Towards the deployment of a fully centralized Cloud-RAN architecture

Veronica Karina Quintuna Rodriguez, Fabrice Guillemin

Towards the deployment of a fully centralized Cloud-RAN architecture

Veronica Quintuna Rodriguez and Fabrice Guillemin
Orange Labs, 2 Avenue Pierre Marzin, 22300 Lannion, France
{veronicakarina.quintunarodriguez,fabrice.guillemin}@orange.com

***Abstract*—In the framework of Network Function Virtualization (NFV), we address in this work the design and sizing of Cloud-RAN architectures. We concretely investigate the execution time of software-based Base Band Units (BBUs) on multi-core systems. Since Cloud-RAN requires a real-time behavior, we use parallel programming techniques in order to minimize the runtime of BBU functions. For an efficient utilization of computing resources, we investigate the relevance of resource pooling where a global scheduling algorithm allocates processing units to runnable BBU-jobs. We specifically examine the gain that can be obtained when applying data parallelism on the channel decoding BBU-function which is the most expensive one in terms of processing time. Performance results show a significant reduction in the runtime of PHY functions which enables the deployment of a fully centralized Cloud-RAN architecture.**

Keywords: Cloud-RAN, NFV, VNF, BBU, channel decoding, multi-core, global scheduling.

I. Introduction

Network Function Virtualization (NFV) [1] is an emerging paradigm which advocates the separation between network functions and hosting hardware. This approach is intended to make the network architecture and operation more flexible since network functions can be instantiated on the fly in cloud infrastructures and no more on dedicated servers via complex procedures. Evolved Packet Core (EPC), IP Multimedia Subsystem (IMS), etc. are typical candidates for virtualization. In this paper, we pay special attention to Radio Access Network (RAN) functions. The cloudification of RAN functions is of utmost interest for several reasons. In the context of network slicing, RAN functions could be instantiated on the fly for a specific use case (e.g., a temporary cellular network for the needs of a company). Cloud-RAN enables the deployment of radio networks with a tailored behavior for a particular service or client, i.e., RAN as a Service (RANaaS).

Cloud-RAN is based on network functions which are defined in software. They run on general purpose hardware, e.g., GPU/CPU based servers. The cloudification of RAN can be seen as a key architectural evolution of mobile networks (especially in areas with a dense mesh of small cells). It allows hundreds of remote Radio Remote Heads (RRHs) to be connected to a centralized BBU-pool hosted in a Central Office (CO).

For a network operator, the main advantage of Cloud-RAN is the native support of collaborative radio technologies such as Coordinated Multi-point (CoMP), coordinated beam-forming, joint transmission, among others. These advanced algorithms are difficult to implement in current mobile architectures due to the latency between cell sites. Resource saving is another important benefit of this evolution. Energy consumption reduction is possible via BBU pooling and even thanks to coordinated ON/OFF switching strategies of eNodeBs (eNBs). Computing resources in data centers can also be saved via resource pooling. The key point is to apply statistical multiplexing when dimensioning future COs.

Centralizing BBUs brings some challenges, a key one being the front-haul sizing. The front-haul dimension represents the distance between the CO (BBUs) and antennas (RRHs). This size is constrained by the Hybrid Automatic Repeat reQuest (HARQ) process defined in LTE, i.e., the Round Trip Time (RTT). As a matter of fact, Cloud-RAN requires a real-time behavior. The BBU should complete the base-band processing within 3 ms where 2 ms are available for the reception process and 1 ms for the transmission process. The front-haul capacity should also be studied. It depends on the configuration of each eNB hosted in the BBU-pool (e.g., MIMO, SISO, LTE bandwidth of 5 MHz, 20 MHz, etc).

The time budget (1 ms and 2 ms for down-link and up-link, respectively) should actually be shared between the propagation time and the processing time. If the processing time is reduced, the front-haul size can be increased and BBU functions can be located higher in the network, typically in Points of Presence at the edge of an IP network.

