Samsung PIM/PNM for Transformer based AI

: Energy Efficiency on PIM/PNM Cluster

Jin Hyun Kim, Yuhwan Ro, Jinin So, Sukhan Lee, Shin-haeng Kang, YeonGon Cho, Hyeonsu Kim, Byeongho Kim, Kyungsoo Kim ,Sangsoo Park, Jin-Seong Kim, Sanghoon Cha, Won-Jo Lee, Jin Jung, Jong-Geon Lee, Jieun Lee, JoonHo Song, Seungwon Lee, Jeonghyeon Cho, Jaehoon Yu, and Kyomin Sohn

Samsung Semiconductor

Index

01

Memory Trends and Bottleneck in HPC/AI

02

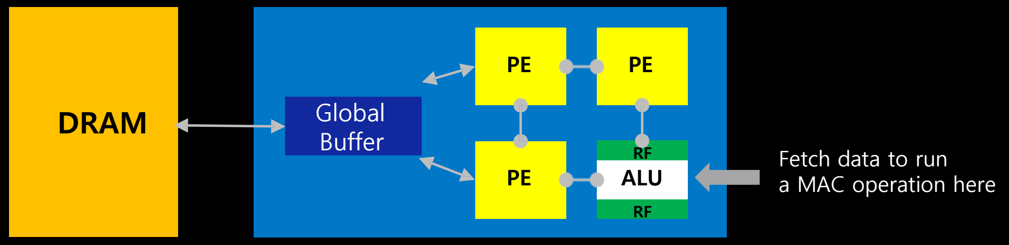
Samsung Memory AI Solution

- Cache Level LLC DRAM, **HBM-PIM (Processing-in-Memory)**
- Memory Expander, **CXL-PNM (Processing-near-Memory)** , CXL-SSD
- Storage : SmartSSD, PBSSD

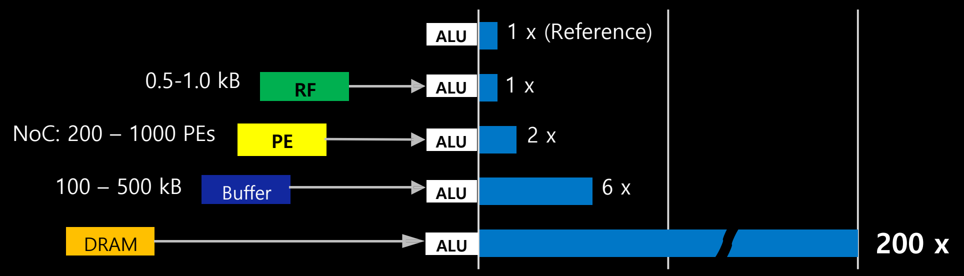
PIM/PNM on Memory Hierarchy and Energy Reduction

- Data movement consumes a lot of energy even for simple computation
- PIM/PNM technology can reduce energy consumption within a typical memory hierarchy
- PIM/PNM device for each layer must meet specific requirements: bandwidth(BW), power, capacity, etc.

