NUMAP: NUMA-aware Multi-core Pinning and Pairing for Network Slicing at the 5G Mobile Edge

Wen-Ping Lai* and Kuan-Chun Chiu
Department of Electrical Engineering
Yuan Ze University
Taoyuan, Taiwan
wpl@saturn.yzu.edu.tw*

***Abstract*—Based on the concept of *network functions virtualization* (NFV) by adopting the virtualized micro-service-based event-driven model, this paper studies the system resource allocation problems of deploying network/service slices on the mobile edge server, performing as a pivotal control/data/information hub in between radio-access and core networks or even the Internet clouds, in the coming era of 5G digital transformation. A *non-uniform memory access* (NUMA)-*aware multi-core pinning-and-pairing* method, called NUMAP, for network/service slicing with respect to different traffic levels is proposed to improve the system performance of a *light-weight* EPC slice (vEPC$_{LW}$) on top of an x86 Dell PowerEdge Server (R740) for the MEC cloudlet platform. This server is equipped with two CPU sockets, each containing 12 physical cores sharing the same *local* memory bank, denoted as a NUMA node; namely *remote* memory access time to another NUMA node is *longer* than the *local* one. The novelty of the proposed NUMAP method lies in the fact that it is aware of three important pairing schemes based on their pairing distances, namely *inter-node-pair*, *intra-node-pair* and *hyper-threaded-pair*, and NUMAP can thus serve as a New Map for multi-core assignment. Preliminary experimental results show that the NUMAP algorithm outperforms the default one which is based on *symmetric multi-processing* (SMP).**

Keywords- 5G; NFV; MEC; NUMA; NUMAP; microservice; network slicing

I. INTRODUCTION

The unprecedentedly vast data volumes generated by the *internet of things* (IoT) have exposed the inefficiency of traditional computing, storage and network models, and thus *multi-access edge computing* (MEC) [1-2] near the 5G [3] mobile end is rapidly emerging as a new technology for reliable, flexible, scalable or even mission-critical demands for multi-tenant differential-service deployments and operations, such as those from *over-the-top* (OTT) media-service content providers.

Hence, in the coming era of 5G digital transformation, a mobile edge cloudlet can be perceived as a pivotal hub of *control/data/information* flows in between the *radio-access network* (RAN) and the *core network* (CN), as shown in Fig. 1, if it is capable of efficient system resource allocations not only to the *service proxies* for far reaches of the core network or even the Internet, but also to the *virtual base-band-unit* (vBBU) *pools* for RAN slicing in C-RAN [4-7],

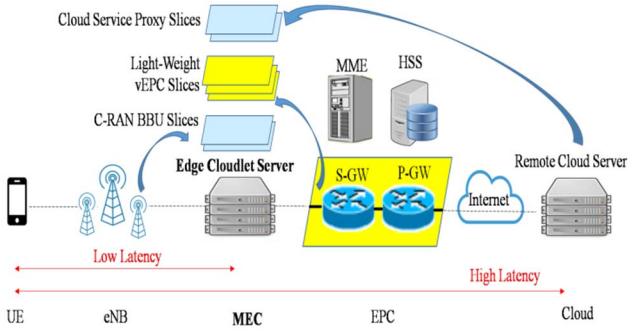

Fig. 1. Typical use cases of MEC as an Edge Cloudlet Server for a data-, control- and information-hub of EPC/RAN/Cloud slicing

close to the data sources generated from the device ends. The network/service slicing and orchestration problems [8-9] of system resources become the major challenges of MEC in order to meet the differential requirements of 5G use cases, such as bandwidth saving for outgoing video content demands in terms of *enhanced mobile broadband* (eMBB), real-time tasking for advanced connected vehicles in terms of *ultra-reliable low-latency communication* (uRLLC), and fast edge intelligence for industrial IoT in terms of *massive machine-type communication* (mMTC).

On the other hand, the ever-increasing demands from both telecom operators and datacom stakeholders are impacting and reshaping the landscape of telecom standardization, such as 5GPPP [10], and this can help the 5G digital transformation reshape from a conventional vendor-lock-in scheme (i.e., with vertically-integrated and proprietary hardware/software for fixed network functions) to an innovative scheme (i.e., with horizontally distributed network functions based on open-source software running on commodity hardware, such as the x86-based architecture), in order to achieve the ETSI network functions virtualization (NFV) [11] for free service chaining, such as the concept of *network store* [12] proposed by the Eurecom OAI project [13] and its extended project called Mosaic-5G [14]. As a result, a trend is happening to re-design and re-architect the conventional center office of telcos to become a small-scale modern data center (DC), such as the ONF CORD project [15]. Certainly, such an open scheme also applies to the MEC design.

