PAPER

Packet Processing Architecture with Off-Chip Last Level Cache Using Interleaved 3D-Stacked DRAM*

Tomohiro KORIKAWA[†a], **Akio KAWABATA**[†b], ***Members***, **Fujun HE**[††c], ***Nonmember***, ***and*** **Eiji OKI**[††d], ***Fellow***

SUMMARY The performance of packet processing applications is dependent on the memory access speed of network systems. Table lookup requires fast memory access and is one of the most common processes in various packet processing applications, which can be a dominant performance bottleneck. Therefore, in Network Function Virtualization (NFV)-aware environments, on-chip fast cache memories of a CPU of general-purpose hardware become critical to achieve high performance packet processing speeds of over tens of Gbps. Also, multiple types of applications and complex applications are executed in the same system simultaneously in carrier network systems, which require adequate cache memory capacities as well. In this paper, we propose a packet processing architecture that utilizes interleaved 3 Dimensional (3D)-stacked Dynamic Random Access Memory (DRAM) devices as off-chip Last Level Cache (LLC) in addition to several levels of dedicated cache memories of each CPU core. Entries of a lookup table are distributed in every bank and vault to utilize both bank interleaving and vault-level memory parallelism. Frequently accessed entries in 3D-stacked DRAM are also cached in on-chip dedicated cache memories of each CPU core. The evaluation results show that the proposed architecture reduces the memory access latency by 57%, and increases the throughput by 100% while reducing the blocking probability but about 10% compared to the architecture with shared on-chip LLC. These results indicate that 3D-stacked DRAM can be practical as off-chip LLC in parallel packet processing systems.

key words: *cache memory, communication system, memory architecture, network function virtualization*

1. Introduction

The performance of packet processing applications is dependent on memory accesses speed of network systems. Table lookup requires fast memory accesses and is one of the most common processes in various packet processing applications. Thus table lookup can be a dominant performance bottleneck in a system without fast memories. Network Function Virtualization (NFV) is transforming the traditional purpose-built hardware systems into commercial-off-the-shelf (COTS) hardware system such as x86 CPU-based general-purposes servers.

Fast cache memories inside a CPU of general-purpose hardware become critical to achieve high performance packet processing over tens of Gbps. Development of virtualization technology such as Intel (R) Data Plane Development Kit (DPDK) [1] and Single Root I/O Virtualization (SR-IOV) [2] and the increased performance of COTS hardware are significantly advancing NFV. There are several software packet forwarding applications that can achieve more than tens of Gbps processing by intensively using cache memories of a CPU [4], [5].

A carrier network comprises complex and various network functions such as the Broadband Network Gateway (BNG) function to terminate Point to Point Protocol (ppp) sessions from end-users, packet filtering functions or mitigation functions to enhance network security level, and Quality of Service (QoS) controlling functions to dynamically satisfy the Service Level Agreement (SLA). Moreover, multiple grades and types of network services are required to support each subscriber's communication demand, which makes a carrier network more complex than typical data center networks.

In NFV-aware carrier network systems, multiple types of applications with different characteristics or applications comprising multiple functions are executed in the same system simultaneously. These multi-tenant, NFV-aware carrier network systems must accommodate huge lookup tables of various packet processing applications in fast on-chip cache memories to achieve over tens of Gbps, which is not possible with the current COTS hardware architecture due to insufficient capacity of on-chip cache memories. Moreover, expanding the capacity of the on-chip cache memories is expensive due to the physical space limitations of the semiconductor chip. Therefore, next NFV-aware carrier network systems require off-chip cache memories with larger capacity in which huge lookup tables of various packet processing applications can be accommodated.

Instead of using on-chip cache memories with small memory capacity, the work in [6], [28] presented the packet processing architecture that uses interleaved 3 Dimensional (3D)-stacked Dynamic Random Access Memory (DRAM) devices, which brings larger memory capacity and more memory parallelism. On the other hand, we need to understand the performance dependency on having or not having the on-chip cache memories when combining each CPU core's on-chip dedicated cache memories, an on-chip shared cache memory of a modern multi-core CPU, and the off-chip 3D-stacked DRAM before we design the hardware/software details to build the real system.

Manuscript received February 17, 2020.
Manuscript revised May 21, 2020.
Manuscript publicized August 6, 2020.

[†]The authors are with Network Service Systems Laboratories, NTT Corporation, Musashino-shi, 180-8585 Japan.
[††]The authors are with Graduate School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.

[*]A part of this paper was presented at IEEE HPSR 2019 [24].

a) E-mail: tomohiro.koorikawa.xa@hco.ntt.co.jp
b) E-mail: akio.kawabata.un@hco.ntt.co.jp
c) E-mail: he.fujun.76z@st.kyoto-u.ac.jp
d) E-mail: oki@i.kyoto-u.ac.jp

DOI: 10.1587/transcom.2020EBP3017

This paper proposes a packet processing architecture that utilizes interleaved 3D-stacked DRAM devices as off-chip Last Level Cache (LLC) in addition to several levels of on-chip dedicated cache memories of each CPU core [24]. Entries of a lookup table are distributed in every bank and vault to utilize both bank interleaving and vault-level memory parallelism. Frequently accessed entries in 3D-stacked DRAM are also cached in on-chip dedicated cache memories of each CPU core. The evaluation results show that the proposed architecture reduces the memory access latency by 57%, and increases the throughput by 100% with reducing blocking probability about 10% compared to the architecture with shared on-chip LLC. These results indicate that 3D-stacked DRAM can be practical as off-chip LLC in parallel packet processing.

The rest of this paper is organized as follows. Section 2 provides the background knowledge of DRAM system, 3D-stacked DRAM devices, and cache memory system. Sections 3 and 4 present the proposed architecture and its modeling. Section 5 presents performance evaluations of the proposed architecture. Section 6 describes related work. Section 7 discusses the limitations and the future directions. Finally, Sect. 8 concludes this paper.

