An Efficient Near-Bank Processing Architecture for Personalized Recommendation System

Yuqing Yang, Weidong Yang, Qin Wang, Naifeng Jing, Jianfei Jiang, Zhigang Mao, Weiguang Sheng

SHANGHAI JIAO TONG UNIVERSITY

Bio

Yuqing Yang

- A 3rd year M.S. student in integrated circuit engineering at Shanghai Jiao Tong Unversity.

- Education

  - Shanghai Jiao Tong Unviersity, China, B.S. (2020)

- Research Interest

  - Near-memory-processing architecture

  - Heterogeneous computing

Outline

Background & Motivations

Key Contributions

Architecture Design

Evaluation Results

Conclusion

Application Scenario of Recommendation Models

Social Network

Online Shopping

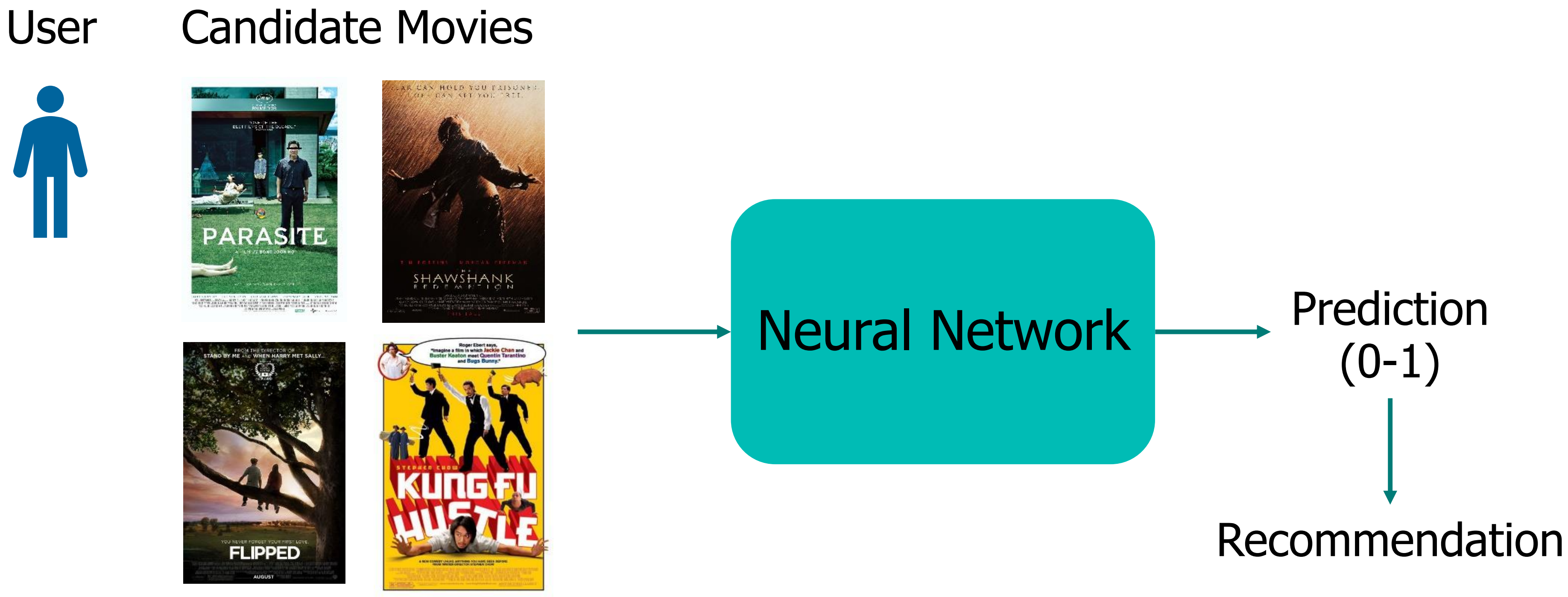
Search Engine

Video/Music

Personalized recommendation models are **widely** used in internet services!

Deep Learning for Personalized Recommendation

- Deep learning can maximize the recommendation **accuracy** and deliver better **user experience**.



User

Candidate Movies

Neural Network

Prediction (0-1)

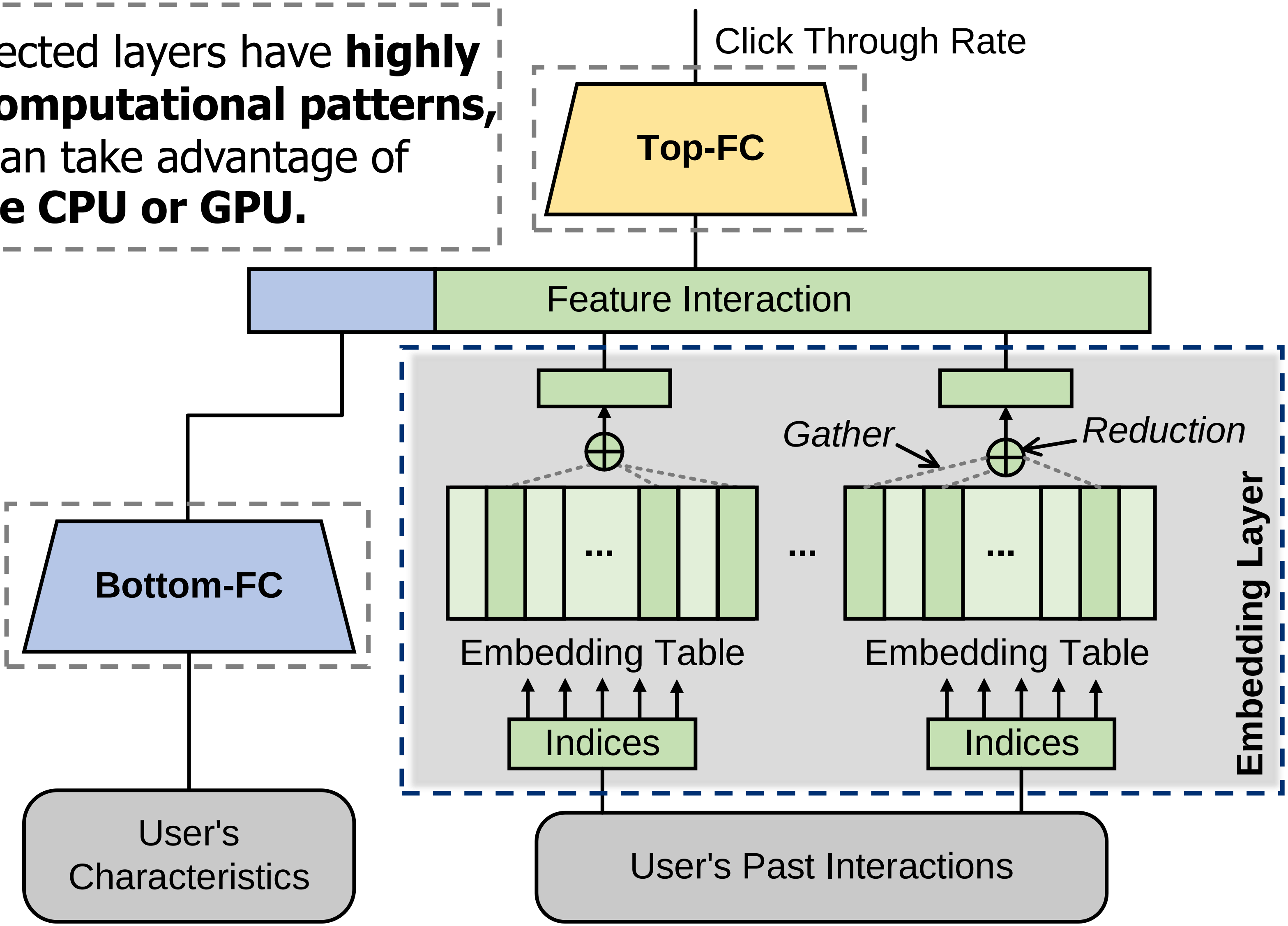
Recommendation

Deep learning-based recommendation models consume the **majority** (**~79%**[1]) resources in AI data centers.