In this paper, we focus on the uRLLC topic of the MEC cloudlet design via studying and understanding the CPU

multi-core assignment issue with respect to a *light-weight evolved packet core* (EPC) network slice realized in a *virtualized* environment, denoted as $vEPC_{LW}$, leading to the associated performance issue of *data-plane* (DP) oriented EPC network slicing [16] on a Dell x86-based MEC server, where the memory access time by an assigned CPU core could be significantly different depending on the accessed memory is *local* or *remote* to the accessing core. Such a phenomenon is recognized as *non-uniform memory access* (NUMA) [17], very common in the current x86-based server machine but usually not yet well considered when designing such a server machine as a MEC cloudlet platform for the common *playground* of CN and RAN network slicing. The main contribution of this paper is thus two-folded: (1) a good understanding of both the *intra-* and *inter-*NUMA-node effects on the packet's routing latency, provided quantitatively via a well-designed experiment layout of virtual machines and networking, and (2) a good *pinning-and-pairing* method for multi-core assignment and its underlying strategy in the context of NUMA, denoted as NUMAP, which is intended for providing a 'New MAP' of NUMA-aware multi-core assignment, obviously applicable to any-number assignment of cores.

The remainder of this paper is organized as follows. Section II details the design principle and strategy of the proposed NUMAP method. Section III describes the proposed design of experiment layout in the virtual-machines-and-networking environment to carry out the latency effects of three NUMAP schemes and SMP under various levels of cross traffic, from *light* to *heavy*, in the context of $vEPC_{LW}$ slicing on the target Dell x86-based MEC server. Finally, Section IV concludes this paper.

II. Proposed Method: NUMAP

As aforementioned, the proposed NUMAP method is intended for serving as a 'New MAP' of NUMA-aware assignment of CPU cores for network slicing on a MEC cloudlet server, assuming the NUMA-based x86 server architecture is enabled, locating in between the *radio-access network* (RAN) and the *core network* (CN) of a telecom operator, to proxy and reduce the on-demand bandwidth pressure for the Internet service access via the CN and meet the low-latency requirements of real-time tasks from the UE end.

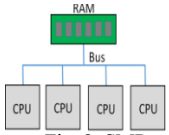

Fig. 2. SMP

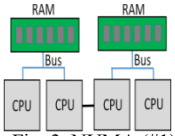

Fig. 3. NUMA (#1)

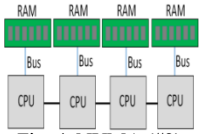

Fig. 4. NUMA (#2)

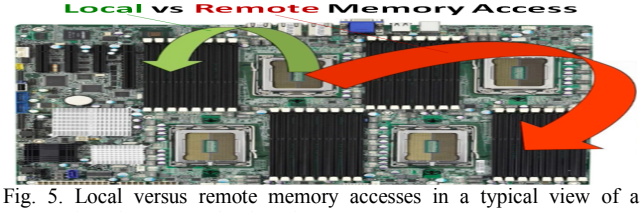

Fig. 5. Local versus remote memory accesses in a typical view of a NUMA-based server mother board

A. SMP

Modern servers evolved from *symmetric multiple processors* (SMP) to NUMA. In SMP, as illustrated by Fig. 2, multiple identical processors share not only a single main memory but also full access to all I/O devices, and thus the memory access time for all the processors is *symmetric* (i.e., *uniform*). SMP also needs an operating system's support (Linux kernel 2.5 started to support SMP) to treat all the processors equally, reserving none for priority. Namely, SMP tries to balance the CPU load among all the processors. However, the memory-access contention bottleneck among the processors over a single system bus becomes the main drawback, and a single-level fast cache system could be introduced such that each processor has its own cache to mitigate the contention and greatly reduce the memory access time. Note that SMP appeared long before the birth of the modern multi-core CPUs.

