TRiM: Tensor Reduction in Memory

Byeongho Kim, Jaehyun Park, Eojin Lee, Minsoo Rhu, and Jung Ho Ahn, *Senior Member, IEEE*

Abstract—Personalized recommendation systems are gaining significant traction due to their industrial importance. An important building block of recommendation systems consists of what is known as the embedding layers, which exhibit a highly memory-intensive characteristics. Fundamental primitives of embedding layers are the embedding vector gathers followed by vector reductions, which exhibit low arithmetic intensity and becomes bottlenecked by the memory throughput. To address this issue, recent proposals in this research space employ a near-data processing (NDP) solution at the DRAM rank-level, achieving a significant performance speedup. We observe that prior NDP solutions based on rank-level parallelism leave significant performance left on the table, as they do not fully reap the abundant data transfer throughput inherent in DRAM datapaths. Based on the observation that the datapath of the DRAM has a hierarchical tree structure, we propose a novel, fine-grained NDP architecture for recommendation systems, which augments the DRAM datapath with an "in-DRAM" reduction unit at the DDR4/5 rank/bank-group/bank level, achieving significant performance improvements over state-of-the-art approaches. We also propose hot embedding-vector replication to alleviate the load imbalance across the reduction units.

Index Terms—DRAM, in-memory processing, near-data processing

1 Deep-Learning-Based Recommendation System

Personalized recommendation systems based on deep learning (RecSys) have recently gained significant attention within the research community due to its industrial importance. For instance, Facebook states that recommendation systems (e.g., the Deep Learning Recommendation Model (DLRM) [7]) account for 80 percent of the AI inference cycles in their datacenters.

RecSys train the correlation between a user and an item as *embedding vectors* to infer the probability of the user clicking the item, recommending the highly ranked ones. Prior work [7] has shown that fully-connected (FC) and tensor gather-and-reduction (GnR) operations, which lookup a number of vectors (entries) from an embedding table and reduce those into one vector by element-wise operations, take up a significant fraction of the inference time of RecSys. While there have been numerous studies focusing on accelerating FC operations, little has been done regarding the acceleration of GnR, whose characteristics differ significantly from those of FC operations.

GnR performs simple reduction operation (e.g., an element-wise sum for SparseLengthsSum (SLS) in Caffe2 [1]) of the embedding vectors collected from multiple embedding table lookups. An embedding table takes the form of a matrix (rank-2 tensor) in which each row holds one embedding vector (rank-1 tensor). The number of elements in a row (hereafter referred to as the vector length) typically ranges from 32 to 256 [14]. As the on-chip storage of a processor is too small to store all of the embedding tables of RecSys, the size of which can exceed hundreds of GBs, embedding vectors are mostly read from the main-memory DRAM. As the compute-to-memory-access ratio of GnR is extremely low with little locality (i.e., several KBs to MBs of DRAM reads over several tens to hundreds of GBs of embedding tables), GnR is highly memory-intensive.

Such properties render GnR a prime candidate for acceleration using *near-data processing* (NDP) at the processor-memory interface. Indeed, TensorDIMM [10] and RecNMP [9] are two recent studies that explored the efficacy of NDP in accelerating GnR. However, we observe that the rank-level parallelism exploited by TensorDIMM and RecNMP does not fully reap the maximum potential of NDP acceleration, leaving significant performance capabilities on the table. In this work, we utilize the features of the DRAM data path structure to extract out additional internal bandwidth compared to NDP architectures based on rank-level parallelism, improving the performance of GnR. Additionally, our novel embedding replication strategy helps to alleviate the load imbalance issues of the prior methods, further improving the performance by enabling finer-grained memory-level parallelism.

2 DRAM Data-Path Structure

The data path of popular main-memory systems (e.g., DDR4 DRAM [3]) has a hierarchical tree structure (see Fig. 1). A memory channel as a root node (depth-0) consists of a primary host memory controller (MC) and multiple, secondary DRAM ranks (depth-1) connected through a depth-1 data bus. Each rank consists of several DRAM devices, all receiving the same command and address information and transferring data accordingly. A few ranks are physically housed in a module. A buffer chip or chipset connects an MC and the ranks in a module to alleviate the signal integrity issues [2].