[1] Gupta U, Wu C J, Wang X, et al. The architectural implications of facebook's dnn-based personalized recommendation[C]//2020 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2020: 488-501.

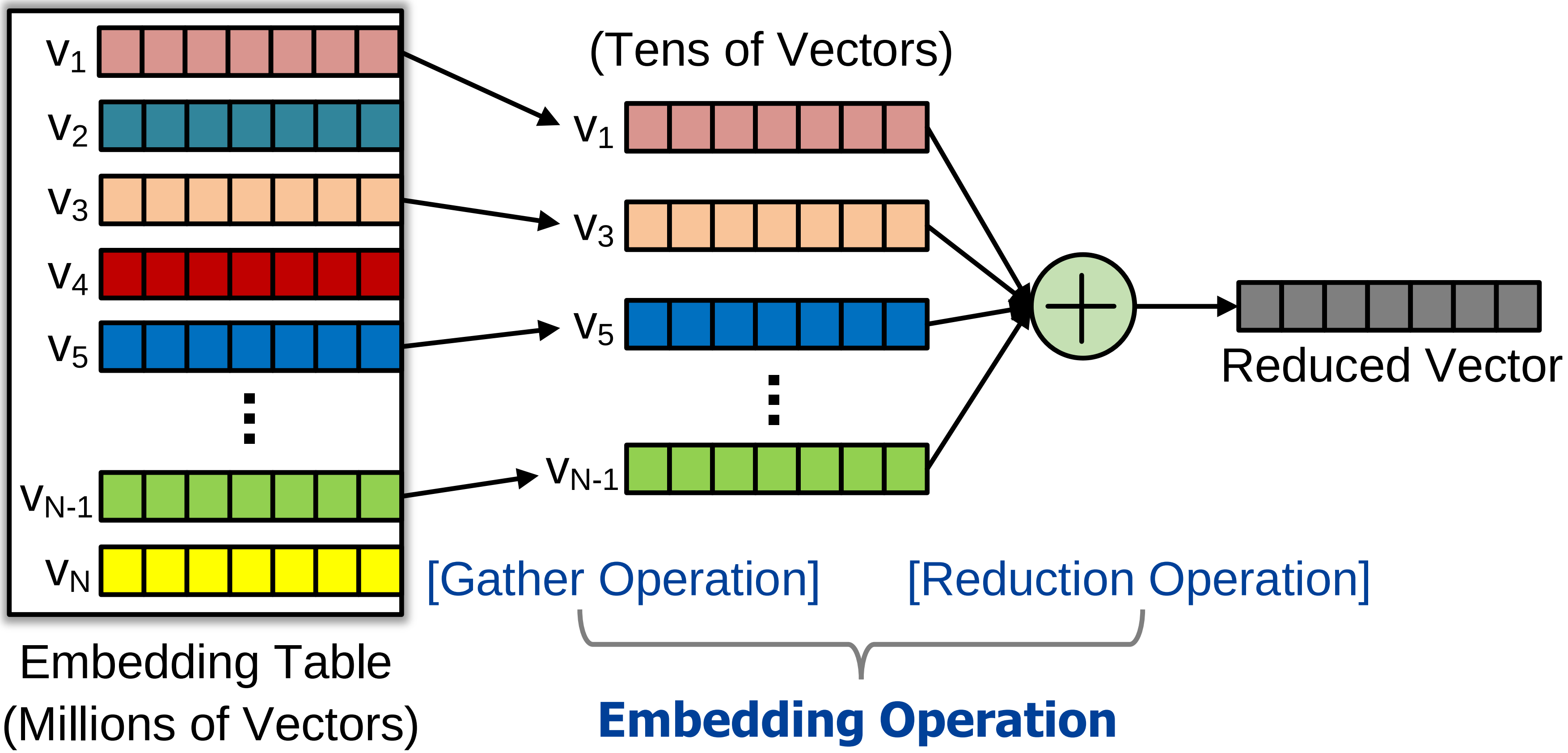
Deep Learning Recommendation Model (DLRM)

Fully-connected layers have **highly regular computational patterns,** and they can take advantage of **multi-core CPU or GPU.**



Embedding layers have **different operation characteristics** with FC!

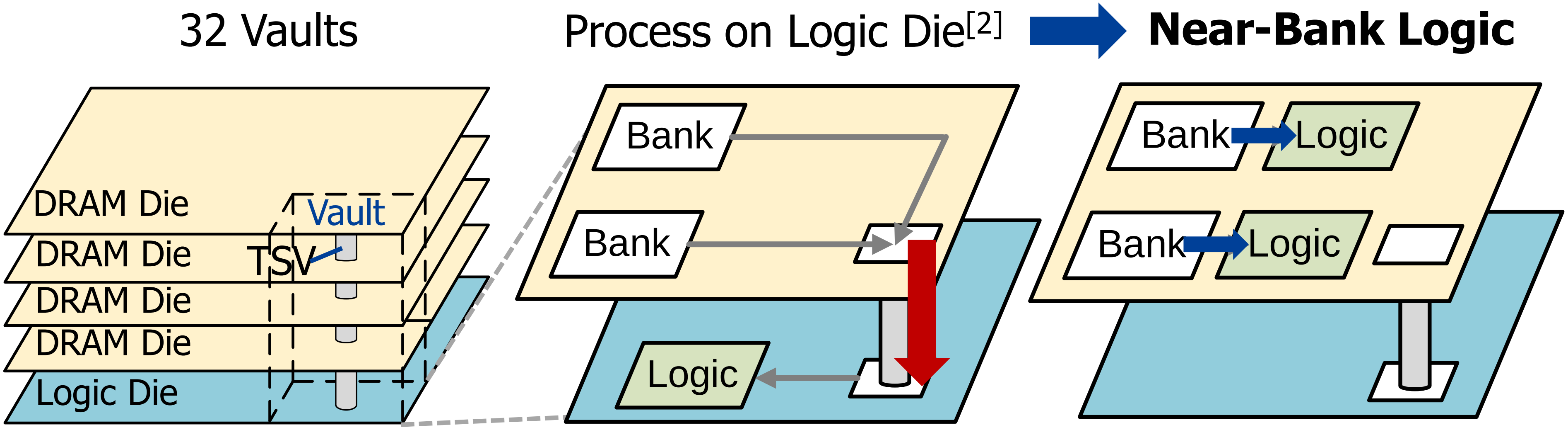
Memory Challenges of Embedding Layers



Gather operations have **sparse and irregular** memory access patterns, reduction operations have **low compute intensity**.

Embedding operations are **memory-bound** and demand **more memory bandwidth**.

3D-Stacked Near-Memory-Processing Architecture



32 Vaults

Process on Logic Die[2] → **Near-Bank Logic**

3D-Stacked Memory

TSV I/O Bound

Higher Bank-Level Bandwidth

Integrate compute-logic closer to where data is stored:

- Greatly increase memory bandwidth (**~8X**)
- Reduce data movement

[2] Kal H, Lee S, Ko G, et al. SPACE: locality-aware processing in heterogeneous memory for personalized recommendations[C]//2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2021: 679-691.

Summary

- Deeping learning-based recommendation models consume the majority resources in AI data centers.

- Memory-bound embedding layers become the bottleneck of DLRM on traditional computing platforms due to irregular memory access patterns.

- Near-bank architecture can provide enormous bank-level bandwidth which is much higher than TSV can provide in 3D-stacked DRAMs.

Target: Design a near-bank processing architecture to accelerate the embedding layer of DLRM.

Outline

Background & Motivations

Key Contributions

Architecture Design

Evaluation Results

Conclusion

Key Contributions

- An efficient near-bank architecture for recommendation models which exploits the enormous bank-level bandwidth of 3D-stacked memory
- A programming model for managing the memory allocation of embedding tables and embedding kernel offloading
- An optimized mapping scheme based on vector partition to improve the utilization of the bank-level bandwidth

Achieves up to **2.10x** performance gain and **31%** energy saving compared to SPACE (ISCA'21).

