IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 18, 2019, accepted May 28, 2019, date of publication June 5, 2019, date of current version June 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2920877

Carrier-Scale Packet Processing Architecture Using Interleaved 3D-Stacked DRAM and Its Analysis

TOMOHIRO KORIKAWA[1], AKIO KAWABATA[1], FUJUN HE[2], (Student Member, IEEE), AND EIJI OKI[2], (Fellow, IEEE)

[1]Network Service Systems Laboratories, NTT Corporation, Musashino-shi, Tokyo 180-8585, Japan
[2]Graduate School of Informatics, Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501, Japan

Corresponding author: Tomohiro Korikawa (tomohiro.koorikawa.xa@hco.ntt.co.jp)

ABSTRACT New network services such as the Internet of Things and edge computing are accelerating the increase in traffic volume, the number of connected devices, and the diversity of communication. Next generation carrier network infrastructure should be much more scalable and adaptive to rapid increase and divergence in network demand with much lower cost. A more virtualization-aware, flexible and inexpensive system based on general-purpose hardware is necessary to transform the traditional carrier network into a more adaptive, next generation network. In this paper, we propose an architecture for carrier-scale packet processing that is based on interleaved 3 dimensional (3D)-stacked dynamic random access memory (DRAM) devices. The proposed architecture enhances memory access concurrency by leveraging vault-level parallelism and bank interleaving of 3D-stacked DRAM. The proposed architecture uses the hash-function-based distribution of memory requests to each set of vault and bank; a significant portion of the full carrier-scale tables. We introduce an analytical model of the proposed architecture for two traffic patterns; one with random memory request arrivals and one with bursty arrivals. By using the model, we calculate the performance of a typical Internet protocol routing application as a benchmark of carrier-scale packet processing wherein main memory accesses are inevitable. The evaluation shows that the proposed architecture achieves around 80 Gbps for carrier-scale packet processing involving both random and bursty request arrivals.

INDEX TERMS Communication systems, memory architecture, network function virtualization, performance analysis, queueing analysis.

I. INTRODUCTION

New network services such as Internet of Things (IoT) and edge computing are driving rapid increases in traffic volume, the number of connected devices and the diversity of communication services [1]–[5]. Next generation carrier network infrastructure should scale well and adapt to the rapid increase and divergence in network demand with much lower capital expenditure (CAPEX) and operating expenditure (OPEX). Network Function Virtualization (NFV) is expected to realize more flexible and lower-cost network infrastructures by replacing traditional purpose-built network equipment with modern, general-purpose hardware.

The associate editor coordinating the review of this manuscript and approving it for publication was Yongpeng Wu.

Development of virtualization technology and the increased performance of commercial off-the-shelf (COTS) hardware are significantly advancing NFV. As one example, software-based packet processing on $\times 86$ CPU-based commodity COTS servers can achieve over several tens of Gbps processing sufficient for data plane packet functions [6]–[8]. Intel Data Plane Development Kit (DPDK) [9] and Single Root I/O Virtualization (SR-IOV) [10] are examples of the latest approaches that provide a framework for more hardware-aware, fast packet processing.

However, there are several significant hurdles to realize such a drastic change in carrier network infrastructure. One of the significant issues addressed in this paper is realizing high-performance packet processing at very large-scale carrier networks without using the traditional dedicated equipment.

A carrier network must accommodate a large number of subscribers across extremely wide areas such as a whole country, i.e. the *carrier-scale* network. 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.

Regarding packet processing performance, the current COTS server architecture has a fatal bottleneck; main memory access is degraded by the frequent cache memory misses imposed by the insufficient CPU cache memory size. The poor main memory access capability of $\times 8$ CPU-based COTS servers is the dominant barrier degrading the carrier-scale packet processing performance as discussed in our previous study [11].

This paper proposes a packet processing architecture that realizes carrier-scale applications without any dedicated hardware. The proposed architecture uses Hybrid Memory Cube (HMC), a sort of 3 Dimensional (3D)-Stacked Dynamic Random Access Memory (DRAM), to achieve acceptable memory access performance. We introduce an analytical model of the proposed architecture for two traffic patterns wherein the memory requests are either random or bursty. As an example of carrier-scale applications, we evaluate the performance of Internet Protocol (IP) address table lookup. The table is held in the HMC instead of CPU cache memory. The evaluations show that the proposed architecture can achieve around 80 Gbps for both random arrival of requests and bursty arrival of requests in carrier-scale packet processing, in which main memory access is inevitable, since CPU cache memory is insufficient to accommodate the huge tables. This proposed architecture achieves both high performance and versatility for carrier network virtualization.

This paper is an extended version of the work in [12]. We detail the background of DRAM and 3D-stacked DRAM devices such as HMC and High Bandwidth Memory (HBM). We extensively describe an analytical model of our proposed architecture. We detail the states to describe the proposed architecture and the state transitions and we formulate the equilibrium equations for random arrivals of requests. We consider bursty arrivals of requests where memory requests concentrate on a particular partial table. We detail the states and the state transitions and we formulate the equilibrium equations for bursty arrivals of requests. We present extensive performance evaluations of the proposed architecture based on our analytical model. We describe the related work on packet processing and present a direction for expanding our analytical model to a general case.

The rest of this paper is organized as follows. Section II provides the background of this work. Section III presents the proposed architecture and its modeling. Sections IV and V provide analyses of the proposed architecture for random and bursty request arrivals, respectively. Section VI presents performance evaluations of the proposed architecture. Section VII describes related work. Section VIII describes a direction in which to expand our analytical model. Finally, Section IX concludes this paper.

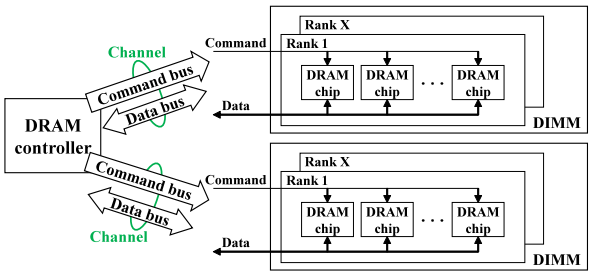


FIGURE 1. DRAM system overview.

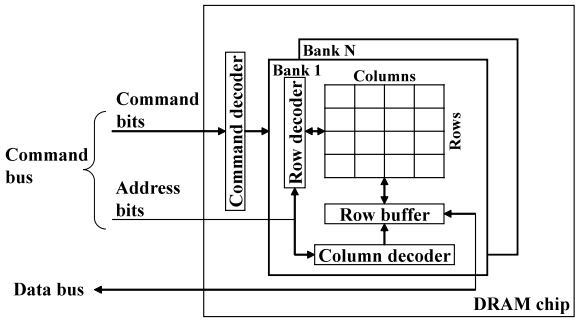


FIGURE 2. Architecture of DRAM chip.

II. BACKGROUND

A. DRAM MEMORY SYSTEM IN COTS SERVERS

The DRAM memory system in COTS servers consists of a memory controller and memory devices as shown in Figure 1. The memory controller handles memory access requests from requestors such as CPUs or Direct Memory Accesses (DMAs) to read the data from memory devices or write the data to memory devices. Note that the memory controller logic is usually integrated inside the latest generation of CPUs. Memory controller and memory devices are connected by 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 at the same time. However, no more than one requestor can use the same bus simultaneously.

Modern DRAM systems have a Dual Inline Memory Module (DIMM) interface with multiple *channels* which allows requestors to access multiple DIMMs simultaneously using multiple command bus and data bus units. Note that multiple DIMMs might be attached to a channel to share the buses in the channel among the DIMMs, which means that the DIMMs attached to the channel cannot be accessed at the same time. A DIMM is organized into *ranks*, and only one rank can be accessed at a time. We only consider DRAMs with only one rank for simplicity.

Each rank consists of multiple DRAM chips. Furthermore, each DRAM chip comprises *banks* that can be accessed in parallel if there are no collisions on either bus. Each bank has a *row buffer* and an array of storage cells organized in *rows* and *columns* as shown in Figure 2. Requestors can only access the content of the row buffer, not the data in the

storage array. To access a specific memory location, the row that contains the desired data must be loaded into the row buffer by an *Activate* command. When the controller wishes to load a different row, the current row buffer has to be written back to the array by a *Precharge* command in advance. The actual *Read* or *Write* commands only handle the data in the row buffer. A row that is cached in the row buffer is usually referred to as an *open row*. On the other hand, a row that is not cached in the row buffer is considered as a *closed row*.

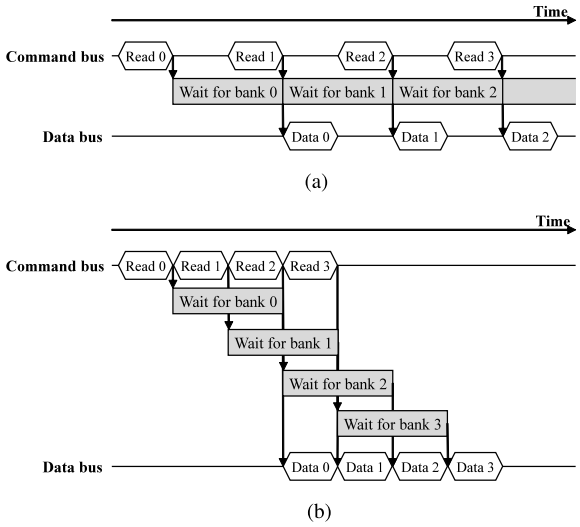


FIGURE 3. Schematic diagram of DRAM bank interleaving. (a) Without bank interleaving. (b) With bank interleaving.

Figure 3 shows the schematic diagram of DRAM bank interleaving. By issuing read commands to an open row at one bank to another, these banks can be *interleaved* to increase memory access performance with no additional hardware modification.

B. 3D-STACKED DRAM

3D-stacked DRAM is a memory device that vertically stacks traditional DRAM devices by using Through Silicon Via (TSV) technology. There are several commercially available 3D-stacked DRAM devices such as Hybrid Memory Cube (HMC) and High Bandwidth Memory (HBM). Since 3D-stacked DRAMs are based on general-purpose DRAMs, they are general-purpose devices but higher performance. The next level in performance is achieved by expensive dedicated devices such as Ternary Content Addressable Memory (TCAM), which is the de facto choice of network search engines [13], [14].