B. NUMA

In order to address the contention bottleneck problem with the *shared-memory via a single bus*, NUMA introduced the idea of *non-uniform memory access*, where each CPU is allowed to have its own *local memory*, leading to much shorter memory-access time, compared to remote *memory* accesses, as shown in Figs. 3 and 4. Meanwhile, along with the birth of multi-core CPUs, each CPU could start to have multiple processor cores, and a *hierarchical caching* system was thus introduced to provide *non-uniform cache access*, where each core has its own private cache, denoted as *level-1 cache* (L1) for both *instruction* (L1i) and *data* (L1d) operations, plus L2 either privately owned by each core or locally shared between a pair of cores (different CPUs have different designs), and L3 globally shared within the same CPU package (i.e., socket). Each CPU package along with its own local memory bank is viewed as a NUMA node, and several nodes form a server. The illustrated server in Fig. 5 consists of 4 NUMA nodes, each with its own local memory bank. The aforementioned *local* and *remote* memory accesses form different NUMA node distances, reflecting *short* and *long* latencies respectively. For convenience, this paper refers such a phenomenon to as *intra-* and *inter-*NUMA-node distances accordingly.

C. NUMAP

In this paper, the proposed NUMAP is a NUMA-aware multi-core *pinning-and-pairing* method for network slicing on a MEC cloudlet server platform. As stated previously, NUMAP can provide a *new map of distance knowledge* in the sense that it does not only deal with the *inter-*NUMA-node distance issue of *non-uniform memory accesses*, but also try to explore the *intra-*NUMA-node distance issues due to *caching hierarchy* and *hyper-threading* (HT), where the former is as aforementioned, and the latter is related to the Intel *virtualization technology* (VT) allowing a physical core be hyper-threaded into two logical cores, with enhanced performance better than one physical core, but not as good as two physical cores.

With such a map of *distance knowledge*, NUMAP helps to specifically perform core pinning from those idle ones

based on the multi-core activities and simultaneously make more efficient *core-pinning-and-pairing* based on their *intra-* or *inter-*NUMA-node distance relationship. *Pinning* stands for *specific individual* selection of cores for the target network slice, in contrast to the *load-balanced* selection in SMP. *Pairing* refers to specific *paired* selection of cores, which consists of three selection schemes:

- *HT-pair*: the paired cores are HT-sibling to each other ($NUMAP_h$).
- *Intra-node-pair*: the paired cores are not an HT-pair, but *collocated* within the same NUMA-node ($NUMAP_c$).
- *Inter-node-pair*: the paired cores are *non-collocated* and belong to different NUMA nodes ($NUMAP_n$).

```
N_cores = the demanded number of cores
int   k = N_cores / 2 ;
int   r = N_cores - 2*k ;      | r = 0 if N_cores = even
                               | r = 1 if N_cores = odd
while (a network-slice core assignment is demanded)
{
    for (i=1; i <= k ; i++)
    {
      pair() ;
    }
    if ( r == 1) then
      pin();
}
function pair(slice-traffic-level)
{
     if (the slice-traffic-level is low) then
     {
         if ( HT-pair is allowed)
            select  NUMAP_h ;
         else if (Intra-node-pair is allowed)
            select  NUMAP_c ;
         else
            select  NUMAP_n ;
     }
     if (the slice-traffic-level is medium or high) then
     {
         if (Intra-node-pair is allowed)
            select  NUMAP_c ;
         else if (HT-pair is allowed)
            select  NUMAP_h ;
         else
            select  NUMAP_n ;
     }
}
function pin( )
{
    if ( collocated pinning after the pairing is allowed)
        select collocated  pinning;
    else
        select non-collocated pinning ;
}
```

Fig. 6. Pseudo code for the proposed NUMAP method

TABLE I. THREE PAIRING SCHEMES OF NUMAP

Scheme	Distance	Definition of *paired* selection		
		HT-pair	*intra-node-pair*	*inter-node-pair*
$NUMAP_h$	*short*	O	O	
$NUMAP_c$	*medium*		O	
$NUMAP_n$	*long*			O

At the first glance of Table I, the design principle for realizing a workable algorithm based on the distance knowledge of NUMAP seems to be clear, namely a reasonable selection priority seems to be: $NUMAP_h$ > $NUMAP_c$ > $NUMAP_n$, according to their distance inequalities. However, based on the quantitative study of this paper, the former inequality $NUMAP_h$ > $NUMAP_c$ is only partly true since it depends on other factors such as the compute ability of *HT-pair* and the *external traffic level to the target network slice*. On the other hand, the latter inequality is certain and $NUMAP_n$ should be avoided unless the number of available cores within the same NUMA node is insufficient.