In [2] a threading model has been proposed for reducing the execution time of BBU functions in the down-link (DL) direction, notably the Channel Coding function. In the present work, we continue along the same line of investigation by introducing a solution to minimize the execution time of BBU functions in the up-link (UL) direction. We concretely investigate the execution of BBU functions on multi-core systems when applying parallel programming. First, we identify the driving factors which determine the runtime of BBU functions. Then, we implement a threading model with data parallelism in order to decrease the execution time of BBU functions in the up-link direction. Finally, we evaluate by simulation the performance of virtualized BBU functions in terms of latency (i.e., processing time). The knowledge of runtime of BBU functions is needed for dimensioning the Cloud-RAN architecture, i.e., the computing resources of data centers, the front-haul size and capacity.

This document is organized as follows: The Cloud-RAN architecture is discussed in Section II. The runtime of BBU functions and parallelization are described in Section III. The performance analysis in terms of latency is presented in Section IV. Concluding remarks are presented in Section V.

II. Cloud-RAN architecture

Current mobile networks have a distributed architecture where BBUs are located near to antennas. BBUs are implemented on proprietary hardware and are provided by a single vendor. On the contrary, Cloud-RAN architectures are based on open-platforms where the base-band functions can be instantiated on demand. In the same way the computing resources can be dynamically allocated.

The Cloud-RAN architecture can support selective centralization of BBU functions. Several functional splits of the BBU have been widely considered in the literature [3], [4], [5]. We can roughly classify Cloud-RAN architectures as fully and partially centralized; in this paper, we focus our study in the case of full centralization which moves all base-band functions (BBUs) higher in the network. See Figure 1 for an illustration.

A. BBU-pool

The BBU-pool is deployed on commodity hardware, i.e., multi-core GPU/CPU-based servers. Base-band functions are defined in software and run as applications. Cloud-RAN employs virtualization technologies which can be based on Virtual Machines (VMs) and/or containers. In this work, we take advantage of the performance provided by containers which, unlike VMs run on a common single kernel. This gives them the benefit of being faster and more resource-efficient. This point has been studied in [6].

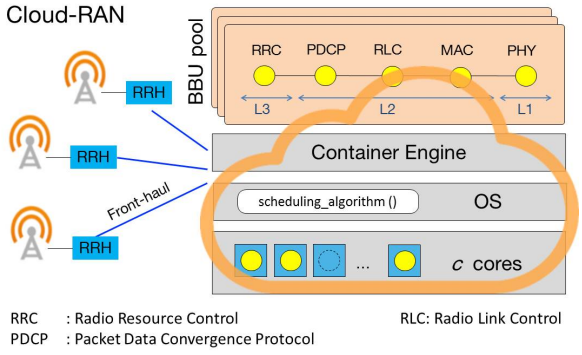


Fig. 1. Fully-centralized Cloud-RAN architecture.

In a container-based Cloud-RAN platform, BBU functions are represented as runnable-tasks (processes or jobs) which are placed in the highest layer of the Cloud-RAN system as shown in Figure 1. These tasks correspond to all physical, MAC and network functionalities, i.e., IFFT/FFT, modulation/demodulation, encoding/decoding, radio scheduling, HARQ management, radio link control, data convergence procedures, and User Equipments (UEs) measurement reporting and paging maintained by the Radio Resource Control (RRC) protocol.

Since strict real-time constraints are present in the execution of BBU functions, the behavior of the Operating System (OS) plays a crucial role in the performance of Cloud-RAN systems. In fact, each BBU-function represents a chunk of jobs or tasks to be scheduled by a global scheduling algorithm embedded into the kernel of the OS. This algorithm determines the order of execution of BBU-jobs and allocates them computing-resources.

B. Front-haul size

The front-haul size, i.e., the distance between the BBU-pool and antennas, is limited by the time-budget of the RTT defined by LTE which includes the acknowledgment of each sub-frame. In LTE, the acknowledgment messages and retransmission procedures in case of errors are handled by the HARQ process.