HMC comprises several DRAM layers on top of the bottom layer, the logic base [15]. Figure 4 shows the schematic structure of HMC. The vertical units called *vaults* correspond to memory channels in the traditional DRAM memory system, and are accessible in parallel. Inside a vault, each DRAM layer has several DRAM banks as with traditional DRAMs.

HBM also has several DRAM layers, channels, banks and a logic layer, similar to HMC [16]. The major differences

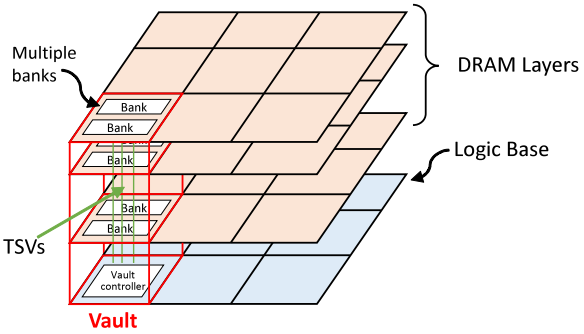


FIGURE 4. Schematic structure of hybrid memory cube.

TABLE 1. Device specifications of DRAM, HMC and HBM.

	DRAM	HMC	HBM
Capacity up to	64GB	8GB	16GB
No. of channels	1	32	8
No. of banks	16	256	256

between HMC and HBM are the number of channels, banks and memory bus width. HMC has more channels and banks than HBM, which means that HMC has more units that can be accessed in parallel. HBM has a wider memory bus, which makes it better-suited for applications such as graphic or image processing [17], [18] and deep neural networks [19], while HMC has packet-based high-speed serial links.

Table 1 describes the device specifications of DRAM, HMC, and HBM, as available at the time of writing [15], [16], [20]. We can see that HMC has the most channels and banks per device. Therefore, we use HMC to accommodate the huge tables used by carrier-scale networks.

III. PROPOSED ARCHITECTURE AND MODELING

A. PROPOSED ARCHITECTURE

Figure 5(a) shows the schematic view of our proposed architecture. It consists of a multicore CPU, an FPGA, an HMC, a DRAM and network interfaces. Although the proposed architecture can have several CPUs, FPGAs, HMCs, and DRAMs, we describe and model the proposed architecture depicted in Figure 5(a) for simplicity.

Basically, incoming packets are processed as follows. (1) Packets entering the network interfaces are directly sent to and buffered in the DRAM by using DMA. (2) A CPU core reads the header information of a packet buffered in the DRAM and looks up tables held in the HMC to determine the next action for the packet. A memory access request is generated for each 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.

A key to the proposed architecture is step (2) above. The HMC must hold huge numbers of table entries. In order to leverage both vault-level and bank-level parallelism, we divide the original table into partial tables in each vault and we copy them across vaults. The original table is divided

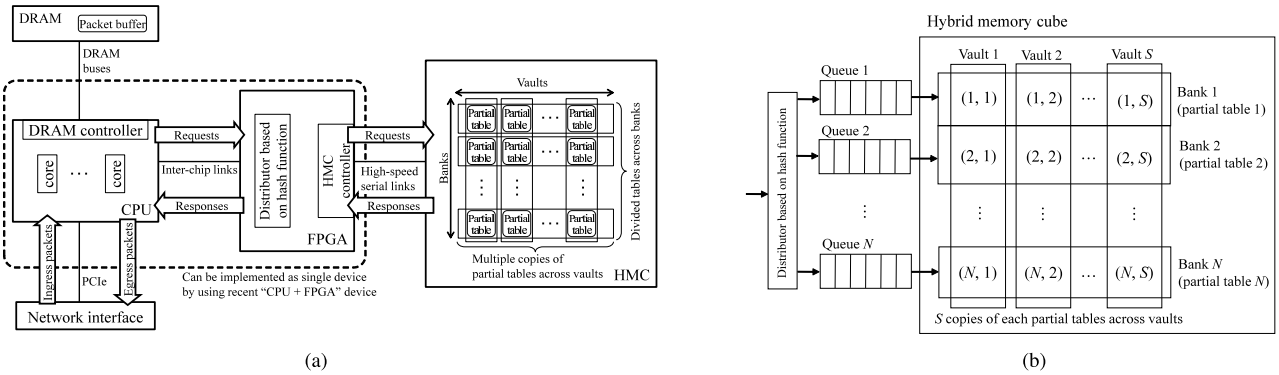


FIGURE 5. Proposed architecture. (a) Overview of proposed architecture. (b) Table lookup subsystem by using an HMC.

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, using a hash function, memory requests to the appropriate vault/bank sets. The FPGA contains two custom circuits for the distributor and an HMC controller. 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 now practical given recent CPU + FPGA device architectures [21]–[23].

As shown in Table 1, an HMC has up to 32 channels. However, the conventional system architecture such as the latest generation Intel Xeon CPU has up to only six channels [24]. In addition, the conventional system architecture has little room to increase the number of channels for DRAM devices due to complex electrical wiring between a CPU and DRAM devices. Therefore, the major advantage of the proposed architecture over the conventional systems is the number of memory channels, which enhances packet processing performance by simultaneously accessing partial tables through multiple vaults of the HMC. The proposed architecture is based on common characteristics among DRAM devices. Thus other types of DRAM devices such as HBM can be used to accommodate tables in the proposed architecture.

B. SYSTEM MODELING

We describe the behavior of the proposed architecture. Figure 5(b) shows the architecture of the table lookup subsystem formed by the HMC and FPGA. The HMC accommodates tables such as IP tables. The subsystem consists of a distributor based on hash function, N queues, and the HMC. The HMC consists of S vaults. Each vault has N banks. A whole lookup table is separated into N partial tables, each of which is allocated to a bank. S copies of each partial table across vaults are made; each vault has the same entries.

When a memory request enters the distributor, the hash function in the distributor classifies the request to one of N queues by using packet information, such as destination IP address. The calculation in this hash function is as simple as to classify the result of an logical operation for some bits of packet header, which is usually finished in one clock clycle in FPGA. Requests entering queue n, where $n \in [1, N]$, are served in a first in first out (FIFO) manner. Queue n has S servers, each of which corresponds to a vault. The sth server for queue n, where $s \in [1, S]$, is denoted by server (n, s). The maximum number of requests that can be accommodated, including all the queues and servers, is K, where $K \geq NS$. A request entering the table lookup subsystem is blocked if the number of requests already being handled by the subsystem is K. The packet generating the blocked request is discarded. The memory resources are shared by N queues under the condition that the total number of accommodated requests in the subsystem does not exceed K. In the worst case, $K - S$ requests are waiting for service in one particular queue and S requests are served by the corresponding S servers. The memory access rate by queue n to bank n at vault s is the service rate of server (n, s); each server serves one request. When more than one server, each of which corresponds to a different bank, at vault s are active, or more than one request is being served, bank interleaving is performed among them. Otherwise, no bank interleaving is performed. When bank interleaving is performed using w banks at vault s, we call the interleaving w-degree bank interleaving; no bank interleaving is performed when $w = 1$.

We describe our introduced analytical model. Each server is in one of three states, idle, busy without bank interleaving, and busy with bank interleaving. A server in idle state does not serve any request. A server in busy state without bank interleaving serves a request without bank interleaving. A server in busy state with bank interleaving serves a request with bank interleaving. When at least a server in idle state for queue n exists, a request at the head of line is served in the following server selection rule. If there is any server in idle state that moves to busy state without bank interleaving, it is selected. Otherwise, a server in idle state that moves to

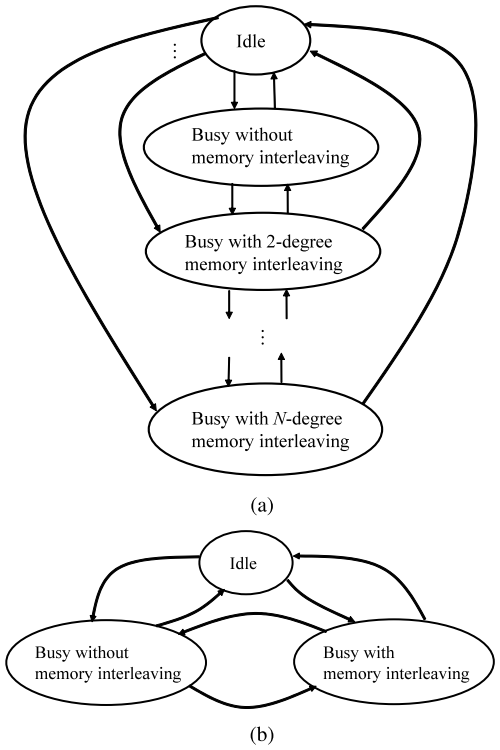


FIGURE 6. State transition for each server. (a) For general N. (b) For $N = 2$.

busy state with bank interleaving that has the least degree of interleaving, is selected.

Figure 6(a) shows a state transition diagram for each server. When sever (n, s) in idle state serves a request, the state moves to busy state with w-degree bank interleaving so as to minimize the value of w. When server (n, s) in busy state with w-degree bank interleaving finishes serving a request and does not serve any request, it enters idle state. When server (n', s) in idle state starts to serve a request, server (n, s) $(n \neq n')$ in busy state with w-degree bank interleaving moves to busy state with $(w + 1)$-degree bank interleaving. When server (n', s) finishes serving a request and does not serve any new request, server (n, s) $(n \neq n')$ in busy state with w-degree bank interleaving enters busy state with $(w - 1)$-degree bank interleaving.

We assume that a request arrives at the table lookup subsystem following a Poisson arrival process with average rate of λ, and the distributor based on a hash function distributes the request among N queues. Therefore, a request is assumed to arrive at each queue based on a Poisson arrival process with average rate of $\frac{\lambda}{N}$. We assume that the service rate of server (s, n) follows an exponential distribution with average service rate of μ_w for w-degree bank interleaving, where $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_N$ with $w\mu_w \geq \mu_1$.

A theoretical model can provide the accurate performance evaluation for the proposed architecture. The result from the theoretical model can also be the reference for that from a simulator. In this paper, we focus on analyzing the case for $N = 2$, as it is the simplest case that includes bank interleaving. The introduced methods can be used to build the analysis models for other cases with $N > 2$. This paper is the first work building two queueing models to analyze the performance of proposed architecture under two types of traffic models. For each traffic model, we describe all feasible states of system with the proposed architecture and analyze the transitions between them with considering the case of $N = 2$.