2. Background

2.1 Memory System in COTS System

The memory system in COTS system consists of memory requestors, memory controllers, and DRAM memory devices. Memory requestors are CPUs or Direct Memory Accesses (DMAs) that read data from memory devices or write data to memory devices. A memory controller and memory devices are connected via a command bus and a data bus. Both buses are accessible in parallel, which means that one requestor can use the command bus while another requestor uses the data bus simultaneously. Modern DRAM systems have multiple channels which can be accessed independently. Each channel comprises banks that can be accessed in parallel if there is no collision on either the command bus or the data bus. Therefore by issuing read commands to one bank to another, these banks can be interleaved to increase memory access performance as shown in Fig. 1.

2.2 3D-Stacked DRAM

3D-stacked DRAM is a memory device that vertically stacks DRAM layers by using Through Silicon Via (TSV) technology. Hybrid Memory Cube (HMC) and High Bandwidth Memory (HBM) are well-known examples of 3D-stacked DRAM devices. As discussed in [6], [28], HMC has more memory channels than HBM. Thus we use HMC in this paper as well. Figure 2 shows the schematic structure of an HMC. The vertical units called *vaults* correspond to channels in the traditional DRAM, and are accessible in parallel. Inside a vault, each DRAM layer has several banks.

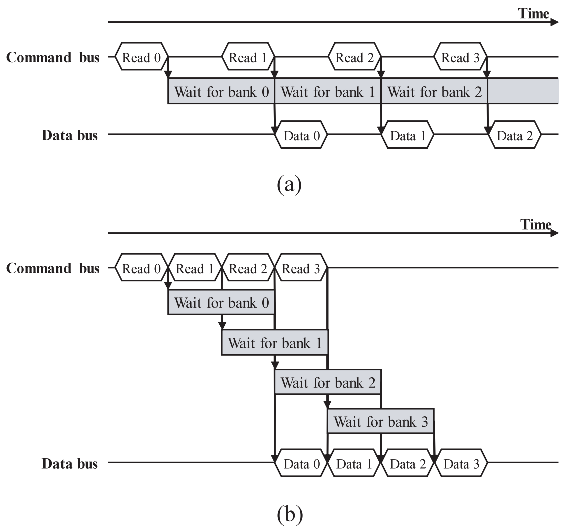


Fig. 1 Schematic diagram of DRAM bank interleaving. (a) Without bank interleaving. (b) With bank interleaving.

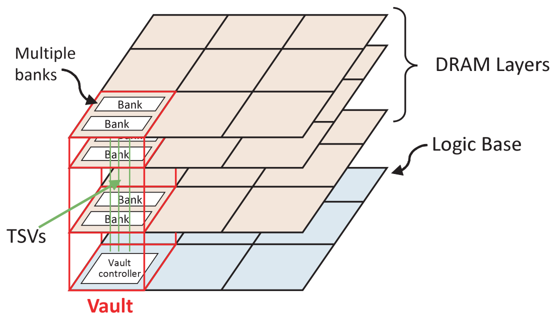


Fig. 2 Schematic structure of Hybrid Memory Cube.

2.3 Cache Memory System

A multi-core CPU has several levels of on-chip cache memories. Usually, there are one or two levels of dedicated cache memories corresponding to each CPU core, and there is a last level cache (LLC) which is shared every CPU core inside the same CPU. For the dedicated caches in different levels, the one with lower level has smaller capacity and faster speed. The LLC has lager capacity and slower speed compared to any dedicated cache. Figure 3 describes a schematic mechanism of cache memory systems and off-chip memory, namely DRAMs in COTS servers or HMCs in the proposed architecture. Any data is transferred between the off-chip memory and each level of cache in a certain block of data, called a *cache line*. The cache line is introduced to enhance the hit probabilities of cache memories by using space locality of the data; the data around the target data are likely to be accessed next. If there is not enough space in each level of the cache memory when loading the data from off-chip memory or other levels of cache memories, a certain cache line is evicted to the next level of cache memory or dropped according to the cache replacement policy of the corresponding system. Usually, the least recently used (LRU) cache line

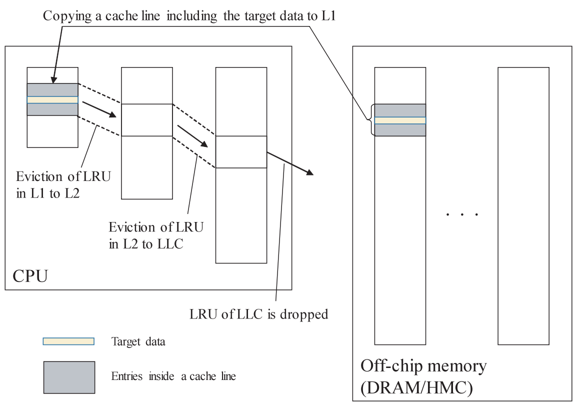

Copying a cache line including the target data to L1
Eviction of LRU in L1 to L2
Eviction of LRU in L2 to LLC
CPU
LRU of LLC is dropped
Target data
Entries inside a cache line
Off-chip memory (DRAM/HMC)

Fig. 3 Schematic mechanism of cache memory systems and main memory.

or a randomly selected cache line is replaced.

3. Proposed Architecture

Figure 4 shows the proposed architecture. It consists of a multi-core CPU that includes on-chip several levels of cache memories dedicated to each CPU core, an FPGA, an HMC, a DRAM, and network interfaces. Incoming packets are processed as follows. (1) Packets entering the network interfaces are directly sent to and buffered in the DRAM by using DMA. The packet descriptor of each incoming packet is randomly distributed to the queue of each CPU core. (2) A CPU core reads the header information of a packet buffered in the DRAM and looks up tables held in the cache memories of the CPU or HMC to determine the next action for the packet. (3) After finishing lookup and determining the next action, the CPU core sends the packet outside the proposed architecture via the network interface that corresponds to the action.