In DDR4 DRAM, a rank consists of four bank-groups (depth-2), with each packed with four banks (depth-3). Similar to the depth-1 data bus, the depth-2 data bus and the depth-3 data bus are shared between one rank and four bank-groups, and between one bank-group and four banks, respectively, thus constituting a hierarchical multi-drop bus among ranks, bank-groups, and banks. While all of the banks in a DRAM device operate independently, only one bank can occupy depth-1/2/3 data buses at any given time, and the frequency of activations (the process of moving a row of a DRAM bank to the sense amplifiers in the bank preparing to read or write data) is limited (e.g., due to tRRD and tFAW) such that it cannot exceed the power limit of the DRAM device [3].

3 Key Challenges of Prior NDP Architecture

Prior NDP Accelerators for GnR. Fig. 2 conceptually explains how the baseline (`Base`) and NDP architectures handle GnR. In `Base`, which uses a conventional module-based DDR4 DRAM, as the MC and all ranks share the depth-1 bus, only one rank can occupy the bus and transfer data in a channel at a time. TensorDIMM and RecNMP, two recently proposed NDP architectures for GnR, employ processing elements (PEs) at the rank level, locating them inside the buffer chips. Each PE performs GnR for (a portion of) an embedding vector in parallel. As the depth-1 bus is not utilized during GnR in each module, each PE in a rank can receive data independently. The aggregated bandwidth for GnR can be as high as the channel bandwidth multiplied by the number of ranks in a channel (rank-level parallelism). Furthermore, data transfer energy is saved as data being reduced within the buffer chips do not consume data-transfer energy in the depth-1 bus. Only the reduced vectors are transferred through the depth-1 bus.

How an embedding table is mapped to DRAM affects the way a system handles GnR. In `Base`, the elements in an embedding vector exist at consecutive addresses in one DRAM row. After

- *Byeongho Kim, Jaehyun Park, Eojin Lee, and Jung Ho Ahn are with the Seoul National University, Seoul 08826, South Korea.*
  E-mail: {bhkim, jhpark}@scale.snu.ac.kr, {yohoyo, gajh}@snu.ac.kr.
- *Minsoo Rhu is with the KAIST, Daejeon 34141, South Korea.*
  E-mail: minsoo.rhu@gmail.com.

Manuscript received 20 Oct. 2020; accepted 18 Nov. 2020. Date of publication 7 Dec. 2020; date of current version 8 Jan. 2021.
(Corresponding author: Jung Ho Ahn.)
Digital Object Identifier no. 10.1109/LCA.2020.3042805

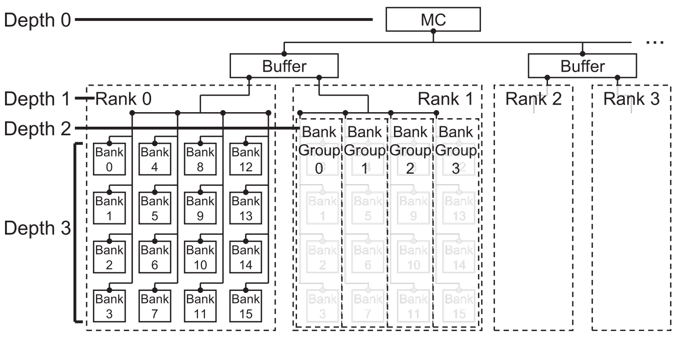

Fig. 1. Simplified data path structure of DDR4 DRAM.

activating the DRAM row containing the target embedding vector, elements are sequentially transferred to the host MC. Two recent NDP architectures adopt different embedding table mapping and reduction strategies. TensorDIMM splits the embedding table vertically. Each partition has a portion of vectors with the same number of elements of whole entries and maps the partitioned vectors of all entries to different ranks (*vertical partitioning*). In contrast, RecNMP evenly distributes the entries of an embedding table to each rank (*horizontal partitioning*). After reducing the vectors in the PEs, the PEs send a partial sum of the vectors to the host, and the host reduces the partial sums. Because the vector is reduced by an element-wise operation, the portion of the vector that arrives at the buffer chip first is reduced before the entire vector arrives at the