Figure 6(b) shows a state transition diagram for each server. When server (n, s) in idle state serves a request, the state moves to busy state with or without bank interleaving. When server (n, s) in busy state with bank interleaving finishes serving a request and does not serve any new request, it enters idle state. When server (n', s) in idle state starts to serve a request, server (n, s) $(n \neq n')$ in busy state without bank interleaving moves to busy state with bank interleaving. When server (n', s) finishes serving a request and does not serve any new request, server (n, s) $(n \neq n')$ in busy state with bank interleaving moves to busy state without bank interleaving.

IV. ANALYSIS OF PROPOSED ARCHITECTURE FOR RANDOM ARRIVAL OF REQUESTS

A. STATES FOR SUBSYSTEM DESCRIPTION

We describe the analytical model of the proposed architecture with $N = 2$ to analyze its performance. Since we assume a Markov process for request arrivals and service times with and without bank interleaving, a state in the subsystem is expressed by (i, j, p), where $i \in [0, K]$ is the number of requests for bank 1, $j \in [0, K]$ is the number of requests for bank 2, and $p \in [0, S]$ is the number of requests being served with 2-interleaving for both banks. The service rates for requests being served without memory interleaving and with 2-interleaving are different. There can be some states with the same (i, j) but different p, for each of which the outgoing transfer rates to the states with $(i - 1, j)$ or $(i, j - 1)$ due to the termination of service of a request depend on the corresponding number of requests being served with 2-interleaving for both banks. Therefore, p is required to be included to identify a state. Since the memory resources are shared by queues 1 and 2, $i + j \leq K$ must be satisfied. Let X denote $[0, K]$.

We describe all possible feasible states to derivate the number of states. The states are divided into three cases for the values of i and j, two of which are further divided to several sub cases with considering the range of p. In case 1, i, j, and S are not equal to each other. In case 2, only two of them are equal. In case 3, all of them are equal. γ_1, γ_2 and γ_3 denote the number of feasible states for case 1, case 2, and case 3, respectively. Γ denotes the total number of feasible states in the subsystem, where $\Gamma = \gamma_1 + \gamma_2 + \gamma_3$.

In case 1, i, j, and S are not equal to each other $(i \neq j, i \neq S, j \neq S)$. As the range of p depends on $i, j, i+j$, and S, case 1 is divided into three sub cases, case 1a, case 1b, and case 1c, which depend on the range of i. γ_1^a, γ_1^b, and γ_1^c denote the number of feasible states for case 1a, case 1b and case 1c, respectively.

In case 1a, $i \in [0, \lfloor S/2 \rfloor]$, where the symbol of $\lfloor x \rfloor$ denotes the maximum integer that does not exceed x. When $i = 0$, we have $j \in (0, S) \cup (S, K]$ and $p = 0$. Therefore, there are $(S-1)+(K-S) = K-1$ feasible states in this situation. When $i \in [1, \lfloor S/2 \rfloor]$, the range of j is $[0, i) \cup (i, S-i] \cup (S-i, S) \cup (S, K-i]$. If $j \in [0, i)$, which means $j < i < S$ and $i + j \leq S$, we get $p \in [0, j]$, which has $j+1$ feasible states for each j. Therefore, there are $\sum_{j=0}^{i-1}(j+1) = \sum_{t=1}^{i} t$ feasible states in total for each i when $i \in [1, \lfloor S/2 \rfloor]$ and $j \in [0, i)$. If $j \in (i, S-i]$, which means $i < j < S$ and $i + j \leq S$, we get $p \in [0, i]$, which has $i+1$ feasible states for each j, where there are $S - i - i$ possibilities. Therefore, there are $(S-2i)(i+1)$ feasible states in total for each i when $i \in [1, \lfloor S/2 \rfloor]$ and $j \in (i, S-i]$. If $j \in (S-i, S)$, which means $i < j < S$ and $i + j > S$, we get $p \in [i+j-S, i]$, which yields $S-j+1$ feasible states for each j. Therefore, there are $\sum_{j=S-i+1}^{S-1}(S-j+1) = \sum_{t=2}^{i} t$ feasible states in total for each i when $i \in [1, \lfloor S/2 \rfloor]$ and $j \in (S-i, S)$. If $j \in (S, K-i]$, which means $i < S < j, p = i$. Therefore, there are $K-S-i$ feasible states in total for each i when $i \in [1, \lfloor S/2 \rfloor]$ and $j \in (S, K-i]$. As a result, by summing all of the number of feasible states for each $i \in [0, \lfloor S/2 \rfloor]$, the total number of feasible states for case 1a is given by

$$\gamma_1^a = K - 1 + \sum_{i=1}^{\lfloor S/2 \rfloor} \left[\sum_{t=1}^{i} t + (S-2i)(i+1) + \sum_{t=2}^{i} t + (K-S-i) \right]. \quad (1)$$

In case 1b, $i \in [\lfloor S/2 \rfloor + 1, S)$ and the range of j is $[0, S-i] \cup (S-i, i) \cup (i, S) \cup (S, K-i]$. If $j \in [0, S-i]$, which means $j < i < S$ and $i + j \leq S$, we get $p \in [0, j]$, which has $j+1$ feasible states for each j. Therefore, there are $\sum_{j=0}^{S-i}(j+1) = \sum_{t=1}^{S-i+1} t$ feasible states in total for each i when $i \in [\lfloor S/2 \rfloor + 1, S)$ and $j \in [0, S-i]$. If $j \in (S-i, i)$, which means $j < i < S$ and $i + j > S$, we get $p \in [i+j-S, j]$, which yields $S-i+1$ feasible states for each j, where there are $i - (S-i) - 1 = 2i - S - 1$ possibilities. Therefore, there are $(2i - S - 1)(S - i + 1)$ feasible states in total for each of i when $i \in [\lfloor S/2 \rfloor + 1, S)$ and $j \in (S-i, i)$. If $j \in (i, S)$, which means $i < j < S$ and $i + j > S$, we get $p \in [i + j - S, i]$, which has $S - j + 1$ feasible states for each j. Therefore, there are $\sum_{j=i+1}^{S-1}(S-j+1) = \sum_{t=2}^{S-i} t$ feasible states in total for each i when $i \in [\lfloor S/2 \rfloor + 1, S)$ and $j \in (i, S)$. If $j \in (S, K-i]$, which means $i < S < j$, $p = i$. Therefore, there are $K - S - i$ feasible states in total for each i when $i \in [\lfloor S/2 \rfloor + 1, S)$ and $j \in (S, K-i]$. As a result, by summing all of the number of feasible states for each $i \in [0, \lfloor S/2 \rfloor]$, the total number of feasible states for case 1b is given by

$$\gamma_1^b = \sum_{i=\lfloor S/2 \rfloor + 1}^{S-1} \left[\sum_{t=1}^{S-i+1} t + (2i - S - 1)(S - i + 1) + \sum_{t=2}^{S-i} t + (K - S - i) \right]. \quad (2)$$

Case 1c is divided into the four cases of $i \in (S, \lfloor K/2 \rfloor]$, $i \in (\lfloor K/2 \rfloor, K-S)$, $i = K-S$, and $i \in (K-S, K]$. If $i \in (S, \lfloor K/2 \rfloor]$, $j \in [0, S) \cup (S, i) \cup (i, K-i]$,[1] and there are $K-i-1$ possibilities of j. If $i \in (\lfloor K/2 \rfloor, K-S)$, $j \in [0, S) \cup (S, K-i]$, and there are $K-i$ possibilities of j. If $i = K-S$, $j \in [0, K-i)$, and there are $K-i$ possibilities of j. If $i \in (K-S, K]$, $j \in [0, K-i]$, and there are $K-i+1$ possibilities of j. As a result, the total number of feasible states for case 1c is given by

$$\gamma_1^c = \sum_{S+1}^{\lfloor K/2 \rfloor} (K-i-1) + \sum_{\lfloor K/2 \rfloor + 1}^{K-S} (K-i) + \sum_{K-S+1}^{K} (K-i+1). \quad (3)$$

Therefore, the total number of feasible states for case 1 is given by

$$\gamma_1 = \gamma_1^a + \gamma_1^b + \gamma_1^c. \quad (4)$$

In case 2, only two of them are equal. There are six sub cases, which are $i = j < S$ for case 2a, $S < i = j$ for case 2b, $i < j = S$ for case 2c, $j = S < i$ for case 2d, $j < i = S$ for case 2e, and $i = S < j$ for case 2f. γ_2^a, γ_2^b, γ_2^c, γ_2^d, γ_2^e, and γ_2^f denote the number of feasible states for case 2a, case 2b, case 2c, case 2d, case 2e, and case 2f, respectively. In case 2a, if $i \in [0, \lfloor S/2 \rfloor]$, which means $i + j \leq S$, we get $p \in [0, i]$, which yields $i+1$ feasible states for each i, so there are $\sum_{t=0}^{\lfloor S/2 \rfloor}(t+1)$ feasible states. If $i \in [\lfloor S/2 \rfloor + 1, S)$, which means $i + j > S$, we get $p \in [2i - S, i]$, which yields $S - i + 1$ feasible states for each i, so there are $\sum_{t=\lfloor S/2 \rfloor + 1}^{S-1}(S - t + 1)$ feasible states. Therefore, the number of feasible states for case 2a is given by $\gamma_2^a = \sum_{t=0}^{\lfloor S/2 \rfloor}(t+1) + \sum_{t=\lfloor S/2 \rfloor + 1}^{S-1}(S-t+1)$. In case 2b with $S < i = j$, clearly, $p = S$ if $i \in (S, \lfloor K/2 \rfloor]$. Therefore, $\lfloor K/2 \rfloor - S$ feasible states can be obtained for case 2b, or $\gamma_2^b = \lfloor K/2 \rfloor - S$. In case 2c with $i < j = S$, clearly, $p = i$ if $i \in [0, S)$. Therefore, S feasible states are obtained for case 2c, or $\gamma_2^c = S$. In case 2d with $j = S < i$, clearly, $p = S$ if $i \in (S, K-S]$. Therefore, $K - 2S$ feasible states are obtained for case 2d, or $\gamma_2^d = K - 2S$. In case 2e with $j < i = S$, clearly, $p = j$ if $j \in [0, S)$. Therefore, S feasible states are obtained for case 2e, or $\gamma_2^e = S$. In case 2f with $i = S < j$, clearly, $p = S$ if $j \in (S, K-S]$. Therefore, $K - 2S$ feasible states are obtained for case 2f, or $\gamma_2^f = K - 2S$. As a result, γ_2 is given by

$$\begin{aligned} \gamma_2 &= \gamma_2^a + \gamma_2^b + \gamma_2^c + \gamma_2^d + \gamma_2^e + \gamma_2^f \\ &= \sum_{t=0}^{\lfloor S/2 \rfloor} (t+1) + \sum_{t=\lfloor S/2 \rfloor + 1}^{S-1} (S-t+1) \\ &\quad + \lfloor K/2 \rfloor + 2K - 3S \end{aligned} \quad (5)$$

[1] When K is an even number and $i = K/2$, $(i, K-i]$ is $(i, i]$, which is an empty set.