The HMC acts like an off-chip LLC with larger capacity in the proposed architecture. In the HMC, lookup table data is distributed as shown in [6], [28]. The original table is divided into some partial tables inside a set of a vault and a bank so that the original table comprises partial tables in a vault. Then, the whole table data in a vault is copied to other vaults. The number of partial tables equals the number of banks in each vault, and the number of copies equals the total number of banks of the HMC.

An FPGA is used to connect the CPUs to the HMC as well as to distribute memory requests using a hash function to the appropriate vault/bank sets. The CPU and the FPGA are linked via inter-chip connections such as Intel Quick Path Interconnect (QPI) or Ultra Path Interconnect (UPI), which is used in the integrated type of CPU + FPGA device and discrete type of FPGA devices, as presented in [7]–[9], [29].

4. System Model

We consider the three system models shown in Fig. 5, including the proposed architecture, to understand the performance dependency on having or not having the on-chip cache memories when the 3D-stacked DRAM is combined with the modern multi-core CPU. Figure 5 shows system models of the proposed architecture, the reference architecture with on-chip LLC and HMC, and reference architecture without any on-chip cache and with HMC. Figure 5(a) is the system model of the proposed architecture. The off-chip HMC is combined with the multi-core CPU, where each CPU core has on-chip dedicated Level 1 (L1) and Level 2 (L2) cache memories. Figure 5(b) shows the system model of the reference architecture, where the off-chip HMC is combied with the multi-core CPU with L1/L2 cache memories of each CPU core and on-chip shared LLC. Figure 5(c) is the system model of the reference architecture, where the off-chip HMC is combined with the multi-core CPU without any on-chip cache memories. The on-chip caches in CPU store the copies of frequently used data based on the least recently used (LRU) approach.

In the proposed architecture, when a packet enters the system, it is randomly assigned to one of the CPU cores regardless of CPU core state. After its assignment to the CPU core, the packet is enqueued into the corresponding queue of the assigned CPU core and is processed following the first come first served (FCFS) policy. The total number of requests, which includes waiting requests for all the queues and the requests being processed by the CPU cores, is limited in the system. The processed request firstly accesses the dedicated caches in increasing order of cache levels, where the L1 cache is firstly accessed. If the corresponding table entry of request is not found, which is called a miss hit, in any level of cache, the request access the next level of cache if there is; otherwise, the request is transferred to a queue to wait for being distributed to the appropriate vault/bank set of the HMC.

In the system model of the architecture with on-chip LLC and HMC, a miss hit request of L2 cache is transferred to a queue to wait for the access to the LLC. If it is a miss hit for a request in LLC, the request is transferred to the HMC. In the system model of the architecture without any cache and with HMC, every request directly transferred to a queue to wait for the access to the HMC.

The table lookup model in HMC was presented in [6], [28]. Note that at any time, only one request is processed by each CPU core; the processing of request is completed when its corresponding table entry is found, which is called a hit, in any part of the system. All the requests from different CPU cores access the HMC or on-chip LLC as the FCFS policy.

After the hit, the copy of cache line, which includes the corresponding table entry of the request, is stored in the L1 cache of corresponding CPU core as the most recently used content. For any level cache of the corresponding CPU core, if it is a miss hit, the LRU content in the cache is evicted to the next level cache. For example, if the corresponding table entry of the request is found in the HMC, the cache line including it is copied to the L1 cache of the corresponding CPU core, and the LRU content in the L1 cache is evicted to the L2 cache. Meanwhile, the LRU content in the L2 cache

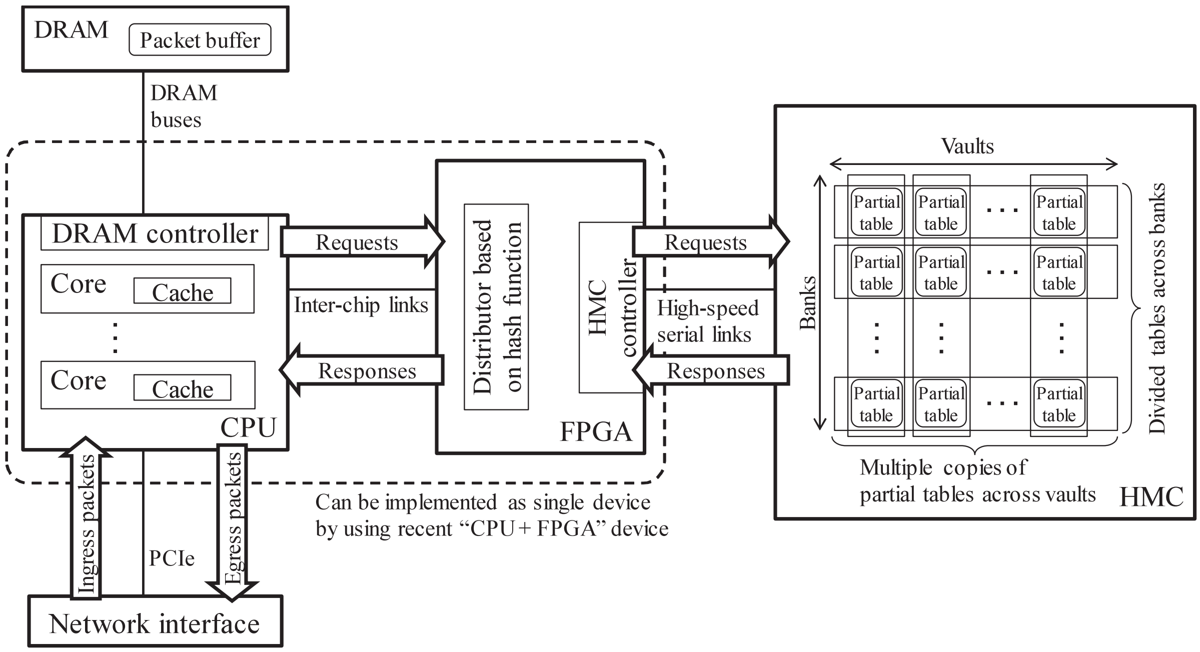


Fig. 4 Proposed architecture.