Fig. 6 illustrates a typical pseudo code of NUMAP. In general, for any demanded number of cores to a target network slice (denoted as N_{cores}), N_{cores} can be expressed in terms of $2k$ or $2k+1$, where $k \in N$. Hence, if N_{cores} is even, the core assignment can be conducted in terms of multiple specific paired selections. If N_{cores} is odd, multiple specific paired selections plus one core-pinning will be needed. Depending the level of slice traffic, HT-pairing and Intra-node-pairing are more preferred than Inter-node-pairing.

III. TEST RESULTS AND ANALYSES

In order to evaluate the three pairing schemes in the proposed NUMAP method, as listed in Table I, this section firstly describes the experiment setup on top of an x86-based Dell EMC PowerEdge R740 server [20], as shown in Fig. 7, followed by the internal view of the R740 mother board layout of 2 NUMA nodes in Fig. 8. A hierarchical-caching topology view of these two NUMA nodes is present in Fig. 9, followed by the adopted examples for the three NUMAP schemes in Fig. 10. Eventually, combined test results are summarized in Fig. 11, followed by some insightful analyses.

A. Experiment setup

In our previous work [16], a *containerized* EPC data-plane slice (i.e., vSPGW, denoted as $vEPC_{LW}$ in this paper) has been successfully realized via the Docker container technology [18-19] based on the Eurecom OpenAirInterface (OAI) software package, where such an OAI-CN-based data-plane slice can successfully run on an x86 PC to jointly work with an OAI-RAN-based small cell running on another x86 PC, equipped with a software-defined-radio transceiver (USRP B210). In this research, such a $vEPC_{LW}$ slice has also been successfully transplanted and deployed on top of the aforementioned R740 as a MEC cloudlet server, located in between the RAN and CN as shown in Fig. 1.

However, the container feature of $vEPC_{LW}$ makes it difficult to reflect itself in terms of the assigned CPUs' loads under different levels of cross traffic if the traffic type routed

through the slice is purely L3/L4-based, which would be totally managed by the Host OS kernel's IP forwarding function, shared among and thus not visible to all the containers. This is because the container is purely application-level virtualized and confined, and thus does not contain any Guest OS, not to mention its own Guest-OS kernel. In order to see the cross traffic effect that helps to magnify the potential latency differences among the three possible schemes of the proposed NUMAP, the $vEPC_{LW}$ slice was thus realized on a hypervisor-based VM adopting the Oracle VirtualBox; namely, the Guest-OS kernel associated with the VM could thus be used to reflect the cross traffic effect. Note that both the Guest and Host OSes are Ubuntu Linux based, running on the R740 server.

Fig. 7 shows a specially designed measurement topology on top of the R740 server, where all the measurement entities are VM-based, including the $vEPC_{LW}$ slice and the auxiliary router. Such a topology is a standard one to let the link between the $vEPC_{LW}$ and auxiliary router become the traffic bottleneck, and thus the $vEPC_{LW}$ slice becomes the bottleneck too. Hence, the routing latency can be reflected by measuring the *round-trip-time* (RTT) between H2 and H5, under different UDP cross-traffic levels, considering that the EPC traffic is UDP-based by default. Two UDP cross traffic flows are thus established between H1 and H4, and also between H3 and H6. Note that the UDP flow is beneficial to generating *constant bit rate* (CBR) and has no flow-rate control, and thus it is possible for two such CBR flows of the same rate to generate a *variable bit rate* (VBR) flow with some short-term burst rates higher than the double rate (i.e., the expected summed rate) to congest the vEPC slice so that the differences among the three schemes in the proposed NUMAP method can be more visible and magnified. Also note that such a measurement topology can get rid of other uncertainty factors coupled into the latency measurement of the $vEPC_{LW}$ slice on the MEC server, such as those from RAN, CN and the Internet. One more thing to note is that all the VMs' virtual ports are only limited up to 1 Gbps, which can be viewed as the *virtual line rate* since the adopted network interface cards are *virtual* too. In this paper, the *iperf3* toolset was adopted for the UDP cross traffic generator , and the *ping* tool for the RTT measurement.