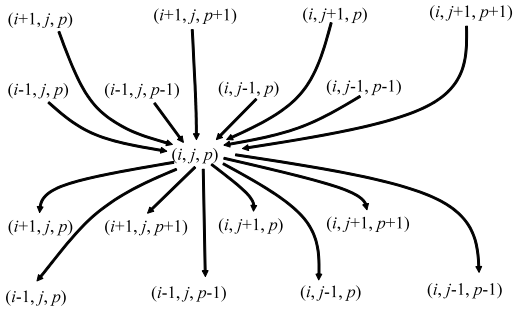


FIGURE 7. State transitions incoming to and outgoing from state (i, j, p).

In case 3, all of i, j and S are equal ($i = j = S$). p is always equal to S and there is just one feasible state for case 3, $\gamma_3 = 1$.

Therefore, by summing all the number of states for each case, the total number of feasible states in the subsystem is given by,

$$\Gamma = \gamma_1 + \gamma_2 + \gamma_3 \tag{6}$$

Based on the discussion on feasible states in the subsystem, we obtain the range of p as $p \in [\min(\max(0, i+j-S), i, j, S), \min(i, j, S)]$. Let $Y(i, j)$ denote $[\min(\max(0, i+j-S), i, j, S), \min(i, j, S)]$.

B. STATE TRANSITION FOR (i, j, p)

Figure 7 shows the state transitions incoming to and outgoing from state (i, j, p), where eight states are incoming to and eight states are outgoing from state (i, j, p). Table 2 describes the rate and condition for each transition. We number the cases from 1 to 16.

C. EQUILIBRIUM STATES

Let $P(i, j, p)$ be the probability that the subsystem is in state (i, j, p). Let U be the set of states (i, j, p), where $i \in X, j \in X$, and $p \in Y(i, j)$. In the equilibrium state, the total incoming flows to state (i, j, p) are equal to the total outgoing flows from state (i, j, p). The equilibrium equations for $(i, j, p) \in U$ are given by,

$$\begin{aligned}(q_1 + q_2 + q_3 + q_4 + q_5 + q_6 + q_7 + q_8)P(i, j, p) \\ = q_9 P(i-1, j, p) + q_{10} P(i-1, j, p-1) \\ + q_{11} P(i, j-1, p) + q_{12} P(i, j-1, p-1) \\ + q_{13} P(i+1, j, p) + q_{14} P(i+1, j, p+1) \\ + q_{15} P(i, j+1, p) + q_{16} P(i, j+1, p+1),\end{aligned} \tag{7}$$

where q_c, $c \in [1, 16]$, equals the transfer rate of case c if the conditions of case c are satisfied and 0 otherwise.

The condition that the sum of all state probabilities equals one is given by,

$$\sum_{(i,j,p) \in U} P(i, j, p) = 1. \tag{8}$$

We can compute the probability of each state $P(i, j, p) \in U$ by solving the multiple equations of (7) and (8).

D. BLOCKING PROBABILITY AND AVERAGE WAITING TIME

We define the blocking probability $P_{\mathrm{b}}^{\mathrm{R}}$ as the probability that a request incoming to the table lookup subsystem is blocked with the condition of $i + j = K$, or the request is not able to enter the queue. $P_{\mathrm{b}}^{\mathrm{R}}$ is given by

$$P_{\mathrm{b}}^{\mathrm{R}} = \sum_{i \in X} \sum_{p \in Y(i,j)} P(i, K-i, p). \tag{9}$$

We define the average waiting time at the subsystem, W^{R}, as the average duration time from when a request enters the subsystem until the request exits the subsystem. The average

TABLE 2. State transitions incoming to and outgoing from state (i, j, p).

Direction	Case	State	Transfer rate	Condition
Incoming states	9	$(i-1, j, p)$	$\lambda/2$	$(0 \le i-1 < S$ and $S-(i-1+j-p) > 0)$ or $i-1 \ge S$
	10	$(i-1, j, p-1)$	$\lambda/2$	$S-(i+j-p) \le 0$ and $0 \le i-1 < S$ and $p > 0$
	11	$(i, j-1, p)$	$\lambda/2$	$(0 \le j-1 < S$ and $S-(i+j-1-p) > 0)$ or $j-1 \ge S$
	12	$(i, j-1, p-1)$	$\lambda/2$	$S-(i+j-p) \le 0$ and $0 \le j-1 < S$ and $p > 0$
	13	$(i+1, j, p)$	$(i+1-p)\mu_1$	$i+1 \le S$ and $(i+1+j) \le K$
			$(S-p)\mu_1 + p\mu_2$	$S < i+1 \le K$ and $(i+1+j) \le K$
	14	$(i+1, j, p+1)$	$(p+1)\mu_2$	$i+1 \le S$ and $(i+1+j) \le K$
	15	$(i, j+1, p)$	$(j+1-p)\mu_1$	$j+1 \le S$ and $(i+j+1) \le K$
			$(S-p)\mu_1 + p\mu_2$	$S < j+1 \le K$ and $(i+j+1) \le K$
	16	$(i, j+1, p+1)$	$(p+1)\mu_2$	$j+1 \le S$ and $(i+j+1) \le K$
Outgoing states	1	$(i-1, j, p)$	$(i-p)\mu_1$	$i \le S$
			$(S-p)\mu_1 + p\mu_2$	$i > S$
	2	$(i, j-1, p)$	$(j-p)\mu_1$	$j \le S$
			$(S-p)\mu_1 + p\mu_2$	$j > S$
	3	$(i-1, j, p-1)$	$p\mu_2$	$0 < i \le S$ and $p > 0$
	4	$(i, j-1, p-1)$	$p\mu_2$	$0 < j \le S$ and $p > 0$
	5	$(i+1, j, p)$	$\lambda/2$	$(i < S$ and $S-(i+j-p) > 0$ and $(i+1+j) \le K)$ or $(S \le i < K$ and $(i+1+j) \le K)$
	6	$(i+1, j, p+1)$	$\lambda/2$	$S-(i+j-p) \le 0$ and $i < S$ and $(i+1+j) \le K$
	7	$(i, j+1, p)$	$\lambda/2$	$(j < S$ and $S-(i+j-p) > 0$ and $(i+j+1) \le K)$ or $(S \le j < K$ and $(i+j+1) \le K)$
	8	$(i, j+1, p+1)$	$\lambda/2$	$(S-(i+j-p) \le 0$ and $j < S$ and $(i+j+1) \le K)$

number of requests in the subsystem, L^{R}, is given by

$$L^{\mathrm{R}} = \sum_{i\in X}\sum_{j\in X}\sum_{p\in Y(i,j)} iP(i,j,p) + \sum_{i\in X}\sum_{j\in X}\sum_{p\in Y(i,j)} jP(i,j,p)$$
$$= 2\sum_{i\in X}\sum_{j\in X}\sum_{p\in Y(i,j)} iP(i,j,p). \tag{10}$$

The first/second terms on the right hand side of the first equality indicate the average number of requests waiting at queue 1/queue 2 and those being served, respectively. The right hand side for the second equality is derived by using $\sum_{i\in X}\sum_{j\in X}\sum_{p\in Y(i,j)} iP(i,j,p) = \sum_{i\in X}\sum_{j\in X}\sum_{p\in Y(i,j)} jP(i,j,p)$.

By using Little's formula [25], $W^{\mathrm{R}} = \frac{L^{\mathrm{R}}}{\lambda}$. Let $\lambda_{\mathrm{e}}^{\mathrm{R}}$ be the throughput and let $W_{\mathrm{e}}^{\mathrm{R}}$ be the average effective average waiting time, which are defined by $\lambda_{\mathrm{e}}^{\mathrm{R}} = \lambda(1 - P_{\mathrm{b}}^{\mathrm{R}})$ and $W_{\mathrm{e}}^{\mathrm{R}} = \frac{L^{\mathrm{R}}}{\lambda_{\mathrm{e}}}$, respectively.

V. ANALYSIS OF PROPOSED ARCHITECTURE FOR BURSTY ARRIVAL OF REQUESTS

We adopt the Interrupted Poisson Process (IPP) as the packet arrival process to analyze the above table lookup subsystem under bursty traffic conditions.

A. OVERVIEW OF IPP

There are two states as regards the arrival of packets, ON and OFF states. The durations of ON and OFF states follow exponential distributions with average inter-arrival times $\frac{1}{\alpha}$ and $\frac{1}{\beta}$, respectively. Once the ON state finishes, the OFF state starts, and vice versa. The ON state includes a Poisson arrival process of packets with average rate λ. There is no request arriving at the table lookup subsystem when the state of the arrival of requests is OFF. The state transition diagram of IPP is shown in Figure 8.

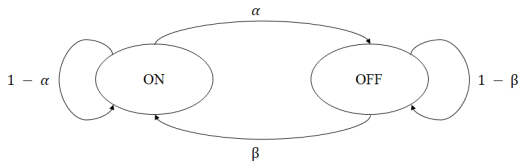


FIGURE 8. State transition diagram of IPP.

We assume that, in the ON state, packets are consecutively destined to the same bank until the ON state finishes. Let $k \in [0, N]$ denote the state of the arrival of a packet; k is set to $n \in [1, N]$ when it is ON state in which the packet is destined to bank $n \in [1, N]$, and zero otherwise. Consequently, for the table lookup subsystem with $N = 2$, a state in the subsystem is expressed as (i, j, p, k).

The total number of feasible states of (i, j, p, k) is 3Γ. Equation (6) gives Γ, which is the total number of feasible states of (i, j, p). In IPP, for each (i, j, p), there are three states where $k = 0, 1, 2$.