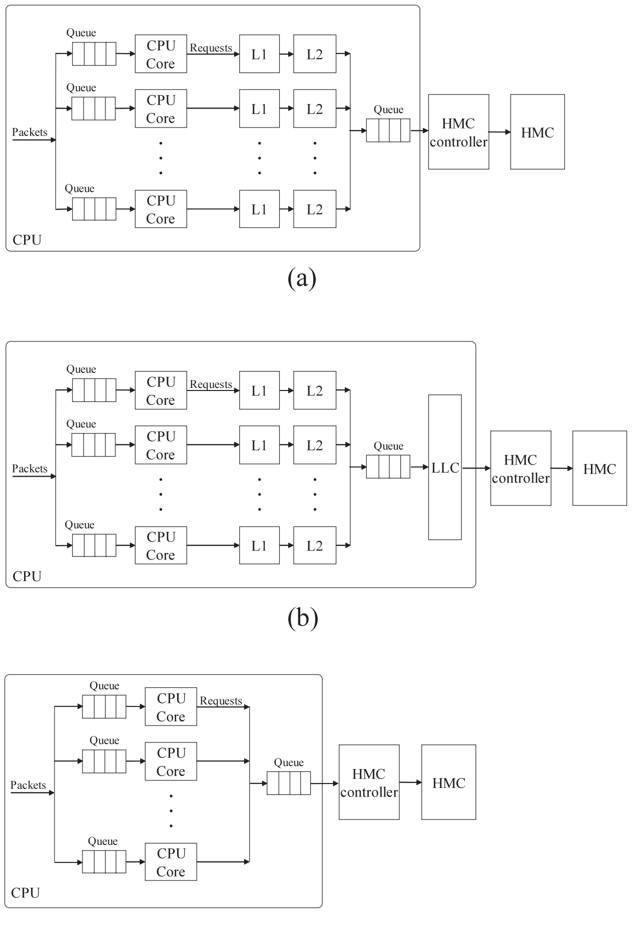


Fig. 5 System model of each architecture. (a) System model of proposed architecture. (b) System model of architecture with on-chip LLC. (c) System model of architecture without any cache and with HMC.

is dropped. In the system model of the architecture with on-chip LLC and HMC, the LRU content in the L2 cache is evicted to the on-chip LLC, and the LRU content in the on-chip LLC is dropped.

Let C denote the number of CPU cores in the system. Let N represent the total number of entries in the system. The maximum number of requests in the system is represented by K. A request incoming to the system is blocked if the number of accommodated requests in the system is K. The size of each queue associated with each CPU core is at least K, which means that no loss of request occurs in each queue. The size of queue associated with the on-chip LLC or HMC controller is considered to be equal with the number of CPU cores, C. The memory capacities of L1 cache, L2 cache, on-chip LLC, and HMC are considered as M_{L1}, M_{L2}, M_{LLC}, and M_{HMC}, respectively. Let B denote the size of cache line. The size of each table entry is represented by b. B/b table entries are copied to the L1 cache when there is a miss hit.

5. Performance Evaluation

5.1 Traffic Model

To evaluate the effectiveness of caching, some studies use the actual traffic traces, such as the works in [14]–[16], and others generate the synthetic traces with considering the content popularity, such as the works in [17], [18], where the content in traffic, such as the Web page requests and the IP addresses lookup requests, is considered to follow a Zipf-like distribution [19]. It indicates a behavior of the traffic that a few most popular contents are requested in high probabilities, and a large proportion of contents are requested in low probabilities.

In our traffic model, we assume that a request arrives at the IP lookup system based on a Poisson arrival process with the average rate of λ. Let I represent the total number of IP addresses stored in the system, which are sorted in decreasing order of popularity. We assume that the requested content follows a Zipf-like distribution, where the relative probability of requesting for the ith most popular IP address is expressed as $\frac{1}{i^\alpha}$, which leads to the probability with normalizing constant as $\frac{\frac{1}{i^\alpha}}{\sum_{i=1}^{I}\frac{1}{i^\alpha}}$. Note that the Poisson and Zipf-like distributions in our traffic model are orthogonal or independent from each other.

The value of α in the Zipf-like distribution varies for different traffic traces [19]. It is reported that the special case of $\alpha = 1$, which is known as the strict Zipf's law, is not appropriate for the content distributions in traffics, such as the Web page requests and the IP addresses lookup requests. Typically, the value of α is in the range of $0 < \alpha < 1$. α in traces from a homogeneous environment appears to be larger than that in traces from a more diversified user population. In other words, as α increases, more requests are concentrated on a few most popular contents. In our paper, we set the value of α as 0.83 [19].

In HMC, we assume that the probability of accessing each HMC bank is the same when the content popularity is considered. We adopt the table lookup model presented in [6], where requests are randomly assigned to HMC banks.

5.2 System Assumption

We introduce our assumptions of the system model for the simplicity of the numerical simulations.

We assume that the processing times of caches and HMC follow exponential distributions. The processing time includes the searching time in cache memories and the DRAM access latency due to DRAM specific behavior such as precharging the row buffer when accessing another row. We assume that the processing rate of the interleaved banks in the off-chip 3D-stacked DRAM is 0.7 times of that of without bank interleaving.

We also assume the steady condition in which there is no memory write request such as updating the table, which allows us to ignore cache coherency. In addition, the CPU reads data from each level of cache or the HMC in 8-byte groups.