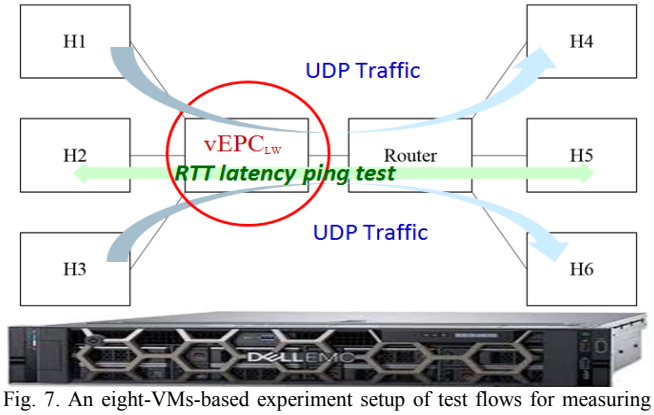

Fig. 7. An eight-VMs-based experiment setup of test flows for measuring the *round-trip-time* (RTT) latencies of a $vEPC_{LW}$ slice on an x86-based Dell PowerEdge Server (R740) running with Ubuntu Linux 16.04.3

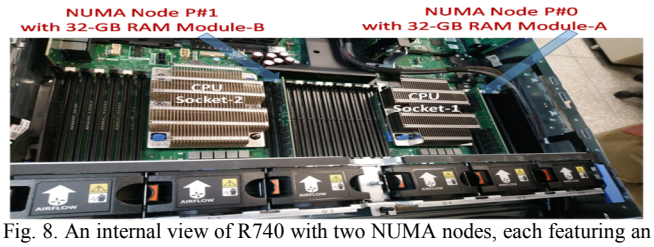

Fig. 8. An internal view of R740 with two NUMA nodes, each featuring an Intel Xeon Gold 5118 CPU package (in its own socket) equipped with one 32-GB DDR4 RAM module, plugged into its own local memory bank

Fig. 8 displays the internal view of the R740 server, consisting of two NUMA nodes, each featuring an Intel Xeon Gold 5118 CPU package (plugged into its own CPU socket). In this study, the R740 server was equipped with two 32-GB DDR4 RAM modules, each plugged into the local memory bank next to the corresponding CPU package as the *local* memory, which is in turn *remote* to the other CPU package.

B. Hierarchical-caching View of NUMA Nodes in the R740

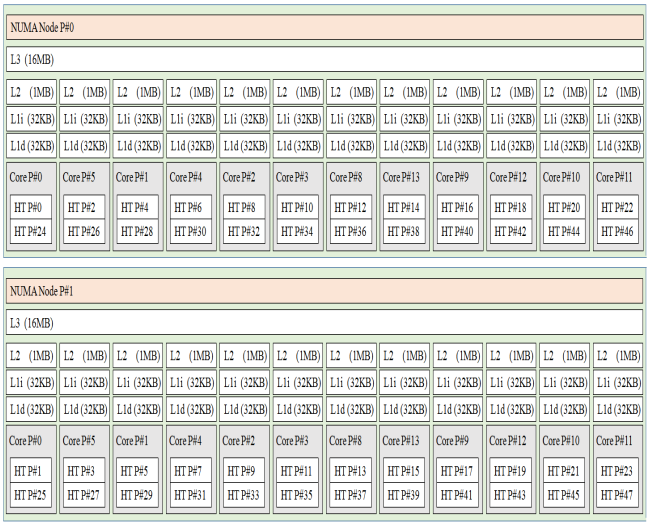

Fig. 9. An expanded view of both the NUMA nodes (P#0 and P#1) of R740, each equipped an Intel Xeon Gold 5118 CPU package consisting of 12 physical core processors, i.e., 24 logical processors if the Intel HT technology is enabled

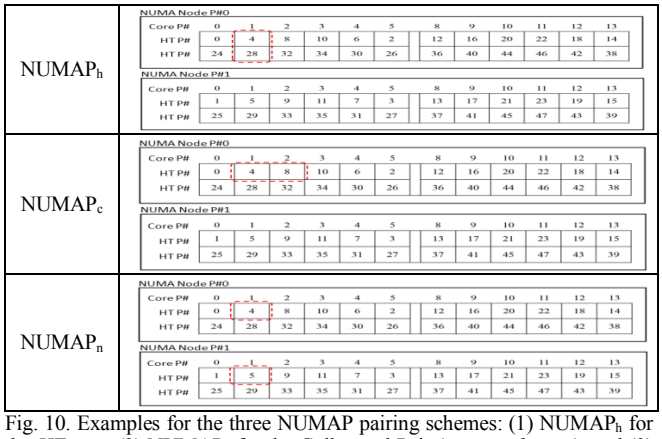

Fig. 10. Examples for the three NUMAP pairing schemes: (1) $NUMAP_h$ for the *HT-pair*, (2) $NUMAP_c$ for the Collocated-Pair (*intra-node-pair*), and (3) $NUMAP_n$ for the Non-collocated-Pair (*inter-node-pair*)