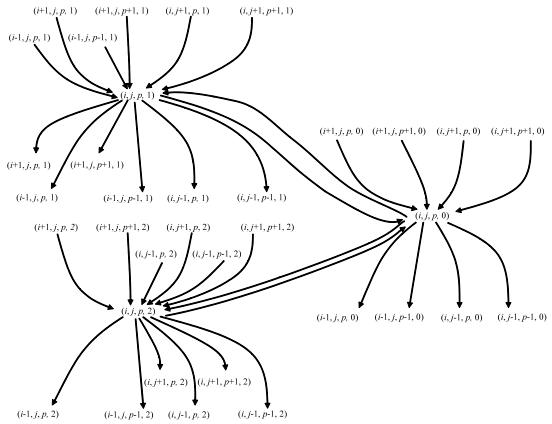


FIGURE 9. State transitions incoming to and outgoing from states $(i, j, p, 0)$, $(i, j, p, 1)$, and $(i, j, p, 2)$.

B. STATE TRANSITION FOR (i, j, p, k)

Figure 9 shows the state transitions incoming to and outgoing from states $(i, j, p, 0)$, $(i, j, p, 1)$, and $(i, j, p, 2)$. Tables 3, 4 and 5 describe the rates and conditions for states $(i, j, p, 0)$, $(i, j, p, 1)$, and $(i, j, p, 2)$, respectively. We number the cases from 1 to 22.

C. EQUILIBRIUM STATES

Let $P(i, j, p, k)$ be the probability that the subsystem is in state (i, j, p, k). Let V be the set of states (i, j, p, k), where $i \in X$, $j \in X$, and $p \in Y(i, j)$. In the equilibrium state, the total incoming flows to state (i, j, p, k) are equal to the total outgoing flows from state (i, j, p, k). The equilibrium equations for $(i, j, p, k) \in V$ are given by,

$$\begin{aligned}(r_1 + r_2 + r_3 + r_4 + r_{10} + r_{11})P(i,j,p,0) &= r_{16}P(i+1,j,p,0) + r_{17}P(i+1,j,p+1,0) \\ &\quad + r_{18}P(i,j+1,p,0) + r_{19}P(i,j+1,p+1,0) \\ &\quad + r_{21}P(i,j,p,1) + r_{22}P(i,j,p,2), \end{aligned} \tag{11a}$$

$$\begin{aligned}(r_1 + r_2 + r_3 + r_4 + r_5 + r_6 + r_9)P(i,j,p,1) &= r_{12}P(i-1,j,p,1) + r_{13}P(i-1,j,p-1,1) \\ &\quad + r_{16}P(i+1,j,p,1) + r_{17}P(i+1,j,p+1,1) \\ &\quad + r_{18}P(i,j+1,p,1) + r_{19}P(i,j+1,p+1,1) \\ &\quad + r_{20}P(i,j,p,0), \end{aligned} \tag{11b}$$

$$\begin{aligned}(r_1 + r_2 + r_3 + r_4 + r_7 + r_8 + r_9)P(i,j,p,2) &= r_{14}P(i,j-1,p,2) + r_{15}P(i,j-1,p-1,2) \\ &\quad + r_{16}P(i+1,j,p,2) + r_{17}P(i+1,j,p+1,2) \\ &\quad + r_{18}P(i,j+1,p,2) + r_{19}P(i,j+1,p+1,2) \\ &\quad + r_{20}P(i,j,p,0), \end{aligned} \tag{11c}$$

where r_c, $c \in [1, 22]$, equals the transfer rate of case c if the conditions of case c are satisfied and 0 otherwise.

The condition that the sum of all state probabilities equals one is given by,

$$\sum_{(i,j,p,k)\in V} P(i,j,p,k) = 1. \tag{12}$$

TABLE 3. State transitions incoming to and outgoing from state $(i, j, p, 0)$ in IPP.

Direction	Case	State	Transfer rate	Condition
Incoming states	16	$(i+1, j, p, 0)$	$(i+1-p)\mu_1$	$i+1 \leq S$ and $(i+1+j) \leq K$
			$(S-p)\mu_1 + p\mu_2$	$S < i+1 \leq K$ and $(i+1+j) \leq K$
	17	$(i+1, j, p+1, 0)$	$(p+1)\mu_2$	$i+1 \leq S$ and $(i+1+j) \leq K$
	18	$(i, j+1, p, 0)$	$(j+1-p)\mu_1$	$j+1 \leq S$ and $(i+j+1) \leq K$
			$(S-p)\mu_1 + p\mu_2$	$S < j+1 \leq K$ and $(i+j+1) \leq K$
	19	$(i, j+1, p+1, 0)$	$(p+1)\mu_2$	$j+1 \leq S$ and $(i+j+1) \leq K$
	21	$(i, j, p, 1)$	α	
	22	$(i, j, p, 2)$	α	
Outgoing states	1	$(i-1, j, p, 0)$	$(i-p)\mu_1$	$i \leq S$
			$(S-p)\mu_1 + p\mu_2$	$i > S$
	2	$(i, j-1, p, 0)$	$(j-p)\mu_1$	$j \leq S$
			$(S-p)\mu_1 + p\mu_2$	$j > S$
	3	$(i-1, j, p-1, 0)$	$p\mu_2$	$0 < i \leq S$ and $p > 0$
	4	$(i, j-1, p-1, 0)$	$p\mu_2$	$0 < j \leq S$ and $p > 0$
	10	$(i, j, p, 1)$	$\beta/2$	
	11	$(i, j, p, 2)$	$\beta/2$	

TABLE 4. State transitions incoming to and outgoing from state $(i, j, p, 1)$ in IPP.

Direction	Case	State	Transfer rate	Condition
Incoming states	12	$(i-1, j, p, 1)$	λ	$(0 \leq i-1 < S$ and $S-(i-1+j-p) > 0)$ or $i-1 \geq S$
	13	$(i-1, j, p-1, 1)$	λ	$S-(i+j-p) \leq 0$ and $0 \leq i-1 < S$ and $p > 0$
	16	$(i+1, j, p, 1)$	$(i+1-p)\mu_1$	$i+1 \leq S$ and $(i+1+j) \leq K$
			$(S-p)\mu_1 + p\mu_2$	$S < i+1 \leq K$ and $(i+1+j) \leq K$
	17	$(i+1, j, p+1, 1)$	$(p+1)\mu_2$	$i+1 \leq S$ and $(i+1+j) \leq K$
	18	$(i, j+1, p, 1)$	$(j+1-p)\mu_1$	$j+1 \leq S$ and $(i+j+1) \leq K$
			$(S-p)\mu_1 + p\mu_2$	$S < j+1 \leq K$ and $(i+j+1) \leq K$
	19	$(i, j+1, p+1, 1)$	$(p+1)\mu_2$	$j+1 \leq S$ and $(i+j+1) \leq K$
	20	$(i, j, p, 0)$	$\beta/2$	
Outgoing states	1	$(i-1, j, p, 1)$	$(i-p)\mu_1$	$i \leq S$
			$(S-p)\mu_1 + p\mu_2$	$i > S$
	2	$(i, j-1, p, 1)$	$(j-p)\mu_1$	$j \leq S$
			$(S-p)\mu_1 + p\mu_2$	$j > S$
	3	$(i-1, j, p-1, 1)$	$p\mu_2$	$0 < i \leq S$ and $p > 0$
	4	$(i, j-1, p-1, 1)$	$p\mu_2$	$0 < j \leq S$ and $p > 0$
	5	$(i+1, j, p, 1)$	λ	$(i < S$ and $S-(i+j-p) > 0$ and $(i+1+j) \leq K)$ or $(S \leq i < K$ and $(i+1+j) \leq K)$
	6	$(i+1, j, p+1, 1)$	λ	$S-(i+j-p) \leq 0$ and $i < S$ and $(i+1+j) \leq K$
	9	$(i, j, p, 0)$	α	

TABLE 5. State transitions incoming to and outgoing from state $(i, j, p, 2)$ in IPP.

Direction	Case	State	Transfer rate	Condition
Incoming states	14	$(i, j-1, p, 2)$	λ	$(0 \leq j-1 < S$ and $S-(i+j-1-p) > 0)$ or $j-1 \geq S$
	15	$(i, j-1, p-1, 2)$	λ	$S-(i+j-p) \leq 0$ and $0 \leq j-1 < S$ and $p > 0$
	16	$(i+1, j, p, 2)$	$(i+1-p)\mu_1$	$i+1 \leq S$ and $(i+1+j) \leq K$
			$(S-p)\mu_1 + p\mu_2$	$S < i+1 \leq K$ and $(i+1+j) \leq K$
	17	$(i+1, j, p+1, 2)$	$(p+1)\mu_2$	$i+1 \leq S$ and $(i+1+j) \leq K$
	18	$(i, j+1, p, 2)$	$(j+1-p)\mu_1$	$j+1 \leq S$ and $(i+j+1) \leq K$
			$(S-p)\mu_1 + p\mu_2$	$S < j+1 \leq K$ and $(i+j+1) \leq K$
	19	$(i, j+1, p+1, 2)$	$(p+1)\mu_2$	$j+1 \leq S$ and $(i+j+1) \leq K$
	20	$(i, j, p, 0)$	$\beta/2$	
Outgoing states	1	$(i-1, j, p, 2)$	$(i-p)\mu_1$	$i \leq S$
			$(S-p)\mu_1 + p\mu_2$	$i > S$
	2	$(i, j-1, p, 2)$	$(j-p)\mu_1$	$j \leq S$
			$(S-p)\mu_1 + p\mu_2$	$j > S$
	3	$(i-1, j, p-1, 2)$	$p\mu_2$	$0 < i \leq S$ and $p > 0$
	4	$(i, j-1, p-1, 2)$	$p\mu_2$	$0 < j \leq S$ and $p > 0$
	7	$(i, j+1, p, 2)$	λ	$(j < S$ and $S-(i+j-p) > 0$ and $(i+j+1) \leq K)$ or $(S \leq j < K$ and $(i+j+1) \leq K)$
	8	$(i, j+1, p+1, 2)$	λ	$(S-(i+j-p) \leq 0$ and $j < S$ and $(i+j+1) \leq K)$
	9	$(i, j, p, 0)$	α	

By considering the symmetric feature of states $(i, j, p, 1)$ and $(j, i, p, 2)$,

$$P(i, j, p, 1) = P(j, i, p, 2) \tag{13}$$

is satisfied. In (11a) and (12), $P(j, i, p, 2)$ is substituted by $P(i, j, p, 1)$ with (13). Then, (11c) can be omitted. The number of decision variables to be solved is reduced from 3Γ to 2Γ.