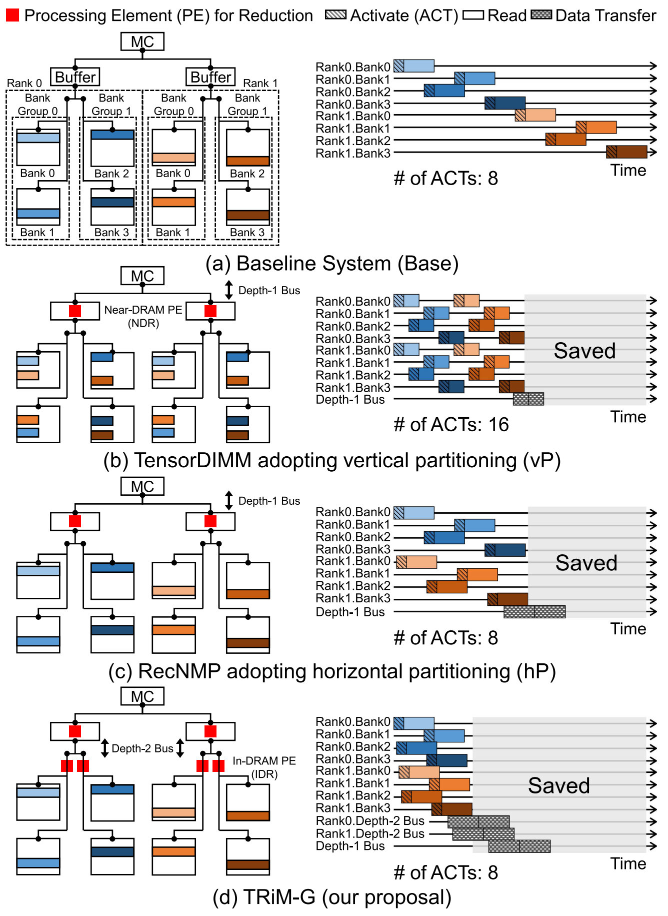

Fig. 2. Exemplar GnR in the baseline and evaluated NDP architectures. Diagrams on the left simplifies conceptual DRAM modules, where each has one rank consisting of 2 bank-groups, each packed with 2 banks, showing how embedding vectors are mapped to the DRAM devices. The timing diagram on the right shows the states of banks and buses while transferring data for or conducting partial GnR near/in DRAM.

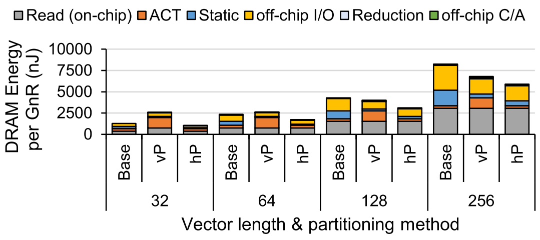

Fig. 3. DRAM energy breakdown of Base and the state-of-the-art NDP architectures when performing GnR.

buffer chip and can be transferred to the host. Moreover, data transfer to the host can be overlapped by the subsequent reduction of vectors by another GnR in the PEs.

An NDP architecture adopting vertical partitioning (vP) evenly distributes one vector among multiple ranks, where the number of row activations (ACTs) is multiplied by the number of ranks compared to Base and an NDP architecture adopting horizontal partitioning (hP). Thus, vP consumes more energy in ACTs for GnR as compared to hP. Moreover, if the partitioned vector in vP is smaller than the DRAM I/O granularity, internal bandwidth is underutilized due to redundant data reads.

Because each rank in hP processes embedding lookups of different entries, hP must transfer different command/address (C/A) signals to each rank. Therefore, hP requires a higher C/A bandwidth than vP. RecNMP mitigates the high C/A bandwidth pressure by compressing a pair of activate/precharge commands and several read commands into one custom instruction. A load balancing issue also exists in hP as each rank may receive a different number of embedding lookups for GnR.