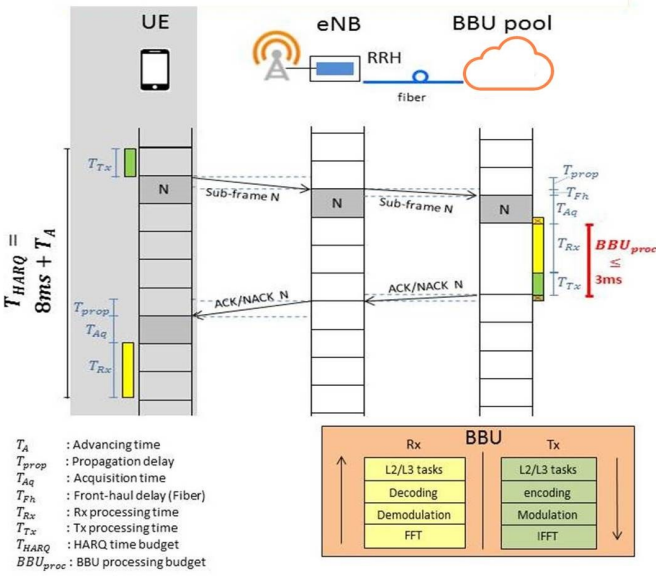


Fig. 2. HARQ process in Cloud-RAN architectures.

As shown in Figure 2, the BBU-pool has less than 3 ms for the whole base-band processing (namely, decoding, checking the Cyclic Redundancy Check (CRC), and encoding the ACK/NACK). The reception process (Rx) has a budget of 2 ms and the transmission process (Tx) 1 ms, denoted by T_{Rx} and T_{Tx}, respectively; the turnaround time is 8 ms.

In fact, to dimension the front-haul size, it is first necessary to know the response time T_r of the BBU-pool, i.e., the required time to execute all BBU functions. Thus, the time-budget for the propagation of IQ signals, i.e., the front-haul delay, is the remaining time after the base-band processing in the BBU-pool. Since the BBU-pool (CO) is linked with antennas by optic-fiber, the front-haul size d can easily be obtained from the front-haul time-budget, so-called front-haul delay T_{Fh}, and the speed of light c, as $d = c * T_{Fh}$.

The HARQ mechanism considers an advancing time T_A in order to align signals in time due to propagation delay between

the UE and the eNB. In LTE there are 8 HARQ processes executed at the same time with an offset of 1 ms each which corresponds to the acquisition time of a sub-frame, T_{Aq}. Thus, $T_{HARQ} = RTT + T_A$, where

$$RTT = T_{Tx} + T_{Rx} + 2 * T_{Aq} + BBU_{proc} - 2 * T_{Fh}.$$

C. Front-haul capacity

The front-haul capacity is defined by the number of eNBs hosted in the CO. The current widely used protocol for data transmission between antennas and BBUs is Common Public Radio Interface (CPRI) which transmits IQ signals. The transmission rate is constant since CPRI is a serial Constant Bit Rate (CBR) interface. It is then independent of the mobile network load [7]. The data throughput depends on the cell configuration and can be obtained as

$$R_{IQ} = 2 * M * f_s * F_{coding} * F_{control} * N_{ant} * N_{sec},$$

where the factor 2 corresponds to I and Q radio signals; M is the number of bits per sample used in the quantisation process (current LTE implementations use $M = 15$ [8]); f_s is the sampling frequency, which is a multiple of the nominal chip rate of LTE, $f_c = 3.84$ MHz, e.g., the sampling frequency for 10 MHz and 20 MHz is 15.36 MHz and 30.72 MHz, respectively; F_{coding} is the line coding factor which can be either $10/8$ or $66/64$ [9]; $F_{control}$ is the control factor which defines the number of data words and control words ($F_{control} = 16/15$ when using CPRI); N_{ant} and N_{sec} are the number of antennas and sectors, respectively.

For example, an eNB with 3 sectors and 8x8 MIMO antennas needs a data rate of 29.49 Gbps for 20 MHz. Research studies are focusing on bandwidth compression techniques and packetisation of CPRI data via Ethernet [10]. Others solutions propose different functional split architectures in order to reduce the required bandwidth [11], [7]. For example, when including the demapping process in the RRH, it is possible to adapt the bandwidth as a function of the traffic load in the cell, then the required front-haul capacity is directly given by the fraction of utilized radio resources [11].

In general, the required data rate significantly decreases when the functional split is shifted after the PHY layer or even after the MAC layer [11].