D. BLOCKING PROBABILITY AND AVERAGE WAITING TIME

Blocking probability $P_{\rm b}^{\rm B}$, which is the probability that a request incoming to the table lookup subsystem is blocked with $i+j=K$, or the request is not able to enter the queue, under the condition of ON state, is given by the following conditional probability.

$$P_{\rm b}^{\rm B} = \sum_{i\in X}\sum_{p\in Y(i,j)}\sum_{k=1}^{2} P(i,K-i,p,k) \Big/ \frac{\frac{1}{\alpha}}{\frac{1}{\alpha}+\frac{1}{\beta}}$$
$$= \frac{\alpha+\beta}{\beta}\sum_{i\in X}\sum_{p\in Y(i,j)}\sum_{k=1}^{2} P(i,K-i,p,k), \qquad (14)$$

note that $\frac{\frac{1}{\alpha}}{\frac{1}{\alpha}+\frac{1}{\beta}} = \frac{\beta}{\alpha+\beta}$ is the probability of ON state.

The average waiting time at the subsystem, $W^{\rm B}$ is the average duration time from when a request enters the subsystem until the request exits the subsystem. The average number of requests in the subsystem, $L^{\rm B}$, is given by

$$L^{\rm B} = \sum_{i\in X}\sum_{j\in X}\sum_{p\in Y(i,j)}\sum_{k=0}^{2} iP(i,j,p,k)$$
$$+\sum_{i\in X}\sum_{j\in X}\sum_{p\in Y(i,j)}\sum_{k=0}^{2} jP(i,j,p,k)$$
$$= 2\sum_{i\in X}\sum_{j\in X}\sum_{p\in Y(i,j)}\sum_{k=0}^{2} iP(i,j,p,k). \qquad (15)$$

The right hand side for the second equality is derived by using $\sum_{i\in X}\sum_{j\in X}\sum_{p\in Y(i,j)}\sum_{k=0}^{2} iP(i,j,p,k) = \sum_{i\in X}\sum_{j\in X}\sum_{p\in Y(i,j)}\sum_{k=0}^{2} jP(i,j,p,k)$.

By using Little's formula [25], $W^{\rm B}$ is given by, $W^{\rm B} = \frac{L^{\rm B}}{\lambda'}$. λ' is the average arrival rate over both ON and OFF states in the subsystem. λ' is given by $\lambda' = \frac{\frac{\lambda}{\alpha}}{\frac{1}{\alpha}+\frac{1}{\beta}} = \frac{\lambda\beta}{\alpha+\beta}$.

As with the analysis of random arrival of requests, let $\lambda_{\rm e}^{\rm B}$ be the throughput and $W_{\rm e}^{\rm B}$ be the average effective waiting time, which are defined by $\lambda_{\rm e}^{\rm B} = \lambda'(1-P_{\rm b}^{\rm B})$ and $W_{\rm e}^{\rm B} = \frac{L^{\rm B}}{\lambda_{\rm e}^{\rm B}}$, respectively.

VI. EVALUATION

Based on the analytical results calculated with the model and its analyses shown in Sections IV and V, we observe performance dependency on each system parameter and arrival pattern of requests of the proposed architecture. The model can incorporate system parameters that reflect a real implementation.

A. NUMERICAL RESULTS FOR RANDOM ARRIVAL OF REQUESTS

We evaluate $P_{\rm b}^{\rm R}$, $\lambda_{\rm e}^{\rm R}$ and $W_{\rm e}^{\rm R}$ of the proposed architecture and investigate their dependency on $\rho^{\rm R}$ and μ_2, by using the analysis presented in Section IV. As a reference model, we use the M/M/S/K model. We set $K=100$, and the arrival rate of λ is the same for both models. We set $S=32$ for both models unless otherwise stated. In M/M/S/K, the service rate is $\mu=1$, and, in the proposed architecture, $\mu_1=\mu=1$. Let $\rho^{\rm R}$ be the traffic load, which is defined by, $\rho^{\rm R}=\frac{\lambda}{S\mu}$. The analytical results are obtained by using a computer with 3.60GHz Intel Core i7-7700 CPU and 32GB memory. In the case of $S=32$ and $K=100$, the average computation time to obtain $P_{\rm b}^{\rm R}$ for each set of μ_2 and $\rho^{\rm R}$ in the proposed architecture is 252 [sec], where the number of states in the analysis is 10607.

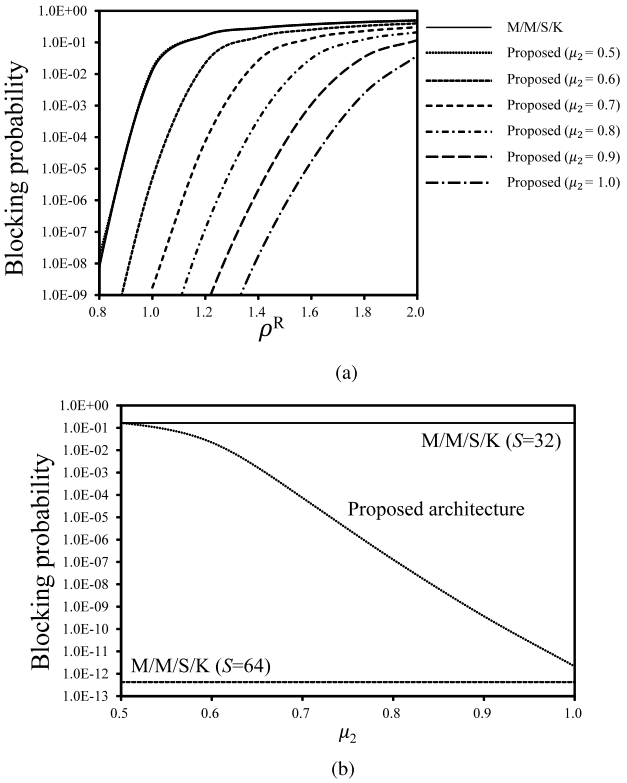


FIGURE 10. Blocking probability. (a) Depending on $\rho^{\rm R}$ with different μ_2. (b) Depending on μ_2 with $\rho^{\rm R}=1.2$.

Figure 10(a) shows the blocking probability dependency on $\rho^{\rm R}$ with different μ_2. In the proposed architecture, the blocking probability increases with $\rho^{\rm R}$, and decreases as μ_2 increases. The blocking probability of the proposed architecture with $\mu_2=0.5$ is close to, but slightly higher than, that of M/M/S/K. This is explained by the observation that, when two subsystems have the same value of the product of the number of servers and the service rate, the one with larger service rate outperforms the other. In addition, 32 servers with service rate $\mu=1$, all requests queued in the subsystem are served, outperform 64 ($=32\times2$) servers with service rate $\mu_2=0.5$, half of which serve requests queued in one of the two separate queues; the former has greater statistical multiplexing effect than the latter.

Figure 10(b) shows the blocking probability dependency on μ_2 with $\rho^{\rm R}=1.2$. In the proposed architecture, the blocking probability decreases as μ_2 increases. The blocking

probability of the proposed architecture with $\mu_2 = 1$ is close to, but slightly higher than, that of M/M/S/K with $S = 64$. This is explained by comparing 64 servers with service rate $\mu = 1$ and 64 ($= 32 \times 2$) servers with service rate $\mu_2 = 1$ as with the observation on Figure 10(a).

Figure 11 show the throughput dependency on ρ^{R} with different μ_2. The throughput in M/M/S/K increases with $\rho^{\mathrm{R}} \leq 1$, and is saturated with $\rho^{\mathrm{R}} > 1$. On the other hand, the throughput of the proposed architecture saturates at a larger point than M/M/S/K. The saturated throughput increases with μ_2.

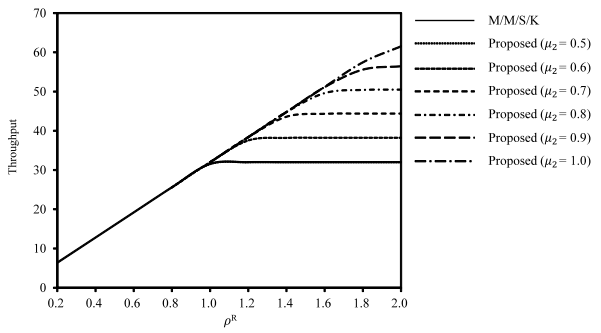


FIGURE 11. Throughput depending on ρ^{R} with different μ_2.

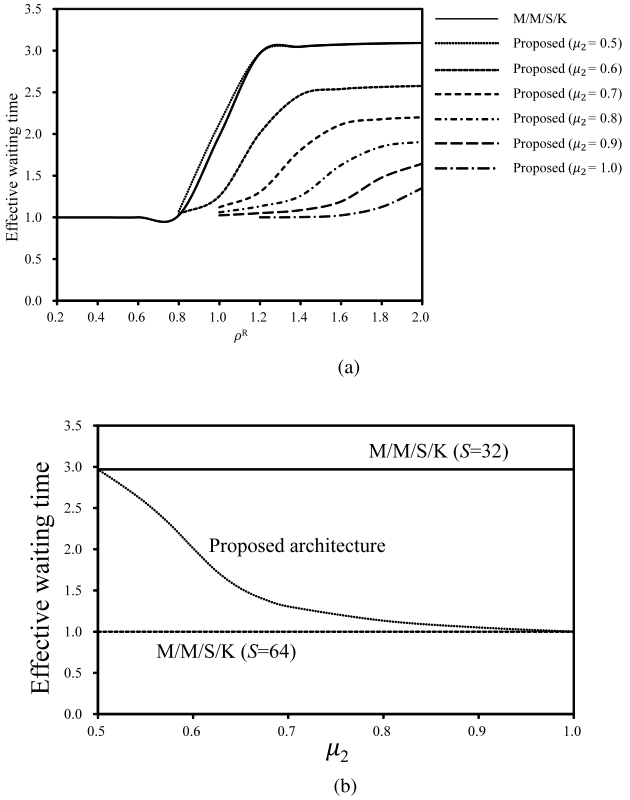


FIGURE 12. Effective waiting time. (a) Depending on ρ^{R} with different μ_2. (b) Depending on μ_2 with $\rho^{\mathrm{R}} = 1.2$.