We assume that the total number of entries, N, the maximum number of memory requests in the system, K, and the memory capacities of each level of cache memory, M_{L1}, M_{L2}, M_{LLC}, are smaller than those of today's COTS systems. These assumptions allow us to finish the numerical simulations within a practical time. Without this assumption on the memory capacity of each cache and the number of table entries, the numerical simulations do not finish in a reasonable time. For instance, if we set $M_{\mathrm{L1}} = 64$ [KiB], $M_{\mathrm{L2}} = 512$ [KiB], $M_{\mathrm{LLC}} = 28$ [MiB], the estimated time to obtain a one-plot result is at least one month by using our simulation environment that is described in Sect. 5.4.

5.3 Blocking Probability and Average Waiting Time

Let R represent a set of requests incoming to the system during a certain period time. $|R|$ denotes the total number of requests in R. The number of rejected requests that come to the system during the period is represented by $|R_{\mathrm{b}}|$, where R_{b} denotes the set of rejected requests. Blocking probability P_{b}, which is a probability that a request incoming to the system is rejected, is defined by $P_{\mathrm{b}} = \frac{|R_{\mathrm{b}}|}{|R|}$. Throughput, λ_{e}, is defined by $\lambda_{\mathrm{e}} = \lambda(1 - P_{\mathrm{b}})$. For accepted request $r \in R\backslash R_{\mathrm{b}}$, let t_r represent its waiting time to be processed by corresponding CPU core. Average effective waiting time, W_{e}, is defined by $W_{\mathrm{e}} = \frac{\sum_{r\in R\backslash R_{\mathrm{b}}} t_r}{|R|-|R_{\mathrm{b}}|}$. The sets of requests which are hit in the L1 caches, the L2 caches, and the LLC, are represented by R_{L1}, R_{L2}, and R_{LLC}, respectively. Hit probabilities in the L1 caches, the L2 caches, and the LLC, are defined by $P_{\mathrm{L1}} = \frac{|R_{\mathrm{L1}}|}{|R|}$, $P_{\mathrm{L2}} = \frac{|R_{\mathrm{L2}}|}{|R|}$, and $P_{\mathrm{LLC}} = \frac{|R_{\mathrm{LLC}}|}{|R|}$, respectively.

In our numerical analysis, we consider an HMC with two banks and S vaults. The processing rates of HMC vault with and without memory interleaving are μ and μ_1, respectively. Let ρ be a traffic load, which is defined by $\rho = \frac{\lambda}{S\mu}$.

We consider the Internet Protocol (IP) address lookup based on DIR-24-8-BASIC [21] for the benchmark of the packet processing, where the size of each entry is 2 byte. Thus we set $b = 2$ [B]. We consider that the multi-core CPU has $C = 28$ CPU cores, and the off-chip HMC has $S = 32$ vaults. The memory capacity of the off-chip HMC is considered to be $M_{\mathrm{HMC}} = 4$ [GiB], which is large enough to store all the entries in the system. We set the service rates of each level of cache memory and that of the off-chip HMC as $\mu_{\mathrm{L1}} = 100$, $\mu_{\mathrm{L2}} = 50$, $\mu_{\mathrm{LLC}} = 10$, $\mu = \mu_1 = 1$, according to the works in [32]–[34]. Based on the aforementioned descriptions and assumptions, we set $K = 100$, $\alpha = 0.83$, $M_{\mathrm{L1}} = 128$ [B], $M_{\mathrm{L2}} = 512$ [B], $M_{\mathrm{LLC}} = 4096$ [B], $N = 2 \times 10^4$, and $\mu_2 = 0.7$. We use $\rho = 0.7$ unless otherwise stated.

5.4 Numerical Simulation Results

We use Intel Xeon Silver 4216 2.10 GHz 16-core CPU with 128 GB memory to run the simulations based on Python 3.7.

Figure 6 shows the blocking probabilities for the proposed architecture, the architecture with on-chip LLC and HMC, and the architecture without any on-chip cache; the set of values of ρ is considered as $\{0.1, 0.2, 0.3, \cdots, 1.8, 1.9, 2.0\}$. We observe that as the traffic load increases, the blocking probability increases for every architecture. The blocking probability of the architecture with on-chip LLC and HMC increases rapidly compared to those of the other two architectures. This indicates that the shared LLC becomes a bottleneck due to the concentration of memory accesses from multiple CPU cores as the traffic load increases. When $\rho = 0.7$, the blocking probabilities for the proposed architecture and the architecture without any on-chip cache are less than 10^{-1}.

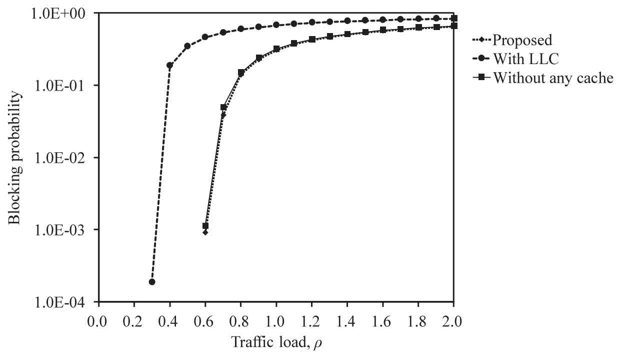


Fig. 6 Blocking probability depending on traffic load for different architectures.

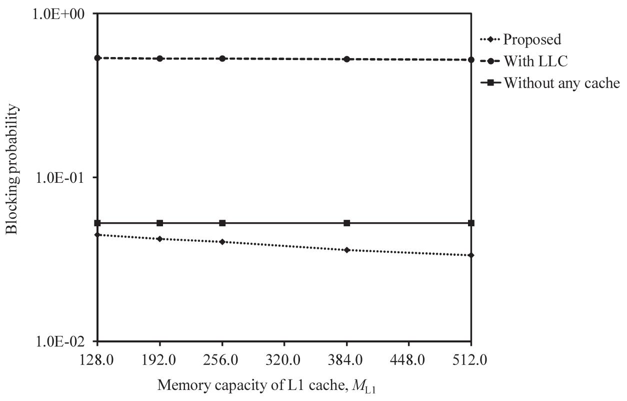


Fig. 7 Blocking probability depending on memory capacity of L1 cache for different architectures.