III. Processing BBU functions in the Cloud

This work is focused on the implementation of BBU functions carried out during the Rx process (up-link). It includes the channel decoding which is the most expensive sub-function in terms of latency [4], [12], [13]. The performance of BBU functions for the Tx process (down-link) is studied in [2].

In order to minimize the runtime of BBU functions, we investigate the relevance of parallel programing and resource pooling in multi-core systems. The main goal of the runtime reduction is to enlarge the distance between the BBU-pool and antennas while saving computing resources.

In [14], namely CloudIQ, is shown that at least 22% of computing resources can be saved only when using statistical multiplexing and resource pooling in Cloud-RAN systems. It exploits the variations in the processing load of individuals BBUs to use fewer computing resources.

While in CloudIQ, an LTE sub-frame is completely processed on a single computing resource, we are interested on splitting the sub-frame processing in parallel runnable tasks in order to decrease the runtime of the entire sub-frame, namely data parallelism. Note that functional parallelism cannot be applied since BBU functions (e.g, decoding, demodulation) need to be executed sequentially.

We propose the parallel processing of sub-frames as follows: The workload of a sub-frame is divided in slices where each of them corresponds to the data of a single UE. These slices, so-called sub-tasks, are executed simultaneously on different cores. For further performance improvement, we propose splitting the workload of a single UE in parallel runnable Code Blocks (CBs).

Hence, the resulting threading model executes one thread per UE, and/or, one thread per CB for the channel decoding function. The threading model is illustrated in Figure 3. A global scheduler allocates a dedicated single core to each sub-task (either per UE or per CB) in order to avoid context switching overhead.

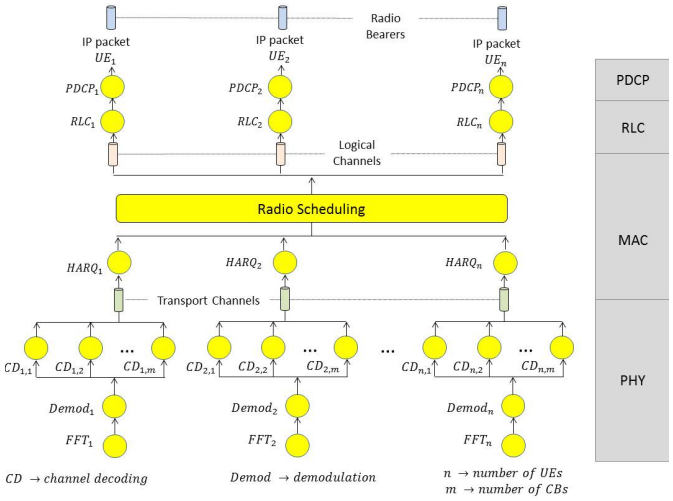


Fig. 3. Threading model of BBU-UL functions using data parallelism.

The Rx base-band processing begins with the demapping process which selects the signal corresponding to a UE in order to demodulate and decode the received data. The MAC layer which handles the radio scheduling and HARQ mechanisms, performs demultiplexing of the Up-Link Shared Channel (UL-SCH) to restore the various logical channels containing the control messages and the IP packets. The size of data per UE in a sub-frame is determined by the radio scheduler which builds the resource grid of a cell. The radio scheduler allocates resources (time and frequency), namely Physical Resource Blocks (PRB), to a UE in function of the data transmission needs and the radio channel quality. The number of allocated PRB together with the Modulation and Coding Scheme (MCS) determine the Transport Block Size (TBS).

The execution time of each PHY function (i.e., FFT, demodulation, and decoding) depends of the TBS attributed to each UE. In other words, when applying data parallelism, the runtime of BBU functions (i.e., the Cloud-RAN performance in terms of latency) depends on the structure of the resource grid which is determined by the radio scheduler. Hence, the radio resource grid represents the workload of a Cloud-RAN system. It is important to note that scheduling strategies are not defined by LTE and as a consequence they are vendor specific. Most common radio schedulers takes into account the Channel Quality Indicator (CQI), the QoS Class Identifier (QCI), the number of spatial layers, Inter-Cell Interference Coordination (ICIC), among others.