Figure 12(a) show the effective waiting time dependency on ρ^{R} with different μ_2, where the effect of blocked requests is eliminated. The effective waiting time decreases as μ_2 increases. Figure 12(b) shows the effective waiting time dependency on μ_2. We have the observation similar to that on Figure 10(b).

B. NUMERICAL RESULTS FOR BURSTY ARRIVAL OF REQUESTS

We evaluate $P_{\mathrm{b}}^{\mathrm{B}}$, $\lambda_{\mathrm{e}}^{\mathrm{B}}$ and $W_{\mathrm{e}}^{\mathrm{B}}$ of the proposed architecture for IPP and investigate their dependency on ρ^{B} and μ_2, by using the analysis presented in section V. We compare the proposed architecture for IPP with a Poisson arrival process. We set $K = 100$ and $S = 32$ unless otherwise stated. Let ρ be the traffic load, which is defined by, $\rho^{\mathrm{B}} = \frac{\lambda'}{S\mu} = \frac{\lambda\beta}{(\alpha+\beta)S\mu}$.

For performance comparison, we use the same ρ^{B} for different models. The analytical results are obtained by using a computer with 3.60GHz Intel Core i7-7700 CPU and 32GB memory. In the case of $S = 32$ and $K = 100$, the average computation time to obtain P_{b} for each set of μ_2 and ρ^{B} in the proposed architecture is 853 [sec], where the number of states in the analysis is 21214.

We introduce parameters, $h > 0$ and $l > 0$, which are defined by $l = \frac{\alpha}{\lambda}$ and $h = \frac{\alpha}{\beta}$. Then, $\lambda' = \frac{\alpha}{(h+1)l}$ and $\rho^{\mathrm{B}} = \frac{\alpha}{(h+1)lS\mu}$. When $h \to 0$, IPP approaches a Poisson arrival process. Note that, with $h \to 0$, each packet continues to have the destination of the same bank, which is different from the model presented in Section IV. For any h, when $l \to \infty$, IPP approaches a Poisson arrival process and the proposed architecture with IPP approaches the model presented in Section IV, where the destination of each packet is randomly assigned to either bank.

Figure 13(a) shows the blocking probability dependency on l and h with $\rho^{\mathrm{B}} = 1.2$ for $\mu_2 = 0.7$. As h becomes large, the blocking probability increases. As l becomes large, the blocking probability decreases. We observe that, when $l \to \infty$, the blocking probability of IPP approaches that of the Poisson arrival process presented in Section IV for any h. The lower h is, the faster the blocking probability of IPP approaches that of the Poisson arrival process. $h \to 0$ indicates that ON state probability is close to 1, and $l \to \infty$ indicates that ON state period is close to zero. Each situation is equivalent to the Poisson arrival process, which is a special case of IPP. Figure 13(b) shows the blocking probability dependency on ρ^{B} and h with $l = 1.0$ for $\mu_2 = 0.7$.

Figures 14 and 15 show the throughput and effective waiting time dependency on l and h with $\rho^{\mathrm{B}} = 1.2$ for $\mu_2 = 0.7$, respectively.

C. PACKET PROCESSING PERFORMANCE

We can calculate the packet processing performance of the proposed architecture by using the numerical results shown in Sections VI-A and VI-B assuming that current carrier-scale packet processing performance is bounded by memory access performance. In the following evaluation of packet processing performance, we focus on IP routing as an example

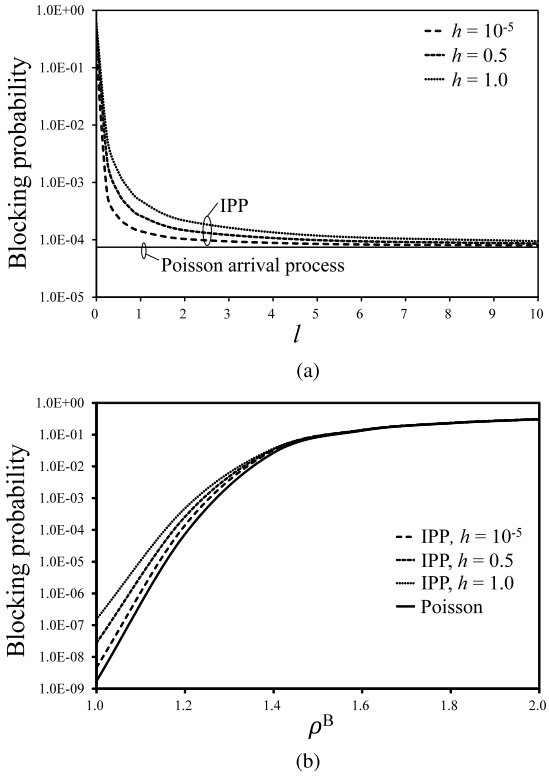


FIGURE 13. Blocking probability. (a) Dependency on l and h with $\rho^{\mathrm{B}} = 1.2$ and $\mu_2 = 0.7$. (b) Dependency on ρ^{B} and h with $l = 1.0$ and $\mu_2 = 0.7$.

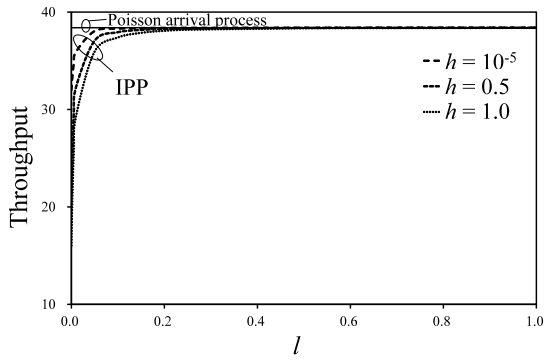


FIGURE 14. Throughput dependency on l and h with $\rho^{\mathrm{B}} = 1.2$ and $\mu_2 = 0.7$.

of carrier-scale packet processing. By using an IP address lookup algorithm such as DIR-24-8-BASIC [26], packet processing of IP routing is finished within one or two memory accesses at most. Its lookup table has in nature 2^{32} entries which correspond to the whole IPv4 address space. The content in each entry is the next hop information corresponding to the prefix of the entry. In detail, based on a real traffic pattern, the lookup tables comprise two lookup tables called TBL24 and TBLlong each of which has the entries corresponding to the upper 24 bits and lower 8 bits, respectively, where the longest prefix match is finished within two memory accesses. Therefore, we can calculate the packet processing performance of IP routing by using the number of achievable

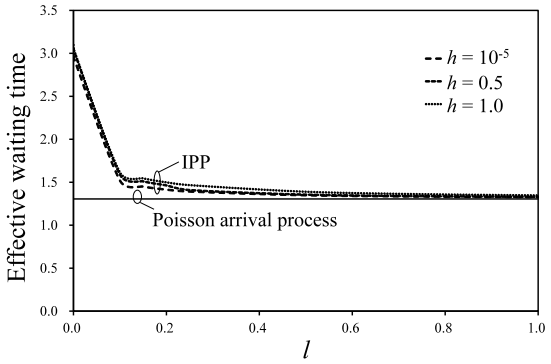


FIGURE 15. Effective waiting time dependency on l and h with $\rho^{\mathrm{B}} = 1.2$ and $\mu_2 = 0.7$.

TABLE 6. Example of packet processing performance.

Model	Throughput (Gbps) at 64B packet	Latency (ns)
M/M/S/K	66	887
Proposed (Poisson)	79	471
Proposed (IPP)	78	542

memory accesses per unit of time as derived from the numerical results in Sections VI-A and VI-B, as well as the number of necessary memory accesses to the IP address lookup per packet, and the additional delay of the distributor based on hash function. The packet processing latency is obtained as the number of required memory accesses times the sum of the average effective waiting time and the additional delay of the distributor. We assume that the hash function works as a pipeline, where it does not affect any other performance metric of the proposed architecture except for the latency.

In NFV-aware carrier network systems, multiple types of carrier-scale packet processing applications in the same system simultaneously. The DIR-24-8-BASIC algorithm requires 33 MB of memory for its routing table which corresponds to the whole IPv4 address space [26]. Therefore, cache memories inside the latest generation of CPUs, such as Intel Skylake, cannot accommodate the large tables for multiple carrier-scale packet processing applications.

For example, let service rate μ be 8 M services per second, which is an estimate since typical latency inside the HMC itself is usually taken to be between 100-180 ns with average of 125 ns [27], [28], and let the traffic load be 1.2. Table 6 lists the calculation results of M/M/S/K, proposed architecture with Poisson arrival, and that with IPP arrival. In this example, we assume the service rate of interleaved bank $\mu_2 = 0.7\mu$, $l = 0.1$, $h = 0.5$, and the additional delay of the distributor is 10 ns which corresponds to one clock cycle at 100 MHz circuits in the FPGA. We also assume that each IP address lookup requires two memory accesses each of which is associated with a request distribution based on hash function.

In the proposed architecture, memory access requests are served simultaneously by using multiple vaults of HMC.

This may change the order of egress packets from the processor, which affect the performance of upper layer such as Transmission Control Protocol (TCP) [29]. In order to eliminate the misordered packets, there are several approaches: to exchange signals among multiple processes or threads so that every packet can be served in order, or to buffer the packets and sort them before transmitted from the processor [30], [31].

VII. RELATED WORK

Several software-based packet processing schemes for COTS server implementation have been proposed [6]–[8], [32]. RouteBricks [6] is the first software-based router application to leverage the parallel processing offered by modern multi-core CPUs. Lagopus [7] is a DPDK-enabled OpenFlow switch that can achieve over 10 Gbps performance at more than 1 M flow entries without any hardware modification. These approaches significantly improved packet processing performance compared to previous software schemes running on single-core CPUs and DPDK. However, their performance directly depends on the high-speed cache memory of the CPUs, which unfortunately is too small to support carrier-scale packet processing given the huge multiple tables involved. PacketShader [32] consolidates parallel processing by using Graphics Processing Unit (GPU) to achieve nearly 40 Gbps packet processing performance. 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 [8] is the latest and fastest 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 [33]. 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. There is a study that utilizes 3D-stacked DRAM devices including HMC as the main memory of a system [34]. The work mainly details the production process of 3D-stacked DRAM devices, and there is no discussion on evaluating system performance. CasHMC [35] is a cycle-accurate simulator for HMC. This simulator does not consider bank interleaving, and the simulation results are valid only current HMC devices.