Figure 7 shows the blocking probabilities for the proposed architectures, the architecture with on-chip LLC and HMC, and the architecture without any on-chip cache; the set of memory capacities of L1 cache M_{L1} is considered as $\{128, 192, 256, 384, 512\}$ [B]. We observe that the proposed architecture outperforms the other two architectures. In addition, the blocking probabilities are almost independent of the memory capacity of L1 cache, which does not need to increase the expensive memory capacity of L1 cache.

Figures 8, 9, and 10 show the blocking probabilities, average effective waiting times, and throughputs for the proposed architecture, the architecture with on-chip LLC and HMC, and the architecture without any on-chip cache, respectively; the set of numbers of entries in each cache line is considered as $\{1, 2, 4, 16, 32, 64\}$. We observe that the proposed architecture reduces the memory access latency by 57%, and increases the throughput 100% with reducing blocking probability about 10% compared to the architecture with on-chip shared LLC when $B/b = 32$. In addition, if the proposed architecture can use smaller cache lines with smaller B/b, the performance of proposed architecture can be improved.

6. Related Work

Several software-based packet processing schemes for COTS

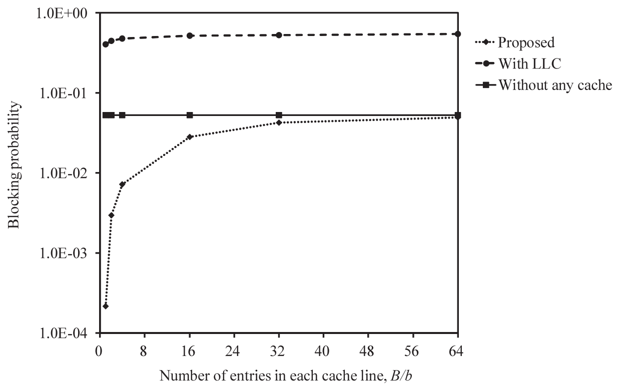


Fig. 8 Blocking probability depending on number of entries in each cache line for different architectures.

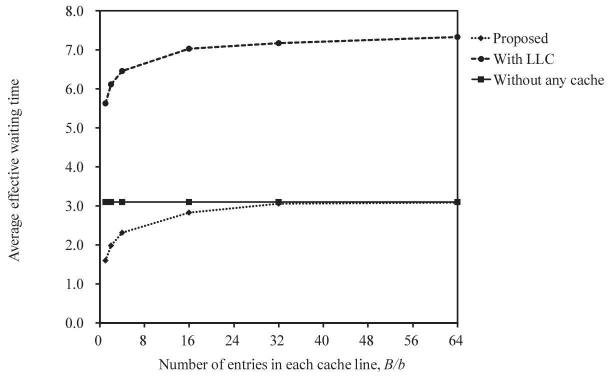


Fig. 9 Average effective waiting time depending on number of entries in each cache line for different architectures.

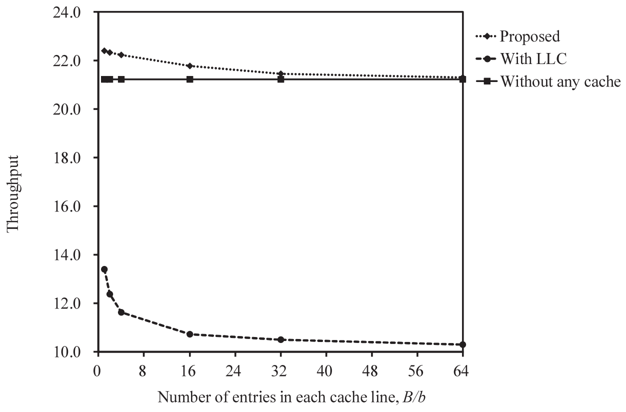


Fig. 10 Throughput depending on number of entries in each cache line for different architectures.

server have been proposed [3]–[5], [20]. RouteBricks [3] is the first software-based router application to leverage the parallel processing offered by modern multi-core CPUs. Lagopus [4] is a DPDK-enabled OpenFlow switch that can achieve over 10 Gbps performance with more than 1 M flow entries. These approaches significantly improve packet processing performance compared to previous software schemes running on single-core CPUs and DPDK. However, their performance directly depends on the cache memory of the

CPUs, which unfortunately is too small to support carrier-scale packet processing given the huge multiple tables involved. PacketShader [20] consolidates parallel processing by using Graphics Processing Unit (GPU). However, their work makes the constraining assumption of homogeneous packet processing to leverage the GPU's Single Instruction Multiple Data (SIMD) performance, and so does not suit carrier-scale packet processing. Poptrie [5] is fast software IP routing table lookup; it offers over 200 M lookups per second with just a single CPU core. The IP address lookup performance itself is sufficient for carrier-scale packet processing. However, this software is also dependent on the small cache memories inside the CPU.

A packet matching system using HMC was studied in [22]. However, no discussion is made on leveraging vault-level parallelism and bank interleaving of HMC, since the main problem targeted by the work is implementing a fast packet matching circuit in FPGA. CasHMC [23] is a cycle-accurate simulator for HMC. This simulator does not consider bank interleaving and cache memories of CPUs, and the simulation results are valid only current HMC devices.

Several studies use cache bypassing approach to reduce memory access latency when the data locality is not high [25]–[27]. The memory requests that bypassed the cache memories directly go to the main memory of the system. However, these studies do not consider using parallelism of main memory, which may increase the blocking probability.