IV. Performance analysis

In this section we investigate the runtime of the PHY BBU sub-functions on a multi-core platform. In order to evaluate the gain in terms of latency, we apply data parallelism for the execution of PHY UL functions, i.e., FFT, demodulation and channel decoding. We use Open Air Interface (OAI) data [13] in order to determine the runtime of BBU sub-functions. OAI is an open-source solution that implements the RAN functionality in software and can be used as a benchmark.

A. Simulation settings

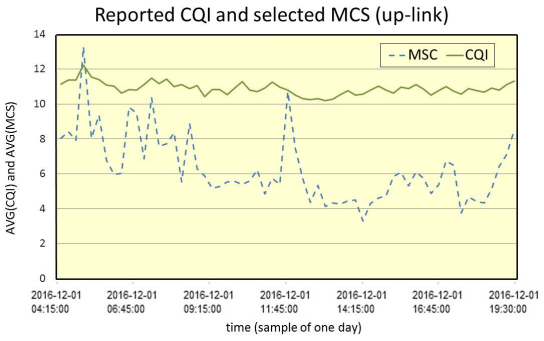
The simulation is performed for a single cell with SISO configuration, 20 MHz of bandwidth and Frequency Division Duplex (FDD) transmission mode. The simulator builds the resource grid considering the behavior of current commercial radio scheduler deployed in large cities. It selects both the MCS and the number of PRBs based on emulated radio conditions, the traffic in the cell, and the data load per UE. The simulator determines the TBS using tables 7.1.7.1-1 and 7.1.7.2.1.1-1 defined in the LTE Physical Layer Specification 3GPP TS 36.213 version 12.4.0 Release 12 [15]. In order to study the worst case, we consider non empty scheduling, i.e., all UL resources designated to a UE are used and need to be processed. We evaluate two scenarios.

1) Scenario A: Real traffic emulation during busy hours: This scenario is carried out under real traffic conditions. We emulate the behavior of current deployed eNBs. We use the MCS and Channel Quality Indicator (CQI) patterns performed by an eNB during busy-hours as shown in Figure 4(a).

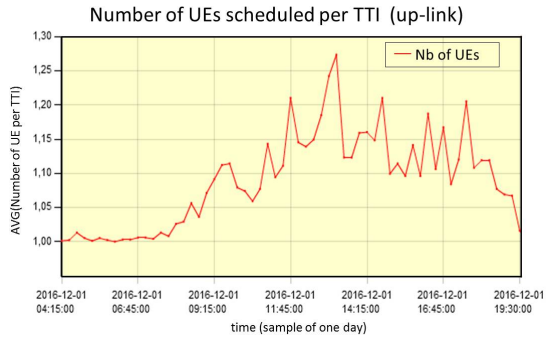
In general, radio schedulers use the CQI to determine the MCS. When the MCS takes values from 0 to 9, data is modulated in QPSK, while for values between $10-16$ the eNB uses 16-QAM modulation. In order to build the resource grid we have also obtained the number of UEs scheduled by Transmission Time Interval (TTI) which is illustrated in Figure 4(b).

Note that only one or two UEs are scheduled each millisecond. A fragment of the obtained resource grid is shown in Figure 5(a).

2) Scenario B: No traffic patterns: In order to investigate the worst case, the simulator does not limit the behavior of eNBs to a specific traffic pattern. The resource grid is built under variable radio conditions and elastic number of connected UEs. The number of PRBs allocated to a UE ranges from 6 to 110 [15], [16]. The MCS varies between 0 and 27 which enables QPSK, 16-QAM, and 64-QAM modulation orders [15]. The obtained resource grid is shown in Figure 5(b).

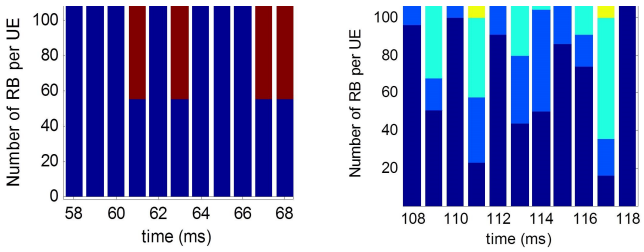


(a)



(b)

Fig. 4. Radio scheduling indicators.



