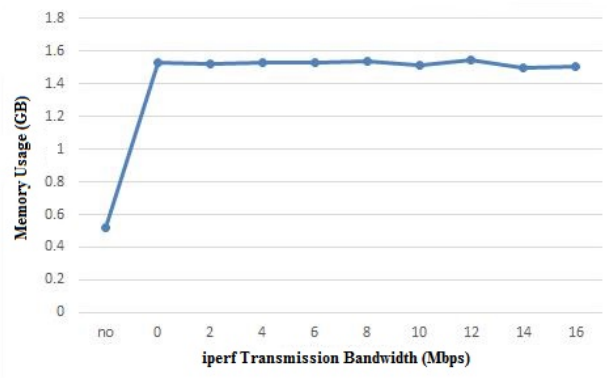

Fig. 7. Memory usage in RCC.

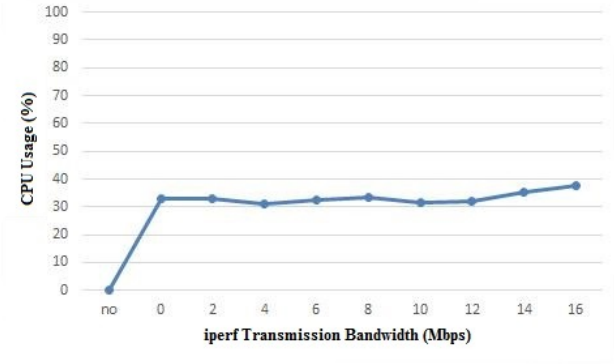

Fig. 8. CPU usage in RRS.

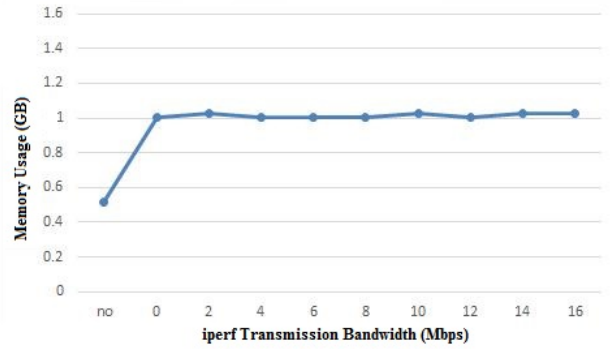

Fig. 9. Memory usage in RRS.

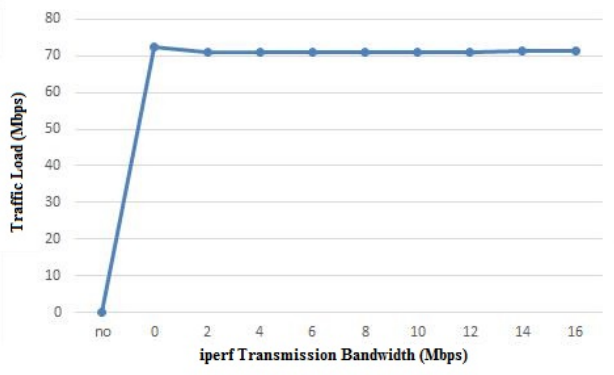

Fig. 10. Traffic load from RCC to RRS.

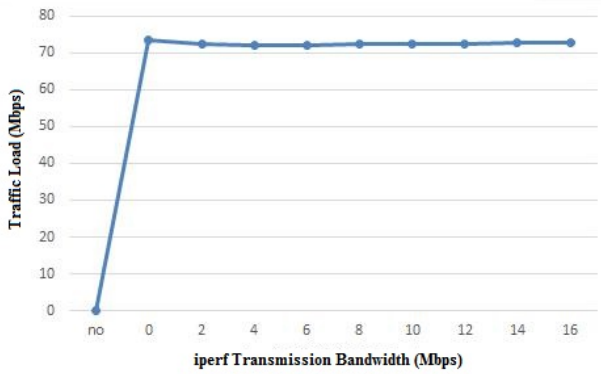

Fig. 11. Traffic load from RRS to RCC.

V. Conclusion

In this paper we investigate the NGFI-based C-RAN. We use OAI open source 4G/5G mobile communication software and GPP-based servers and personal computers to build an OAI C-RAN testbed. We also use the source codes of OAI to run the performance profiling on this OAI C-RAN testbed to understand the behavior of this testbed. The purpose of this paper is to build the comprehensive performance profiling methods and results on the OAI C-RAN system, and to use these results to help designing and optimizing the OAI C-RAN system. Based on the results, we could decide which part of the system software to optimize to improve the system speed and the efficiency of memory usage.

The results of our work show that the CPU and memory usages of RRS in OAI C-RAN are not huge. For mass deployment of RRS in the future, the RRS could be installed and run on cost-effective low-end computers. The CPU and memory usages of RCC are significant. Therefore, it requires suitable design, virtualization technologies, and resource allocation algorithms to effectively handle the RCC processing. The fronthaul bandwidth occupation is fixed in the current implementation of OAI C-RAN. Making the fronthaul bandwidth occupation dynamically adjust to user traffic would be a critical issue.

References

[1] M. Jaber, M. A. Imran, R. Tafazolli, and A. Tukmanov, "5G Backhaul Challenges and Emerging Research Directions: A Survey," *IEEE Access*, vol. 4, pp. 1743–1766, 2016.

[2] *White Paper of Next Generation Fronthaul Interface, Version 1.0*, China Mobile Research Institute, 2015.

[3] Next Generation Fronthaul Interface (1914) Working Group. http://sites.ieee.org/sagroups-1914. Accessed: 2018-05-31.

[4] OpenAirInterface. http://www.openairinterface.org. Accessed: 2018-05-31.

[5] N. Nikaein, E. Schiller, R. Favraud, R. Knopp, I. Alyafawi, and T. Braun, *Towards a Cloud-Native Radio Access Network*. Springer International Publishing, 2017, pp. 171–202.

[6] A. Virdis, N. Iardella, G. Stea, and D. Sabella, "Performance Analysis of OpenAirInterface System Emulation," in *2015 3rd International Conference on Future Internet of Things and Cloud*, Aug 2015, pp. 662–669.

[7] perf. https://perf.wiki.kernel.org. Accessed: 2018-05-31.

[8] iperf. https://iperf.fr. Accessed: 2018-05-31.
[9] Dstat. https://github.com/dagwieers/dstat. Accessed: 2018-05-31.
[10] top. http://man7.org/linux/man-pages/man1/top.1.html. Accessed: 2018-05-31.