Outline

Background & Motivations

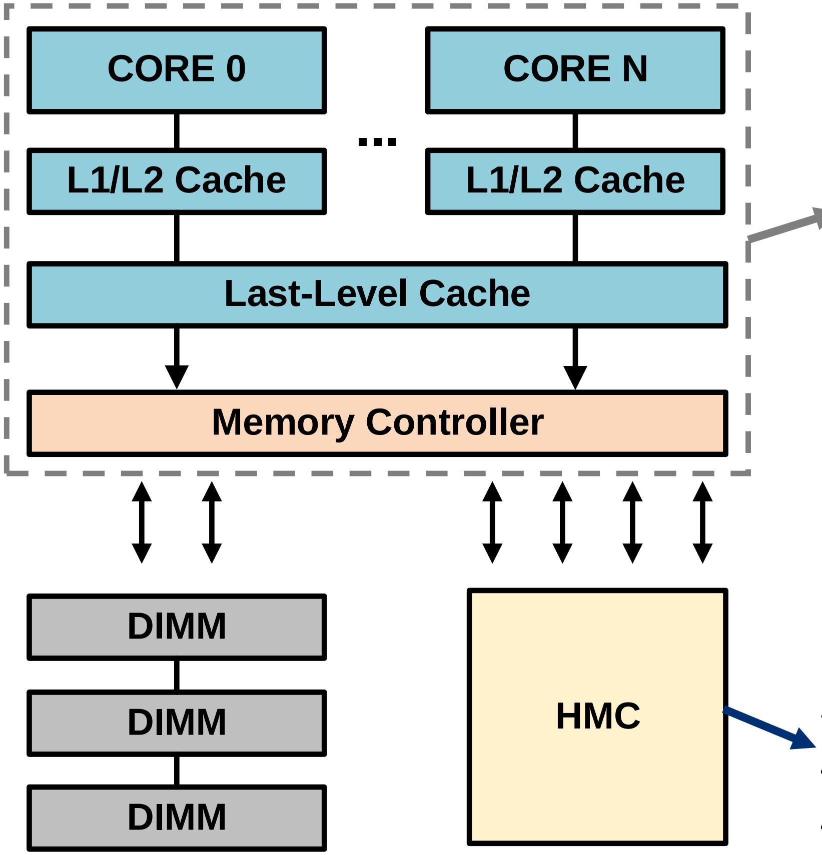
Key Contributions

Architecture Design

Evaluation Results

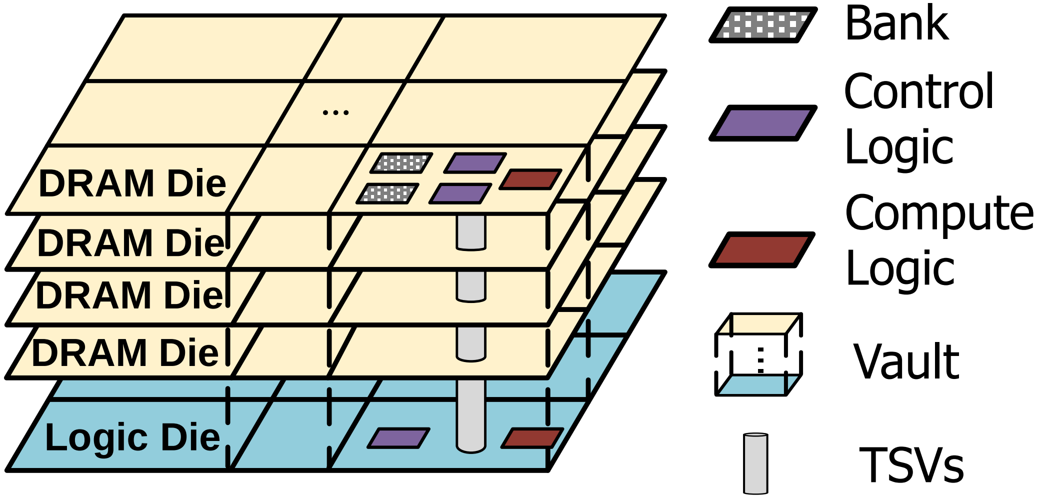
Conclusion

System Overview



Host Processor:

- Allocate embedding items between HMC and DIMMs
- Offload embedding kernels
  1. encode embedding instructions
  2. send instructions to HMC



Embedding instructions are **dispatched** and **decoded**, then gather and reduction operations are **performed** on logic units in HMC.

Embedding Instructions (Emb-Inst)

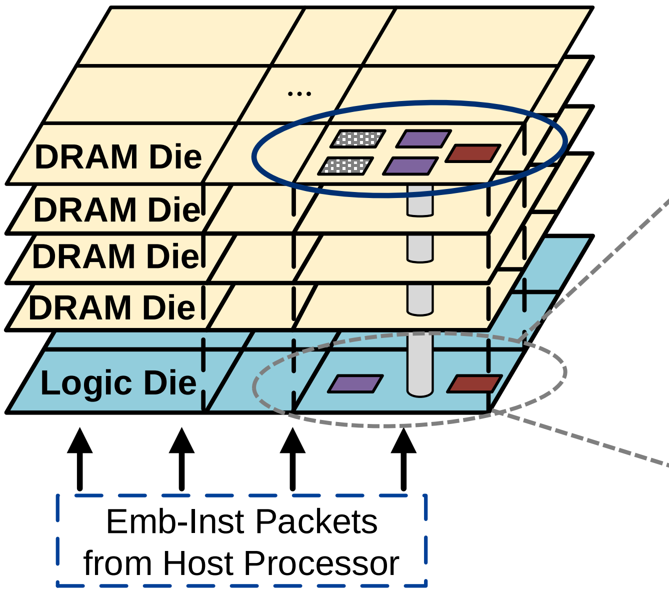
- Instruction Design

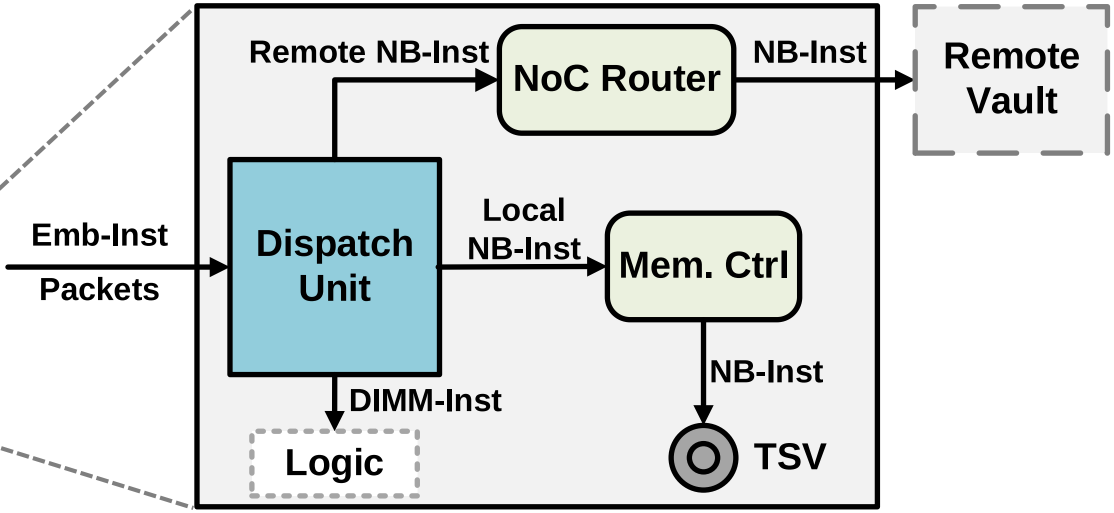
	Opcode	Source	Vector Size	Destination	Reduction Indicator
NB-Inst	Add.NB	HMC Addr	vsize	Vault ID	Psum Tag
DIMM-Inst	Add.D	DIMM Addr	vsize	Vault ID	Psum Tag