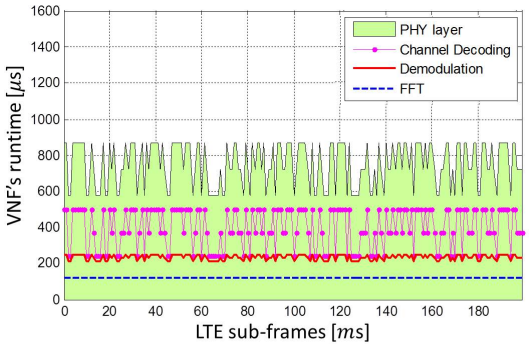
(a) $ScenarioA$ (b) $ScenarioB$

Fig. 5. LTE resource grid (Cloud-RAN workload).

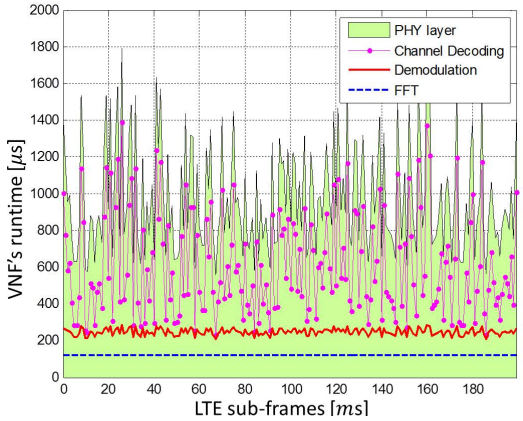
B. PHY UL runtime

We evaluate the runtime of the PHY layer during the reception process. In a first step, we do not consider the data parallel model presented in Section III. Figure 6(a) and Figure 6(b) show the execution time for the demodulation, channel decoding and FFT sub-functions for Scenarios A and B, respectively. Besides the fact that the FFT runtime is constant and that the demodulation processing time is less fluctuating than the decoding process, it is worth noting that

the runtime of the whole PHY layer is essentially determined by the decoding processing time.



(a) Scenario A



(b) Scenario B

Fig. 6. Performance of virtual-BBUs (PHY layer, up-link).

The FFT runtime only depends on the number of PRBs. The demodulation sub-function is directly influenced by the modulation order and the number of PRBs. The decoding process depends on the TBS, i.e., the CQI, the data load by UE and the traffic in the cell.

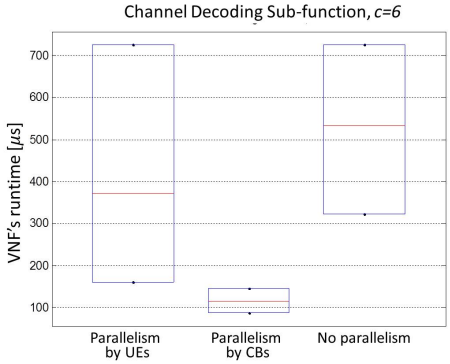
Simulation results show that the execution time of the PHY UL sub-functions can reach values of up to 1.8 ms (Scenario B). It does not leave enough margin to deploy a fully centralized RAN architecture which should additionally consider the front-haul delay and the processing time of L2 and L3 functions within the LTE Rx budget, i.e., 2 ms. In order to enable the deployment of a fully centralized RAN architecture, we evaluate in the following the performance gain obtained when using data parallelism.

C. Data parallelism: Performance gain

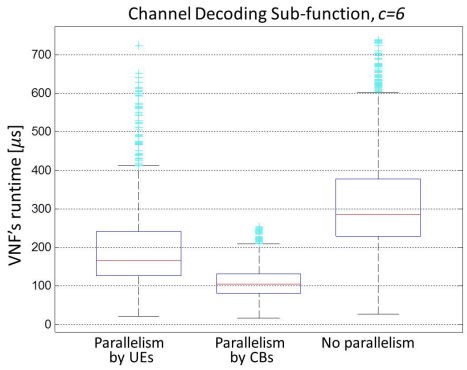
We focus our analysis on the most expensive function in terms of latency, i.e., the channel decoding process. We evaluate the performance gain when using data parallelism on a multi-core platform with c cores. We decompose the channel decoding function in a set of jobs where each of them processes a portion of data. In our proposed Cloud-RAN system, each runnable job accesses to computing resources according to the scheduling strategy. The scheduler selects them in first-in-first-out order. We use a non-preemptive scheduling algorithm which allocates a single core to each job. After allocation, a core is busy until that the job is completed, i.e., a job cannot be interrupted.