Comparison of the NDP Accelerators. We measure the DRAM energy consumption of vP and hP with Base while changing the vector length from 32 to 256 (see Fig. 3). We set the Base and the NDP architectures to use commodity DRAM modules, DDR4-2400, with four ranks (2 DIMMs $\times$ 2 ranks) and estimate the power consumption of the DDR4 DRAM with industry datasheets [11] and off-chip I/O with CACTI [8]. We set the reduction operation of GnR to the element-wise sum (SLS) and the number of lookups for GnR (N_{lookup}) to 80, following the same configurations in DLRM [7].

Because the ACT energy of vP is four (the number of ranks per channel) times larger than that of Base and hP, hP is more energy-efficient than vP. When the vector lengths are 32 and 64, vP consumes more DRAM energy compared to Base and hP because the ACT energy accounts for a large portion of the DRAM energy consumption. As the vector length increases, the ACT energy is amortized over the off-chip I/O energy, DRAM read energy, and DRAM static energy. When the vector length is 256, off-chip data transfer from a buffer chip to a host MC is significantly reduced in vP and hP compared to Base. Moreover, the DRAM static energy in the NDP architectures is reduced due to the speedup in GnR. Thus, the energy consumption of vP and hP decreases by 18 and 30 percent over Base, respectively. Because half of the data are wasted for vP when the vector length is 32 (the size of a partitioned vector (8 $\times$ 4B for 32-bit floating-point) is smaller than the DRAM I/O granularity (64B)), there is no significant difference in energy consumption between the two vector lengths, 32 and 64. This waste causes the speedup in vP to be half of that in hP for a vector length of 32 (see Fig. 6). Energy consumed by the C/A signals and reduction operation slightly affects total energy consumption.

Proposed Approach. Because the data path of DRAM is organized as a tree structure, if there are PEs dedicated to memory nodes (e.g., ranks, bank groups, and banks) above a certain depth in an NDP architecture, embedding vectors can be hierarchically reduced from the depth. The number of ranks (up to several) is much smaller than N_{lookup} (dozens) for GnR such that the speedup of GnR by exploiting rank-level parallelism is limited; there is

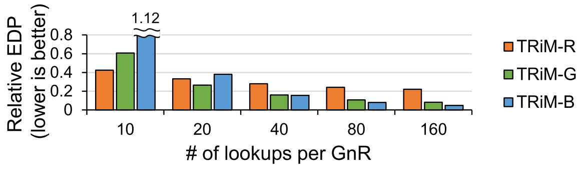

Fig. 4. Comparing the energy-delay product (EDP) of TRiM architectures according to N_{lookup}.

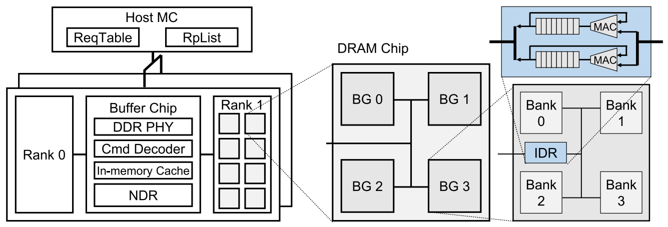

Fig. 5. TRiM-G architecture. Assuming ×8 data I/O bit-width, two 32-bit MACs are placed in IDR.

room to improve the performance further by exploiting finer-grained memory-level parallelism.

4 TRiM Architecture

We propose a finer-grained NDP architecture that maximally utilizes the abundant, *internal* DRAM bandwidth to accelerate the RecSys inference. Because TensorDIMM's vertical partitioning based table mapping function suffers from low internal bandwidth utilization with aggravated energy efficiency (due to the excessive partitioning of the vector and the significant increase in the number of DRAM row ACTs), our proposal assumes a horizontal partitioning scheme for maximal efficiency. As we dedicate more PEs to perform GnR in parallel compared to the prior NDP architectures, it is possible to utilize the vertical and horizontal partitioning schemes together. However, because this strategy inherits the disadvantages of both architectures, we do not take this direction.