- Psum Tag: indicate which reduction operation the vector belongs to
- Vault ID: indicate which vault the psum is sent to

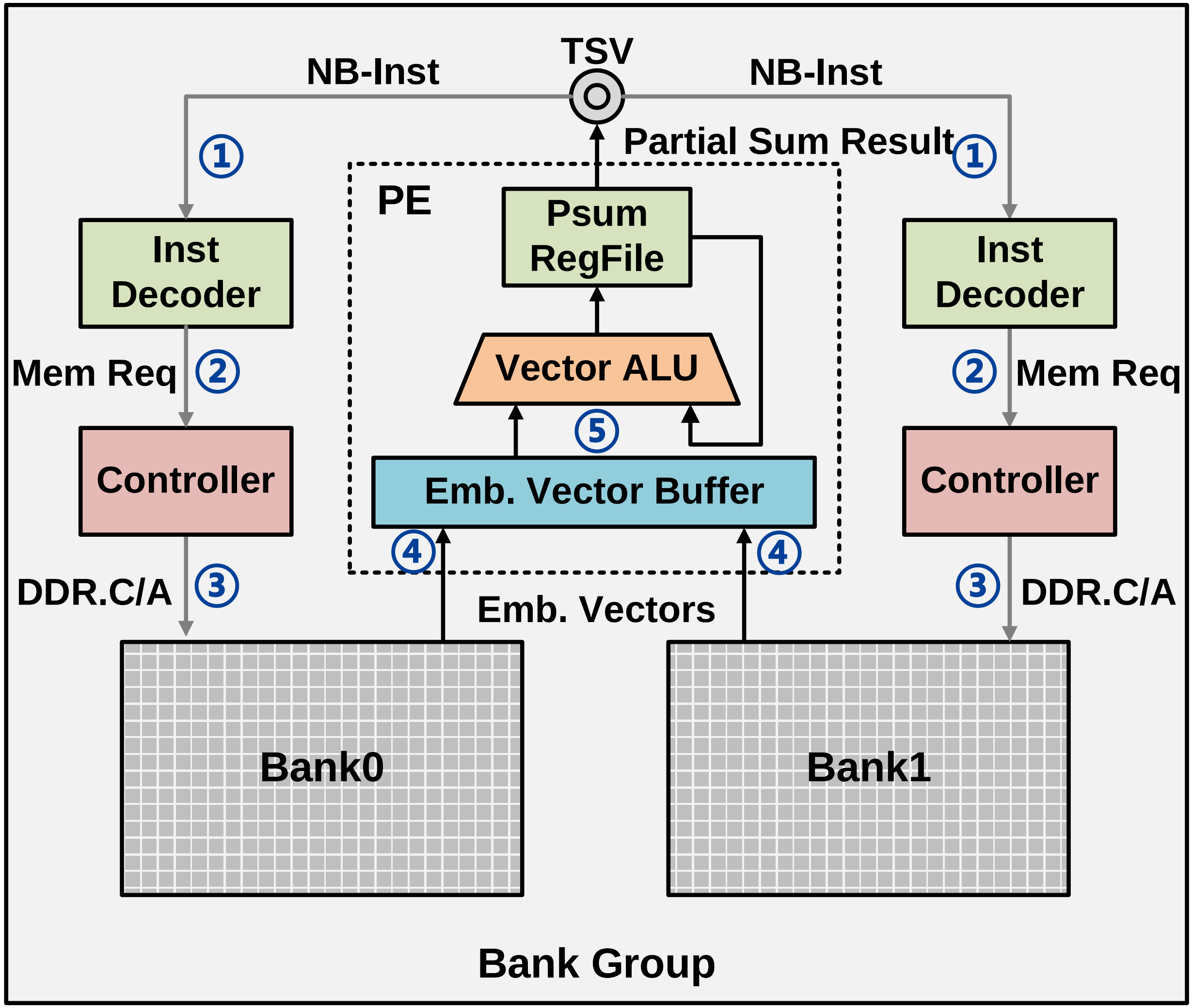
According to the location of embedding items, **Opcode** is set as add.NB (item is stored in HMC) or add.D (item is stored in DIMM) .

- Instruction Dispatch





Near-bank Processing Logic Design



- **Gather**

① Receive NB-Inst

② Instruction Decode

③ Send DDR.C/A

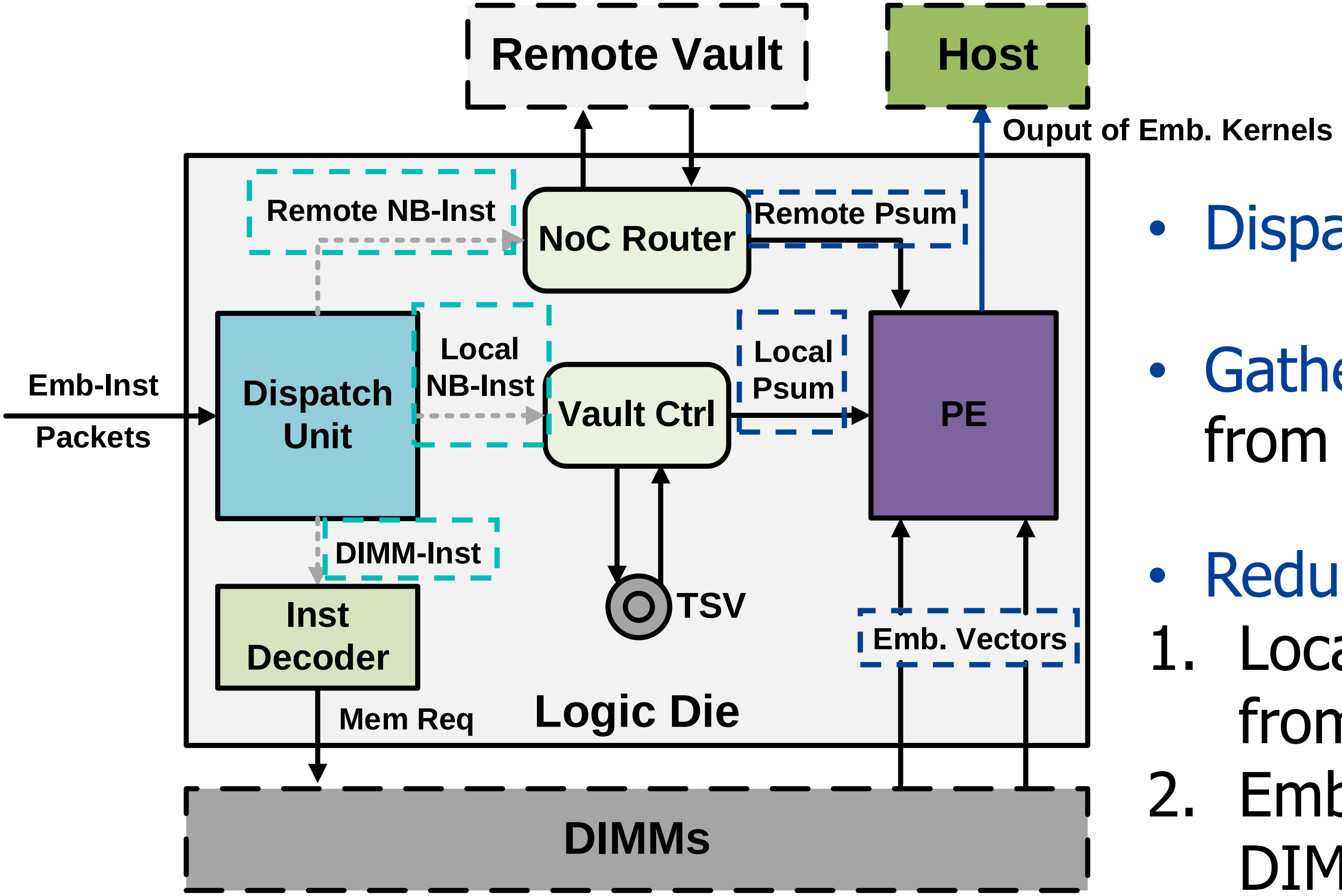
④ Load Emb. Vectors

- **Reduction**

⑤ Element-Wise Summation for Emb. Vectors with Same Psum Tags

Finally, partial sum result with no more element-wise summation to be done are sent to the logic die for further reduction (If need).