The thermal feasibility of a system with 3D-stacked DRAM is studied in [36]–[38]. They consider thermal feasibility of Processing in Memory (PIM). PIM uses logic layer functionality more aggressively, which produces more heat and requires stronger cooling systems. They conclude that PIM use cases with 3D-stacked DRAM are feasible if the system has high-end active cooling. Therefore, the proposed architecture is more feasible since it uses HMC for simple memory access, and so can use the commodity coolers of COTS systems.

Dividing an original table into several partial tables is studied in [39]–[41]. They divide a table to make table data more memory efficient; the goal is to make the table fit inside a device with fixed memory capacity such as a TCAM, on-chip memories in FPGA, and external SRAM.

Table update schemes are studied in studies that introduce table data structures such as [8], [26], [39]. In our proposed architecture, the HMC has multiple vaults to accommodate whole tables. Tables inside the HMC are updated sequentially for each vault by using a table update scheme corresponding to the table data structure.

VIII. DIRECTION TO EXPANSION OF ANALYTICAL MODEL FOR GENERAL $N \geq 2$

We consider the analytical model for a system with general $N \geq 2$. A state in the system is expressed by a vector, that consists of the following components. First, $i_n \in [0, K]$ is the number of requests for bank $n \in [1, N]$. Second, $p_{i_n i_{n'}} \in [0, S]$, where $n, n' \in [1, N]$ and $n \neq n'$, is the number of requests being served with 2-interleaving for banks n and n'. The number of $p_{i_n i_{n'}}$ is ${}_N C_2$, where ${}_n C_m = \frac{n!}{(n-m)!m!}$. Third, $p_{nn'n''} \in [0, S]$, where $n, n', n'' \in [1, N]$, $n \neq n'$, $n' \neq n''$, and $n'' \neq n$, is the number of requests being served with 3-interleaving for banks n, n', and n''. The number of $p_{nn'n''}$ is ${}_N C_3$. In the same way, we define $p_{nn'n''n'''\ldots}$, which is the number of requests being served with $(N-1)$-interleaving. The number of $p_{nn'n''n'''\ldots}$ for $(N-1)$-interleaving is ${}_N C_{N-1} = N-1$. Finally, $p_{123\cdots N} \in [0, S]$ is the number of being served requests with N-interleaving for all banks.

IX. CONCLUSION

We proposed an architecture that allows an HMC to support carrier-scale packet processing. The proposed architecture enhances memory access concurrency by leveraging the vault-level parallelism and bank interleaving offered by HMC. The architecture uses a hash-function-based distributor of memory requests to among the sets of vault and bank, each of which accommodates a portion of the original huge (carrier-scale) tables. We introduced an analytical model of a bank-interleaved HMC subsystem for two traffic patterns where the arrivals of memory requests are either random of bursty. The analytical results for random arrival of requests showed the performance of the proposed architecture and its dependency on traffic load and bank interleaving. The analytical result for bursty arrival of requests detailed the performance dependency on the burstiness of the input traffic. The evaluation result of packet processing performance showed that our proposed architecture achieves around 80 Gbps in carrier-scale packet processing wherein main memory accesses are inevitable; CPU cache memory is too small to accommodate the huge tables even if request arrivals are bursty.

REFERENCES

[1] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[2] *Mobile-Edge Computing—Introductory Technical White Paper*. Accessed: Jul. 1, 2018. [Online]. Available: https://portal.etsi.org

[3] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[4] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[5] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.

[6] 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," in *Proc. ACM SIGOPS*, 2009, pp. 15–28.

[7] *Lagopus Switch, a High Performance Software OpenFlow 1.3 Switch*. Accessed: Sep. 15, 2017. [Online]. Available: http://www.lagopus.org/

[8] 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.

[9] *Intel Data Plane Development Kit*. Accessed: Sep. 15, 2017. [Online]. Available: http://dpdk.org/

[10] *PCI-SIG Single Root I/O Virtualization (SR-IOV) Support in Intel Virtualization Technology for Connectivity*. Accessed: Sep. 15, 2017. [Online]. Available: https://www.intel.com

[11] T. Korikawa, A. Kawabata, and A. Masuda, "Toward carrier-scale general-purpose node," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Jan. 2017, pp. 536–540.

[12] T. Korikawa, A. Kawabata, F. He, and E. Oki, "Carrier-scale packet processing system using interleaved 3D-stacked DRAM," in *Proc. IEEE ICC*, May 2018, pp. 1–6.

[13] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Comput. Surv.*, vol. 37, no. 3, pp. 238–275, 2005.

[14] M. J. Akhbarizadeh, M. Nourani, R. Panigrahy, and S. Sharma, "A TCAM-based parallel architecture for high-speed packet forwarding," *IEEE Trans. Comput.*, vol. 56, no. 1, pp. 58–72, Jan. 2007.

[15] *Hybrid Memory Cube Specification 2.1*. Accessed: Jan. 21, 2019. [Online]. Available: http://www.hybridmemorycube.org/

[16] H. Jun, J. Cho, K. Lee, H.-Y. Son, K. Kim, H. Jin, and K. Kim, "HBM (high bandwidth memory) DRAM technology and architecture," in *Proc. IEEE Int. Memory Workshop (IMW)*, May 2017, pp. 1–4.

[17] N. Chatterjee, M. O'Connor, D. Lee, D. R. Johnson, S. W. Keckler, M. Rhu, and W. J. Dally, "Architecting an energy-efficient DRAM system for GPUs," in *Proc. IEEE HPCA*, Feb. 2017, pp. 73–84.

[18] *High-Bandwidth Memory (HBM) Reinventing Memory Technology*. Accessed: Jan. 21, 2019. [Online]. Available: https://www.amd.com/Documents/High-Bandwidth-Memory-HBM.pdf

[19] M. Zhu, Y. Zhuo, C. Wang, W. Chen, and Y. Xie, "Performance evaluation and optimization of HBM-enabled GPU for data-intensive applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 5, pp. 831–840, May 2018.

[20] *DDR4 SDRAM, Micron*. Accessed: Jan. 21, 2019. [Online]. Available: https://www.micron.com/products/dram/ddr4-sdram/

[21] 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," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs*, Nov./Dec. 2011, pp. 80–85.

[22] Y. Watanabe, Y. Kobayashi, T. Takenaka, T. Hosomi, and Y. Nakamura, "Accelerating NFV application using CPU-FPGA tightly coupled architecture," in *Proc. ICFPT*, Dec. 2017, pp. 136–143.

[23] D. J. M. Moss, S. Krishnan, E. Nurvitadhi, P. Ratuszniak, C. Johnson, J. Sim, A. Mishra, D. Marr, S. Subhaschandra, and P. H. W. Leong, "A customizable matrix multiplication framework for the intel HARPv2 Xeon+FPGA platform: A deep learning case study," in *Proc. ACM/SIGDA FPGA*, 2018, pp. 107–116.

[24] *2nd Generation Intel Xeon Scalable Processors Brief*. [Online]. Available: https://www.intel.com/content/www/us/en/products/docs/processors/xeon/ 2nd-gen-xeon-scalable-processors-brief.html

[25] J. D. C. Little, "A proof for the queuing formula: $L = \lambda W$," *Oper. Res.*, vol. 9, no. 3, pp. 383–387, Jun. 1961.

[26] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 1998, pp. 1240–1247.

[27] 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," in *Proc. IEEE IISWC*, Oct. 2017, pp. 66–75.

[28] 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," in *Proc. IEEE ISPASS*, Apr. 2018, pp. 99–108.

[29] K. C. Leung, V. O. K. Li, and D. Yang, "An overview of packet reordering in transmission control protocol (TCP): Problems, solutions, and challenges," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 4, pp. 522–535, Apr. 2007.

[30] S. Govind, R. Govindarajan, and J. Kuri, "Packet reordering in network processors," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, Mar. 2007, pp. 1–10.

[31] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling Internet routers using optics," in *Proc. ACM SIGCOMM*, 2003, pp. 189–200.

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

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

[34] T. Kirihata, J. Golz, M. Wordeman, P. Batra, G. W. Maier, N. Robson, T. L. Graves-Abe, D. Berger, and S. S. Iyer, "Three-dimensional dynamic random access memories using through-silicon-vias," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 3, pp. 373–384, Sep. 2016.

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

[36] M. J. Khurshid and M. Lipasti, "Data compression for thermal mitigation in the hybrid memory cube," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, Oct. 2013, pp. 185–192.

[37] Y. Zhu, B. Wang, D. Li, and J. Zhao, "Integrated thermal analysis for processing in die-stacking memory," in *Proc. 2nd MEMSYS*, 2016, pp. 402–414.

[38] Y. Eckert, N. Jayasena, and G. H. Loh, "Thermal feasibility of die-stacked processing in memory," in *Proc. 2nd Workshop Near-Data Process.*, 2014, pp. 1–5.

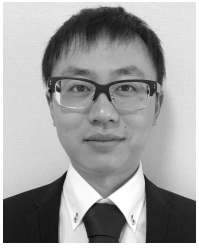
[39] Y. Wu and G. Nong, "A scalable pipeline architecture for IPv4/IPv6 route lookup," in *Proc. 18th IEEE ICON*, Dec. 2012, pp. 416–421.

[40] J.-Y. Huang and P.-C. Wang, "TCAM-based IP address lookup using longest suffix split," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 976–989, Apr. 2018.

[41] H. Le and V. K. Prasanna, "Scalable tree-based architectures for IPv4/v6 lookup using prefix partitioning," *IEEE Trans. Comput.*, vol. 61, no. 7, pp. 1026–1039, Jul. 2012.

TOMOHIRO KORIKAWA received the B.S. and M.S. degrees from Waseda University, Tokyo, Japan, in 2012 and 2014, respectively. In 2014, he joined as a Researcher of Network Service Systems Laboratories in Nippon Telegraph and Telephone (NTT) Corporation, Tokyo, Japan, where he is doing research in network system architecture and network design.

AKIO KAWABATA 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 (NTT) Corporation Communication Switching Laboratories, 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 from 2011 to 2014. He is an Executive Research Engineer and Project Manager of Network Service Systems Laboratories at NTT.



FUJUN HE received the B.E. and M.E. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with Kyoto University, Kyoto, Japan.

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 (M'95–SM'05–F'13) 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 with the University of Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar with 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.