It was necessary to obtain a clear map of NUMA (i.e., the objective of the NUMAP method) from the viewpoint of OS before any *paired* selection of the *logical cores* could become meaningful. In this study, the map information was collected from the queries to the Linux *sys* virtual filesystem about the NUMA caching- and topology-information, i.e.,

/sys/devices/system/cpu/cpu*/cache

and

/sys/devices/system/cpu/cpu*/topology

and further bitmap decoding and reasoning was needed to obtain a map like Fig. 9, in terms of a hierarchical caching topology view of the two individual Intel Xeon Gold 5118 CPU packages (NUMA Node Processors), in order to understand the possible pairing relations of HT Processors (i.e., at the level of *logical* cores). As seen, the *even-number* HT Processors belong to NUMA Node P#0, and the *odd-number* ones to NUMA Node P#1. With the knowledge of this map and the reordering based on Core P# (i.e., at the level of *physical* cores), Fig. 10 illustrates the examples of the three NUMAP pairing schemes, adopted by Fig. 11.

C. *Latency Effects of NUMAP Pairing Schemes*

Fig. 11 summarizes the RTT latency effects of the three NUAMP schemes versus the default scheme (i.e., SMP) under different levels of UDP cross traffic, generated by two CBR UDP flows, with their summed rate running from 0 to 500 Mbps. For simplicity, two logical cores were assigned to the VM-based $vEPC_{LW}$ slice. Recall, from the previous discussion for the experiment setup, that such a summed rate could well be expected to be a VBR of short-term bursts with an increasing possibility of *longer* congestion than expected as the demanded UDP cross traffic runs *higher*, considering the fact that all ports of the VM-based $vEPC_{LW}$ slice are only limited to 1 Gbps, not high enough so that the bursts could easily touch the ceiling. It is clearly seen that such an expectation becomes true, and it helps to magnify the differences among the measured latencies of the three pairing schemes of NUMAP, as well as the SMP. Note that each data point is the mean value of 1000 *ping*'s RTT measures, with its *standard error* small enough to be included within the marker size. Also note that the horizontal position of each data point presents its *measured* value too, and its misalignment with the *demanded* level of cross traffic (i.e., the expected vertical dotted grid line) can thus potentially reveal some early sign of congestion, which will be discussed below. The summarized results from Fig. 11 deliver the following major messages:

- At the zero level of cross traffic (i.e., no UDP traffic), all look similarly around 0.55 ms except that SMP is somewhat higher than 0.6 ms because it has to pay some extra cost for frequent context switching among different cores. The above latencies seem to be very typical responses via a single-hop routing without contention, as expected for the fact that the $vEPC_{LW}$ slice is made up of an OAI-based combined gateway of S-GW and P-GW, omitting their S5/S8 interface. Although the differences among the NUMAP schemes are small at the *ms* scale, it can still be seen clearly that they are all well separated at the μs scale, considering that their associated standard errors are much smaller than their mutual differences. At the μs scale, the latency effects of the three NUMAP schemes are fully dominated by their pairing distances: the longer the distance, the higher the latency. Such an effect has also been seen clearly and magnified in case the *ping* test is initiated from the UE side. But it needs more future efforts to control and understand the uncertainty brought by the radio, and thus not yet shown here.
- At a *light*-level of cross traffic, say 100 Mbps, $NUMAP_h$ (*HT-paired*) can take the lead as expected since the cache-distance between the *HT-pair* is the shortest one. However, up to the level of 200 Mbps, all the differences are within 0.2 ms, and the default SMP seems to be the all-time loser.
- *Beyond* the *medium* level of cross traffic, say 300 Mbps, the *inter-node-paired* scheme $NUMAP_n$ (*non-collocated pair*) becomes the true victim due to the fact that its longest NUMA distance starts to dominate the cause of the largest latency.
- When the cross traffic level gets really *heavy* passing 400 Mbps and eventually reaches 500 Mbps, all the NUMAP target schemes and the SMP become well separated in their latency performances. It is interesting but not surprising to see that $NUMAP_h$ is defeated not only by $NUMAP_c$ but even by SMP. This can be reasoned by the fact that the *HT-pair* of $NUMAP_h$ actually consumes only 1 physical core, whereas the others 2 physical cores. Hence, it is not really a shame to be defeated by SMP only around the heavy-level close to congestion. Note that the horizontal positions of the data points are the *measured* throughputs of cross traffic, not the *demanded* ones shown by the vertical dotted grid lines. This is a special design in order to see any sign of early congestion in terms of the misalignment between the data point and the expected