Logic Die Design



- Dispatch Emb-Insts
- Gather Emb. Vectors from DIMMs
- Reduction

1. Local/Remote Psum from DRAM Dies
2. Emb. Vectors from DIMMs

After reduction operations, final outputs of embedding kernels would be sent back to the host processor.

Programming Model: Purposes

- **Memory Allocation of Embedding Tables**

1. Rearrange embedding tables according to item accesses
2. Set border line of embedding tables according to HMC bandwidth (bank-level) ratio of the total memory bandwidth
3. Distribute embedding items between HMC (index < border line) and DIMMs (index >= border line)

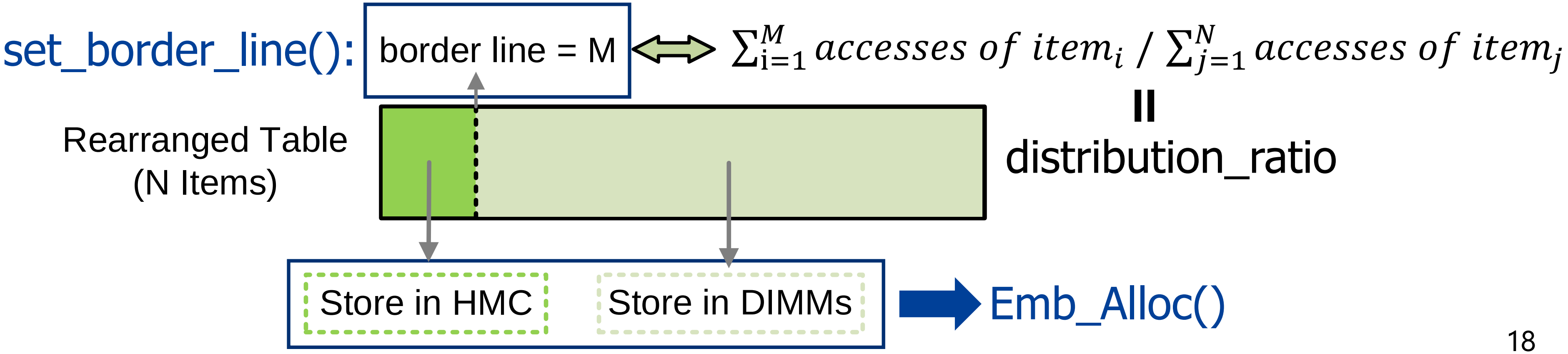
- **Embedding Kernels Offloading**

1. Initialization
2. Embedding instructions generation

Programming Model: Embedding Table Allocation

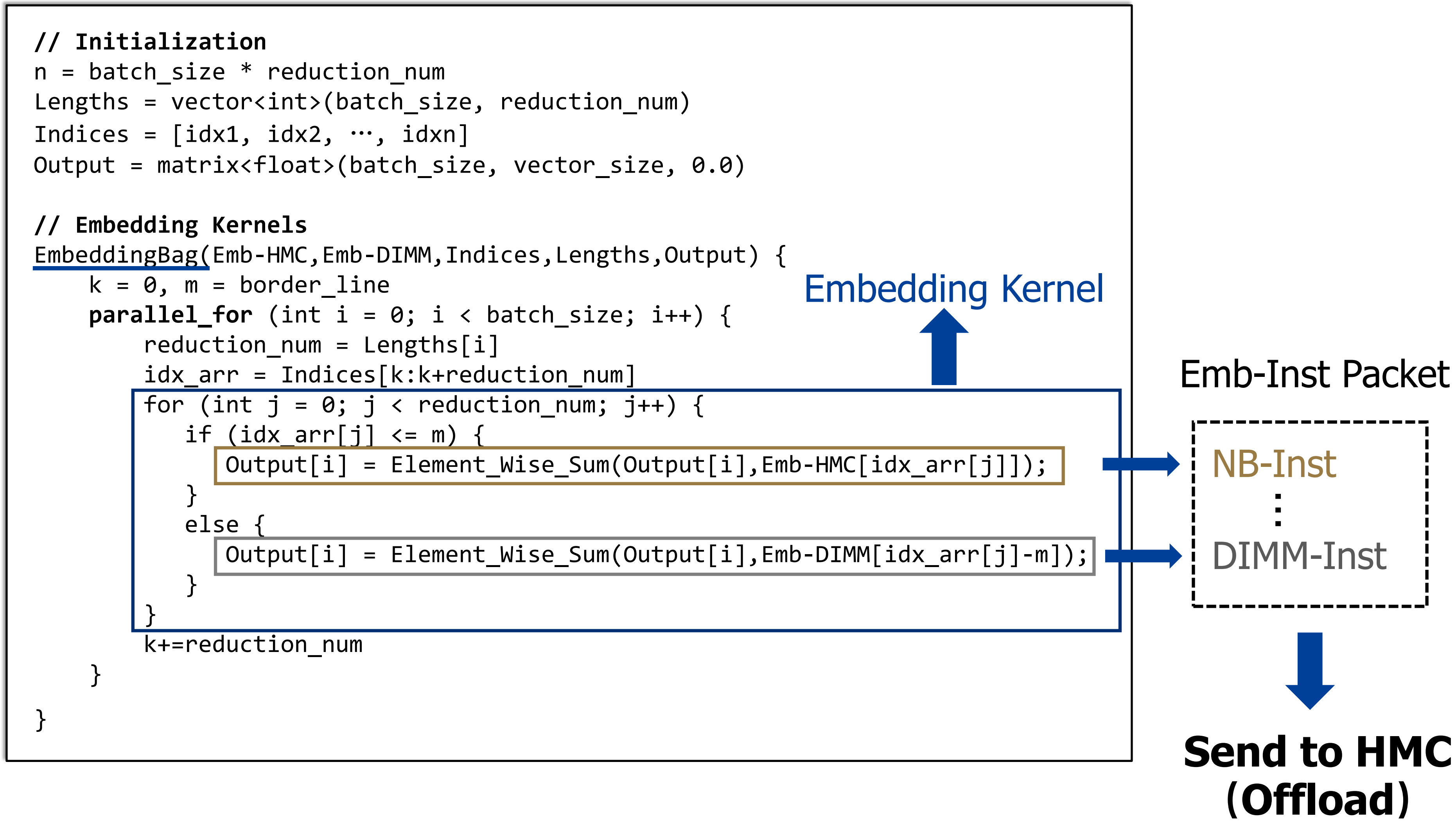
```
// Embedding Table Allocation
access_hash_table = count_access(past_inference_set);
Emb = rearrange_emb(Emb, access_hash_table);
distribution_ratio = HMC_bandwidth / total_bandwidth;
border_line = set_border_line(Emb, access_hash_table, distribution_ratio);
Emb-HMC, Emb-DIMM = Emb_Alloc(Emb, border_line);
```

- count_access(): Count number of accesses per embedding item
- rearrange_emb(): Rearrange embedding tables by ranking items according to number of accesses



set_border_line(): border line = M ⟺ $\sum_{i=1}^{M} accesses\ of\ item_i\ /\ \sum_{j=1}^{N} accesses\ of\ item_j$ = distribution_ratio

Programming Model: Embedding Kernel Offloading



```
// Initialization
n = batch_size * reduction_num
Lengths = vector<int>(batch_size, reduction_num)
Indices = [idx1, idx2, ⋯, idxn]
Output = matrix<float>(batch_size, vector_size, 0.0)

// Embedding Kernels
EmbeddingBag(Emb-HMC,Emb-DIMM,Indices,Lengths,Output) {
    k = 0, m = border_line
    parallel_for (int i = 0; i < batch_size; i++) {
        reduction_num = Lengths[i]
        idx_arr = Indices[k:k+reduction_num]
        for (int j = 0; j < reduction_num; j++) {
           if (idx_arr[j] <= m) {
              Output[i] = Element_Wise_Sum(Output[i],Emb-HMC[idx_arr[j]]);
           }
           else {
              Output[i] = Element_Wise_Sum(Output[i],Emb-DIMM[idx_arr[j]-m]);
           }
        }
        k+=reduction_num
    }
}
```