Proposed NDP Architecture. We present TRiM (Tensor Reduction in Memory), a DRAM-based NDP architecture tailored to GnR operations. Our proposal is based on the key observation that the datapath constituting any given rank/bank-group/bank exhibits a tree-like interconnect topology, which opens up opportunities to conduct the GnR operation in a hierarchical fashion by employing the NDP-based PE unit per bank, bank group, or rank. Embodiments of the DDR4/5-based TRiM architecture include TRiM-R/G/B, corresponding to the depth **R**ank/bank-**G**roup/**B**ank to which a PE is allocated. From this perspective, RecNMP can be regarded as TRiM-R.

Fig. 2d highlights the key benefits of TRiM-G against prior NDP proposals for RecSys. First, In-DRAM PE for Reduction (IDR) gathers a series of embedding vectors from the corresponding DRAM banks within its local bank group, generating the final (partially) reduced vector in an accelerated manner. Multiples of IDR-reduced vectors are collected by Near-DRAM PE for Reduction (NDR) in aggregate for the next level of reduction at the rank level. The final output is eventually transferred back to the host MC. Because the entire in-memory reduction operation (at both IDR and NDR) is conducted within the shared datapath, the proposed NDP architecture fully reaps the benefits of the abundant memory read throughput unlocked with in-memory processing, thus achieving superior performance and energy efficiency.

The proper TRiM architecture for GnR differs depending on the characteristics of the GnR workload. Allocating the NDP PE for a certain memory depth allows GnR operations to exploit the internal bandwidth as much as the channel bandwidth multiplied by the number of the memory nodes (each equipped with a PE) in a memory channel. However, if N_{lookup} does not sufficiently exceed the number of PEs, this architecture could suffer from a significant load imbalance issue, resulting in low utilization of the internal DRAM data transfer bandwidth.

Exploring NDP Unit Placement. We measure the relative energy-delay product (EDP) of GnR, which is the average of the relative EDP values measured as we double the vector length from 32 to 256, of TRiM-R/G/B while changing N_{lookup} compared to the base (see Fig. 4). To estimate the performance of GnR at a production-scale workload, we generate an embedding table access trace with a Zipf distribution according to the studies [4] because the real trace is not open to the public. The detailed parameters are set based on the information in the prior work [6], [9].

As N_{lookup} increases, the relative EDP improves (decreases) for all TRiM architectures, but EDP improves more for those adopting finer-grained memory-level parallelism. This occurs because the larger the N_{lookup} is, the higher the utilization of the internal DRAM data transfer bandwidth becomes, increasing the performance of GnR. TRiM-G and TRiM-B outperform TRiM-R when N_{lookup} is larger than 20. When N_{lookup} is 40 or 80, the relative EDP of TRiM-B is slightly better than that of TRiM-G. However, considering that TRiM-B incurs 4× more area overhead than TRiM-G as it populates a PE per bank, not a bank group, TRiM-G is a better option compared to TRiM-B in the range of N_{lookup} (between 20 and 80) covered by DLRM [9]. Hereafter, we detail the microarchitecture for TRiM-G.

Mitigating Load Imbalances Through Replication. At a given N_{lookup}, a memory node with a PE receives fewer embedding vectors to reduce when TRiM exploits finer-grained parallelism, potentially experiencing load imbalance problems. RecNMP alleviates this problem by processing multiple GnR operations simultaneously. However, this induces area overhead as the PE of each node needs dedicated registers to hold a partial sum for each overlapped GnR operation, which can be problematic as more PEs are deployed for GnR.

We propose to tackle the load imbalance challenges of prior NDP designs with a novel *hot entry replication* scheme, which copies frequently accessed (hot) entries to each memory node. We can balance the load between the nodes by distributing hot requests to access the replicated entries at the nodes with lower loads. An MC receives requests to access entries for a GnR operation, counts the number of requests heading to each memory node at a table (ReqTable), and sends the DRAM commands to the nodes. The MC has a list of (index, address) pairs of the replicated entries (RpList), where the replicated entries exist at the same relative locations across all memory nodes. When processing a request to access an entry in RpList, the MC sends a command to the node with the minimal load at the ReqTable, after which it updates the table.

Replicating more entries further alleviates the load imbalance, but the capacity overhead increases as more entries must be stored in DRAM. Therefore, a trade-off between the number of replicated entries and the degree of load imbalance mitigation should be considered. As we detail later, our proposal can reduce the load imbalance problem by 39.2 percent, with only a 1.6 percent increase in the memory capacity due to replicated entries.

Putting it All Together. Fig. 5 illustrates the proposed hierarchical NDP architecture for RecSys, TRiM-G, where an NDP PE tailored to GnR is employed per bank group as well as per rank. The IDR, dedicated to each bank group, is located between the bank-group I/O multiplexer (MUX) and global I/O MUX. The NDR, dedicated to each rank, is placed on the buffer chip. Both IDR and NDR consist of a 32-bit floating-point multiply-add units (MACs) for vector reductions and includes registers that temporarily store the partial sums of the reduced vectors. The buffer chip includes a lightweight command decoder that processes compressed instructions and an in-memory cache for NDR to process embedding entries with high locality effectively [9]. TRiM leverages the method of compressing the instructions

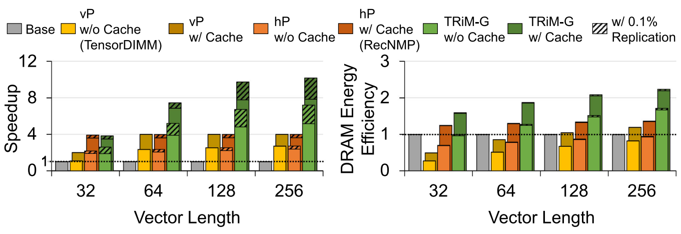

Fig. 6. Performance and DRAM energy efficiency of different architectures. Cache denotes in-memory cache [9].

sent from a host MC into ranks from RecNMP [9], allowing each rank to send DRAM commands to the bank group independently. Commands for GnR (e.g., reading a vector from a DRAM and reducing it with the vector in the IDR registers) are defined using the reserved-for-future-use (RFU) command of DDR4.

Design Overhead. We synthesized both the NDR and IDR units using the Synopsys Design Compiler with 40 nm CMOS technology at a frequency of 300 and 200 MHz and scaled the result of IDR to a 20 nm DRAM process referring to [5], [12]. The total area overhead of IDRs is 0.903 mm^2 per DRAM chip, assuming that each chip has $\times 8$ data I/O bit-width. Using an 8 Gb DDR4 die, the reduction units increase the chip area by 1.6 percent, incurring a small overhead; thus, increasing the driver strength of inter-bank datapath by a small degree keeps DRAM access latency unchanged with a minimal increase in read/write energy [13]. The additional area overhead of NDR and a 128 KB in-memory cache per rank modeled by CACTI [8] is 0.658 mm^2, which is lower than that of the prior NDP architecture [2].

Evaluation of TRiM on the RecSys Workload. We compared TRiM-G with previous studies on the workload mentioned above. We set the replication rate to 0.1 percent, which incurs a 1.6 percent capacity overhead. The cache hit rate of `Base`, which processes GnR on the host, is set to that of in-memory cache in the NDP architectures. Fig. 6 shows the performance and energy efficiency of `Base`, `vP`, `hP`, and TRiM-G, with or without applying the in-memory cache and the replication scheme for various vector lengths. The energy consumption of GnR in `Base` is conservatively evaluated as the energy of `Base` for cache access, with reduction in the CPU not included. TRiM-G processes 8 GnRs (640 lookups) at a time, as in prior work [9]. Also, we allocate an entire embedding table in one MC. `Base` uses a DDR4-2400 DRAM module exploiting its peak bandwidth.