We evaluate the performance of the decoding sub-function under two degrees of data parallelism (1) by UEs and (2) by Code Blocks (CBs). In general, a sub-frame contains more than one UE, hence the PHY processing can be executed in parallel threads where each job corresponds to a UE. In the same way, when a Transport Block allocated to a UE exceeds 6120 bits, it is split in CBs of 6120 bits which can be executed in parallel. Hence, BBU-jobs arrive to the computing platform in batches. The batch size when applying data parallelism by UEs corresponds to the number of UEs per TTI. The batch size when processing CBs in parallel is the product of active UEs and the number of CBs per UE. The number of parallel threads for both (1) and (2) varies every millisecond in function of the number of UEs allocated by TTI and the data load per UE.

We study the runtime of the decoding sub-function when using a multi-core platform with 6 cores for both above described scenarios A and B. We evaluate the gain of the parallel execution by UEs and by CBs with respect to no-parallelism, i.e., the serial execution of jobs.



(a) Scenario A



(b) Scenario B

Fig. 7. Performance of channel decoding when using data parallelism.

Let us consider the Scenario A which reflects the behavior of current deployed eNBs in busy hours. Here, the mean runtime of the decoding process can be reduced by a factor of five when using data parallelism by CBs. It is shown

in Figure 7(a). When more than one UE is scheduled by TTI, data parallelism by UEs can also considerably decrease the execution time. Parallelism by UEs requires notably less developing effort than parallelism by CBs. Parallelism by UEs can then be considered as advantageous when taking into account the complexity in the implementation.

Let us now consider the Scenario B, performance results are shown in Figure 7(b). The runtime of the decoding function is divided by 3 when executing CBs in parallel and halved when applying threads per UEs.

The study of the probability density function of the channel decoding runtime shows that when executing one thread per CB, runtime values are more concentrated around the mean. Even if the reduction experimented when running UEs in parallel is already very interesting in terms of latency, the parallelism by CBs offers better performance, especially when comparing the tail of both probability density functions. See Figure 8 for an illustration. Note that when applying parallelism by UEs, the probability density function exhibits a long tail that is similar to that obtained when there is no parallelism.

In other words, parallelism by CBs presents less statistical dispersion, i.e., the execution of the channel decoding function is near to a specific value (e.g., 300 ms for a multi-core platform with 6 cores). This fact is crucial for the cloudification of RAN functions.

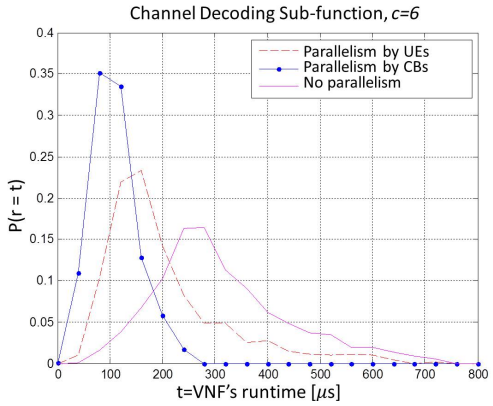


Fig. 8. Probability density of channel decoding runtime, Scenario B.

V. Conclusion

In this work, we have studied a fully centralized Cloud-RAN architecture which meets the need for improving the performance of software-based BBUs in terms of latency. To reduce the runtime of the most expensive BBU function, i.e., the channel decoding process, we have applied data parallelism and resource pooling in a multi-core system.

We have evaluated two methods of parallelism. The first one uses one thread per UE, and the second one goes further, employing one thread per CB. We have implemented a global scheduler to determine which core executes a particular job and in what order the BBU-jobs are processed.

Performance results show that a finer granularity in the parallelization model of the decoding function can significantly reduce the runtime and the variability. This performance gain enables longer distances between the BBU-pool and antennas, as well as, the efficient utilization of computing resources.