Embedding Kernel

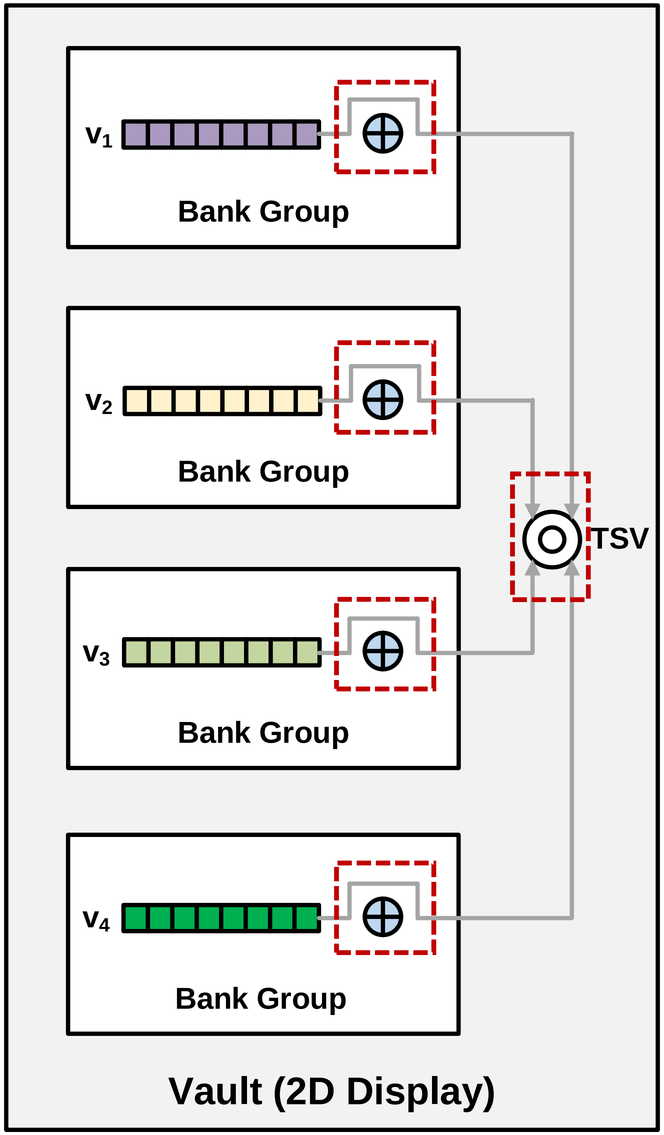
Emb-Inst Packet

NB-Inst
⋮
DIMM-Inst

Send to HMC (Offload)

Kernel Offloading Stage: Embedding kernels are compiled into embedding instructions which are packed into packets to be sent to HMC.

Mapping Scheme: Problem Analysis



Typical Case
(v_1, v_2, v_3, v_4 need to be reduced)

Problem

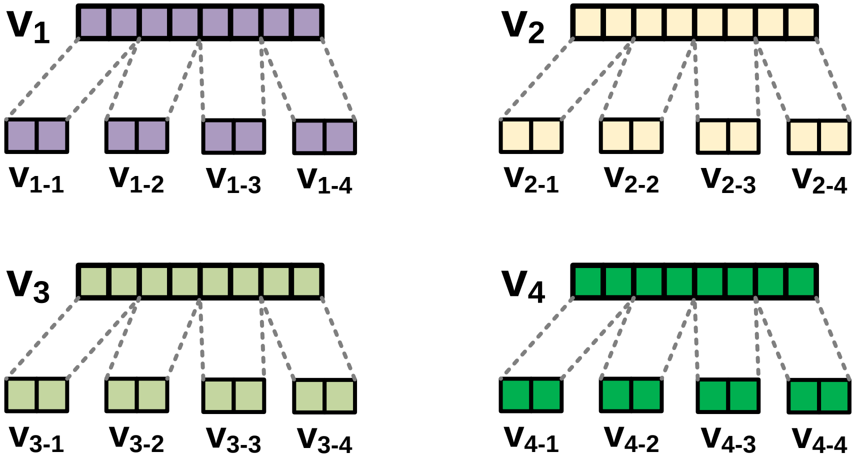
- Embedding vectors can not be reduced on near-bank logic
- All the vectors are transferred to the logic die to perform reduction operations through TSVs

1. TSV bandwidth becomes the limit.

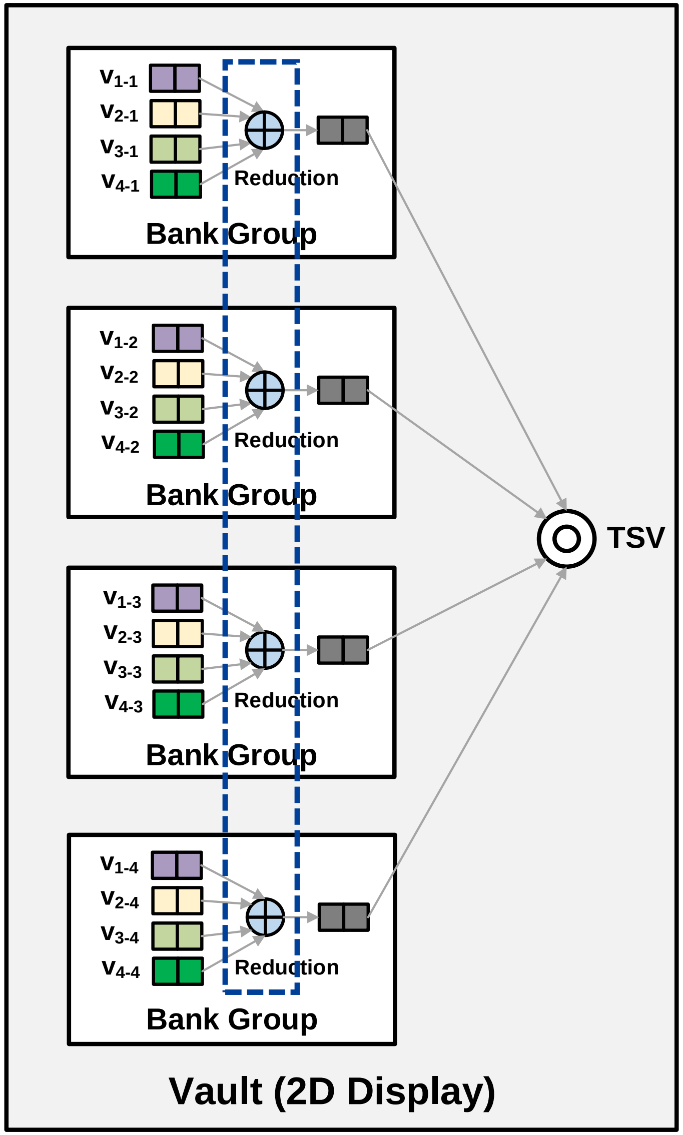
2. Bank-level bandwidth can not be exploited.

Mapping Scheme: Partition and Mapping

[Partition]



[Map]



- Embedding operations are executed on near-bank logic **in parallel**.
- **Less data (1/4)** is transferred through TSVs, bank-level bandwidth is fully exploited.

Outline

Background & Motivations

Key Contributions

Architecture Design

Evaluation Results

Conclusion

Experiment Setup

- Recommendation System

  - Model: DLRM

  - Datasets: Anime, MovieLens, LastFM

- System Simulation

  - Develop a cycle-accurate model for near-bank processing architecture based on Ramulator[3]

  - Use Cacti-3DD[4] to estimate the energy consumption

- Baseline

  - SPACE (ISCA'21) for recommendation models

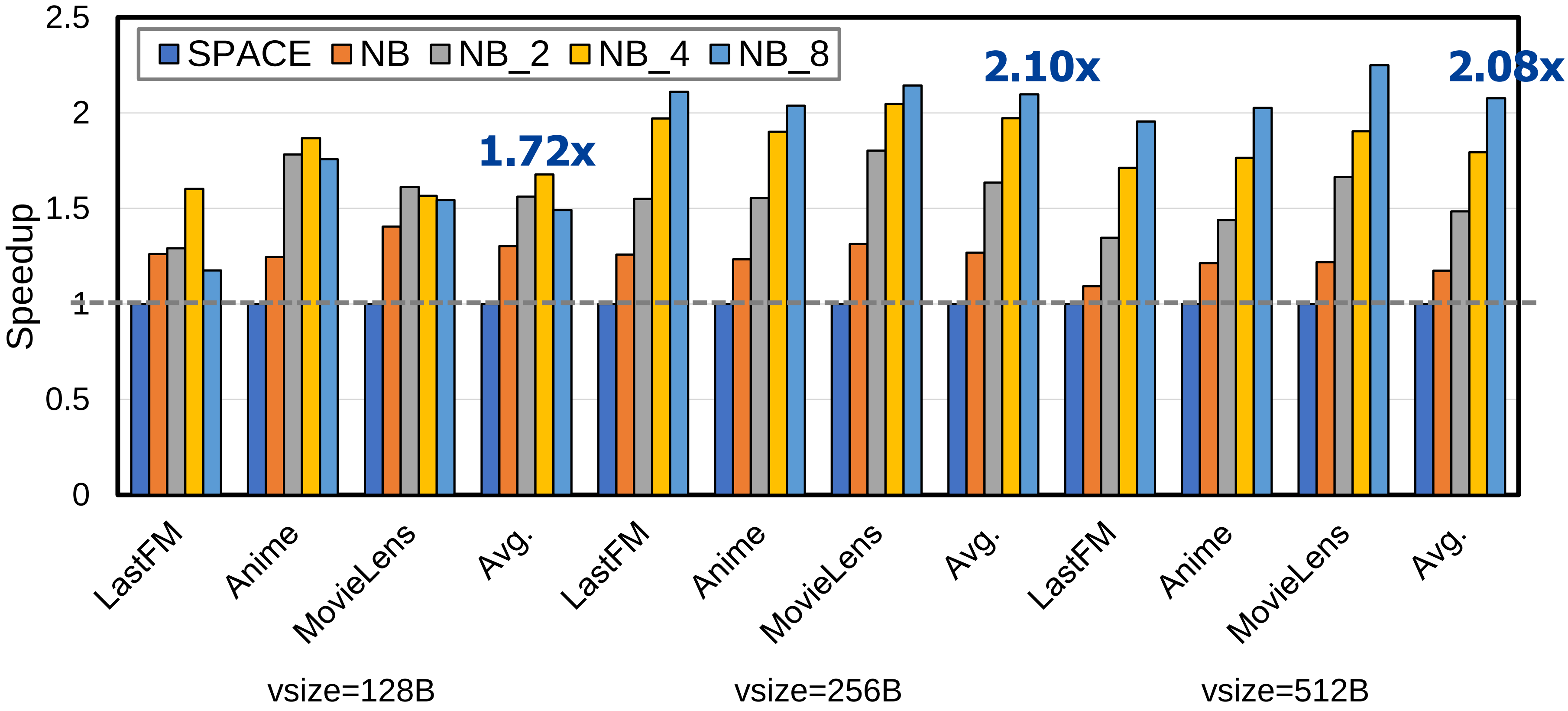
[3] Kim Y, Yang W, Mutlu O. Ramulator: A fast and extensible DRAM simulator[J]. IEEE Computer architecture letters, 2015, 15(1): 45-49.

[4] Chen K, Li S, Muralimanohar N, et al. CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory[C]//2012 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2012: 33-38.

Performance Evaluation

Our Work (NB) V.S. SPACE

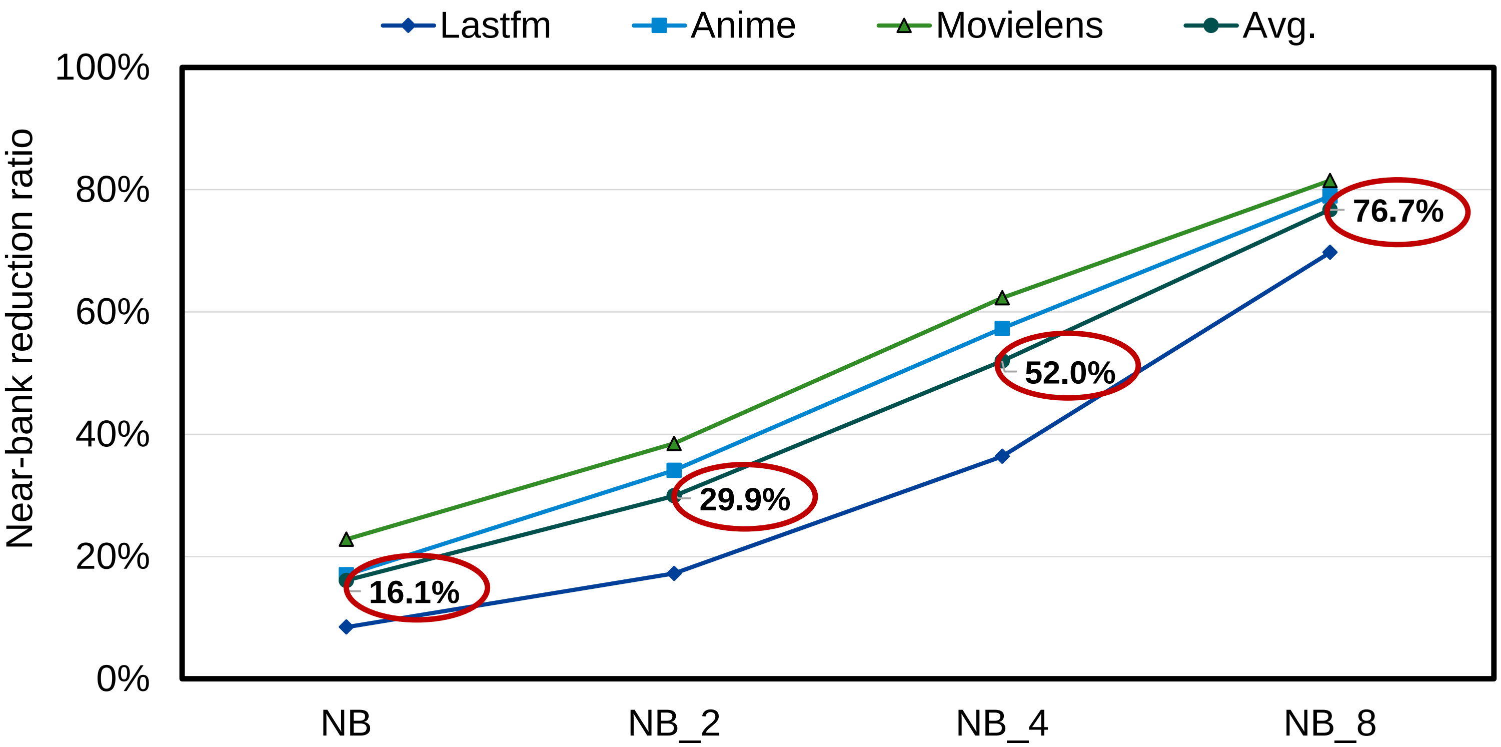
- NB_x: Apply mapping scheme (x is the partition number)



Compared to SPACE, our work can achieve **1.97x** speedup on average and up to **2.10x** speedup (vsize = 256B) .