There are several approaches to skip the on-chip LLC as presented in [30], [31]. These approaches allow the requests that miss the on-chip L2 cache to directly access the off-chip 3D-stacked DRAM, which may be used to implement the proposed architecture in the absence of on-chip LLC.

7. Discussion

As we describe in Sect. 1, the 3D-stacked DRAM in the proposed architecture improves packet processing performance based on the memory parallelism of the device. Thus, before we design the hardware/software details, we firstly focus on the memory parallelism of the system architectures, in which the CPU cache memories and the 3D-stacked DRAM are combined. In particular, we build the simulator to understand the performance dependency on the combination of each CPU core's on-chip dedicated cache memories, an on-chip shared cache memory, and the off-chip 3D-stacked DRAM, in terms of the memory parallelism. Therefore, although the cache sizes in the simulator are smaller than those of today's CPUs, we observe the advantage of the proposed architecture over the conventional architecture in terms of the memory parallelism by using the queueing model-based simulator.

Additionally, while we showed the table lookup as an example of packet processing whose performance depends on the memory access speed, we observe that the proposed architecture increases the overall memory access parallelism, which usually improves the performance of NFV applications. We also suppose that the performance improvement of a specific NFV application by using the proposed architecture depends on how data of the NFV application is located in the memory and the memory access characteristics of the NFV application.

In our future study, we plan to incorporate the hardware/software details of the CPUs, the 3D-stacked DRAM, and the operating system in the packet processing architecture, where the results of this work will be a reference for more complex models and simulators that require lots of system parameters and complex parameter tuning. Also, the data structure and the memory access characteristics of a specific NFV application will become worth incorporated.

8. Conclusion

This paper proposed an architecture that utilizes 3D-stacked DRAM as an off-chip LLC in addition to several levels of on-chip dedicated cache memories of each CPU core to support fast packet processing operations such as table lookup. In the proposed architecture, entries of a lookup table are distributed in every bank and vault to utilize both bank interleaving and vault-level memory parallelism. Frequently accessed entries in 3D-stacked DRAM are also cached in on-chip dedicated cache memories of each CPU core. We evaluated the performance of proposed architecture with considering several system parameters. The evaluation results showed that the proposed architecture reduced the memory access latency by 57%, and increased the throughput 100% with reducing blocking probability about 10% compared to the architecture with shared on-chip LLC. This indicated that 3D-stacked DRAM can be practical as off-chip LLC in parallel packet processing.

References

[1] Intel Data Plane Development Kit. http://dpdk.org/
[2] PCI-SIG Single Root I/O Virtualization (SR-IOV) Support in Intel Virtualization Technology for Connectivity. https://www.intel.com
[3] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: Exploiting parallelism to scale software routers," Proc. ACM SIGOPS, pp.15–28, 2009.
[4] Lagopus switch, a high performance software OpenFlow 1.3 switch. http://www.lagopus.org/
[5] H. Asai and Y. Ohara, "Poptrie: A compressed trie with population count for fast and scalable software IP routing table lookup," SIGCOMM Comput. Commun. Rev., vol.45, no.4, pp.57–70, Aug. 2015.
[6] T. Korikawa, A. Kawabata, F. He, and E. Oki, "Carrier-scale packet processing system using interleaved 3D-stacked DRAM," IEEE International Conference on Communications, 2018.
[7] N. Oliver, R.R. Sharma, S. Chang, B. Chitlur, E. Garcia, J. Grecco, A. Grier, N. Ijih, Y. Liu, P. Marolia, H. Mitchel, S. Subhaschandra, A. Sheiman, T. Whisonant, and P. Gupta, "A reconfigurable computing system based on a cache-coherent fabric," 2011 International Conference on Reconfigurable Computing and FPGAs, pp.80–85, Nov. 2011.
[8] Y. Watanabe, Y. Kobayashi, T. Takenaka, T. Hosomi, and Y. Nakamura, "Accelerating NFV application using CPU-FPGA tightly coupled architecture," ICFPT, pp.136–143, Dec. 2017.
[9] D.J. Moss, S. Krishnan, E. Nurvitadhi, P. Ratuszniak, C. Johnson,

J. Sim, A. Mishra, D. Marr, S. Subhaschandra, and P.H. Leong, "A customizable matrix multiplication framework for the Intel HARPv2 Xeon+FPGA platform: A deep learning case study," Proc. ACM/SIGDA FPGA, pp.107–116, 2018.

[10] K. Lahiri, A. Raghunathan, and S. Dey, "Evaluation of the traffic-performance characteristics of system-on-chip communication architectures," VLSI Design 2001 Fourteenth International Conference on VLSI Design, pp.29–35, 2001.

[11] I.Y. Bucher and D.A. Calahan, "Models of access delays in multiprocessor memories," IEEE Trans. Parallel Distrib. Syst., vol.3, no.3, pp.270–280, 1992.

[12] G.V. Varatkar and R. Marculescu, "On-chip traffic modeling and synthesis for MPEG-2 video applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.12, no.1, pp.108–119, 2004.

[13] V. Soteriou, H. Wang, and L. Peh, "A statistical traffic model for on-chip interconnection networks," 14th IEEE International Symposium on Modeling, Analysis, and Simulation, pp.104–106, 2006.

[14] E. Cohen and H. Kaplan, "Proactive caching of DNS records: Addressing a performance bottleneck," Computer Networks, vol.41, no.6, pp.707–726, 2003.

[15] T. Chiueh and P. Pradhan, "High-performance IP routing table lookup using CPU caching," IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol.41, no.6, pp.707–726, 2003.

[16] S. Ihm and V.S. Pai, "Towards understanding modern web traffic," Proc. 2011 ACM SIGCOMM conference on Internet measurement conference, pp.295–312, 2011.