The main conclusion is that the cloudification of RAN functions for the up-link is feasible in dense areas, which is in accordance with [2].

References

[1] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[2] Veronica Quintuna and Fabrice Guillemin. VNF modeling towards the Cloud-RAN implementation. In *Conference on Networked Systems (NetSys 2017), Gottingen, Germany*, 2017.

[3] Aleksandra Checko, Henrik L Christiansen, Ying Yan, Lara Scolari, Georgios Kardaras, Michael S Berger, and Lars Dittmann. Cloud RAN for mobile networks technology overview. *IEEE Communications surveys & tutorials*, 17(1):405–426, 2015.

[4] Navid Nikaein. Processing radio access network functions in the cloud: Critical issues and modeling. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, pages 36–43. ACM, 2015.

[5] Dario Sabella, Peter Rost, Yingli Sheng, Emmanouil Pateromichelakis, Umer Salim, Patricia Guitton-Ouhamou, Marco Di Girolamo, and Giovanni Giuliani. RAN as a service: Challenges of designing a flexible RAN architecture in a cloud-based heterogeneous mobile network. In *Future Network and Mobile Summit (FutureNetworkSummit), 2013*, pages 1–8. IEEE, 2013.

[6] Islam Alyafawi, Eryk Schiller, Torsten Braun, Desislava Dimitrova, Andre Gomes, and Navid Nikaein. Critical issues of centralized and cloudified lte-fdd radio access networks. In *Communications (ICC), 2015 IEEE International Conference on*, pages 5523–5528. IEEE, 2015.

[7] Jialong Duan, Xavier Lagrange, and Frederic Guilloud. Performance analysis of several functional splits in C-RAN. In *Vehicular Technology Conference (VTC Spring), 2016 IEEE 83rd*, pages 1–5. IEEE, 2016.

[8] Christian Fabio Alessandro Lanzani, Lars Dittmann, and Michael Stübert Berger. *4G mobile networks: An analysis of spectrum allocation, software radio architectures and interfacing technology*. PhD thesis, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Electromagnetic SystemsInstitut for Elektromagnetiske Systemer, 2012.

[9] CPRI specification v6.1, 2014. Version 6.1.

[10] Bin Guo, Wei Cao, An Tao, and Dragan Samardzija. LTE/LTE-A signal compression on the CPRI interface. *Bell Labs Technical Journal*, 18(2):117–133, 2013.

[11] Dirk Wubben, Peter Rost, Jens Steven Bartelt, Massinissa Lalam, Valentin Savin, Matteo Gorgoglione, Armin Dekorsy, and Gerhard Fettweis. Benefits and impact of cloud computing on 5G signal processing: Flexible centralization through cloud-ran. *IEEE signal processing magazine*, 31(6):35–44, 2014.

[12] Islam Alyafawi, Eryk Schiller, Torsten Braun, Desislava Dimitrova, Andre Gomes, and Navid Nikaein. Critical issues of centralized and cloudified LTE-FDD radio access networks. In *2015 IEEE International Conference on Communications (ICC)*, pages 5523–5528. IEEE, 2015.

[13] Navid Nikaein, Raymond Knopp, Florian Kaltenberger, Lionel Gauthier, Christian Bonnet, Dominique Nussbaum, and Riadh Ghaddab. Openair-interface 4G: an open LTE network in a PC. In *International Conference on Mobile Computing and Networking*, 2014.

[14] Sourjya Bhaumik, Shoban Preeth Chandrabose, Manjunath Kashyap Jataprolu, Gautam Kumar, Anand Muralidhar, Paul Polakos, Vikram Srinivasan, and Thomas Woo. Cloudiq: A framework for processing base stations in a data center. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 125–136. ACM, 2012.

[15] LTE, evolved universal terrestrial radio access, physical layer procedures (3GPP TS 36.213 version 12.4.0 release 12). Standard, European Telecommunications Standards Institute, 2015.

[16] LTE, evolved universal terrestrial radio access (e-utra) and evolved universal terrestrial radio access network(e-utran), overall description, stage 2 (3gpp ts 36.300 version 13.6.0 release 13). Standard, European Telecommunications Standards Institute, February 2017.