Effectiveness of Mapping Scheme

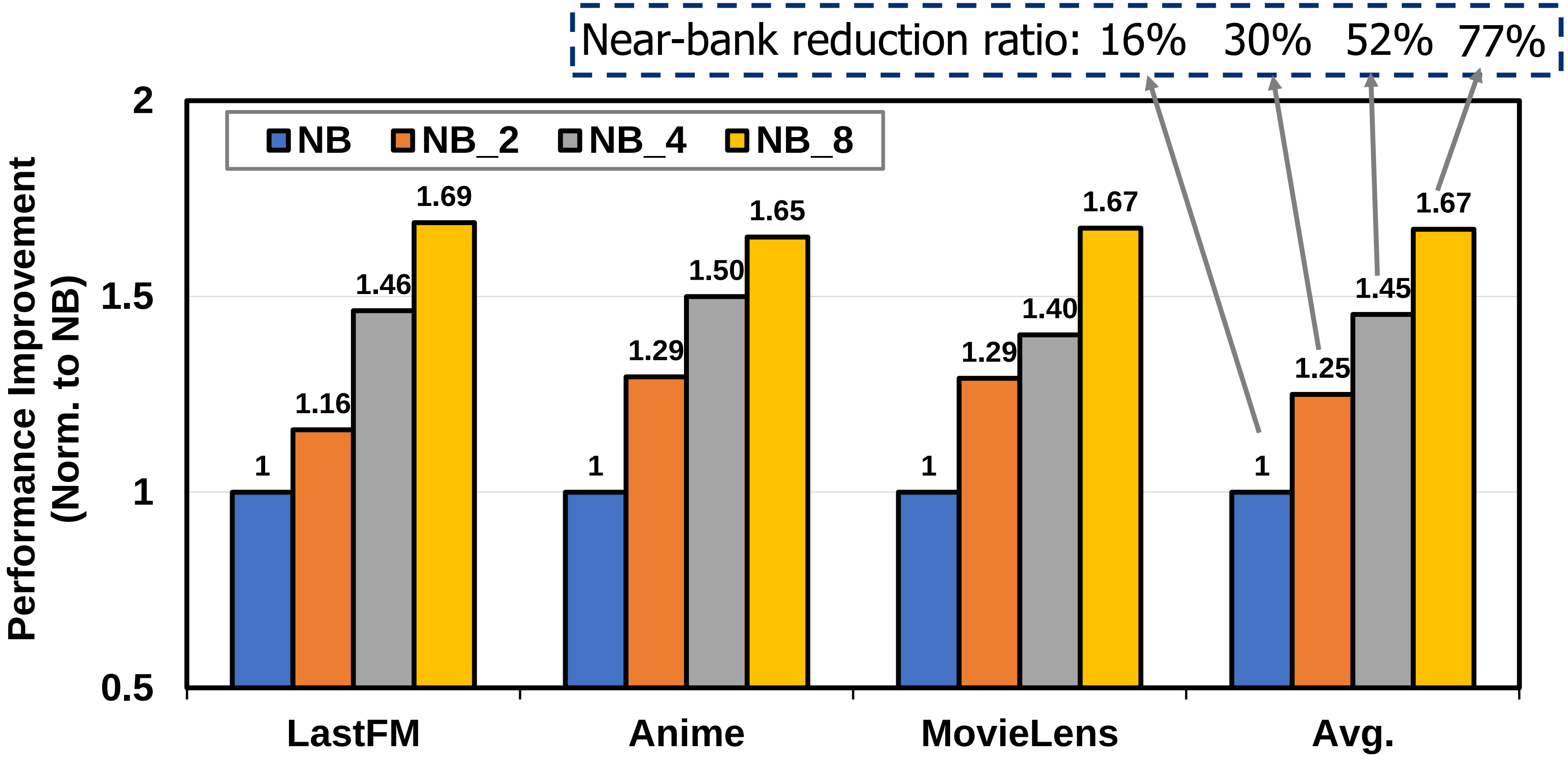
- NB (Near-Bank Arch): Our work
- NB_x: Apply mapping scheme (x is the partition number)



As the partition number increases, the ratio of near-bank reduction rises and bank-level parallelism is exploited.

Effectiveness of Mapping Scheme

- NB (Near-Bank Arch): Our work
- NB_x: Apply mapping scheme (x is the partition number)

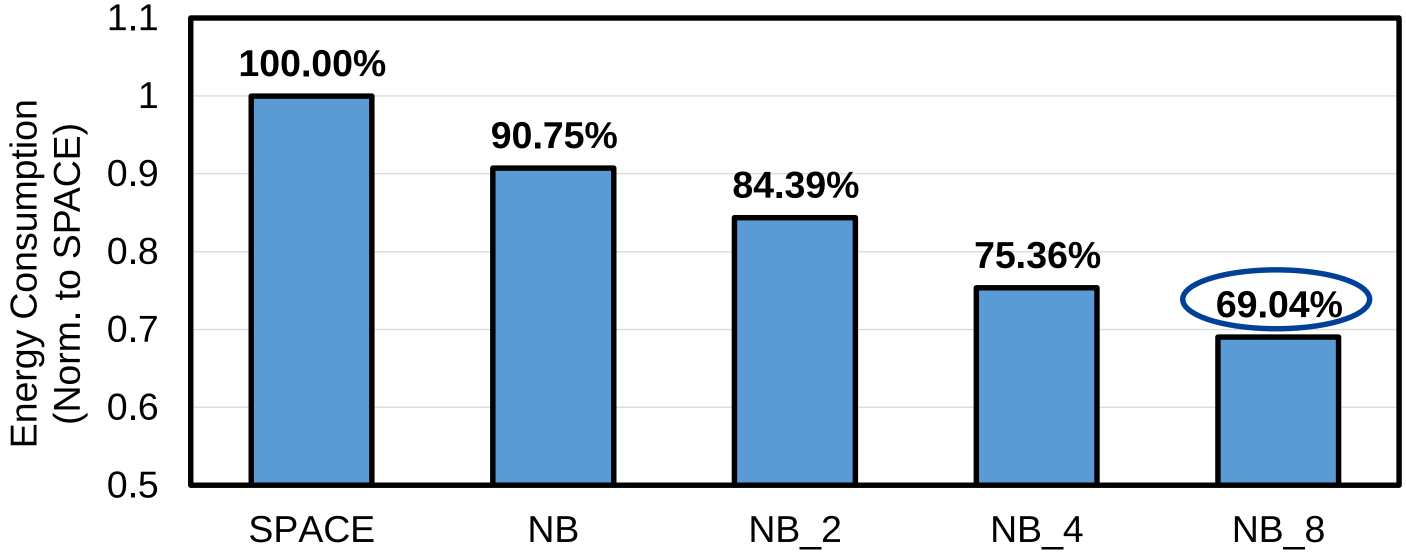


Mapping scheme effectively improves the utilization of bank-level bandwidth and thus performance of near-bank architecture is improved.

Energy Consumption Evaluation

Our Work (NB) V.S. SPACE

- NB_x: Apply mapping scheme (x is the partition number)



Our work can save at most **31%** energy for data movement.

- **Advantages** of near-bank processing for reducing data movement

1. **Less** amount of data is transferred through TSVs
2. **Off-chip data movement** between HMC and DIMMs **decreases** as more items are stored in HMC

Outline

Background & Motivations

Key Contributions

Architecture Design

Evaluation Results

Conclusion

Conclusion

- **We propose an efficient near-bank architecture** for DLRM that provides:

  - Bank-level parallelism in processing embedding layers

  - A specialized programming model

  - An optimized mapping scheme

- **We evaluate our design** for DLRM using three different real-world recommendation datasets:

  - Compared to SPACE, our design achieves **2.10x** speedup.

  - Compared to SPACE, our design saves **31%** energy consumption.

  - Our mapping scheme improves near-bank reduction ratio from **16%** to **77%.**

An Efficient Near-Bank Processing Architecture for Personalized Recommendation System

Thank you !

Q&A

Presenter: Yuqing Yang

Email: yangyuqing@sjtu.edu.cn

SHANGHAI JIAO TONG UNIVERSITY