[17] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching," IEEE/ACM Trans. Netw., vol.10, no.5, pp.589–603, 2002.

[18] C. Wang, L. Xiao, Y. Liu, and P. Zheng, "DiCAS: An efficient distributed caching mechanism for P2P systems," IEEE Trans. Parallel Distrib. Syst., vol.17, no.10, pp.1097–1109, 2006.

[19] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol.1, pp.126–134, 1999.

[20] S. Han, K. Jang, K. Park, and S. Moon, "PacketShader: A GPU-accelerated software router," SIGCOMM Comput. Commun. Rev., vol.40, no.4, pp.195–206, Aug. 2010.

[21] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," IEEE INFOCOM'98, vol.3, pp.1240–1247, March 1998.

[22] D. Rozhko, G. Elliott, D. Ly-Ma, P. Chow, and H.-A. Jacobsen, "Packet matching on FPGAs using HMC memory: Towards one million rules," Proc. ACM/SIGDA FPGA, pp.201–206, 2017.

[23] D.I. Jeon and K.S. Chung, "CasHMC: A cycle-accurate simulator for hybrid memory cube," IEEE Comput. Arch. Lett., vol.16, no.1, pp.10–13, Jan. 2017.

[24] T. Korikawa, A. Kawabata, F. He, and E. Oki, "Packet processing architecture with off-chip LLC using interleaved 3D-stacked DRAM," 2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR), Xi'An, China, pp.1–6, 2019.

[25] Y. Huangfu and W. Zhang, "Hardware-based and hybrid L1 data cache bypassing to improve GPU performance," 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, New York, NY, pp.972–976, 2015.

[26] S. Gupta, H. Gao, and H. Zhou, "Adaptive cache bypassing for inclusive last level caches," 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, Boston, MA, pp.1243–1253, 2013.

[27] G. Sun, C. Zhang, P. Li, T. Wang, and Y. Chen, "Statistical cache bypassing for non-volatile memory," IEEE Trans. Comput., vol.65, no.11, pp.3427–3440, Nov. 2016.

[28] T. Korikawa, A. Kawabata, F. He, and E. Oki, "Carrier-scale packet processing architecture using interleaved 3D-stacked DRAM and its analysis," IEEE Access, vol.7, pp.75500–75514, 2019.

[29] Intel Product Brief, "Enabling UPI and PCIe Gen4 accelerators," https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/solution-sheets/stratix-10-dx-product-brief.pdf

[30] T.S. Warrier, K. Raghavendra, and M. Mutyam, "SkipCache: Application aware cache management for chip multi-processors," IET Computers & Digital Techniques, vol.9, no.6, pp.293–299, 2015.

[31] J. Kong and K. Lee, "A DVFS-aware cache bypassing technique for multiple clock domain mobile SoCs," IEICE Electron. Express, vol.14, no.11, pp.1–12, 2017.

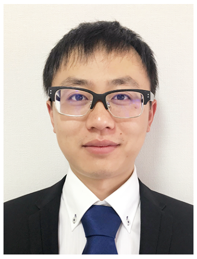
[32] R. Hadidi, B. Asgari, B.A. Mudassar, S. Mukhopadhyay, S. Yalamanchili, and H. Kim, "Demystifying the characteristics of 3D-stacked memories: A case study for hybrid memory cube," IEEE International Symposium on Workload Characterization (IISWC), Seattle, WA, pp.66–75, 2017.

[33] R. Hadidi, B. Asgari, J. Young, B.A. Mudassar, K. Garg, T. Krishna, and H. Kim, "Performance implications of NoCs on 3D-stacked memories: Insights from the hybrid memory cube," 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Belfast, pp.99–108, 2018.

[34] Intel(R) 64 and IA-32 Architectures Optimization Reference Manual, https://software.intel.com/sites/default/files/managed/9e/bc/64-ia-32-architectures-optimization-manual.pdf

Tomohiro Korikawa is a researcher of Network Service Systems Laboratories in Nippon Telegraph and Telephone Corporation (NTT), Tokyo, Japan. He received the B.S., M.S. degrees from Waseda University, Tokyo, Japan, in 2012, 2014, respectively. In 2014, he joined Nippon Telegraph and Telephone Corporation (NTT), Tokyo, Japan, where he is researching network system architecture and network design.

Akio Kawabata is an executive research engineer, project manager of Network Service Systems Laboratories in Nippon Telegraph and Telephone Corporation (NTT), Tokyo, Japan. He received the B.E., M.E. and Ph.D. degrees from the University of Electro-Communications, Tokyo, Japan, in 1991, 1993 and 2016, respectively. He is also associated with the Department of Communication Engineering and Informatics at the University of Electro-Communications in Tokyo, Japan for research activities. In 1993, he joined Nippon Telegraph and Telephone Corporation (NTT) Communication Switching Laboratories, Tokyo, Japan, where he has been engaging to develop switching systems, and researching network design and switching system architecture. He served as a senior manager of R&D Department at NTT East since 2011 until 2014.



Fujun He is currently pursuing the Ph.D. degree at Kyoto University, Kyoto, Japan. He received the B.E. and M.E. degrees from University of Electronic Science and Technology of China, Chengdu, China, in 2014 and 2017, respectively. He was an exchange student in The University of Electro-Communications, Tokyo, Japan, from 2015 to 2016. His research interests include modeling, algorithm, optimization, resource allocation, survivability, and optical networks.

Eiji Oki received the B.E. and M.E. degrees in instrumentation engineering and the Ph.D. degree in electrical engineering from Keio University, Yokohama, Japan, in 1991, 1993, and 1999, respectively. He was with Nippon Telegraph and Telephone Corporation (NTT) Laboratories, Tokyo, from 1993 to 2008, and The University of Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar at the Polytechnic Institute of New York University, Brooklyn. In 2017, he joined Kyoto University, Japan, where he is currently a Professor. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks.