TensorDIMM: A Practical Near-Memory Processing Architecture for Embeddings and Tensor Operations in Deep Learning

KAIST

Youngeun Kwon, Yunjae Lee, and Minsoo Rhu

Research Scope

DL architecture research so far

Primarily focused on "dense" DNN layers (e.g., CNNs, RNNs, …)

* Chen et al., "DaDianNao: A Machine-Learning Supercomputer", ISCA-2014

* Chen et al., "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks", ISCA-2016

* Han et al., "EIE: Efficient Inference Engine on Compressed Deep Neural Network", ISCA-2016

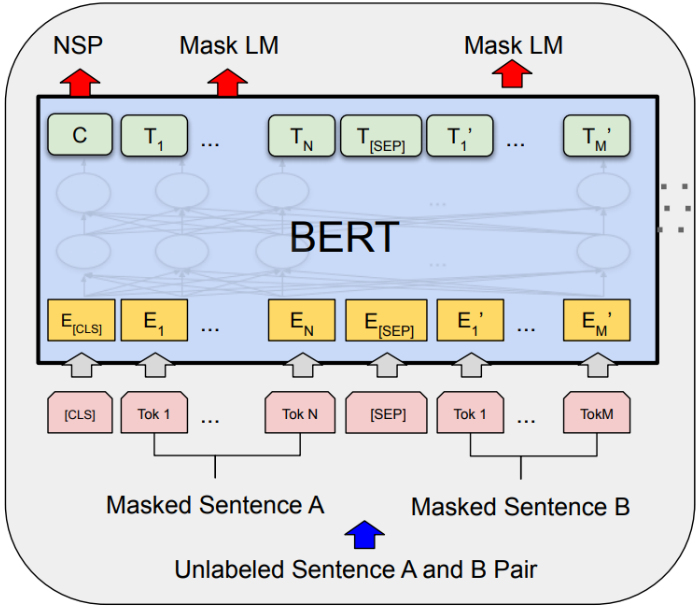
* Parashar et al., "SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks", ISCA-2017

* Yazdanbakhsh et al., "GANAX: A Unified MIMD-SIMD Acceleration for Generative Adversarial Networks", ISCA-2018

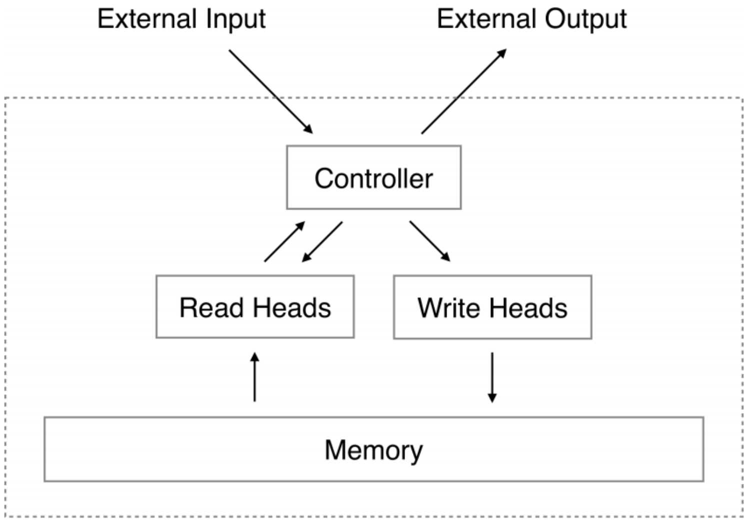
KAIST

Emerging DL applications?

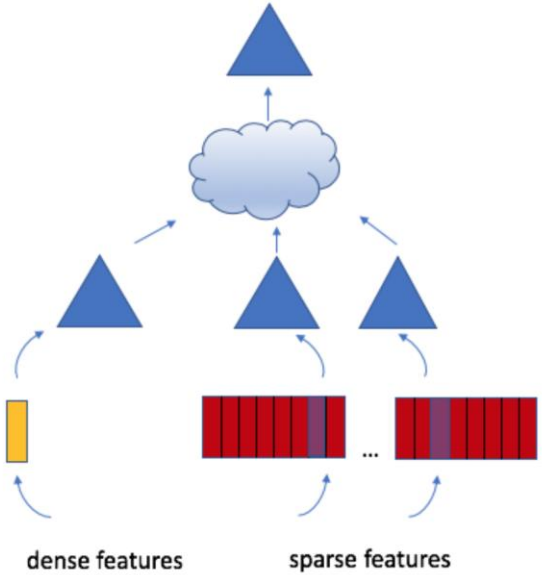
"Non" conventional DNN layers are causing a bottleneck



BERT



Neural Turing Machine

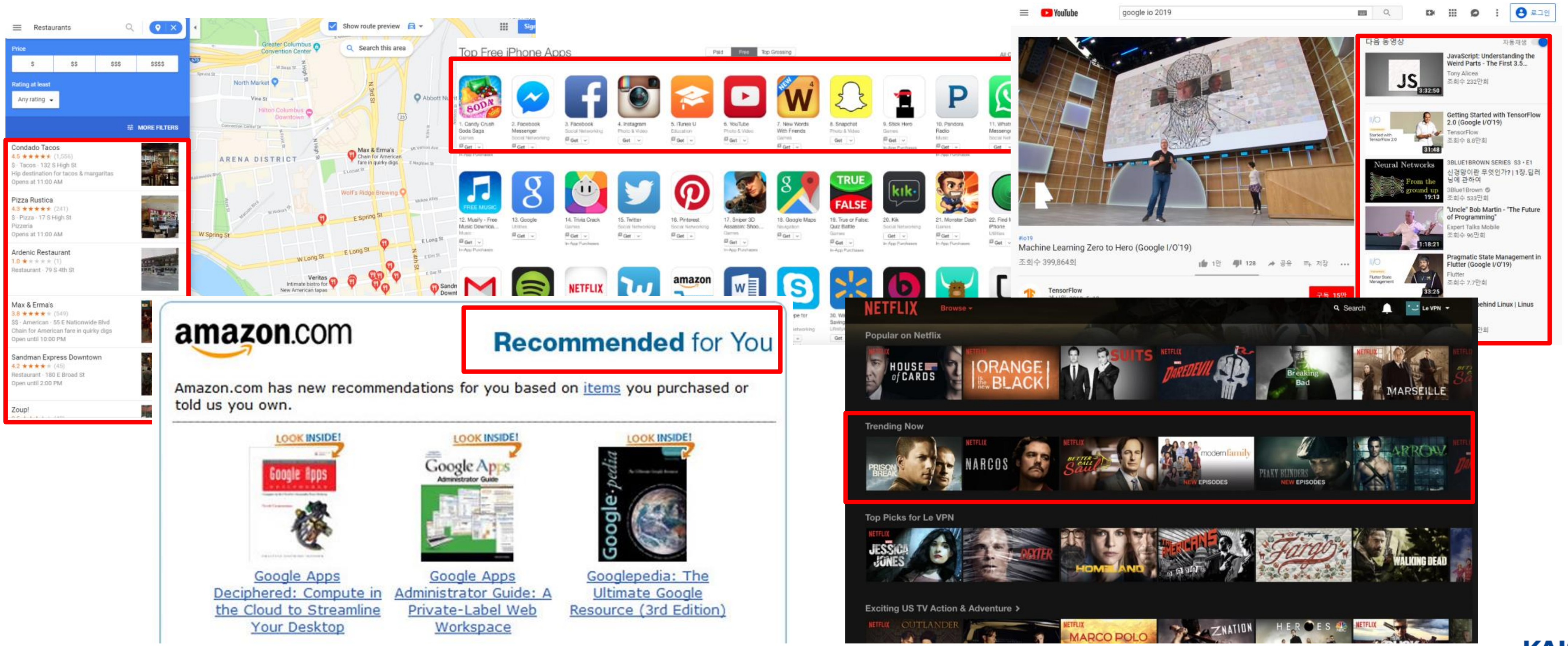


Recommendation

* Devlin et al., "Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding", arxiv.org, 2018
* Graves et al., "Neural Turing Machines", arxiv.org, 2014
* Naumov et al., "Deep Learning Recommendation Model for Personalization and Recommendation Systems", arxiv.org, 2019

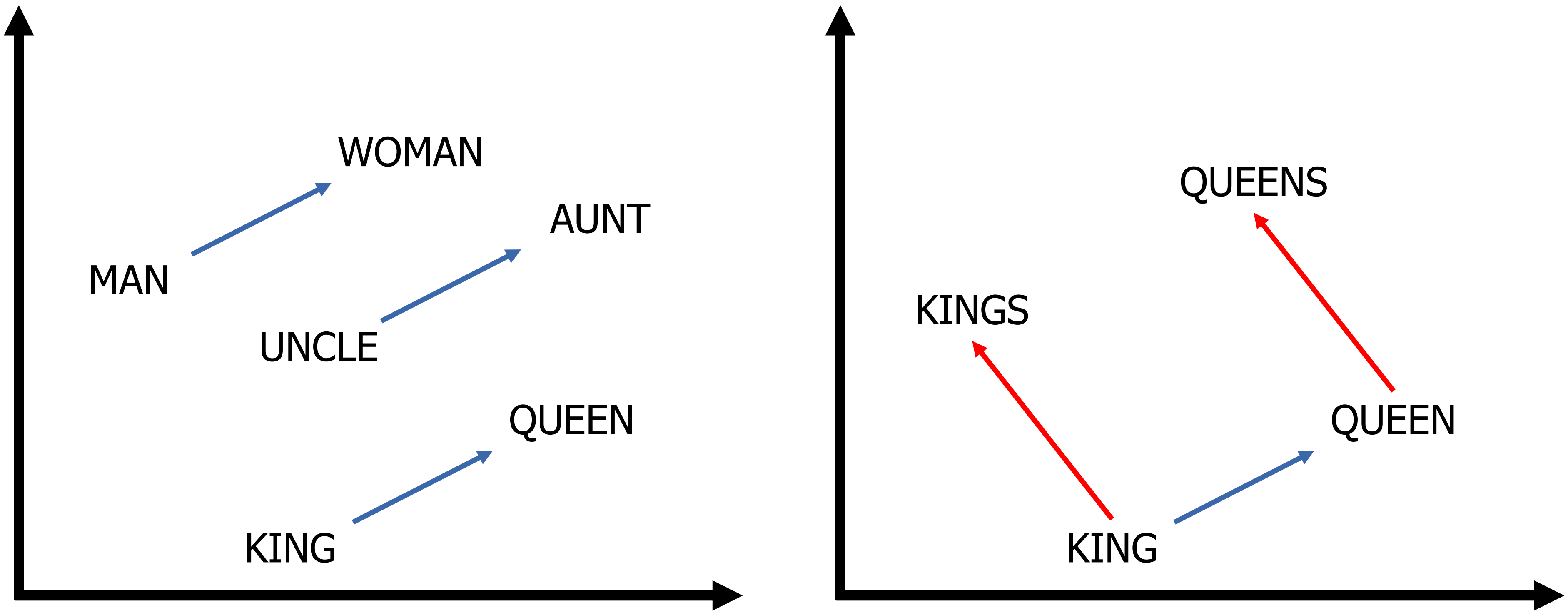
Personalized recommendation models

"Sparse" embedding layers (rather than dense DNNs) are the bottleneck



What is an embedding?

Projection of sparse features into dense vector dimension (e.g., word2vec)



* Mikolov et al., "Distributed Representations of Words and Phrases and their Compositionality", NIPS-2013

What is an embedding?

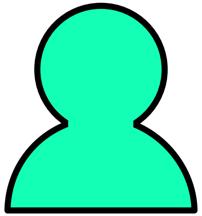
Stored as a large look-up table containing millions-to-billions of entries

User ID	Embedding (vector)
0: Sam	[0.49, 0.52, 0.23, 0.69, 0.32, …]
1: Harry	[0.24, 0.27, 0.13, 0.09, 0.79, …]
2: Matt	[0.31, 0.71, 0.46, 0.91, 0.07, …]
3: John	[0.83, 0.43, 0.81, 0.57, 0.09, …]
4: Elicia	[0.31, 0.83, 0.23, 0.69, 0.86, …]
…	…
N: Danny	[0.77, 0.18, 0.71, 0.59, 0.46, …]

N: can be millions

Recommendation model 101

Goal: predict a preference of user-item pair



Sam

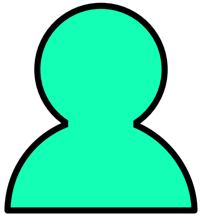
KAIST

Recommendation model 101

Goal: predict a preference of user-item pair

Movie 0
Movie 1
Movie 2
Movie 3
Movie 4
Movie 5
Movie 6
Movie 7
…
Movie N

[Embedding table]

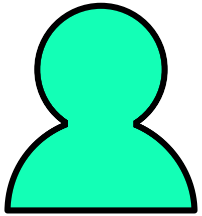


Sam

Recommendation model 101

Goal: predict a preference of user-item pair

Movie 0
Movie 1
Movie 2
Movie 3
Movie 4
Movie 5
Movie 6
Movie 7
…
Movie N

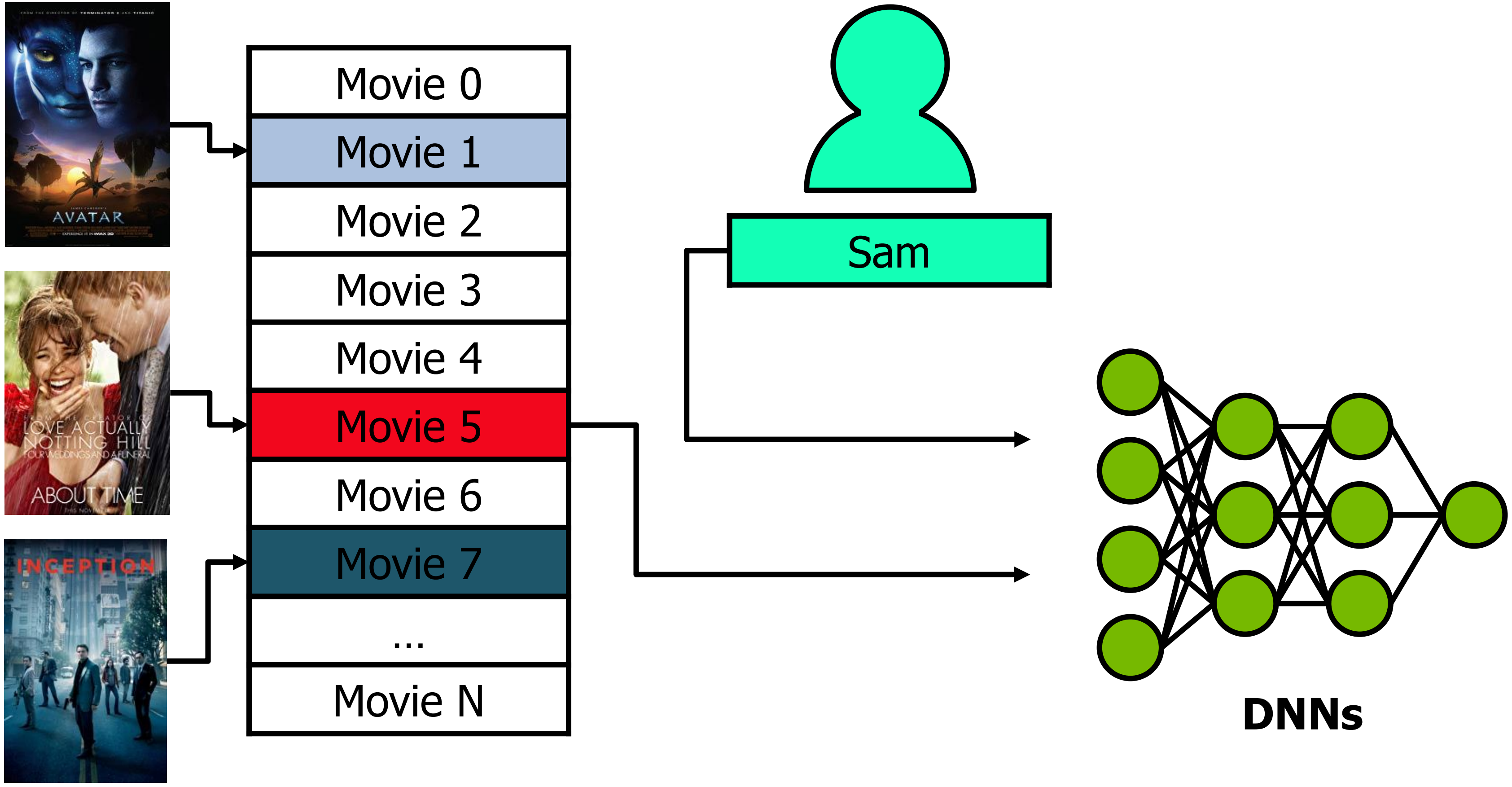


Sam

[Embedding table]

Recommendation model 101

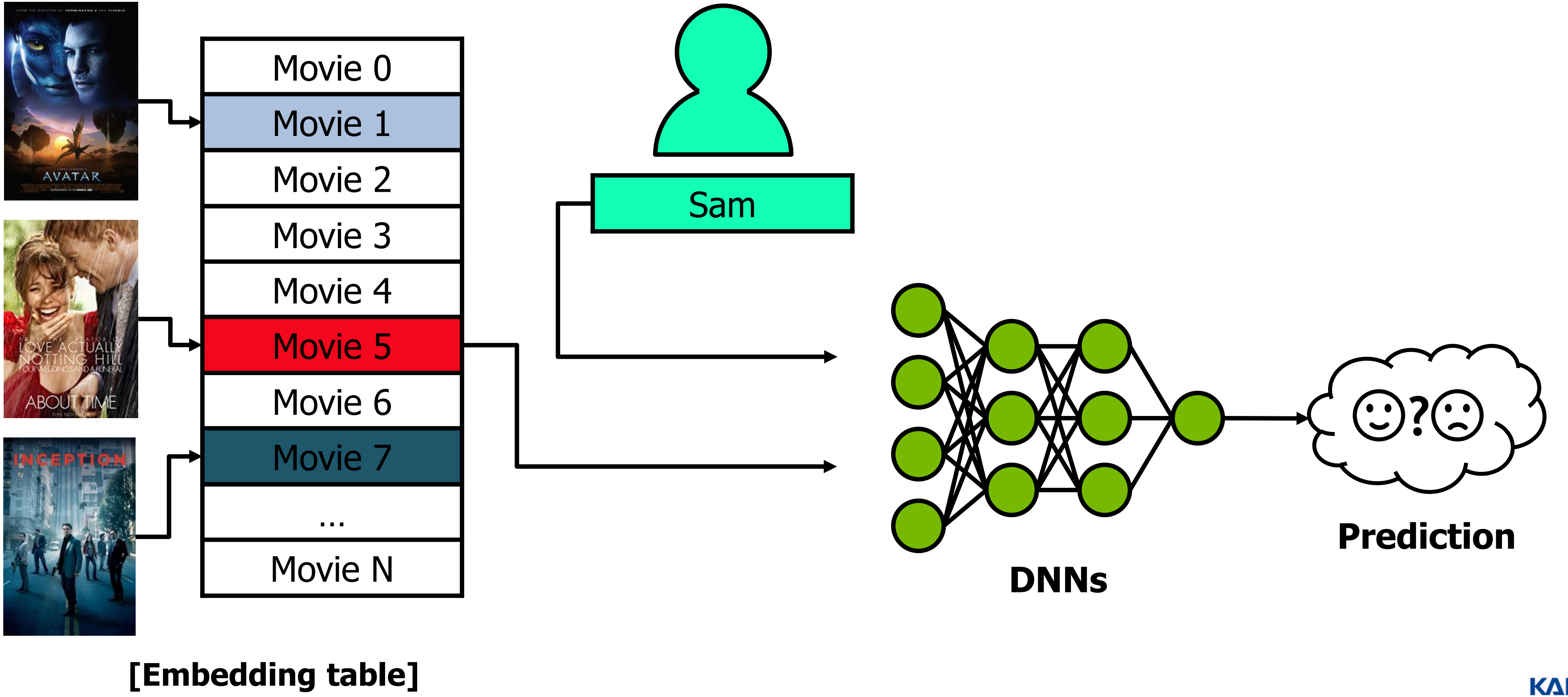
Goal: predict a preference of user-item pair



[Embedding table]

Recommendation model 101

Goal: predict a preference of user-item pair



KAIST

Key Primitives in Embedding Layers

#1: Embedding lookup (gather)

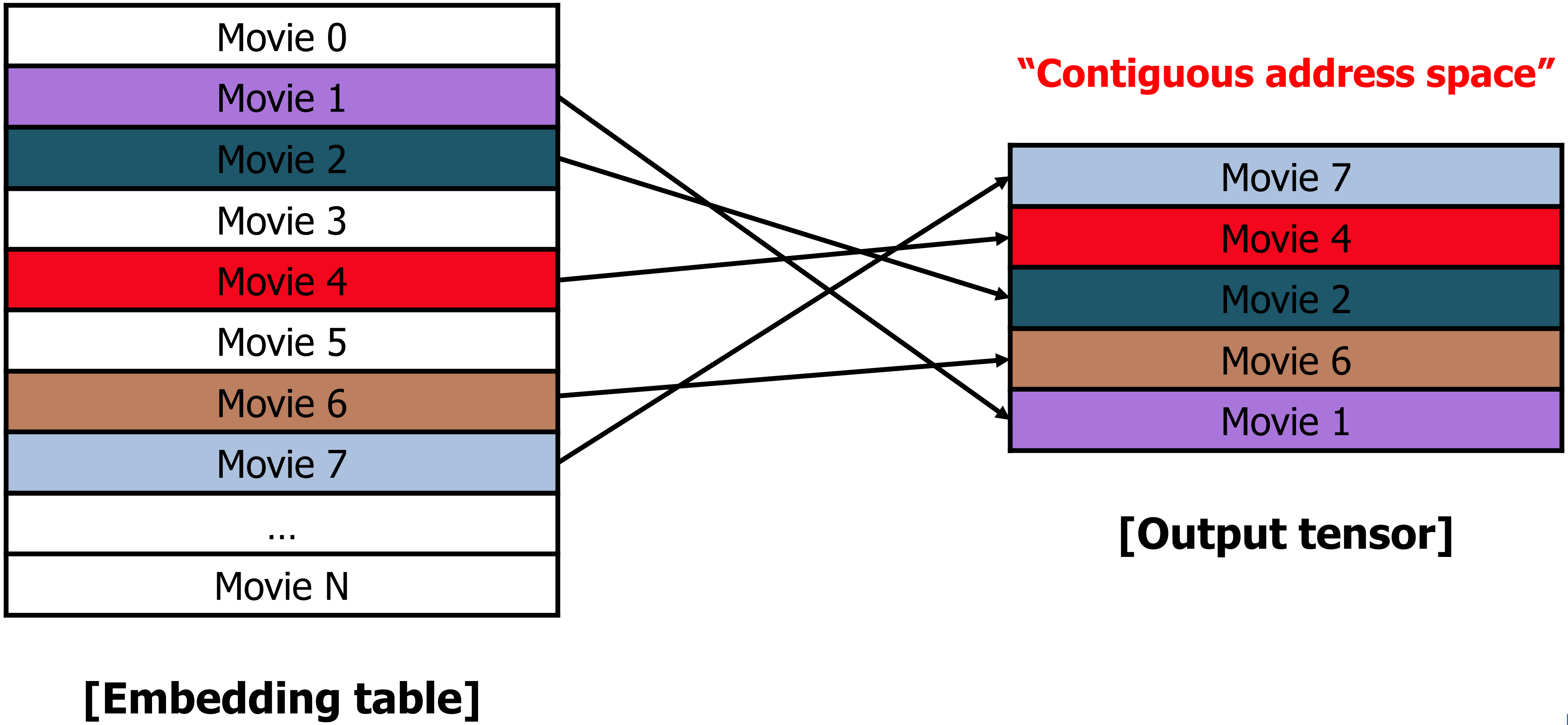
Copying target embeddings into contiguous address space

Movie 0
Movie 1
Movie 2
Movie 3
Movie 4
Movie 5
Movie 6
Movie 7
…
Movie N

[Embedding table]

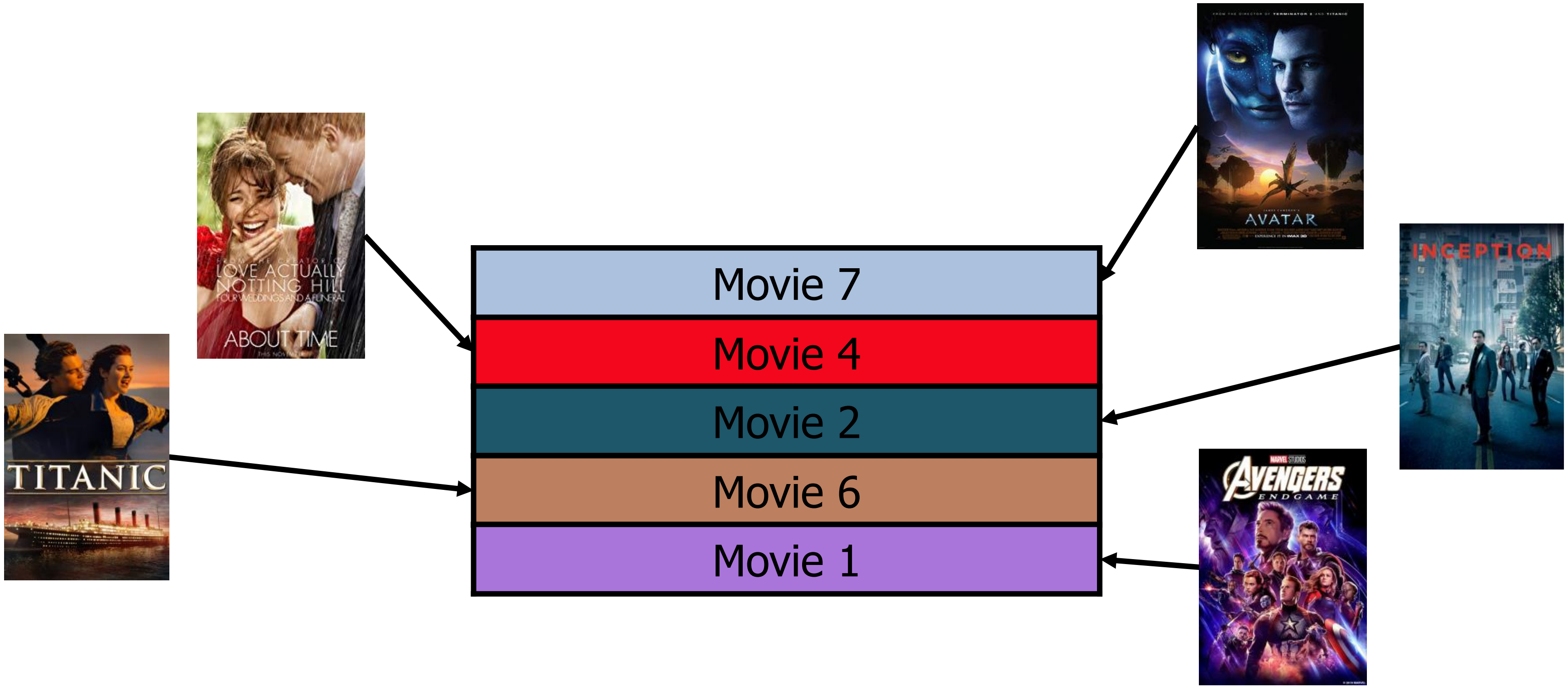
#1: Embedding lookup (gather)

Copying target embeddings into contiguous address space



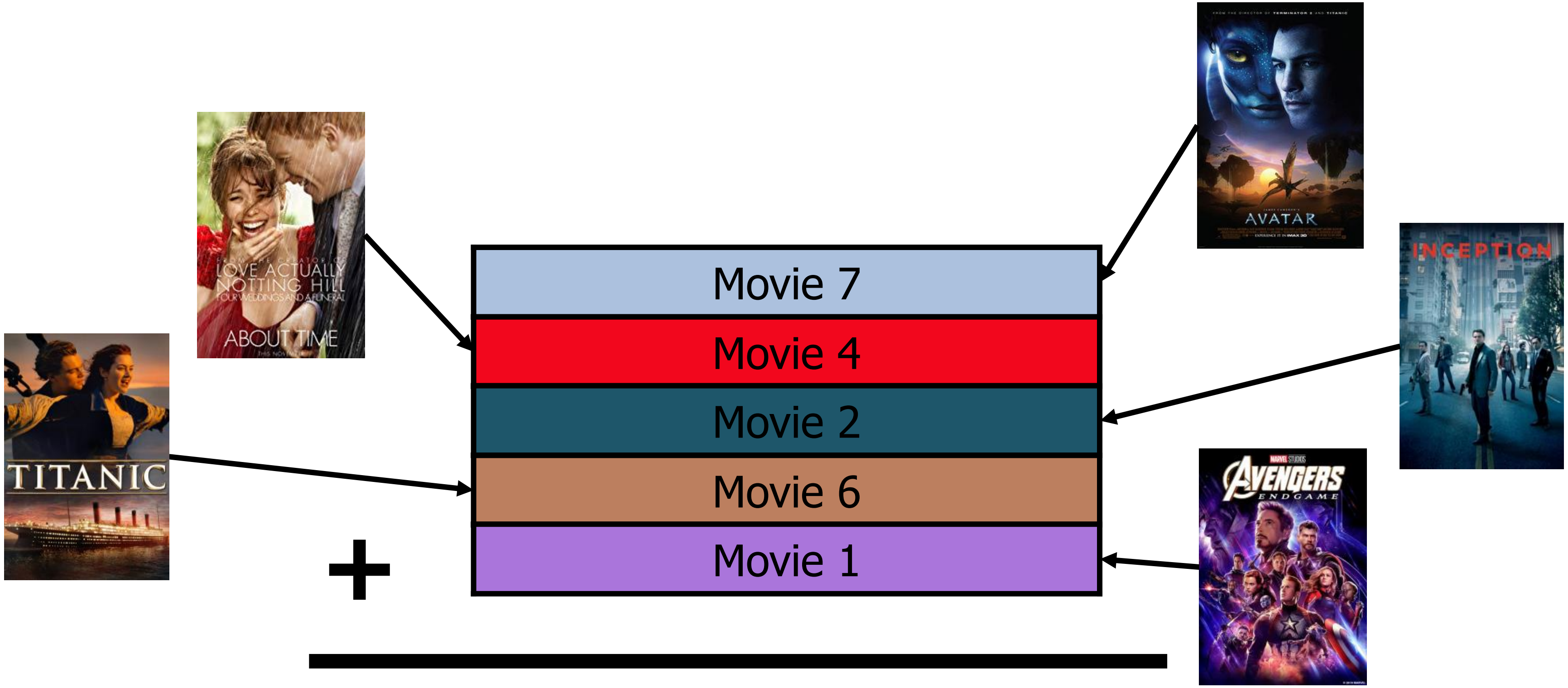
#2: Tensor operation (reduction)

e.g., Averaging multiple embeddings, element-wise addition/multiplication



#2: Tensor operation (reduction)

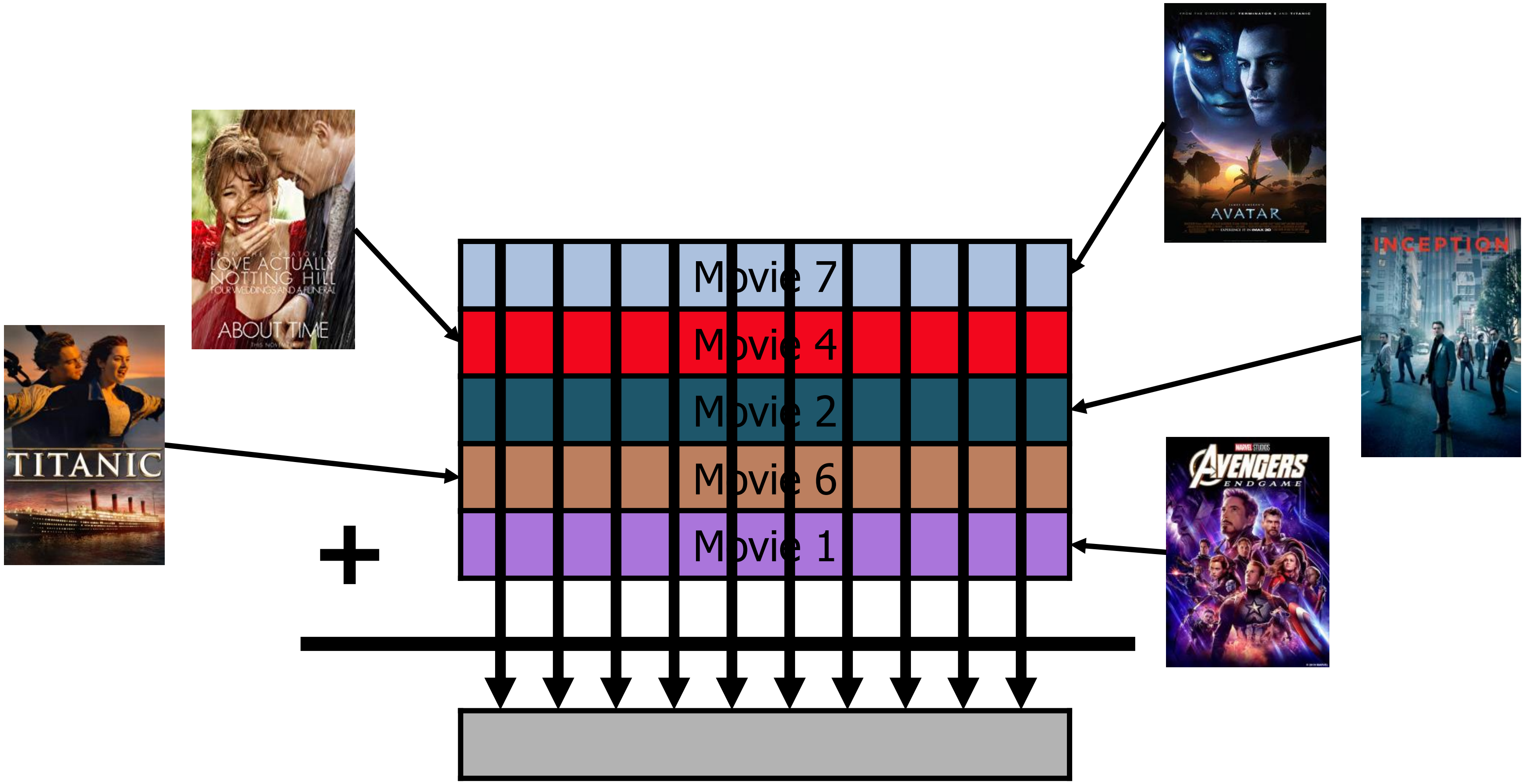
e.g., Averaging multiple embeddings, element-wise addition/multiplication



KAIST

#2: Tensor operation (reduction)

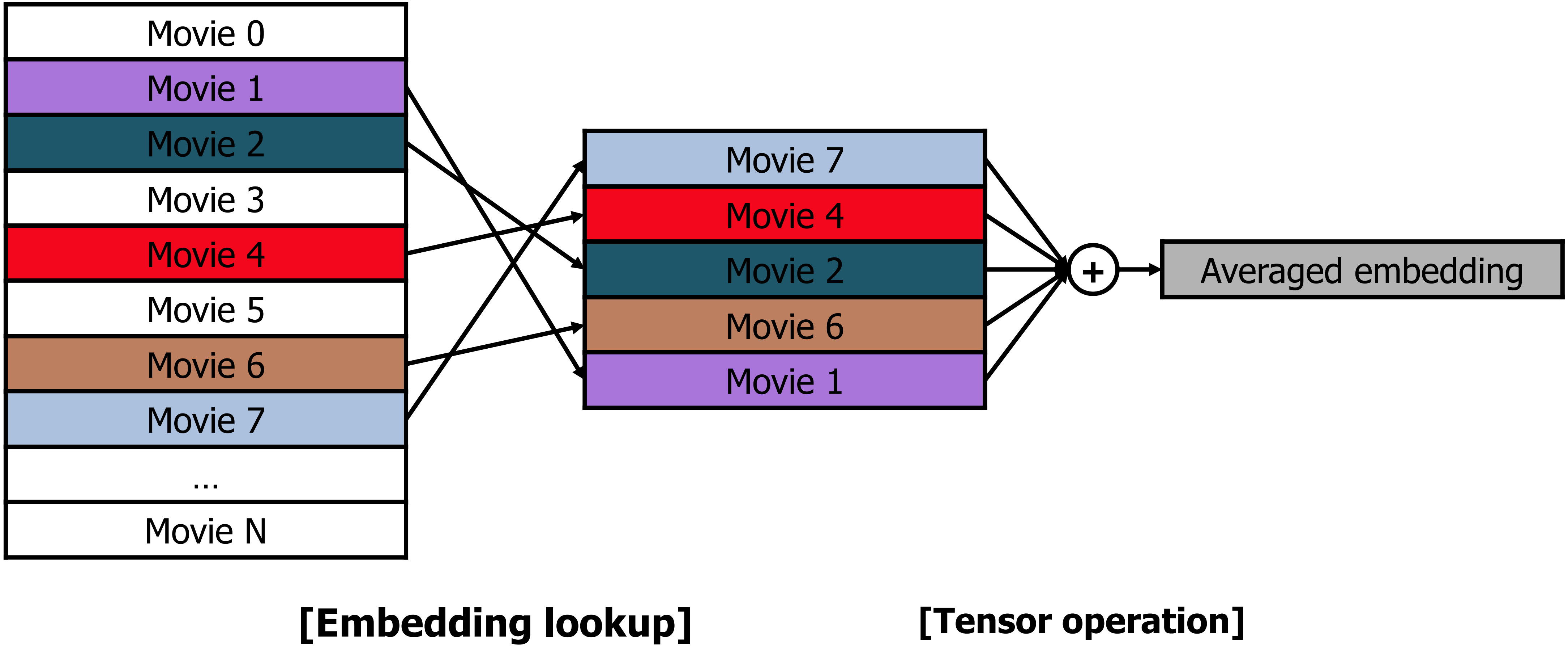
e.g., Averaging multiple embeddings, element-wise addition/multiplication



<Average movie embedding>

Key challenges of embedding layers

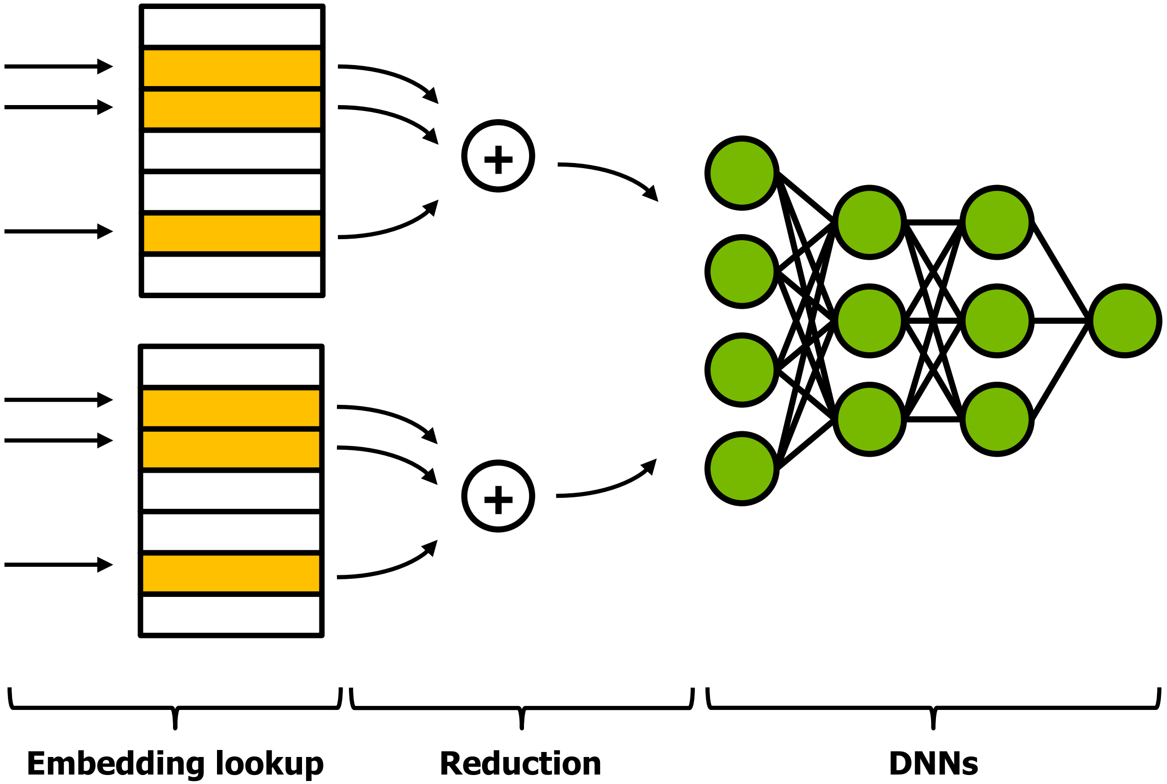
Embedding gathers/reductions are extremely memory-bandwidth sensitive



Current Solutions for Recommendation Systems

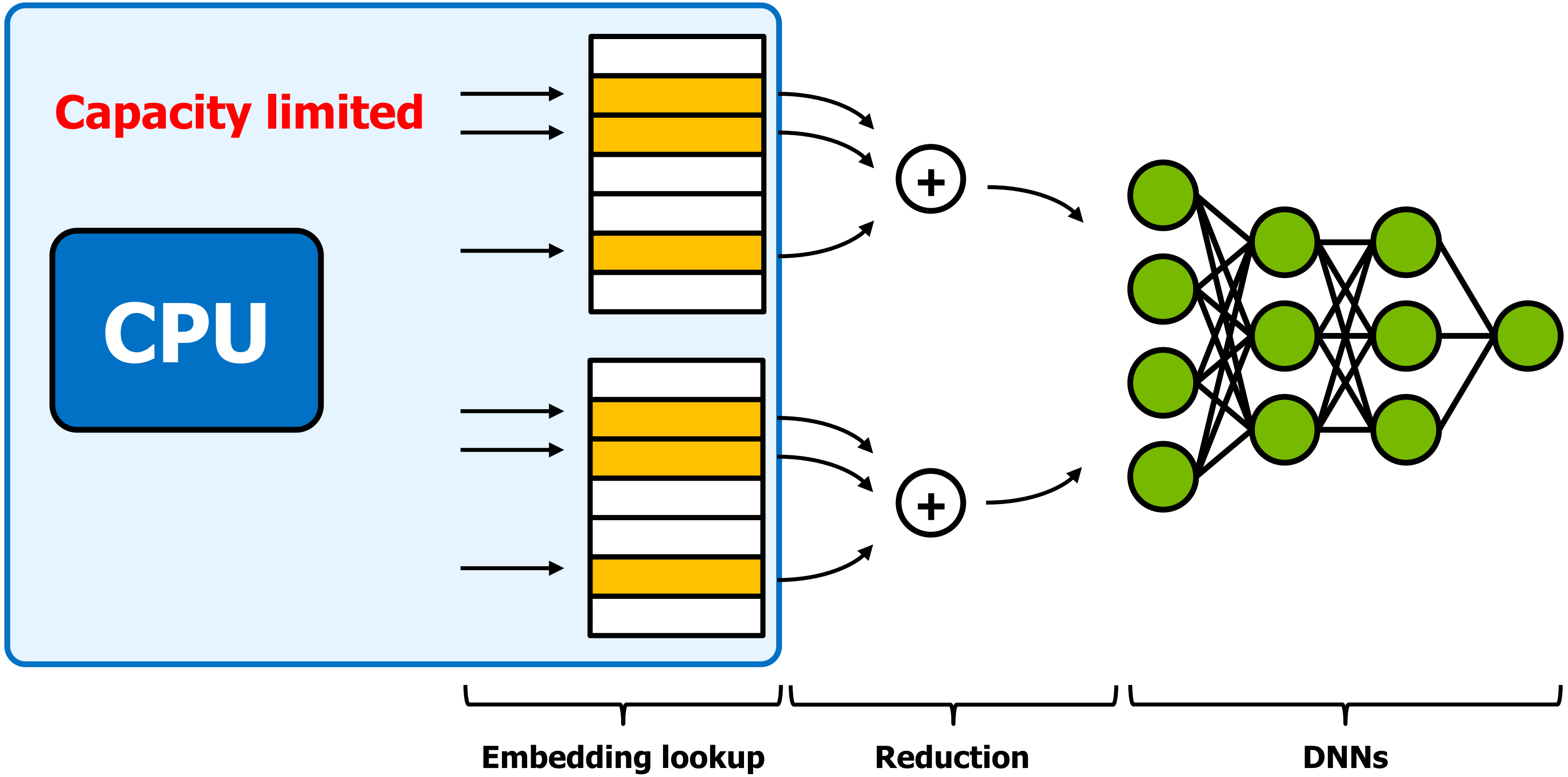
The memory wall for "Inference"

Size of embedding tables can reach hundreds of GBs



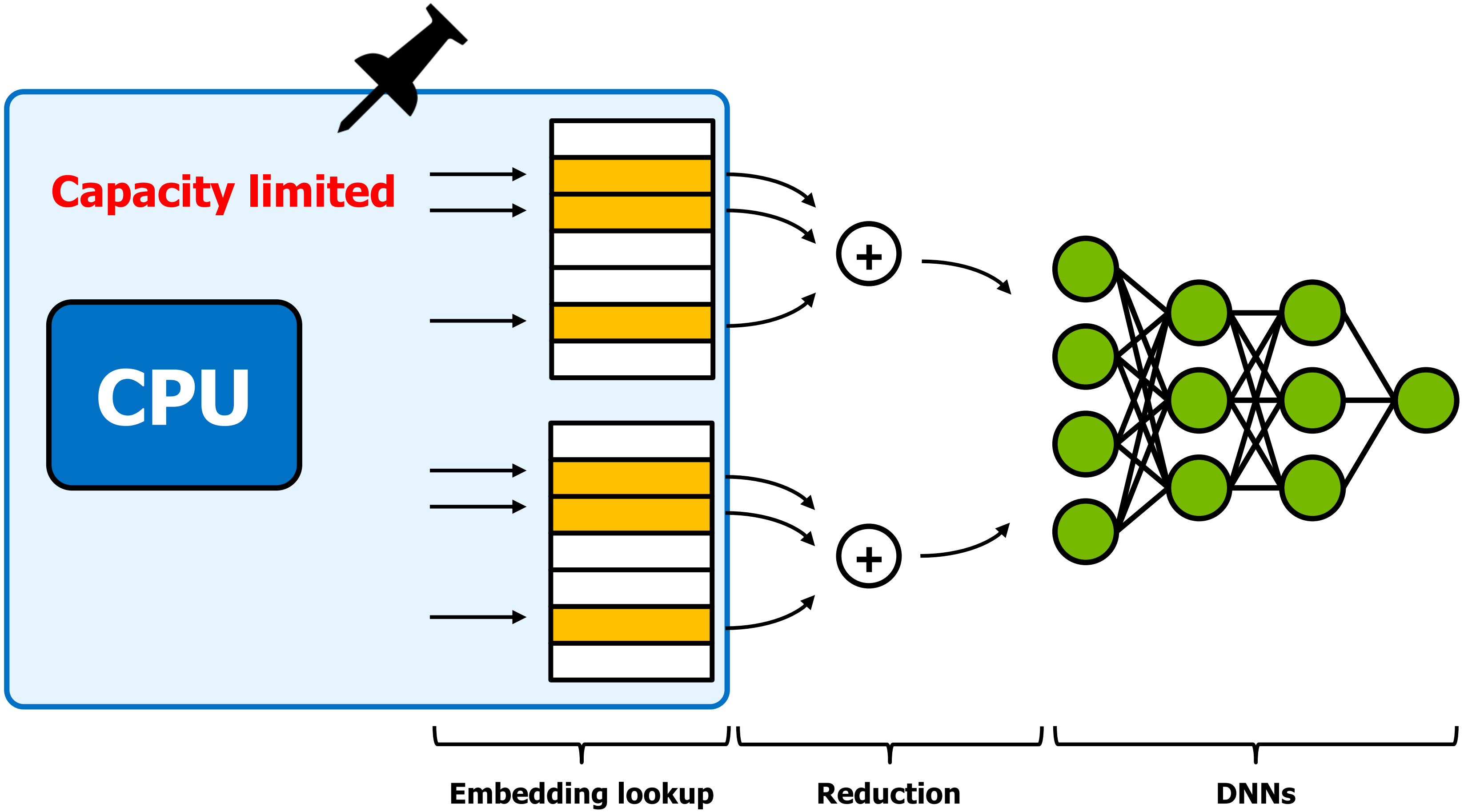
The memory wall for "Inference"

Size of embedding tables can reach hundreds of GBs



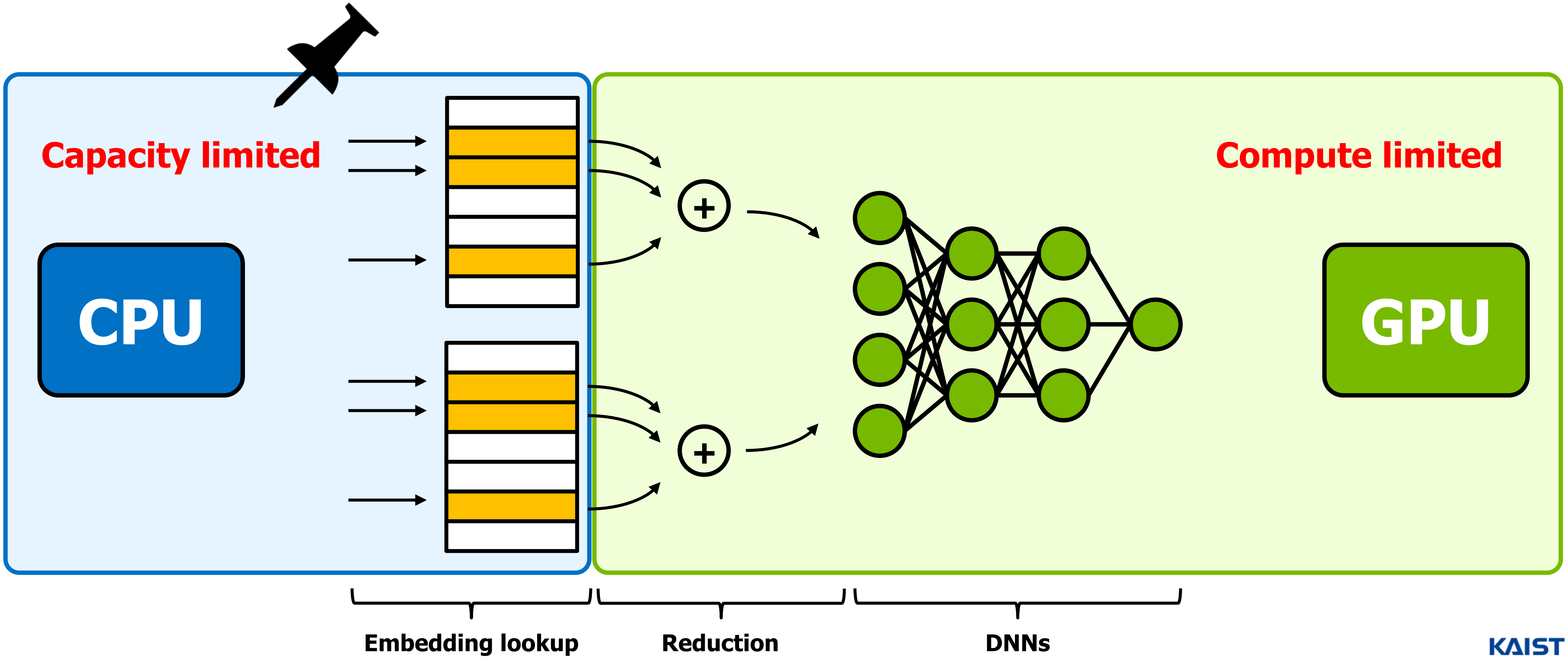
The memory wall for "Inference"

Size of embedding tables can reach hundreds of GBs



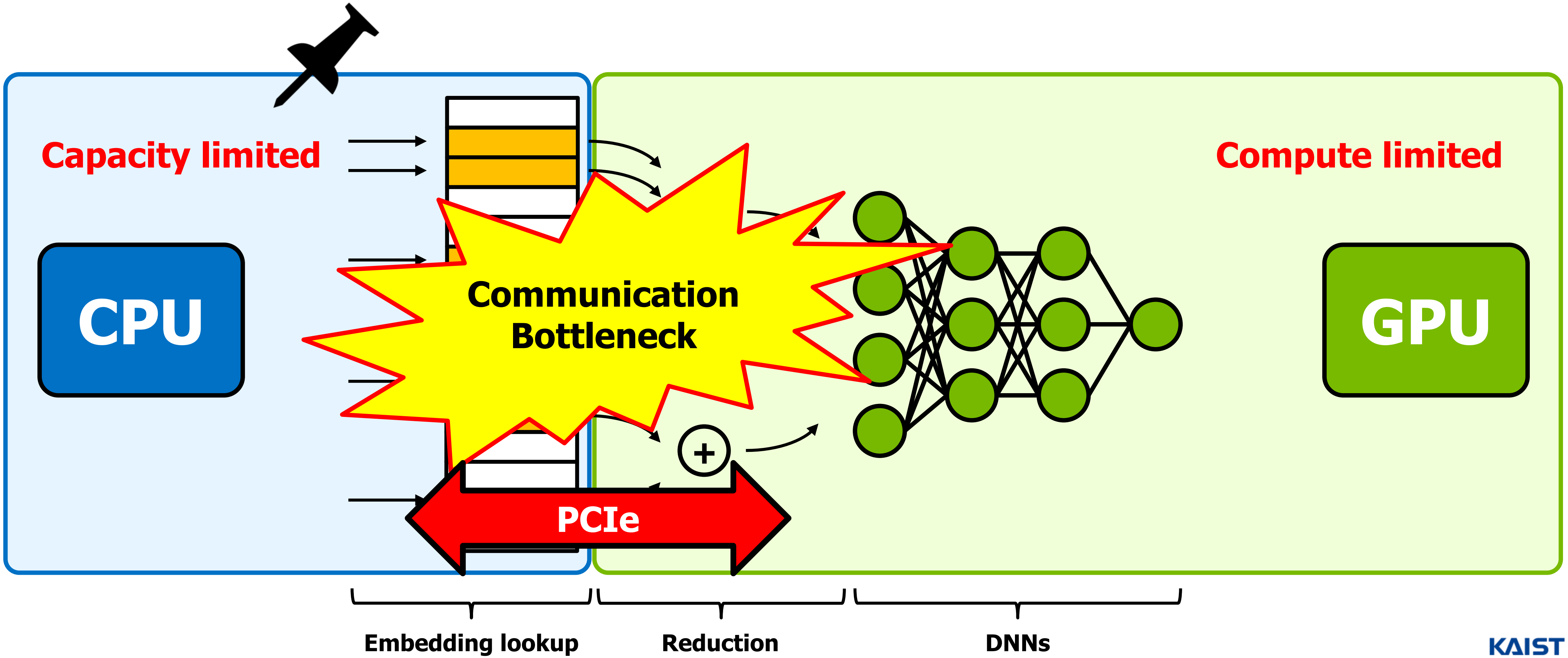
Design#1: Hybrid CPU-GPU approach

CPU stores entire embedding tables, but DNNs executed using GPUs



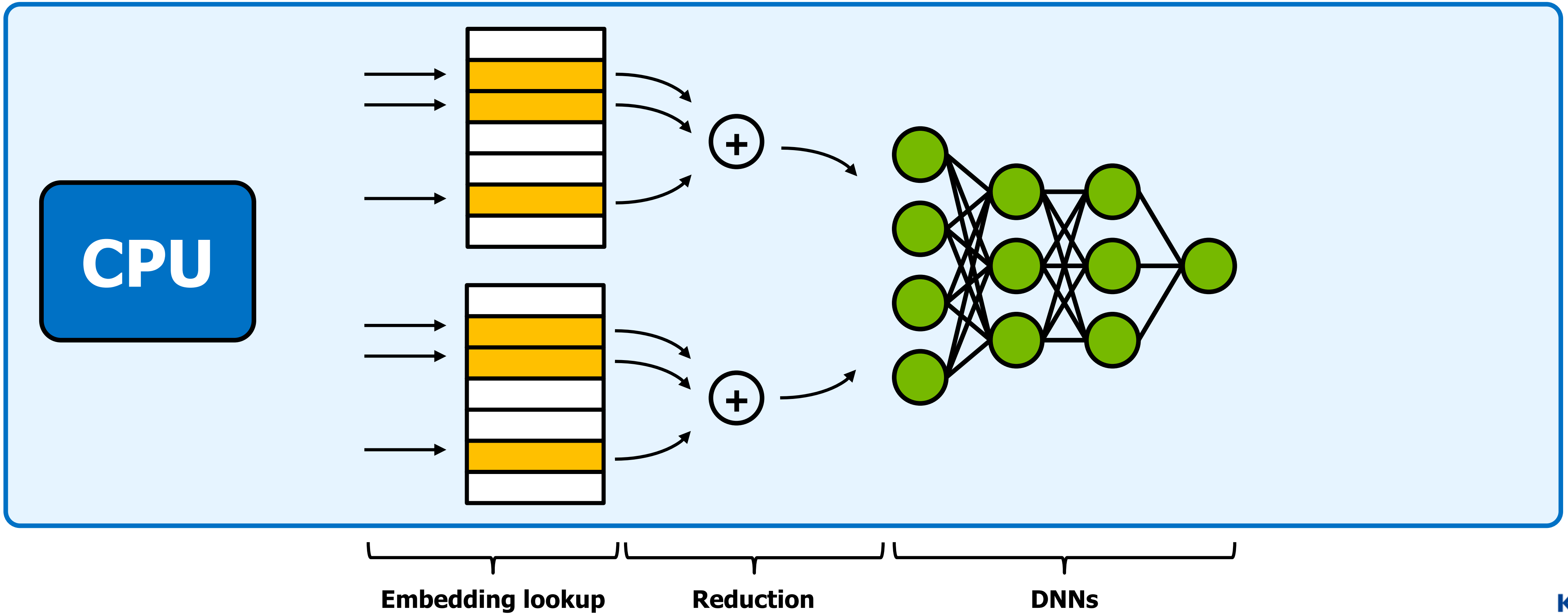
Design#1: Hybrid CPU-GPU approach

Challenges: need to copy multiple embeddings via narrow PCIe channel



Design#2: CPU-only approach

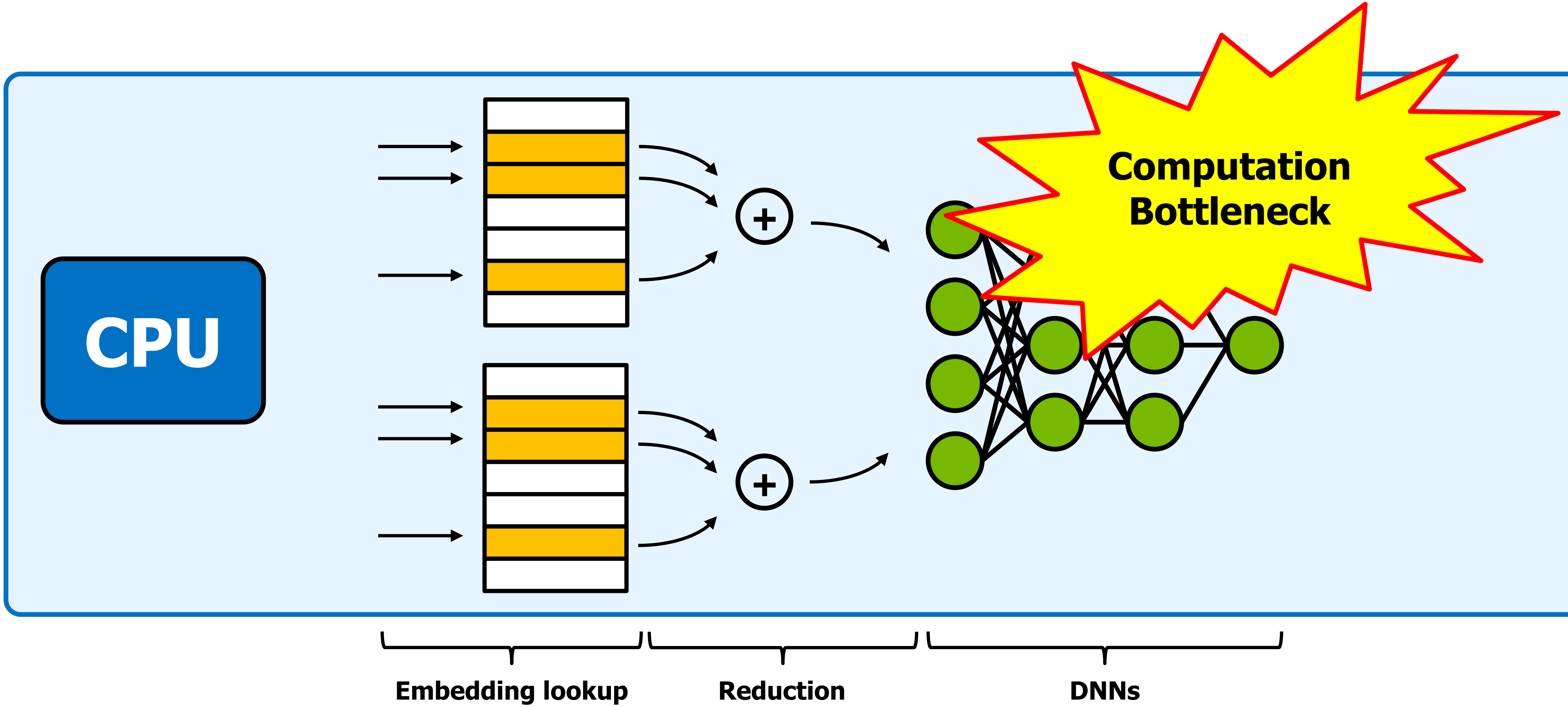
The CPU handles the entire steps of inference



KAIST

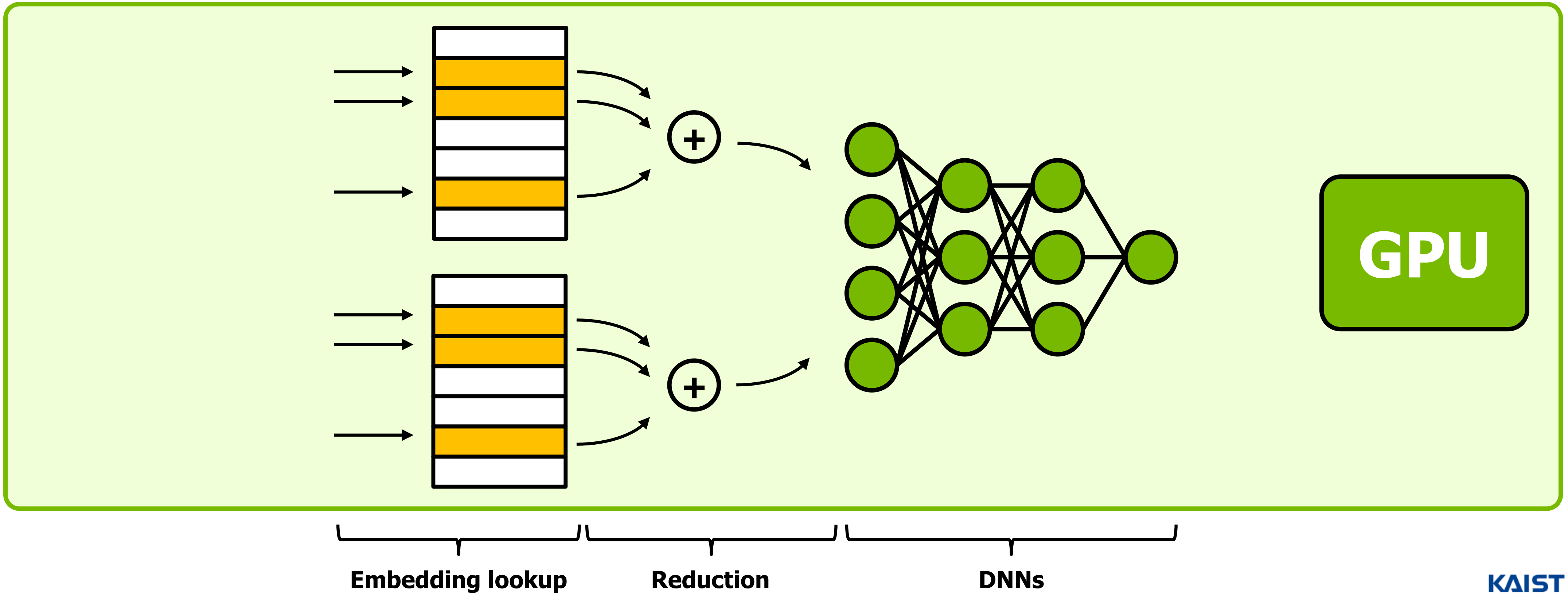
Design#2: CPU-only approach

Challenges: low throughput of CPUs slows down DNN computation



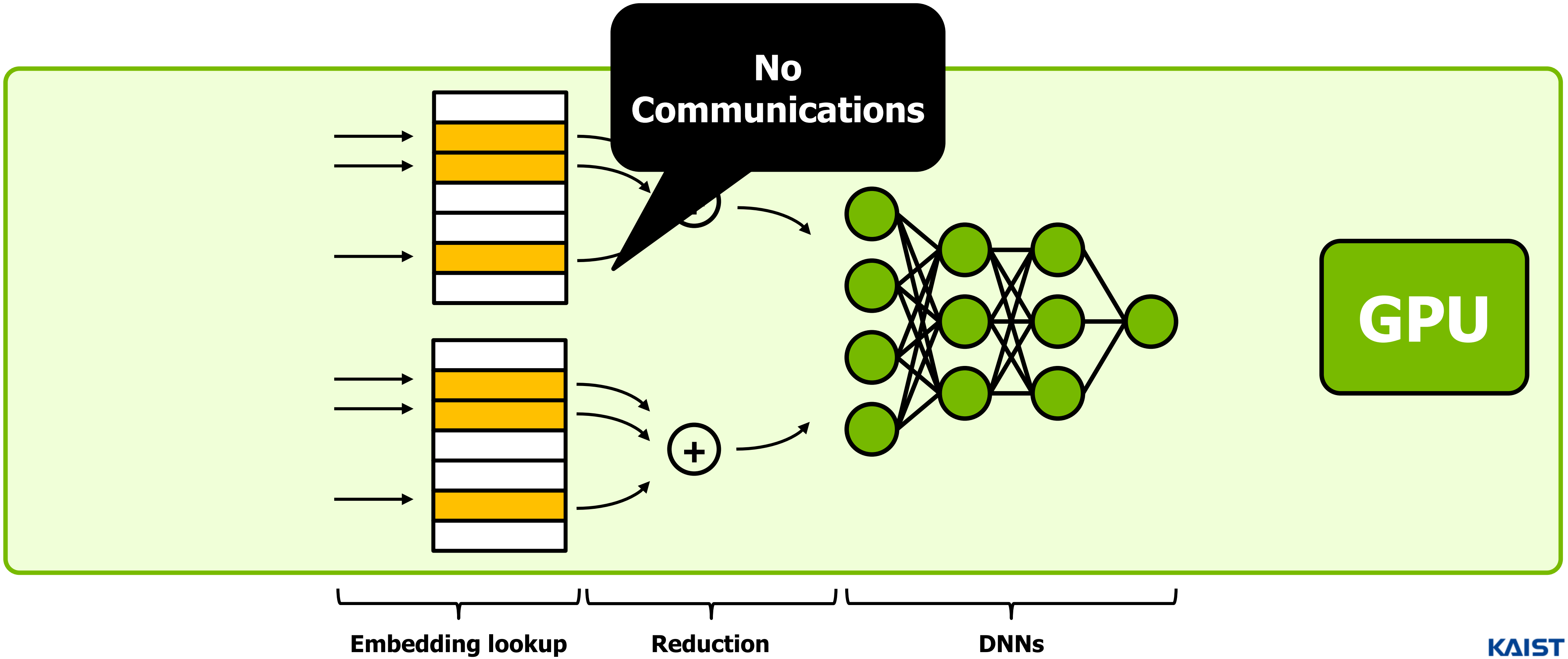
Design#3: GPU-only approach?

Unbuildable, oracular solution assuming infinite GPU memory capacity



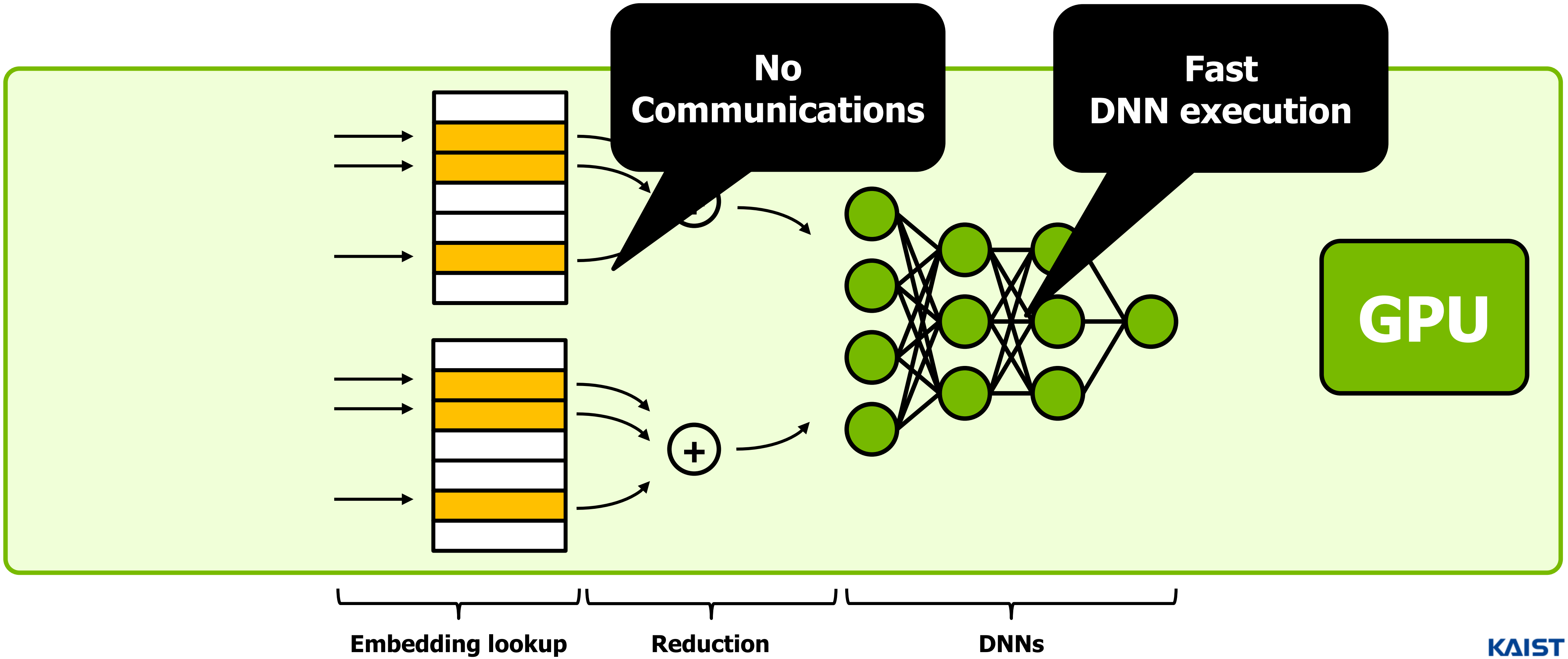
Design#3: GPU-only approach?

Unbuildable, oracular solution assuming infinite GPU memory capacity



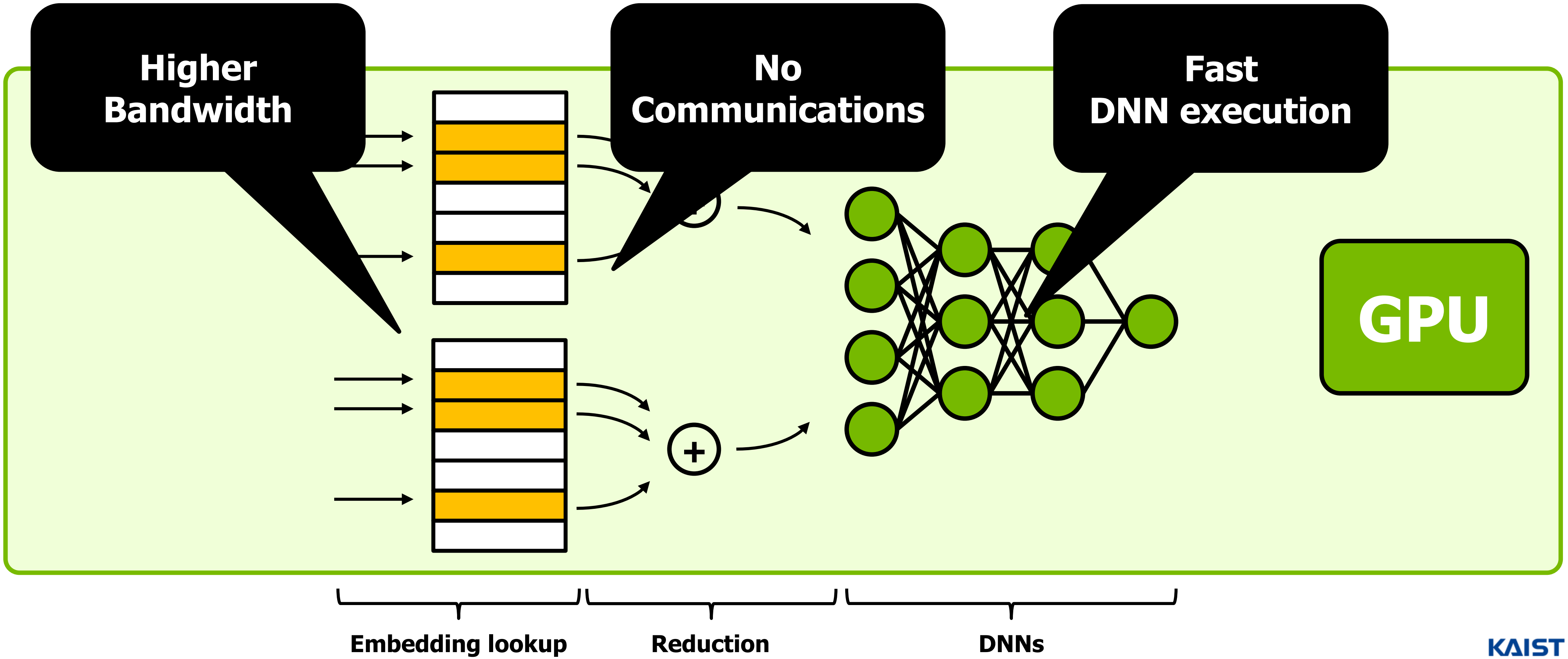
Design#3: GPU-only approach?

Unbuildable, oracular solution assuming infinite GPU memory capacity



Design#3: GPU-only approach?

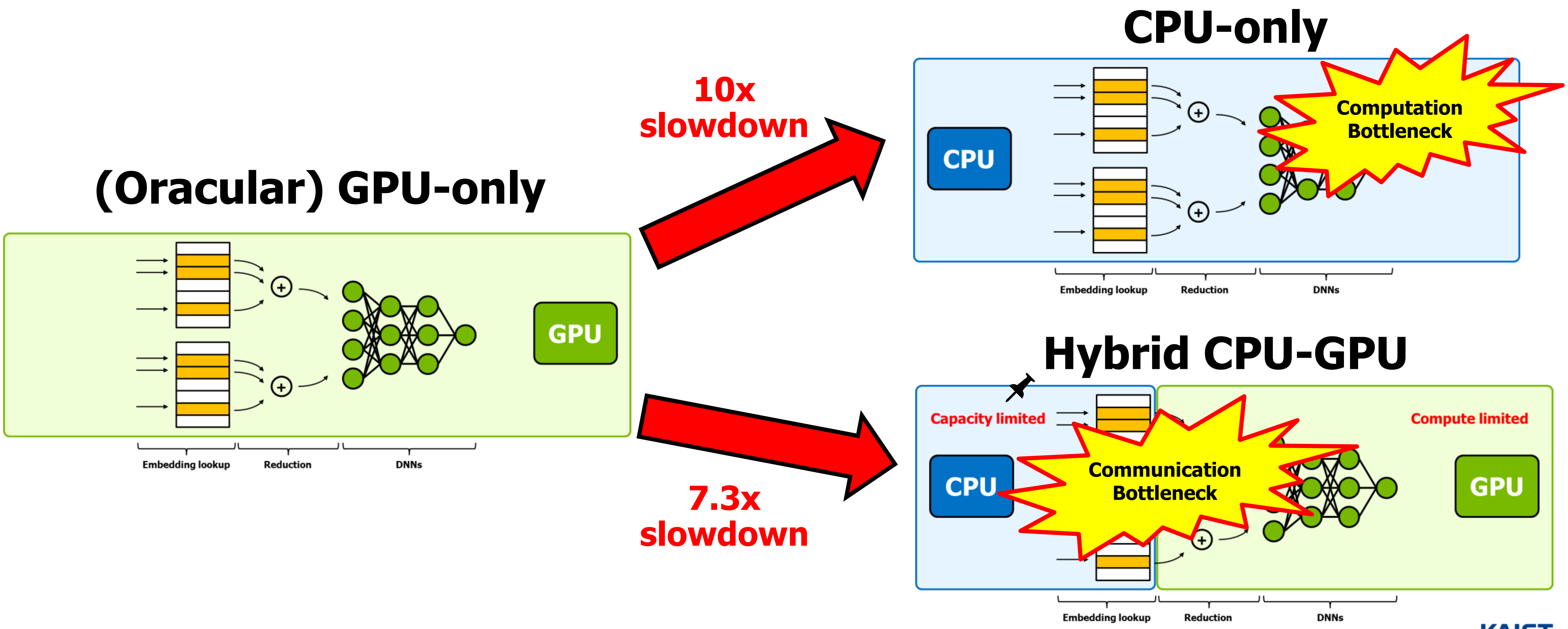
Unbuildable, oracular solution assuming infinite GPU memory capacity



KAIST

Key challenges of existing solutions?

CPU-only and hybrid CPU-GPU (vs. GPU-only)

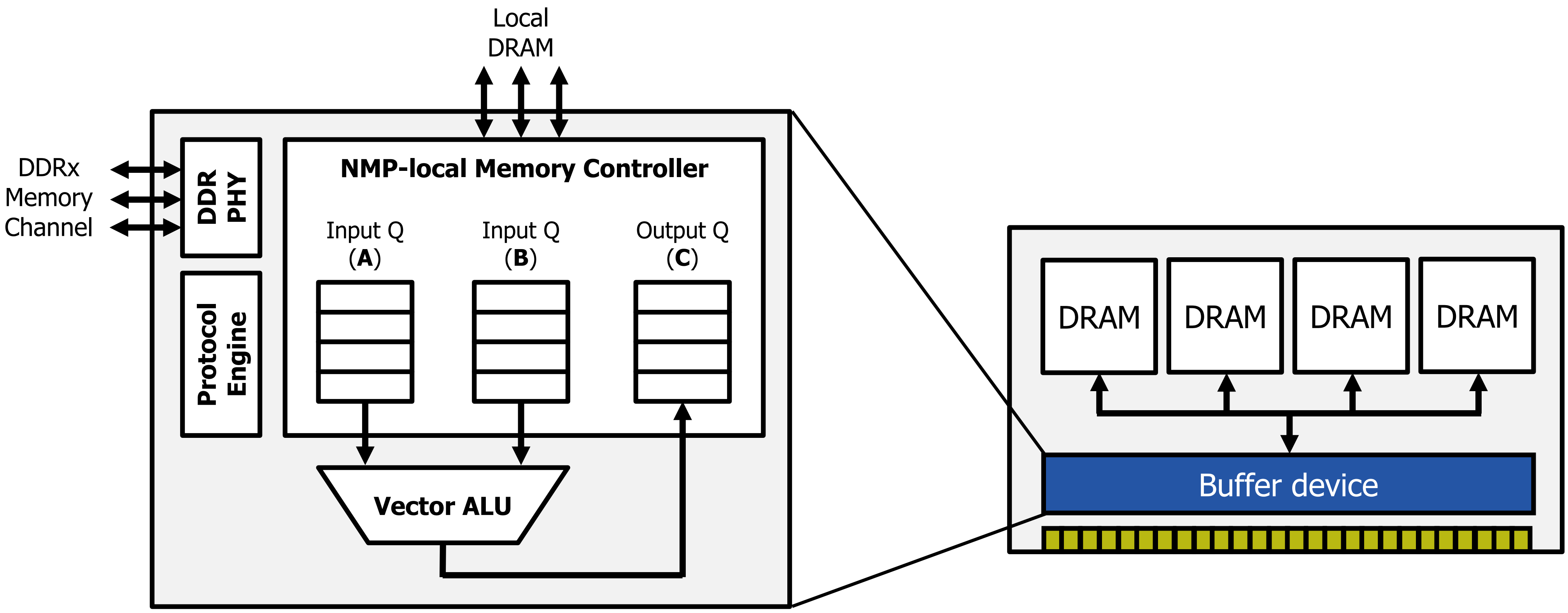


Our Approach: “Near”-Memory Acceleration

(so Near-Memory Processing, NMP)

TensorDIMM: a NMP for embeddings

Augment buffer device to add NMP cores for embedding gathers/reductions

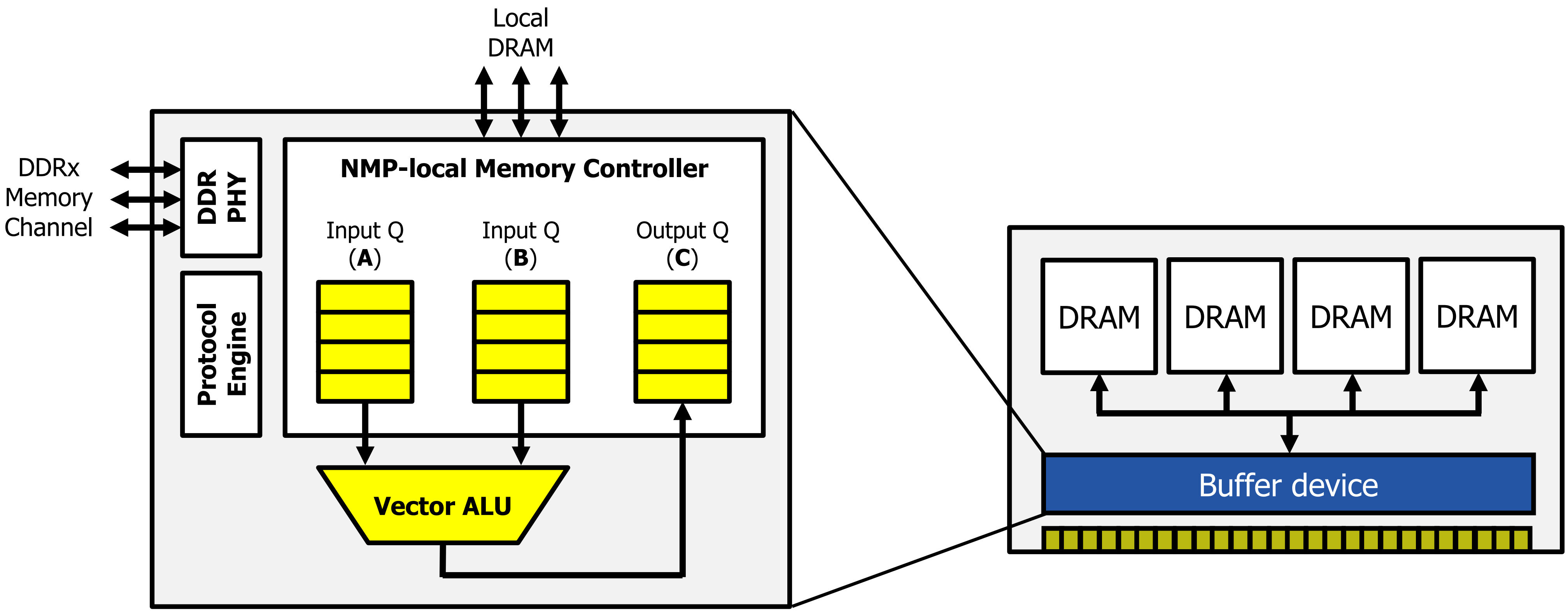


(a) NMP core

(b) TensorDIMM

TensorDIMM: a NMP for embeddings

Augment buffer device to add NMP cores for embedding gathers/reductions

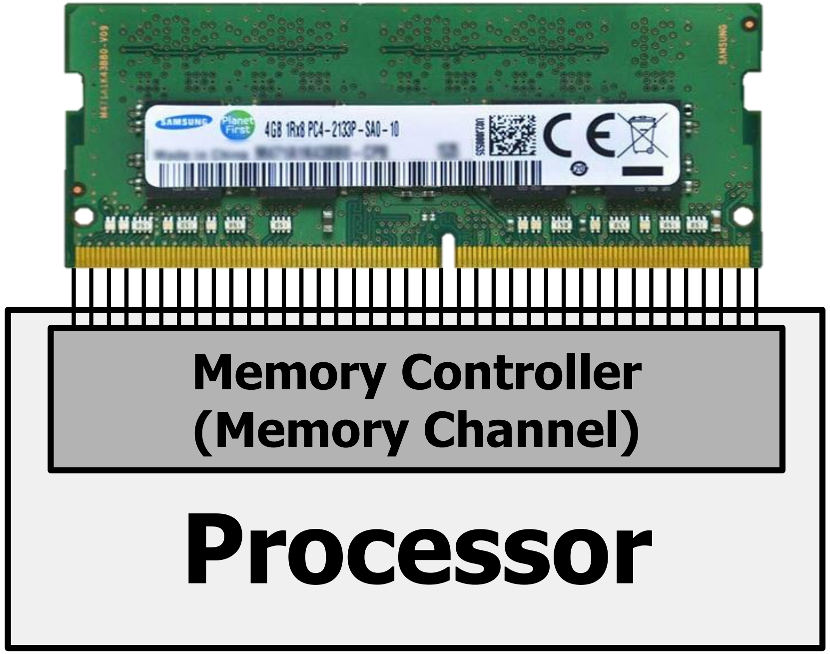


(a) NMP core

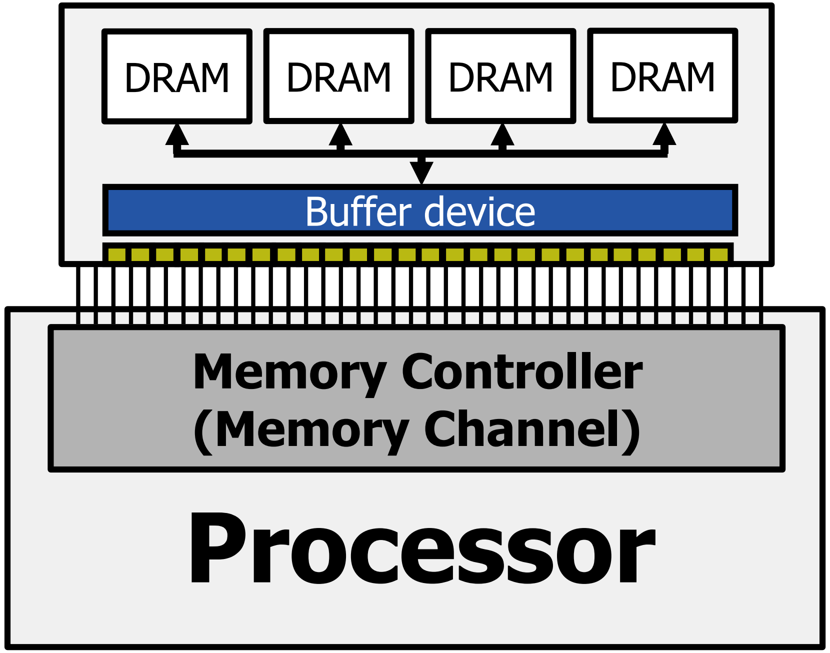
(b) TensorDIMM

Key advantage of TensorDIMM

"Effective" memory bandwidth scales proportional to the # of DIMMs/ranks



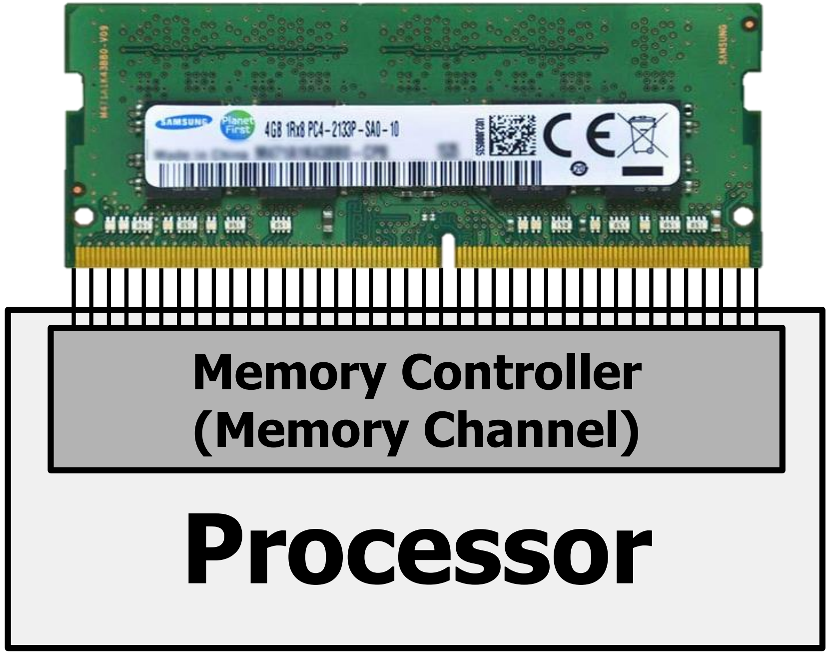
Current system



TensorDIMM approach

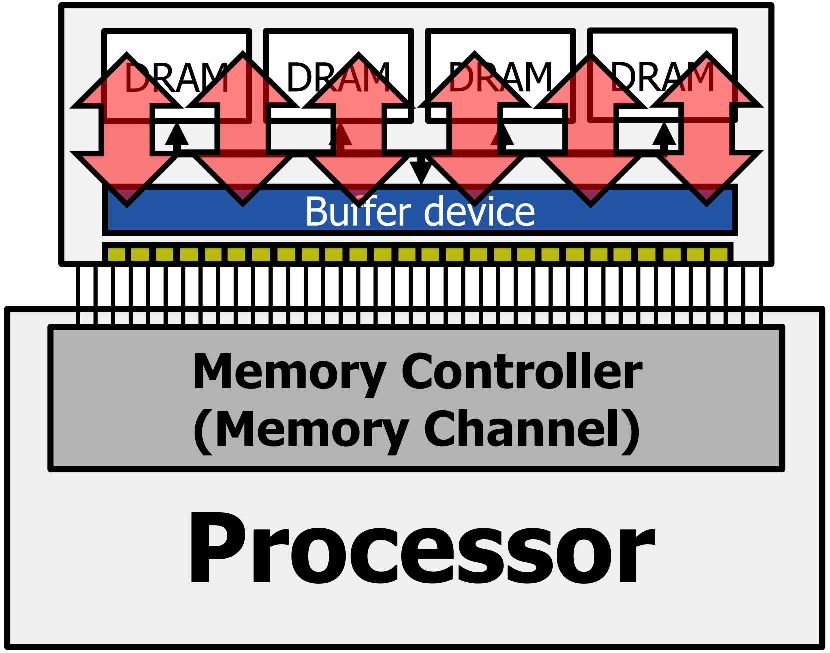
Key advantage of TensorDIMM

"Effective" memory bandwidth scales proportional to the # of DIMMs/ranks



Current system

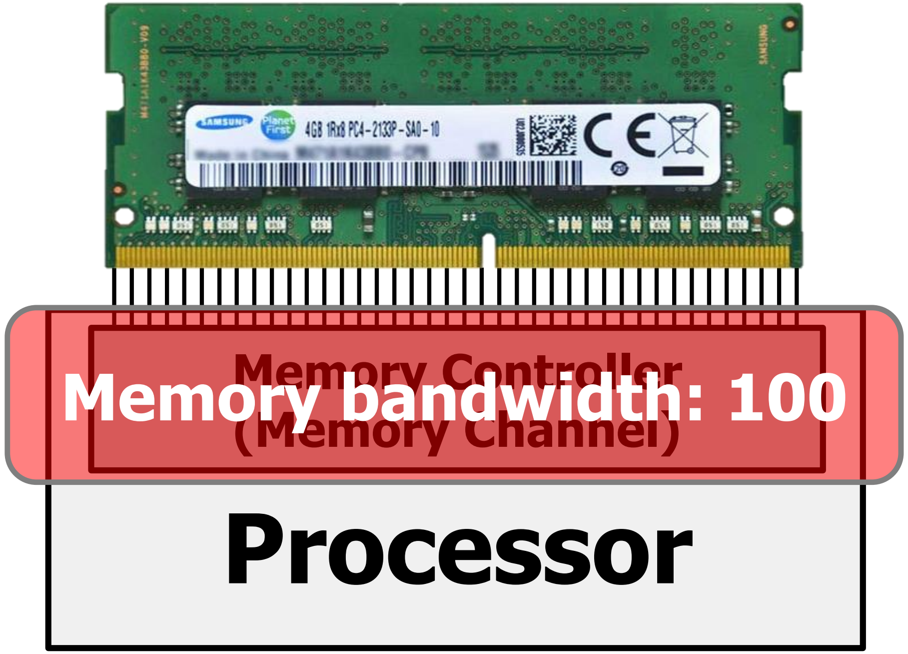
Embedding **gathers/reductions** are done **"locally"** within a DIMM



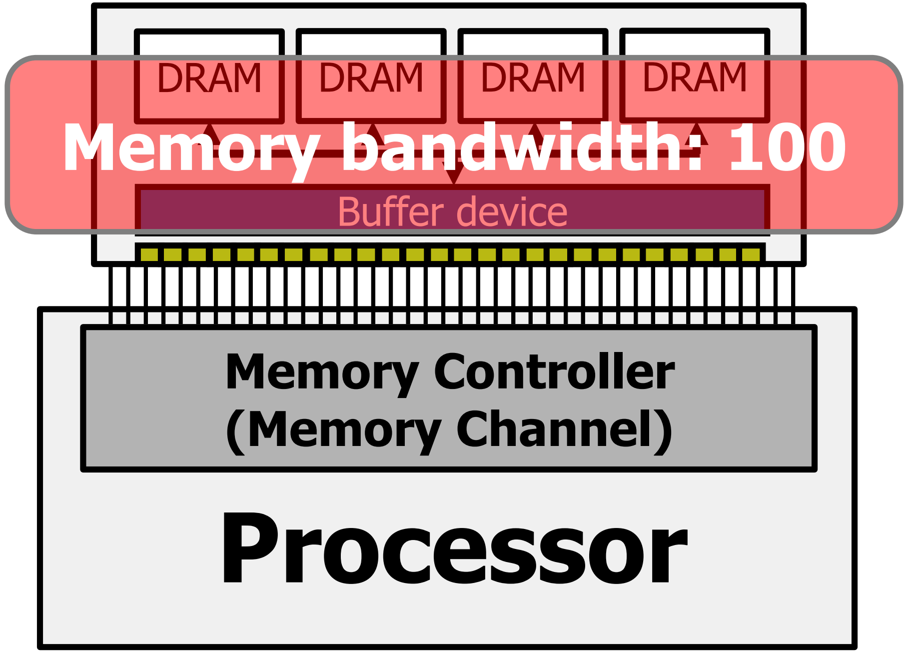
TensorDIMM approach

Key advantage of TensorDIMM

"Effective" memory bandwidth scales proportional to the # of DIMMs/ranks



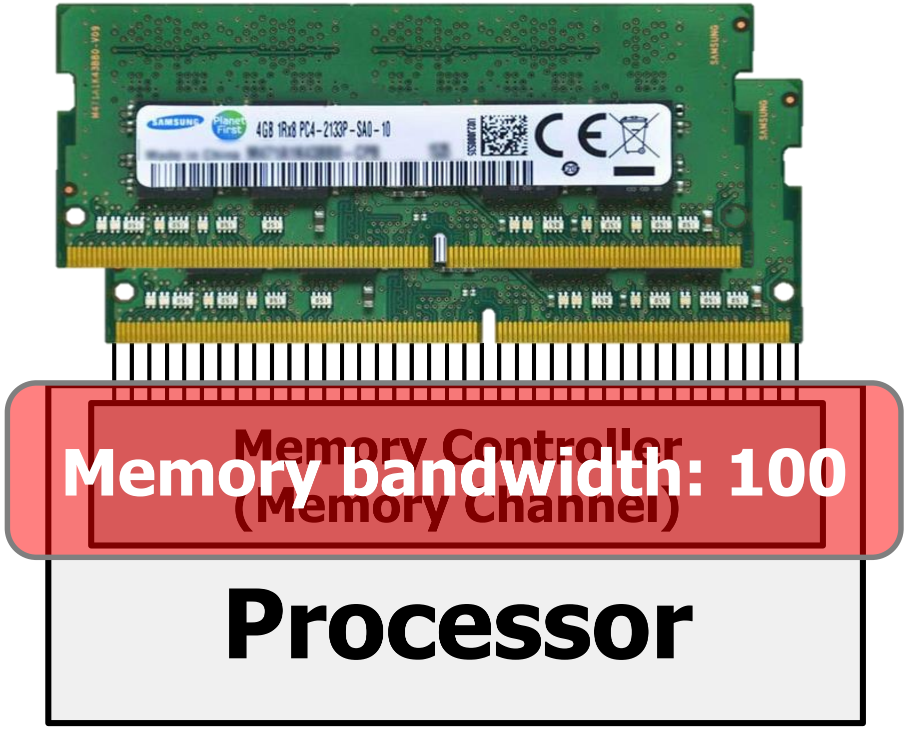
Current system



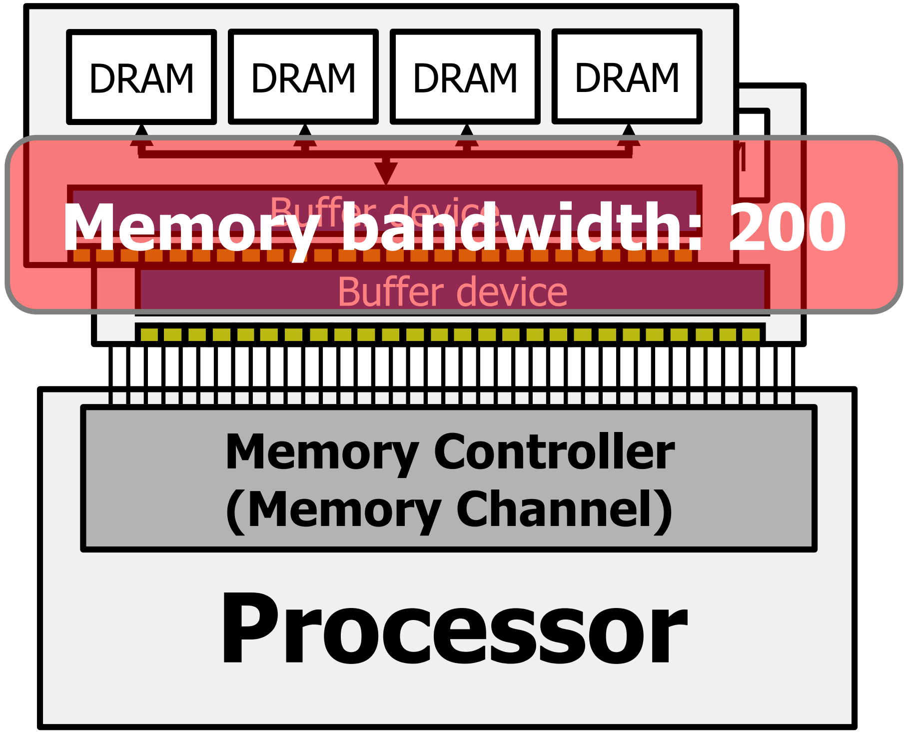
TensorDIMM approach

Key advantage of TensorDIMM

"Effective" memory bandwidth scales proportional to the # of DIMMs/ranks



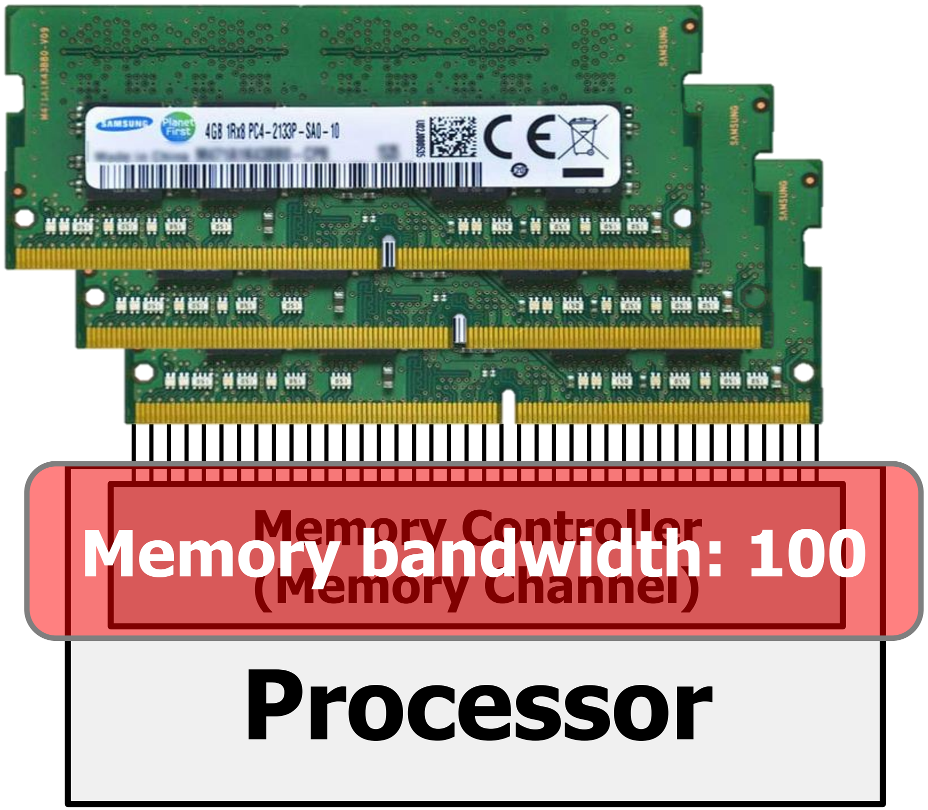
Current system



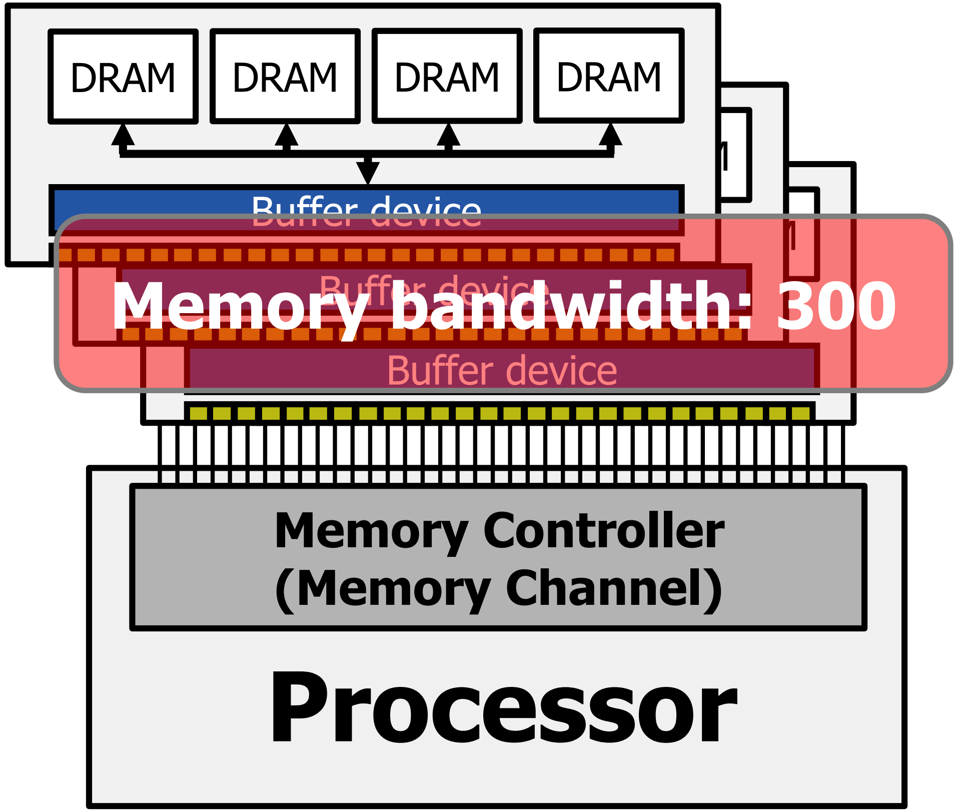
TensorDIMM approach

Key advantage of TensorDIMM

"Effective" memory bandwidth scales proportional to the # of DIMMs/ranks



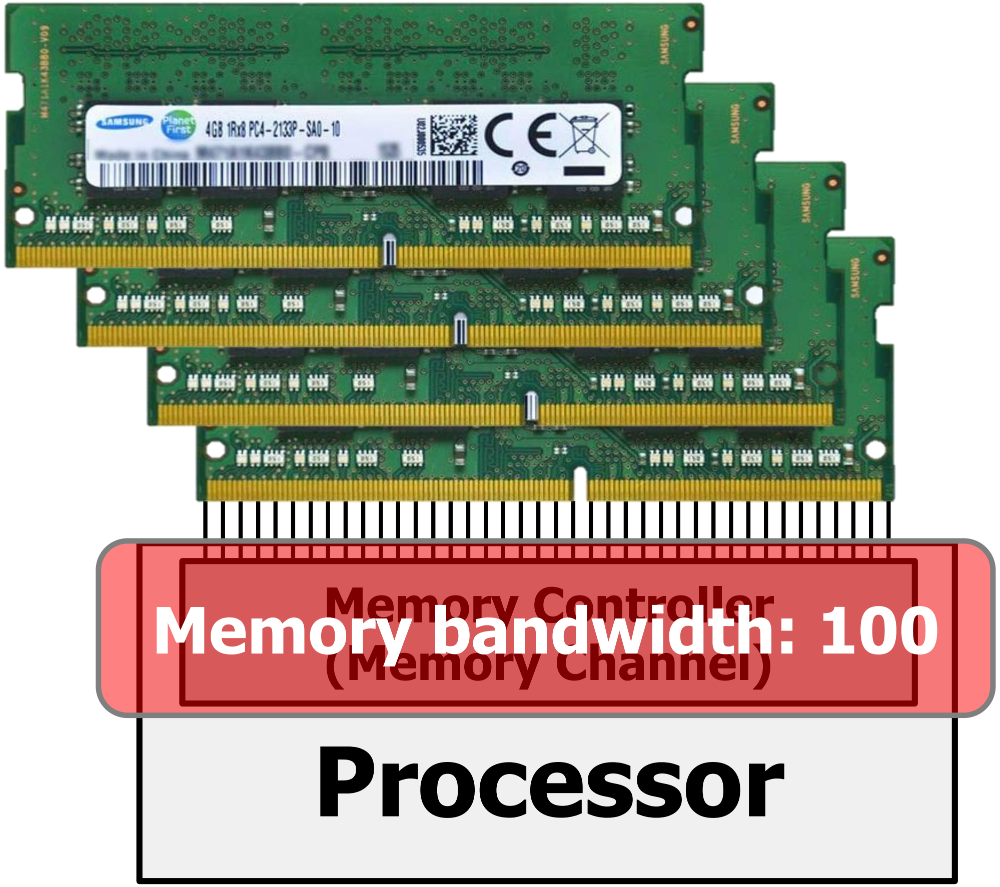
Current system



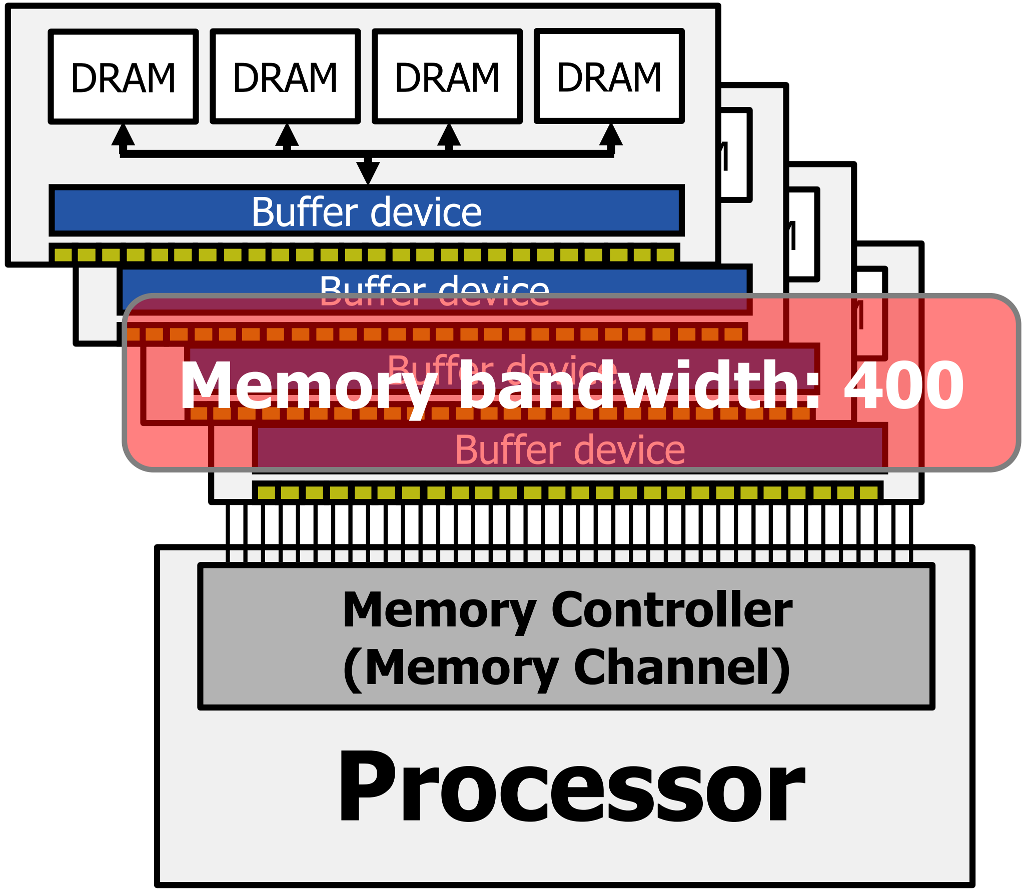
TensorDIMM approach

Key advantage of TensorDIMM

"Effective" memory bandwidth scales proportional to the # of DIMMs/ranks



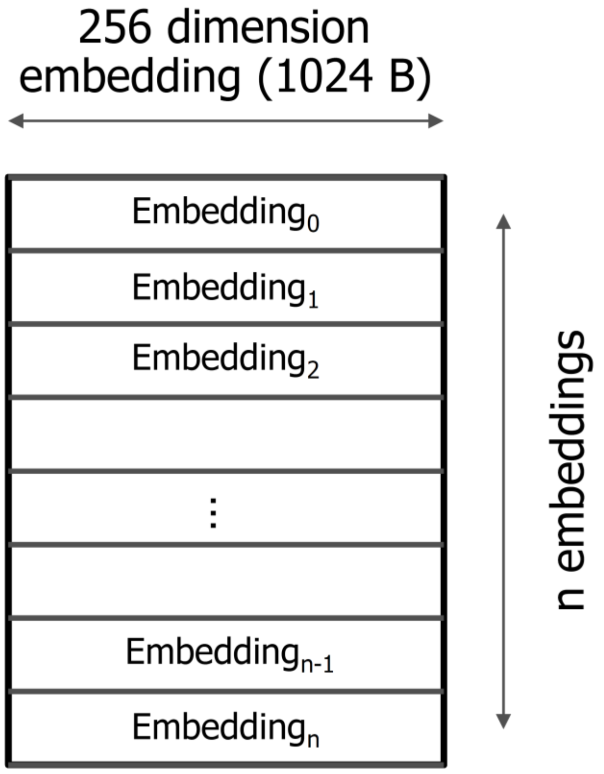
Current system



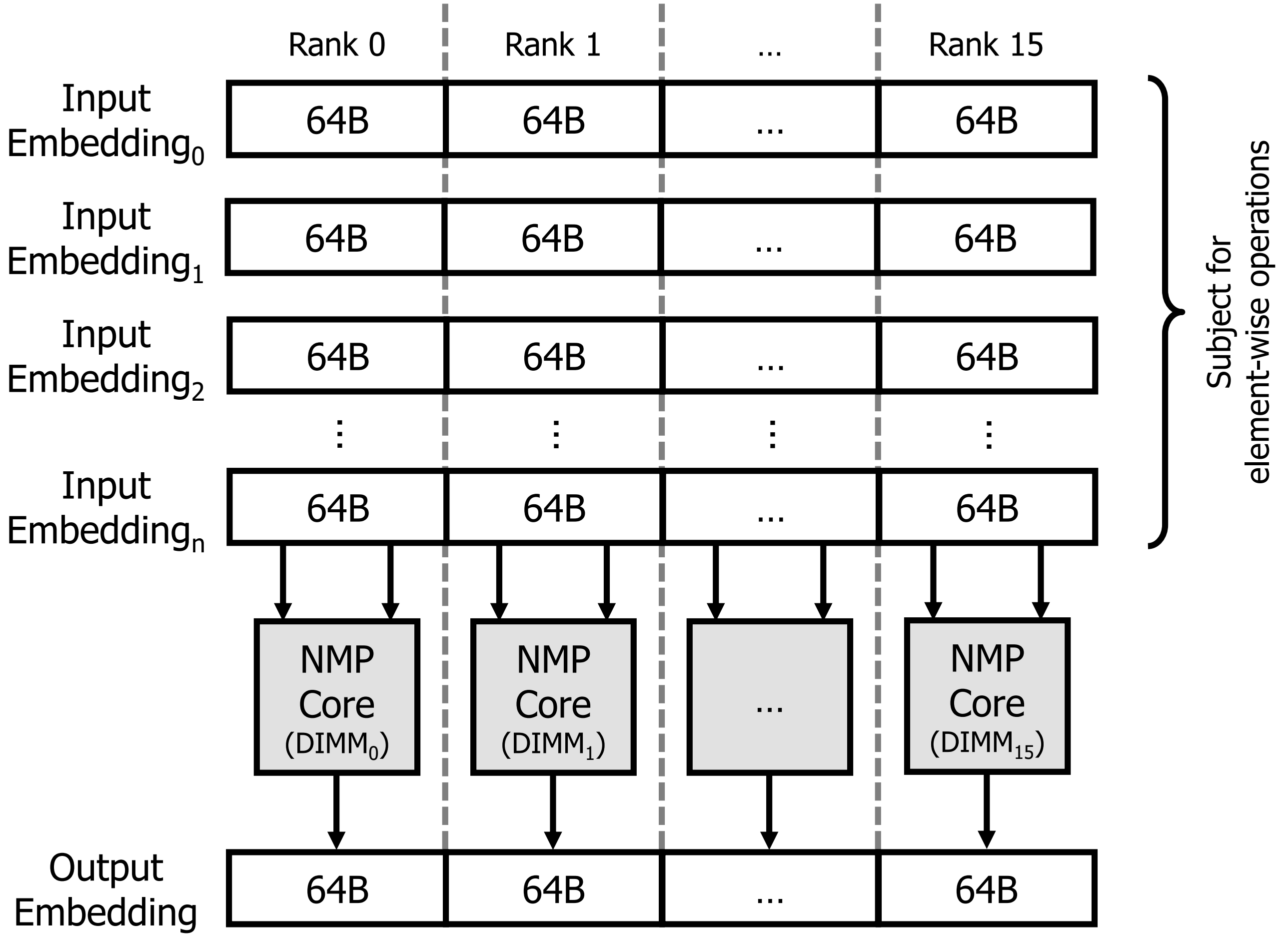
TensorDIMM approach

Mapping embedding tables in DRAMs

Leverage rank-level parallelism for maximal bandwidth utilization

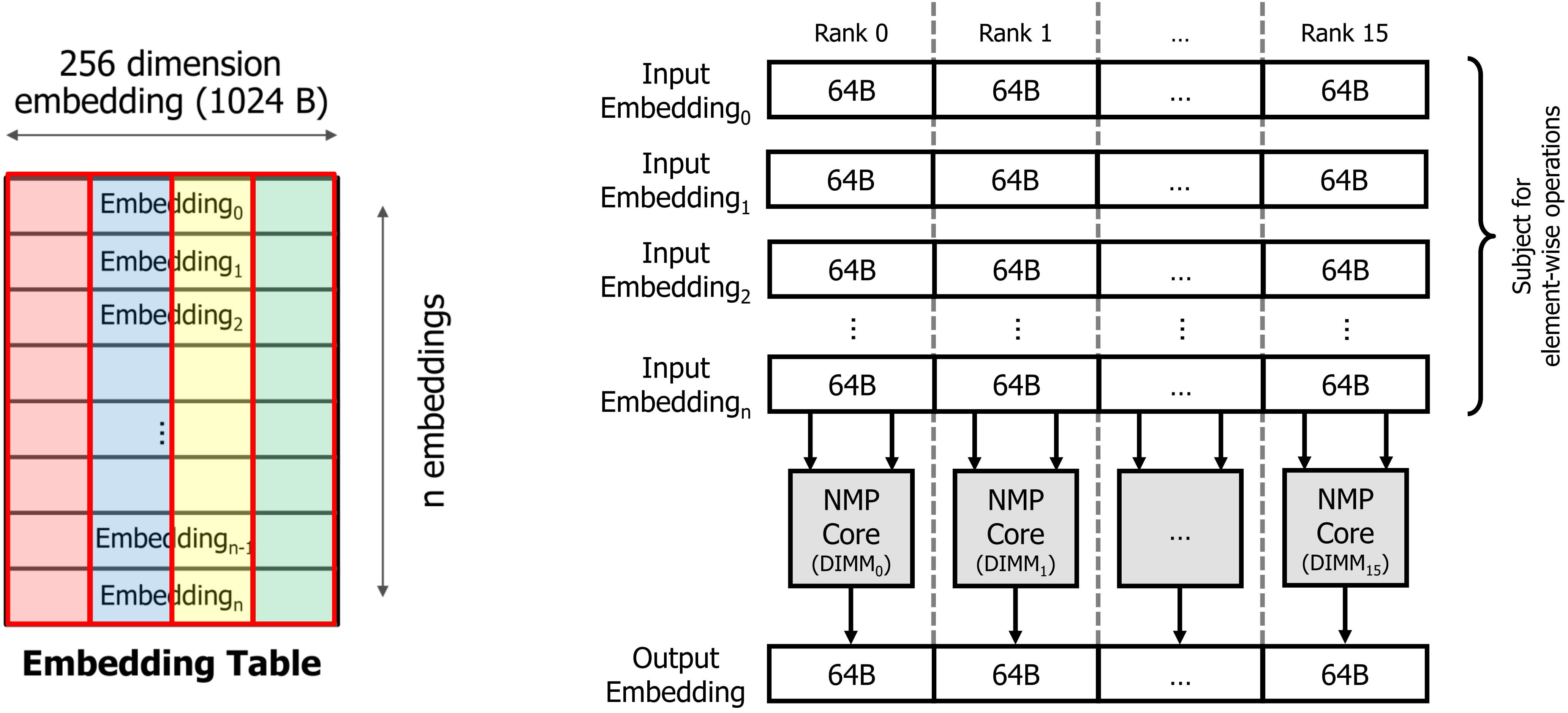


Embedding Table



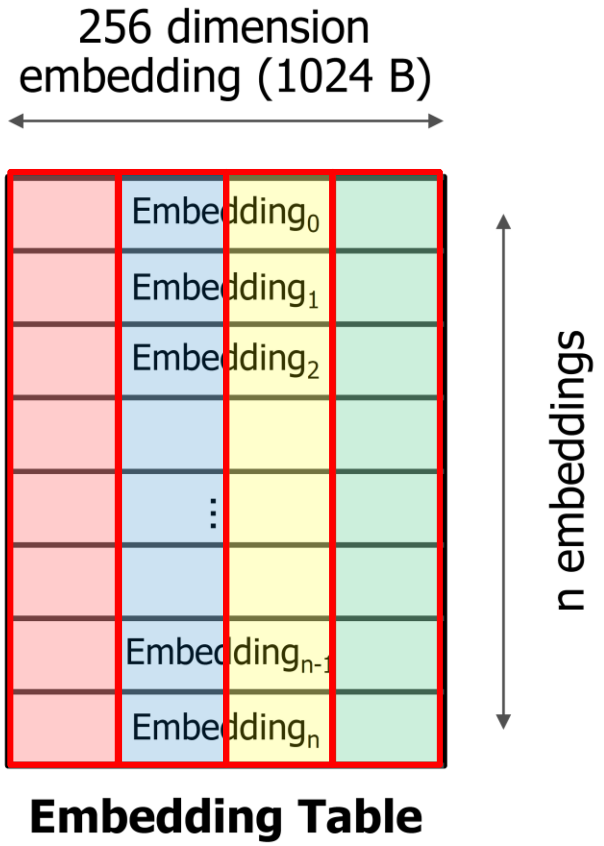
Mapping embedding tables in DRAMs

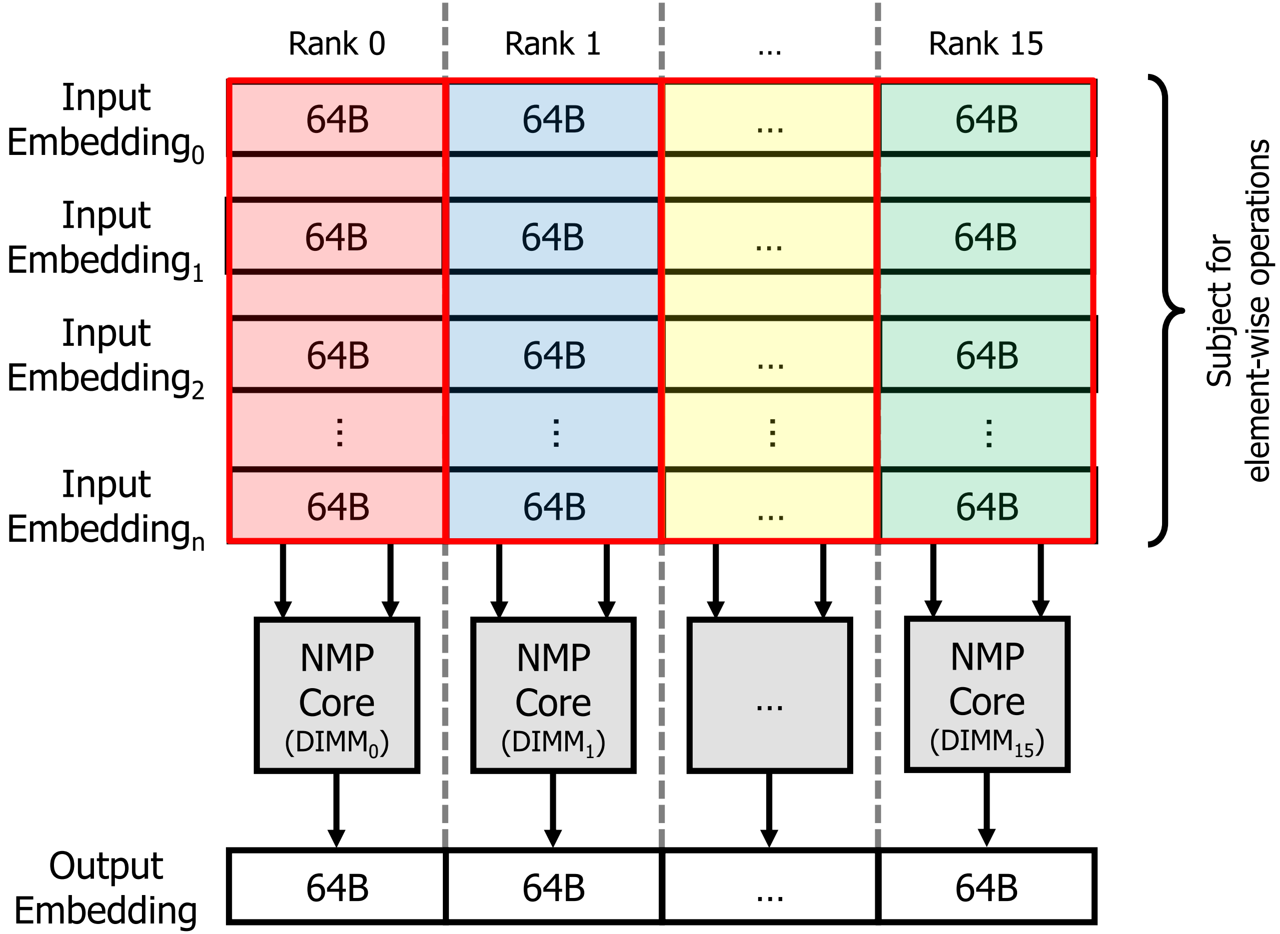
Leverage rank-level parallelism for maximal bandwidth utilization



Mapping embedding tables in DRAMs

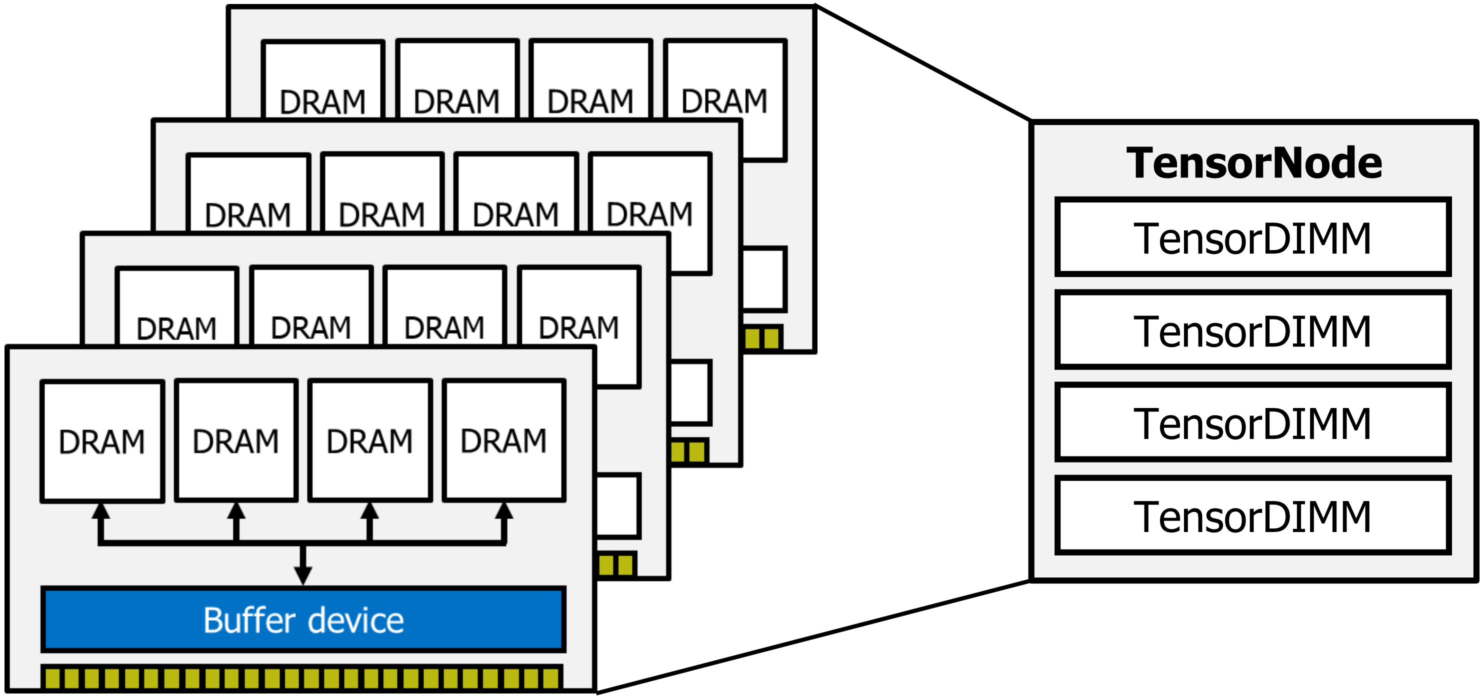
Leverage rank-level parallelism for maximal bandwidth utilization





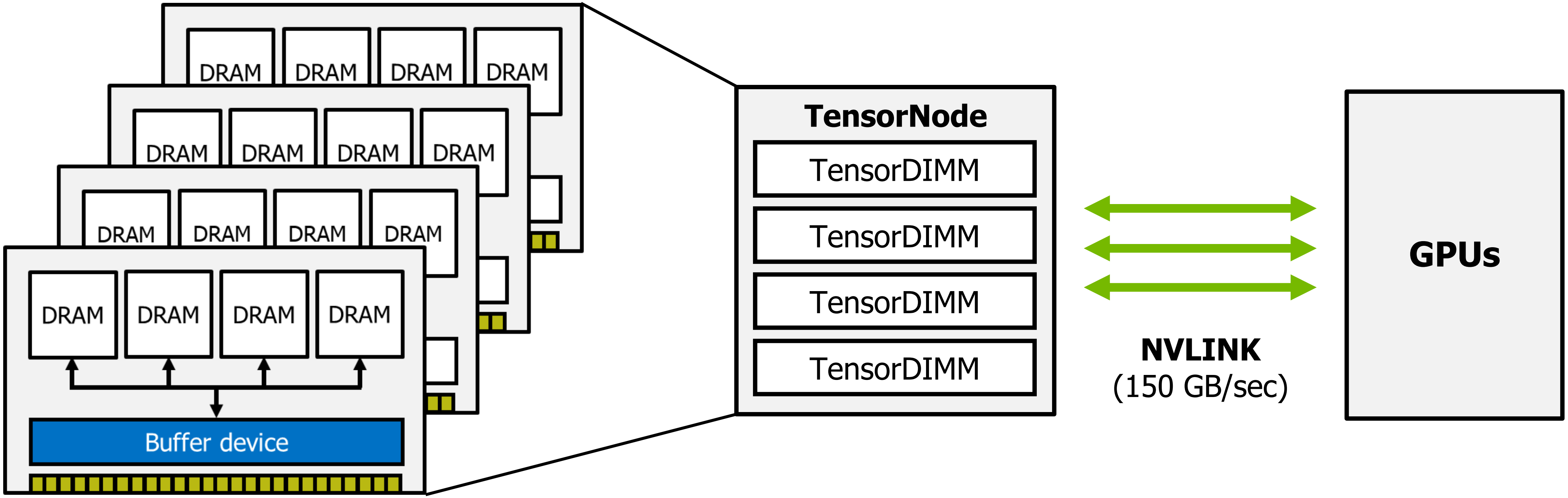
Tensor“Node” using TensorDIMMs

A pooled memory architecture aggregated with multiple TensorDIMMs



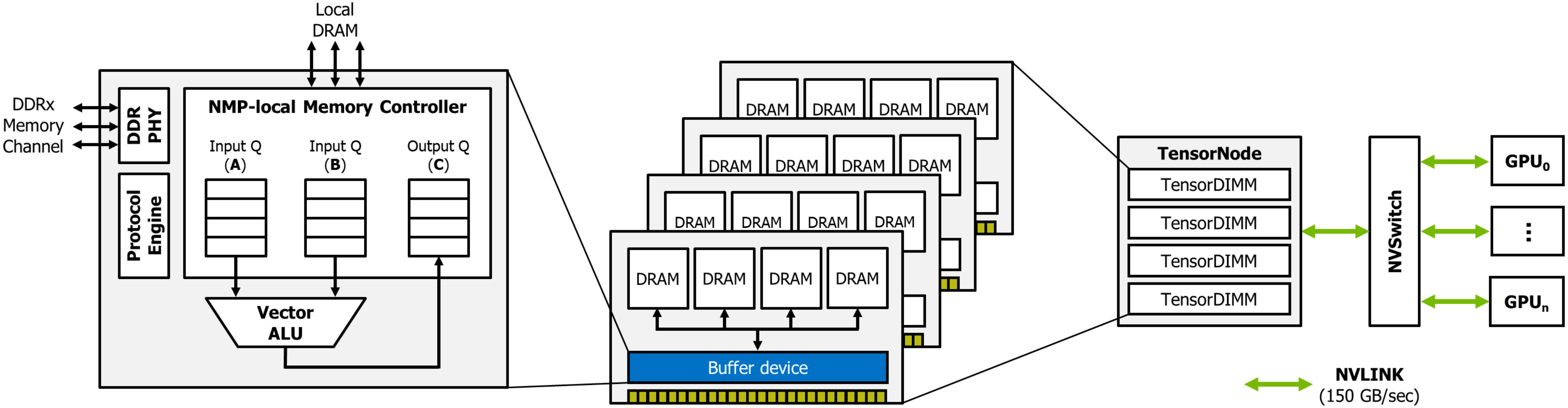
TensorNode as "remote" memory pools

Utilize high-speed links (e.g. NVLINK) for inter-device communication



Putting everything together

A platform for scalable expansion of both memory bandwidth and capacity



(a) NMP core

(b) TensorDIMM

(c) System architecture with TensorNode

Addresses the **memory bandwidth** challenge

Addresses the **memory capacity** challenge

Addresses the **compute** & **communication** challenges

Evaluation

Evaluation methodology

Combination of cycle-level simulation and emulation on real ML systems

- ❑ Cycle-level DRAM simulator (Ramulator*)

- ❑ Proof-of-concept software prototype on real ML systems (NVIDIA DGX-1V)

* Kim et al., "Ramulator: A Fast and Extensible DRAM Simulator", IEEE Computer Architecture Letters, 2015

Evaluation methodology

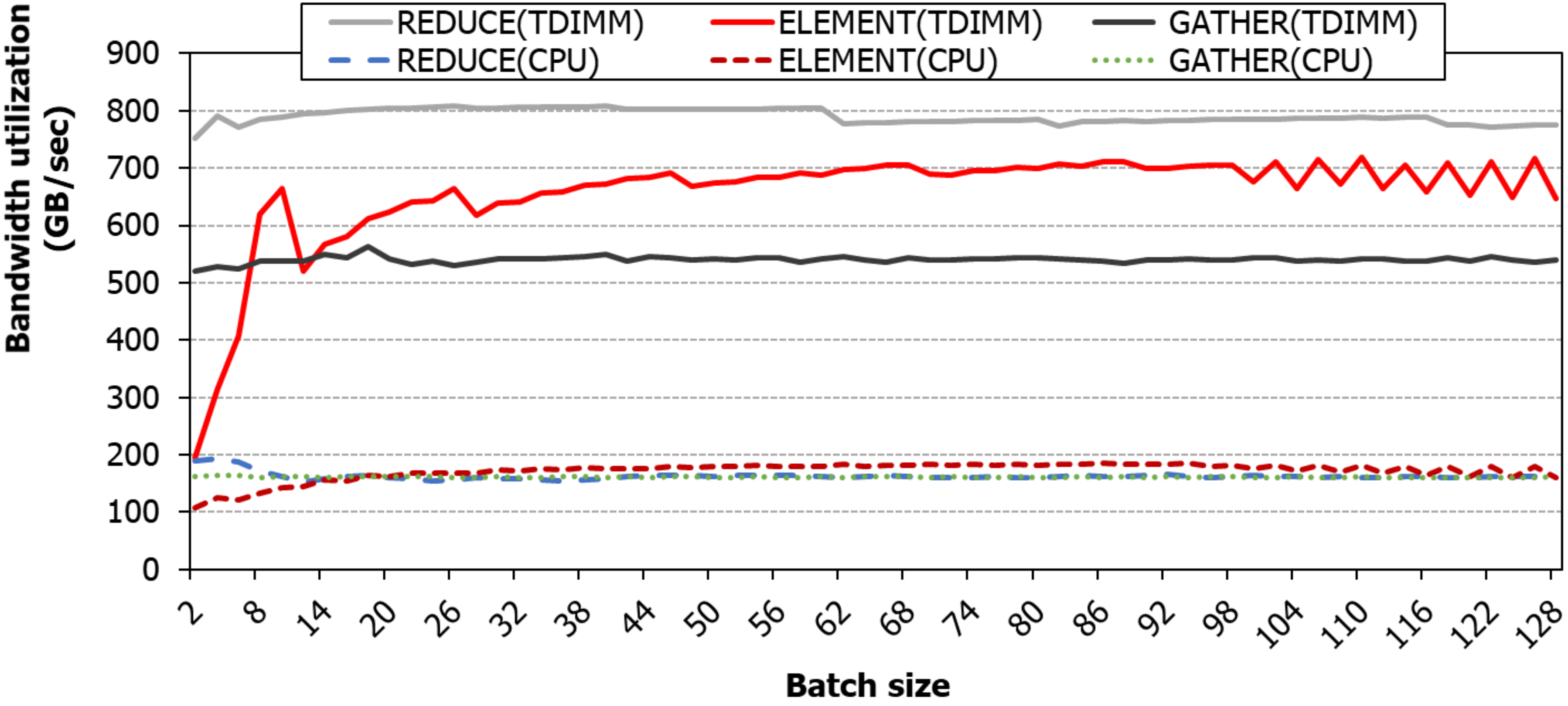
Combination of cycle-level simulation and emulation on real ML systems

- Cycle-level DRAM simulator (Ramulator*)
  - Memory bandwidth for embedding gathers/reductions under our address mapping
- Proof-of-concept software prototype on real ML systems (NVIDIA DGX-1V)

* Kim et al., “Ramulator: A Fast and Extensible DRAM Simulator”, IEEE Computer Architecture Letters, 2015

Memory bandwidth utilization

Effective bandwidth scales proportional to number of ranks (avg 4x↑)



KAIST

Evaluation methodology

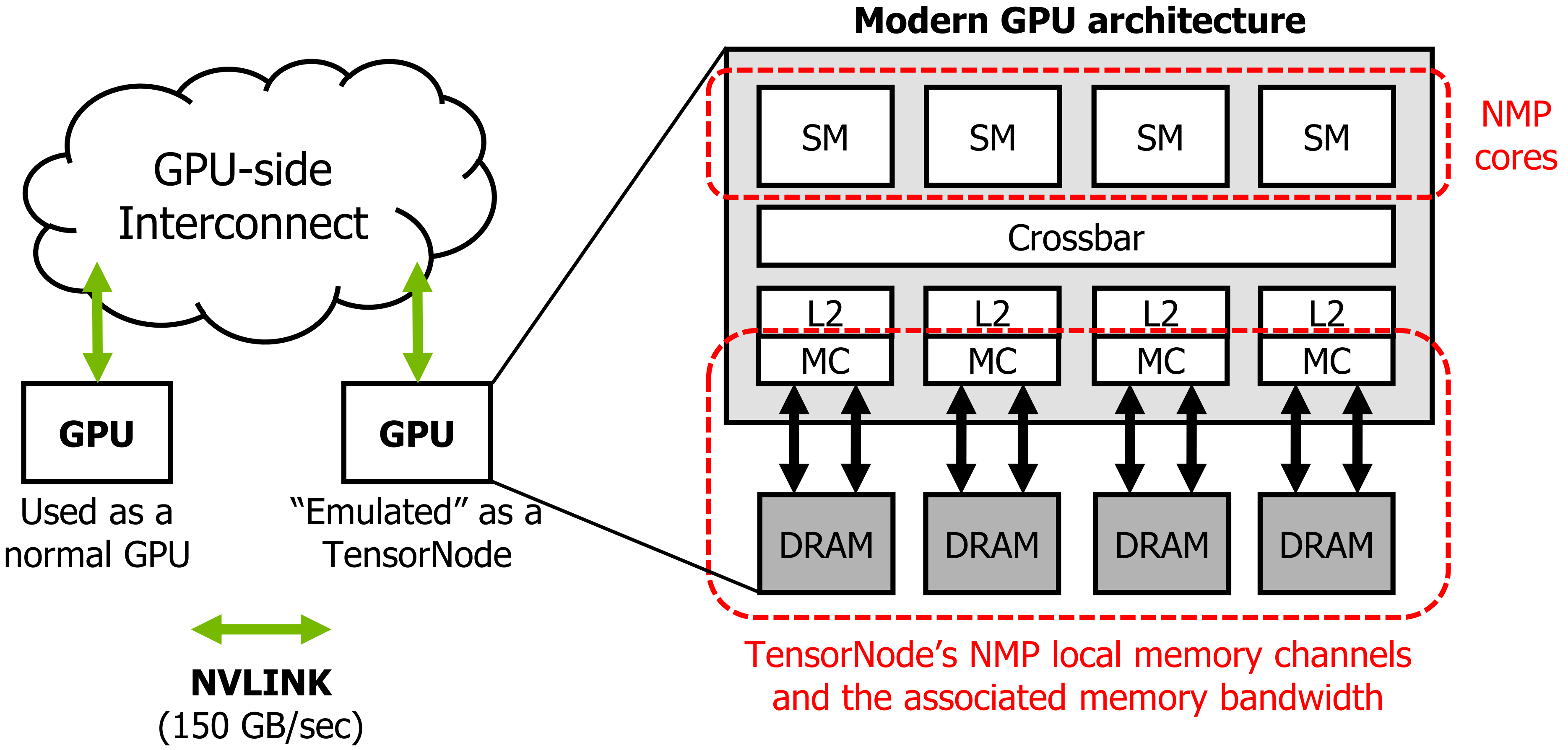
Combination of cycle-level simulation and emulation on real ML systems

- Cycle-level DRAM simulator (Ramulator*)
  - Memory bandwidth for embedding gathers/reductions under our address mapping
- Proof-of-concept software prototype on real ML systems (NVIDIA DGX-1V)
  - Intel’s Math Kernel Library (MKL)
  - NVIDIA cuDNN / cuBLAS
  - In-house CUDA implementation of other layers
  - NVIDIA DGX-1V
    - Eight NVIDIA V100 GPUs
    - Two Intel Xeon E5-2698 v4

* Kim et al., “Ramulator: A Fast and Extensible DRAM Simulator”, IEEE Computer Architecture Letters, 2015

TensorNode system modeling

A proof-of-concept software prototype to emulate TensorDIMM



Evaluation methodology

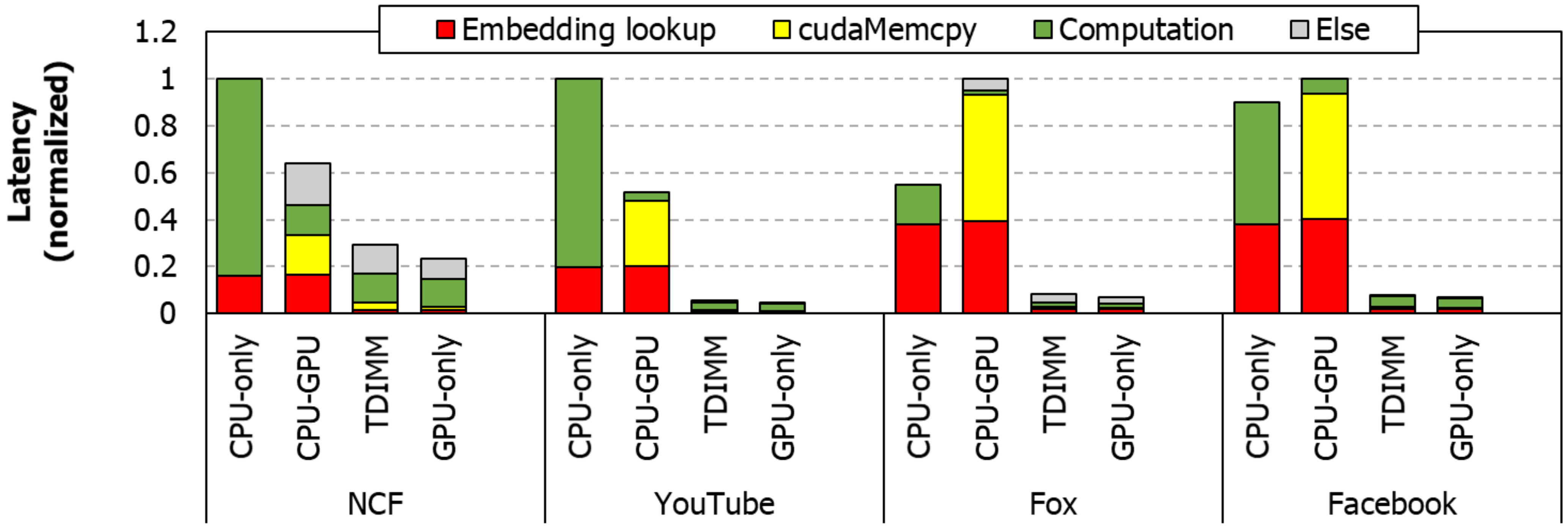
A proof-of-concept software prototype to emulate TensorDIMM

Four system design points

- CPU-only
- Hybrid CPU-GPU
- TensorDIMM (ours)
- GPU-only (oracle)

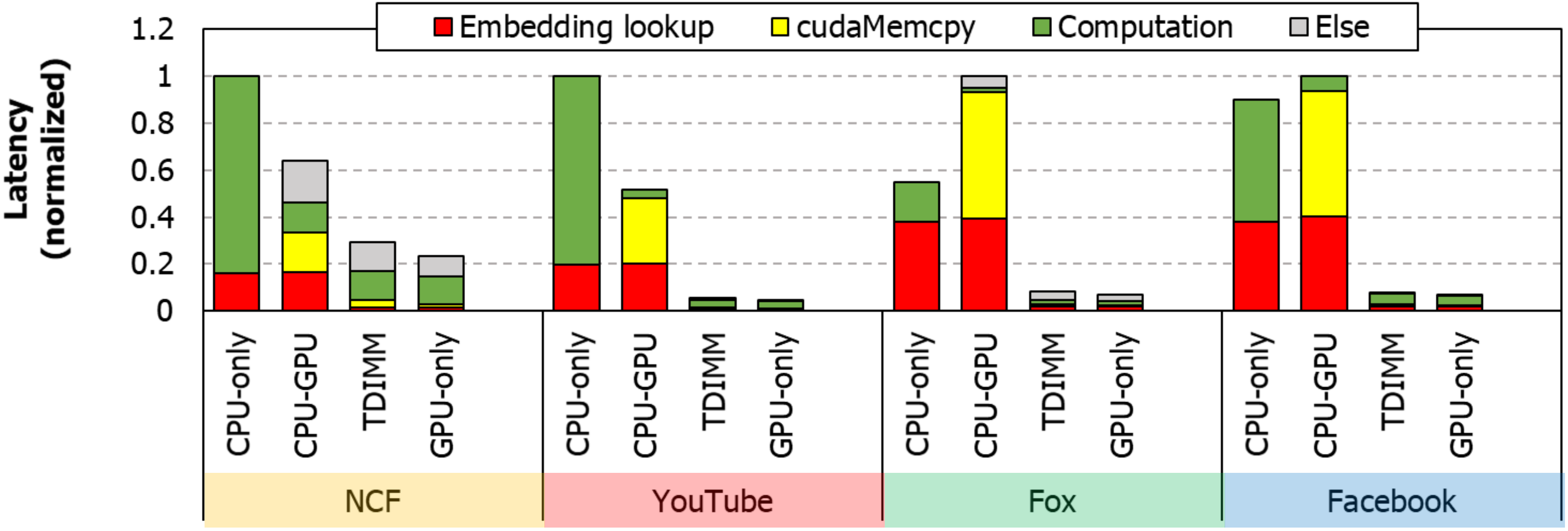
Latency breakdown

TensorDIMM helps reduce both embedding/MLP latency



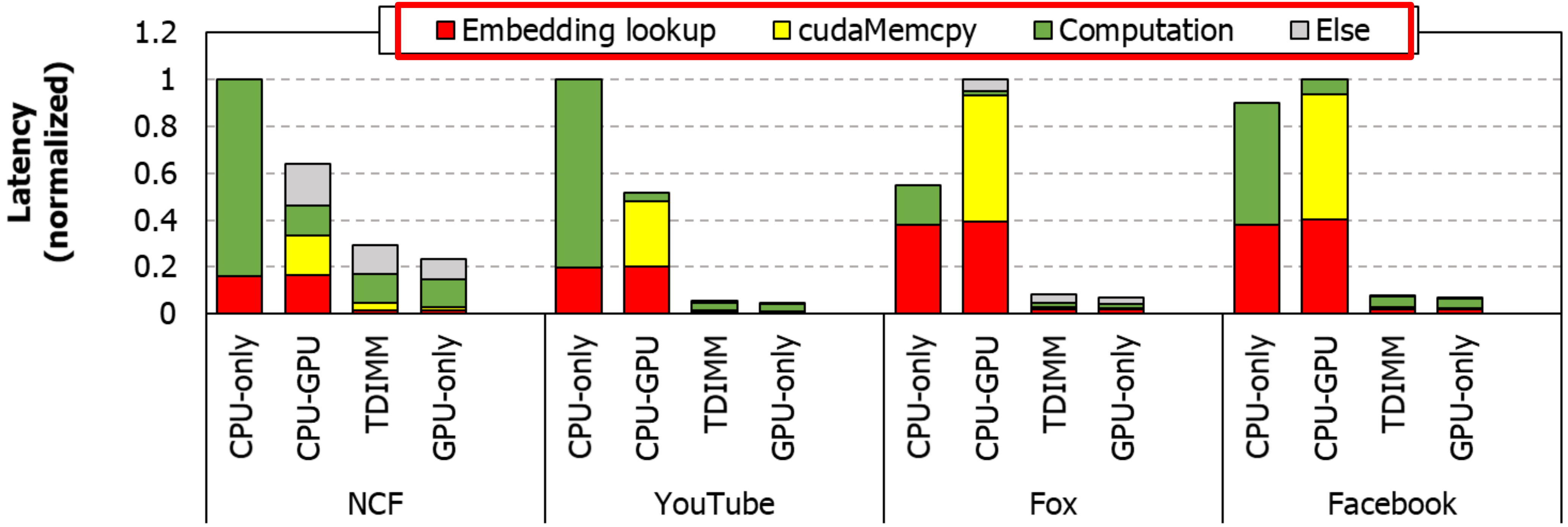
Latency breakdown

TensorDIMM helps reduce both embedding/MLP latency



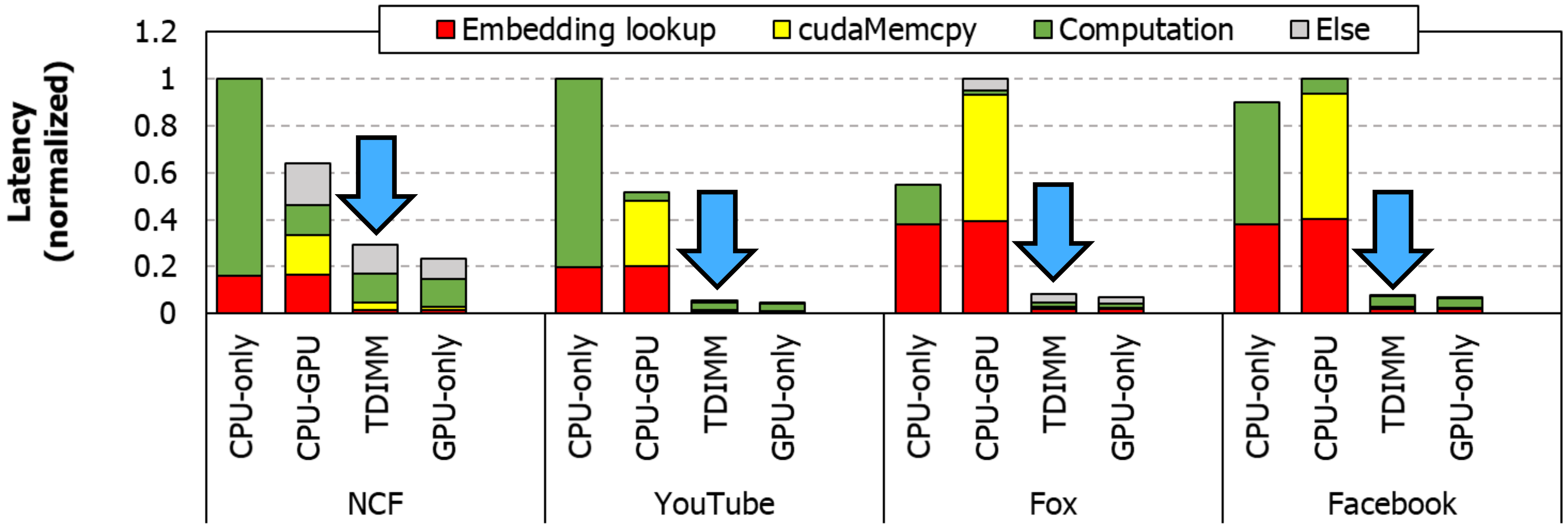
Latency breakdown

TensorDIMM helps reduce both embedding/MLP latency



Latency breakdown

TensorDIMM achieves overall 6-9x speedup against the baselines



TensorDIMM:
A Near-Memory Processing Architecture for Sparse Embedding Layers

The **"first"** architectural solution tackling sparse embedding layers

A **"practical"** near-memory processing solution for an important AI workload

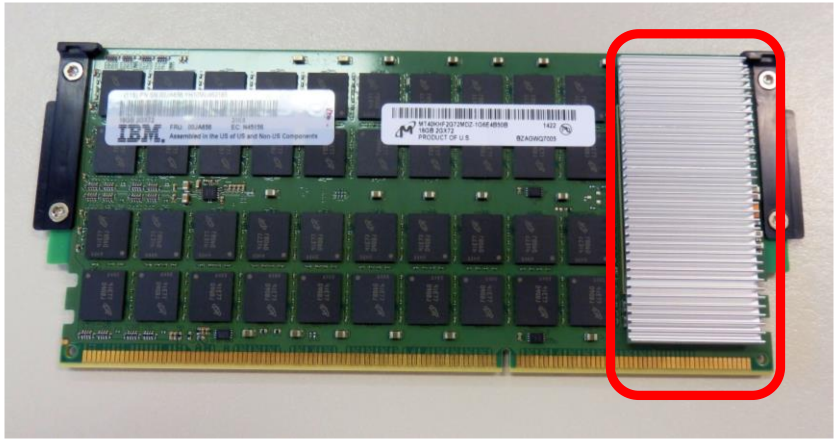
Average **"6~9x"** performance improvement on state-of-the-art DNN-based recommendation models

Questions?

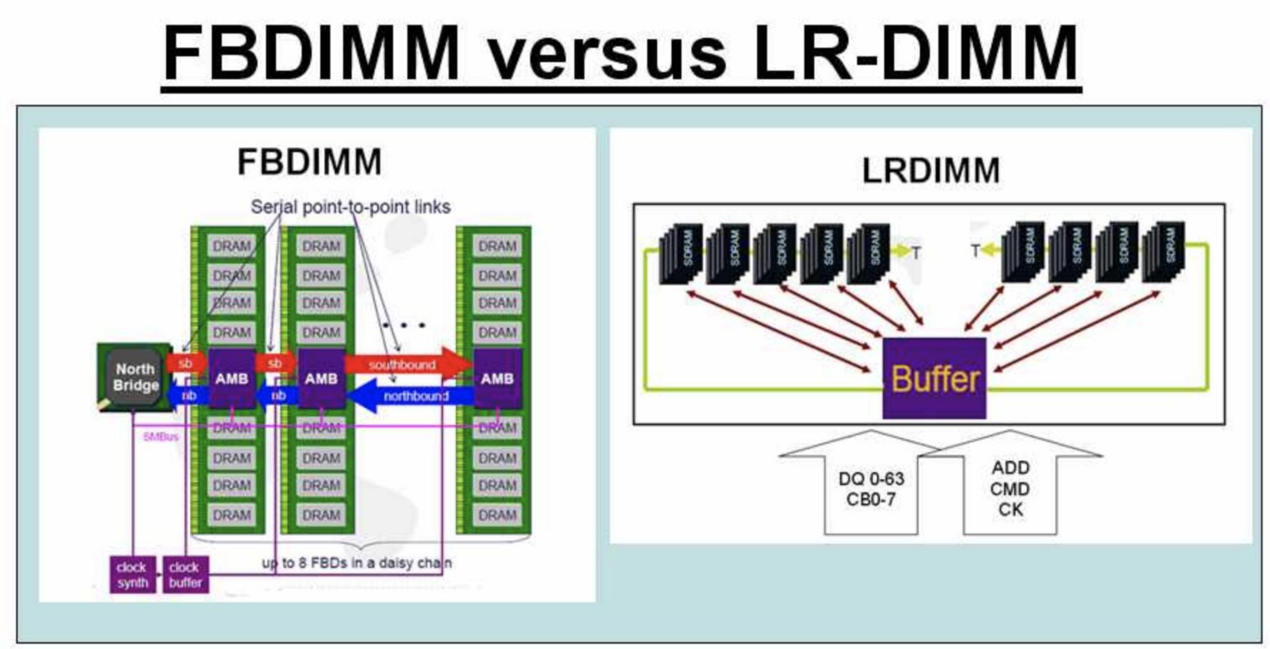
Backup Slides

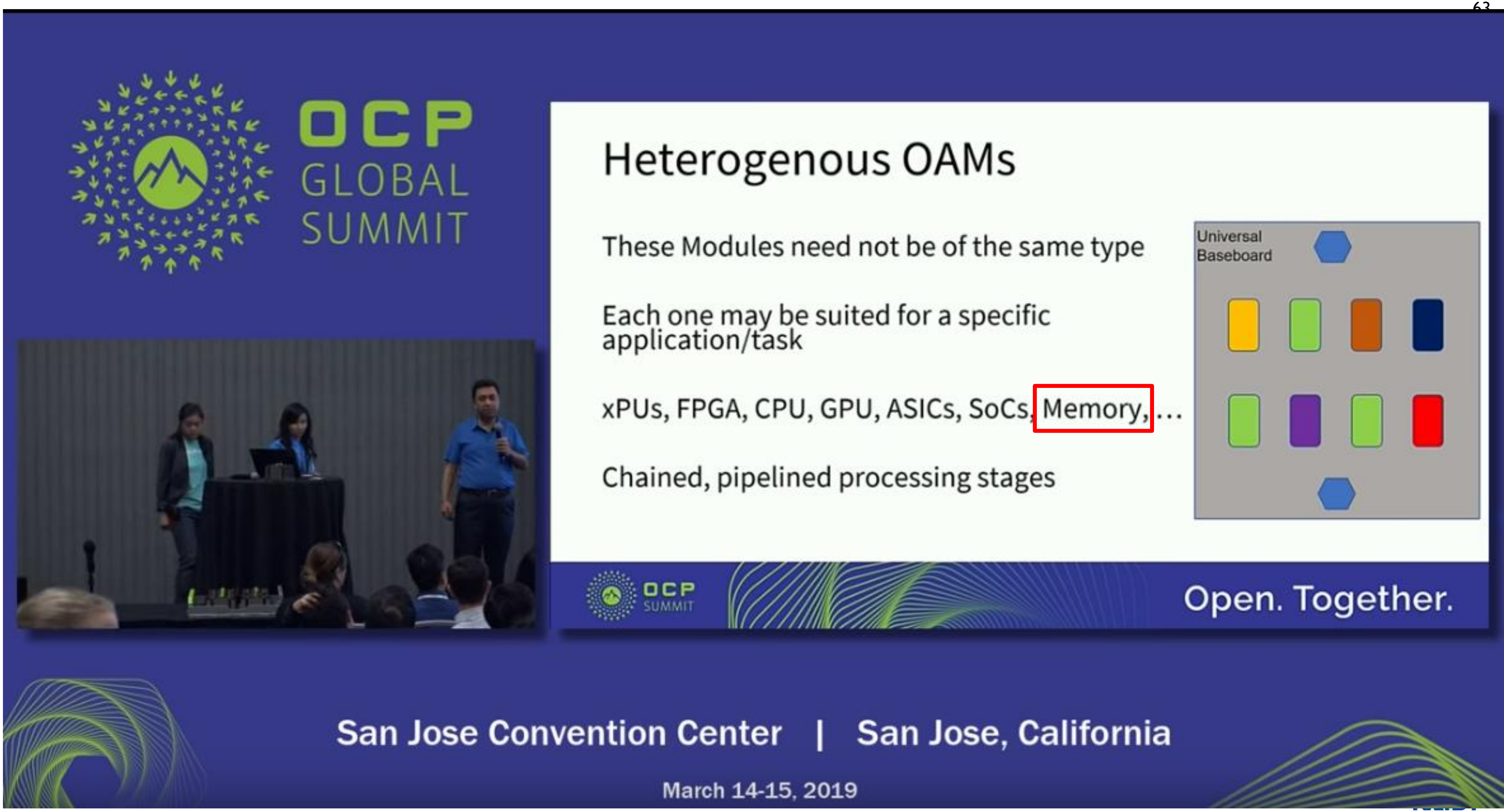
TensorDIMM design overheads

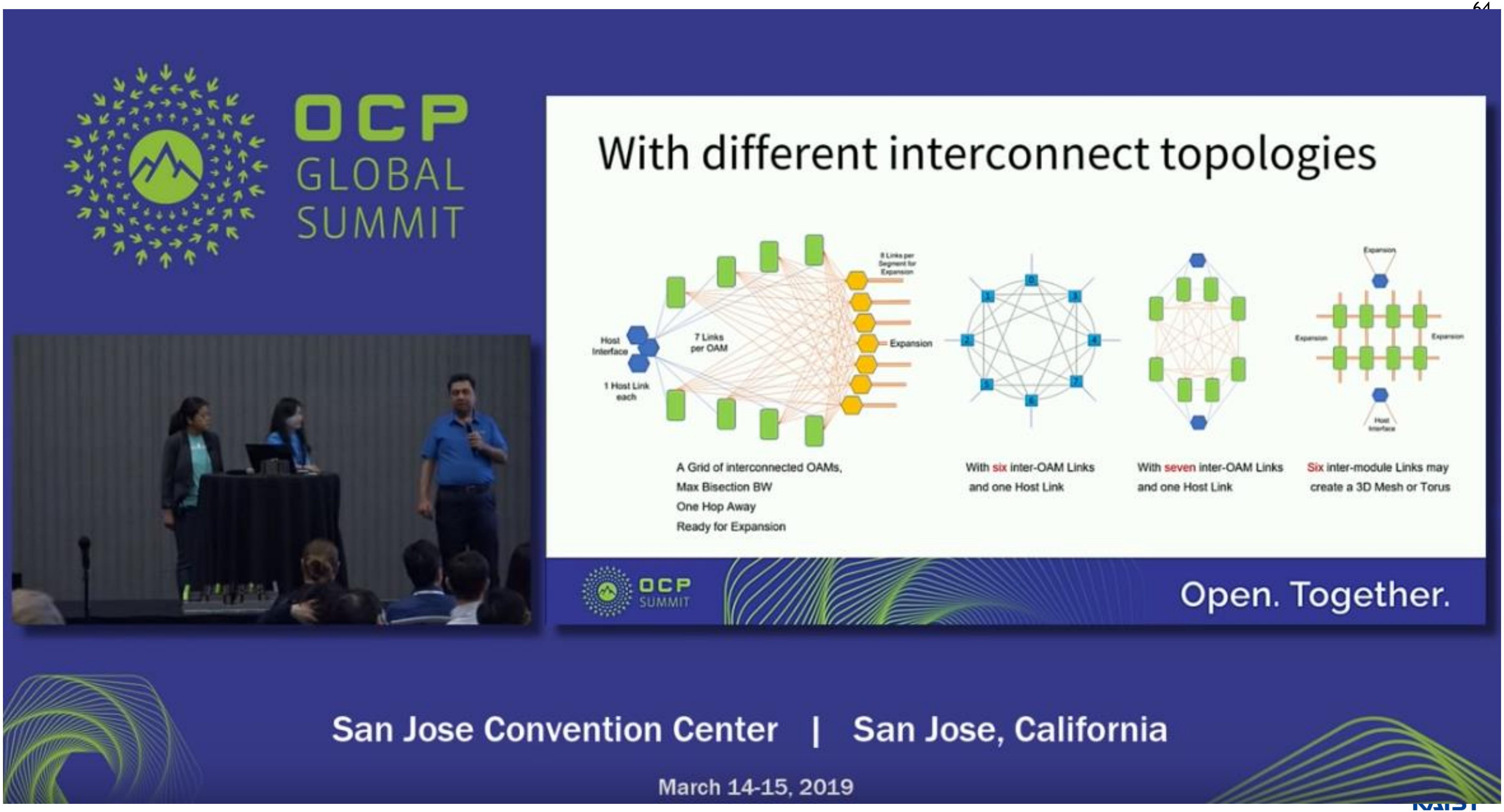
It's not free, but adding custom logics within DIMM has been done before



IBM centaur DIMM




OCP GLOBAL SUMMIT
Heterogenous OAMs
These Modules need not be of the same type
Each one may be suited for a specific application/task
xPUs, FPGA, CPU, GPU, ASICs, SoCs, Memory, ...
Chained, pipelined processing stages
Universal Baseboard
OCP SUMMIT
Open. Together.
San Jose Convention Center | San Jose, California
March 14-15, 2019


OCP
GLOBAL
SUMMIT
With different interconnect topologies
Host Interface
7 Links per OAM
1 Host Link each
Expansion
A Grid of interconnected OAMs,
Max Bisection BW
One Hop Away
Ready for Expansion
With six inter-OAM Links
and one Host Link
With seven inter-OAM Links
and one Host Link
Six inter-module Links may
create a 3D Mesh or Torus
OCP
SUMMIT
Open. Together.
San Jose Convention Center | San Jose, California
March 14-15, 2019

OCP
GLOBAL
SUMMIT
OCP Accelerator Module Spec
Support both 12V and 48V as input
Up to 350w(12V) and up to 700w(48V) TDP
102mm x 165mm
Support single or multiple ASIC(s) per Module
Up to eight x16 Links (Host + inter-module Links)
Support one or two x16 High speed link(s) to Host
Up to seven x16 high speed interconnect links
Expect to support up to 450W (air-cooled) and 700W (liquid-cooled)
Up to 8* Modules per system
System management and debug interfaces
OCP SUMMIT
Open. Together.
San Jose Convention Center | San Jose, California
March 14-15, 2019