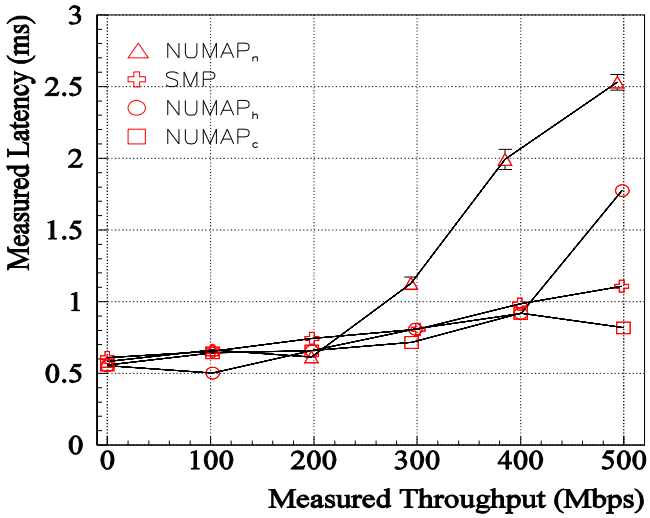

Fig. 11. Measured RTT *ping*-latency effects of a VM-based $vEPC_{LW}$ slice under different external UDP cross-traffic levels (with 2 flow rates summed up) running from 0 to 500 Mbps, where the vertical dotted grid lines stand for the demanded values, while the markers for the corresponding measured throughputs

vertical grid line. Such a misalignment behavior is most serious and visible in the context of $NUMAP_n$ due to its longest NUMA distance, starting from *medium*-traffic up to *heavy*-traffic. However, this is also within the expectation since the traffic level is a *relative* term, namely the long distance between the *non-collocated pair* of $NUMAP_n$ changes the sense of traffic level, and *early congestion* could well start to happen around others' *medium* traffic level.

IV. Conclusion and Outlook

In this paper, a NUMA-aware multi-core *pinning-and-pairing* method called NUMAP has been proposed for studying the system resource allocation problems of deploying network/service slices on a mobile edge server (x86-based Dell PowerEdge R740), which could perform as a pivotal control/data/information hub in between the RANs and the CN or even the Internet clouds, in the coming era of 5G digital transformation. NUMAP can provide a *new map of distance knowledge* in the sense that it does not only deal with the *inter*-NUMA-node distance issue of *non-uniform memory accesses* (i.e., the $NUMAP_n$ scheme), but also try to explore the *intra*-NUMA-node distance issues due to *caching hierarchy* and *hyper-threading* (HT), which can further split into two more schemes, namely $NUMAP_c$ and $NUMAP_h$.

The experimental results reveal that the NUMA distance starts to dominate the cause of latency effects starting from the *medium* level of cross traffic, and thus the $NUMAP_n$ scheme should be avoided unless necessary.

Overall speaking, the *intra-node-paired* $NUMAP_c$ seems to be the all-time winner due to its relatively short NUMA distance. However, if the power-saving issue needs to be considered, $NUMAP_h$ seems to be a better choice than $NUMAP_c$ from *zero*- to *light*-traffic, or even up to *medium*-traffic, since the *HT-pair* consumes only 1 physical core, in contrast to 2 physical cores in all the other schemes. Hence, a *mixed-selection* scheme between $NUMAP_h$ and $NUMAP_c$ would be preferred for the $vEPC_{LW}$ slice deployment on the R740 server, if such a scheme can dynamically adapt to the slice traffic level, as shown in Fig. 6. However, the traffic level threshold for switching between these two schemes remains to be decided by the design goal.

As an outlook, how to make the deployed $vEPC_{LW}$ slice really *light weight* remains a dilemma for the virtualization-technology choice between the hypervisor-VM and the container, remarkable by their *heavy*- and *light*-weight use of system resources respectively. As aforementioned, the hypervisor-VM was chosen for this study because the Layer 3/4 routing traffic of the $vEPC_{LW}$ slice is handled by the Host OS Kernel of the MEC server, and thus invisible to the container due to its feature of *application-layer confinement.* In other words, unless the routing traffic is *deeply investigated* by the application layer of SPGW, the container-based $vEPC_{LW}$ slice will not experience substantial routing-traffic stress. Fortunately, this issue is only unique to the routing-type network slice. For other network/service slice types involving any application layer, the container is definitely the top choice to provide light-weight micro-services. A newly emerging technology called HyperContainer [21] might be a potential solution for this unique problem of OS kernel's traffic-level detection and system resource adaptation for $vEPC_{LW}$ slicing.

Acknowledgment

This work was partly supported by Taiwan's funding agencies, including Ministry of Science and Technology under grants MOST 107-2221-E-155-013 and MOST 108-2221-E-155-022-MY2, and Ministry of Education under grants PEE107022 and PEE1080252.

References

[1] Heli Zhang, Jun Guo, Lichao Yang, et al., "Computation Offloading Considering Fronthaul and Backhaul in Small-Cell Networks Integrated with MEC," *IEEE Conf. on Computer Communications Workshops*, Atlanta, GA, USA, May 2017, pp. 115–120.

[2] M. Patel et al., Mobile-edge computing - introductory technical white paper. ETSI White Paper, Sep. 2014.

[3] *System Architecture for the 5G system Stage 2*, 3GPP TS 23.501 v15.00, Dec. 2017.

[4] Y. Lin, L. Shao, Z. Zhu, Q.Wang, and R. K. Sabhikhi, "Wireless network cloud: Architecture and system requirements," *IBM J. Res. Develop.*, vol. 54, no. 1, pp. 4:1–4:12, Jan./Feb. 2010.

[5] *C-RAN the road towards green ran*, China Mobile Research Institute White Paper, Oct. 2011.

[6] A. S. Thyagaturu, Z. Alharbi, and M. Reisslein, "R-FFT: Function split at IFFT/FFT in unified LTE CRAN and cable access network," *IEEE Trans. Broadcasting*, vol. 64, no. 3, pp. 648–665, Sep. 2018.

[7] N. Nikaein et al., "Towards a cloud-native radio access network," in *Advances in Mobile Cloud Computing and Big Data in the 5G Era*, vol. 22, C. X. Mavromoustakis et al. Ed. Cham: Springer, 2017, pp. 171–202.

[8] *Description of Network Slicing Concept*, NGMN White Paper, Jan. 2016.

[9] X. Foukas, G. Patounas, A. Elmokashfi and M. K. Marina, "Network slicing in 5G," *IEEE Comm. Magazine*, vol. 55, no. 5, pp. 94–100, May 2017.

[10] *View on 5G Architecture*, 5GPPP White Paper v2.0, Dec. 2017.

[11] *Network Functions Virtualization* (NFV) *Ecosystem Report on SDN Usage in NFV Architectural Framework*, ETSI NFV-EVE White Paper 005, Dec. 2015.

[12] N. Nikaein et al., "Network store: exploring slicing in future 5G networks," *Proc. 10th Int. Workshop on Mobility in the Evolving Internet Architecture* (*MobiArch '15*), Paris, France, Sep. 2015, pp. 8–13.

[13] N. Nikaein, M. K. Marina, S. Manickam et al., "OpenAirInterface: A flexible platform for 5G research," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 33-38, 2014.

[14] Eurecom, the Mosaic-5G Project, http://mosaic-5g.io/

[15] ONF, the M-CORD project, https://www.opennetworking.org/

[16] W.P. Lai*, Y.H. Wang and K.C. Chiu, "Containerized Design and Realization of Network Functions Virtualization for a Light-Weight Evolved Packet Core Using OpenAirInterface," *Proc. APSIPA Annual Summit and Conference 2018* (*APSIPA ASC'18*), Honolulu, Haiwaii, USA, Dec. 2018, pp. 472–477.

[17] Ulrich Drepper. What every programmer should know about Memory [Online]. Available: https://people.freebsd.org/~lstewart/articles/cpumemory.pdf

[18] I. Miell and A. H. Sayers, *Docker in Practice*. New York: Manning, 2016.

[19] D. Bernstein, "Containers and cloud: from LXC to Docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.

[20] *Dell EMC PowerEdge R740 and R740xd*, Dell EMC Technical Guide, 2018.

[21] The HyperContainer Project. [Online]. Available: https://github.com/hyperhq/hyperd