TRiM-G (with in-memory cache) achieves a speedup of up to 5.19× (7.86×) over `Base` and up to 2.17× (2.16×) over `vP` and `hP` owing to the increase in the internal bandwidth caused by exploiting finer bank-group-level parallelism. 0.1 percent replication improves the performance by up to 29.3 and 39.3 percent with and without in-memory cache, respectively. Our hot-entry replication scheme is more effective in TRiM-G than `vP` and `hP` as it works better for TRiM with finer-grained memory-level parallelism. For the vector length of 32, performance improvement is relatively low as the DRAM timing parameters constrain the frequency of ACT to limit power dissipation, whereas a smaller vector length needs fewer reads per ACT, demanding more frequent ACT for higher performance. TRiM-G consumes up to 69.7 percent lower DRAM energy than `Base`, and up to 83.1 and 39.5 percent lower DRAM energy than TensorDIMM and RecNMP, respectively, because the amount of data transferred from the IDRs in DRAM chips to the buffer chips of TRiM-G, which is one vector for each bank-group per GnR, is smaller than that of `vP` and `hP`. The impact of hot-entry replication on the energy efficiency is negligible because this scheme does not change the total number of accesses in the table entries.

5 Conclusion

We have proposed TRiM, a fine-grained NDP architecture that accelerates the embedding vector GnR in recommendation systems. First, we identified the challenges of the state-of-the-art NDP architectures for accelerating GnR and the potential for further energy-efficiency improvements by unlocking the inherent bandwidth amplification opportunities within the DRAM chip's tree-topology-based datapath. Our results demonstrated that TRiM improves the embedding-lookup performance by up to 3.85× and 2.79× over the state-of-the-art NDP architectures, TensorDIMM and RecNMP, respectively.

Acknowledgment

This work was supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT under Grant NRF-2018R1A5A1059921.

References

[1] Caffe2, 2017. [Online]. Available: https://caffe2.ai
[2] H. Asghari-Moghaddam, Y. H. Son, J. Ahn, and N. S. Kim, "Chameleon: Versatile and practical near-DRAM acceleration architecture for large memory systems," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2016, pp. 1–13.
[3] R. Balasubramonian, *Innovations in the Memory System*, vol. 14. San Rafael, CA, USA: Morgan & Claypool, 2019.
[4] B. Chen and C. Yang, "Caching policy optimization for D2D communications by learning user preference," in *Proc. IEEE 85th Veh. Technol. Conf.*, 2017, pp. 1–6.
[5] F. Devaux, "The true processing in memory accelerator," in *Proc. IEEE Hot Chips 31 Symp.*, 2019, pp. 1–24.
[6] A. Eisenman *et al.* "Bandana: Using non-volatile memory for storing deep learning models," 2018, *arXiv: 1811.05922*.
[7] U. Gupta *et al.*, "The architectural implications of Facebook's DNN-based personalized recommendation," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, 2020, pp. 488–501.
[8] N. P. Jouppi, A. B. Kahng, N. Muralimanohar, and V. Srinivas, "CACTI-IO: CACTI with OFF-chip power-area-timing models," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 7, pp. 1254–1267, Jul. 2015.
[9] L. Ke *et al.*, "RecNMP: Accelerating personalized recommendation with near-memory processing," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit.*, 2020, pp. 790–803.
[10] Y. Kwon, Y. Lee, and M. Rhu, "TensorDIMM: A practical near-memory processing architecture for embeddings and tensor operations in deep learning," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2019, pp. 740–753.
[11] Micron, "Calculating memory power for DDR4 SDRAM," 2017. [Online]. Available: https://www.micron.com/-/media/client/global/documents/products/technical-note/dram/tn4007_ddr4_power_calculation.pdf
[12] H. Shin, D. Kim, E. Park, S. Park, Y. Park, and S. Yoo, "McDRAM: Low latency and energy-efficient matrix computations in DRAM," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2613–2622, Nov. 2018.
[13] Y. H. Son, O. Seongil, Y. Ro, J. W. Lee, and J. Ahn, "Reducing memory access latency with asymmetric DRAM bank organizations," in *Proc. 40th Annu. Int. Symp. Comput. Archit.*, 2013, pp. 380–391.
[14] C.-J. Wu *et al.*, "Developing a recommendation benchmark for MLPerf training and inference," 2020, *arXiv: 2003.07336*[cs.LG].