Energy Cost of Data Transfer

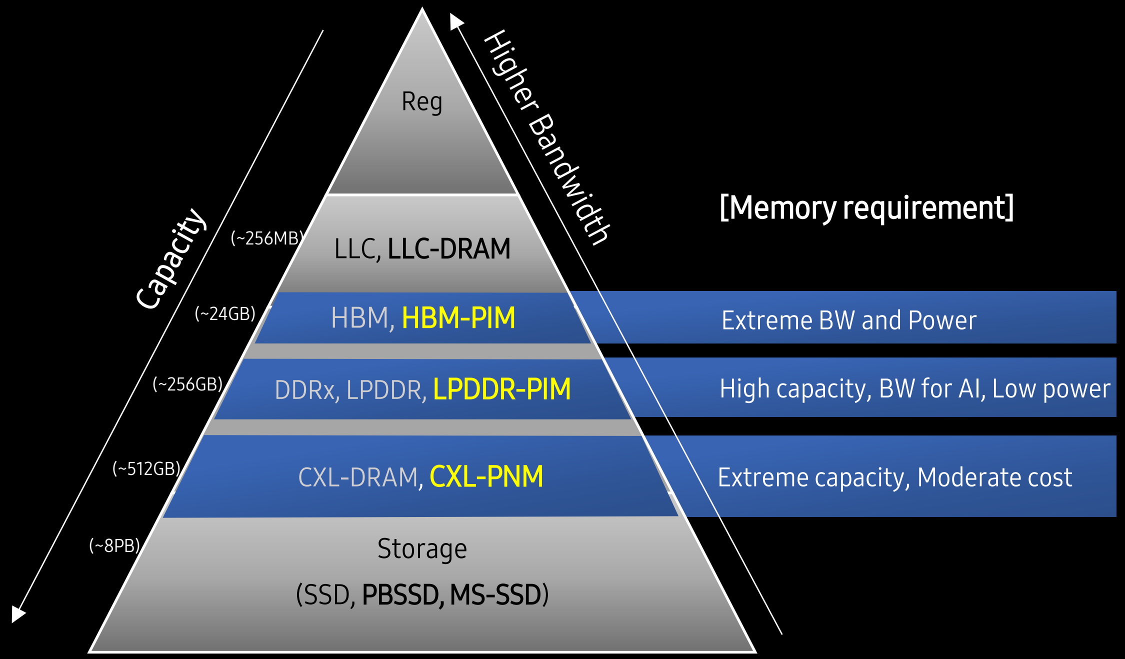


Normalized Energy Cost*



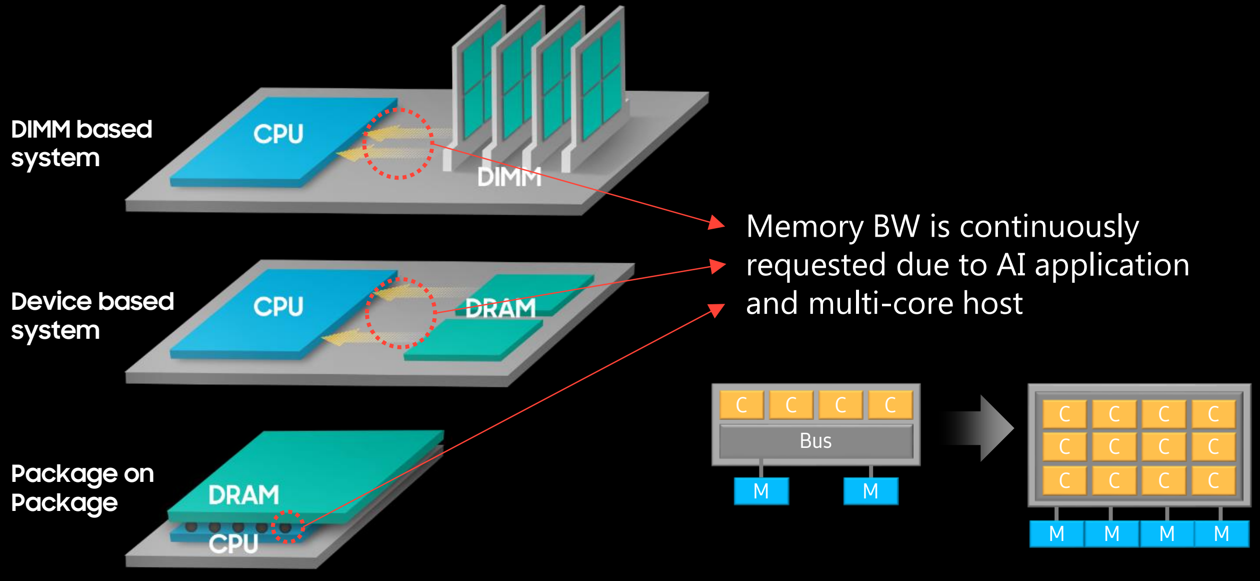
Source: * Y.-H. Eyeriss, 2016 ISCA

Computing Coupled Memory Hierarchy



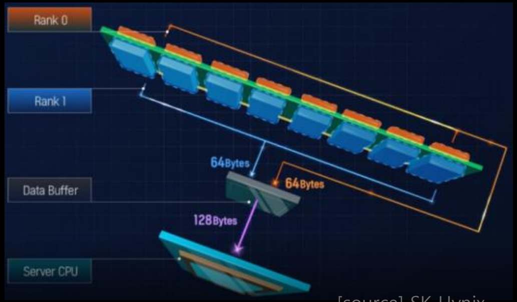
Traditional Approach to Overcome Memory Bottleneck

- While there are various methods to increase BW, it is difficult to achieve a dramatic increase
  - Limited by # of PCB wires, # of CPU ball, and thermal constraints
- Increasing # of the balls and PCB wires is physically and thermally bounded and is a expensive solution
  - MCR-DIMM, PAM3/4 signaling IO, 2K-IO or 3D stacking



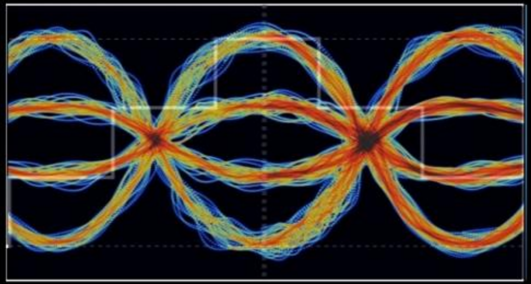
Memory BW is continuously requested due to AI application and multi-core host

MCR-DIMM(Multiplexer Combined Ranks DIMM)



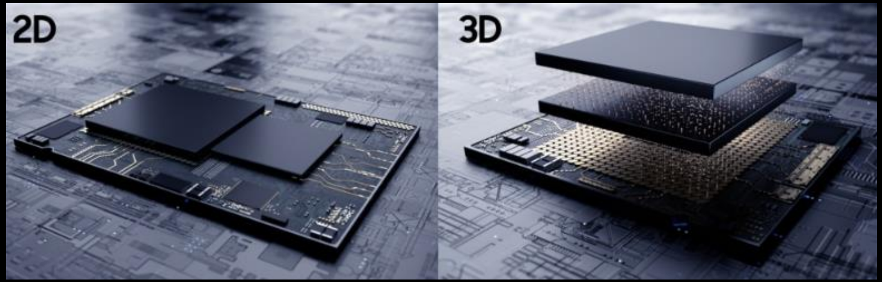
[source] SK Hynix

PAM 3/4(Pulse Amplitude Modulation)



[source] Micron

2K-IO 2D or 3D stacking

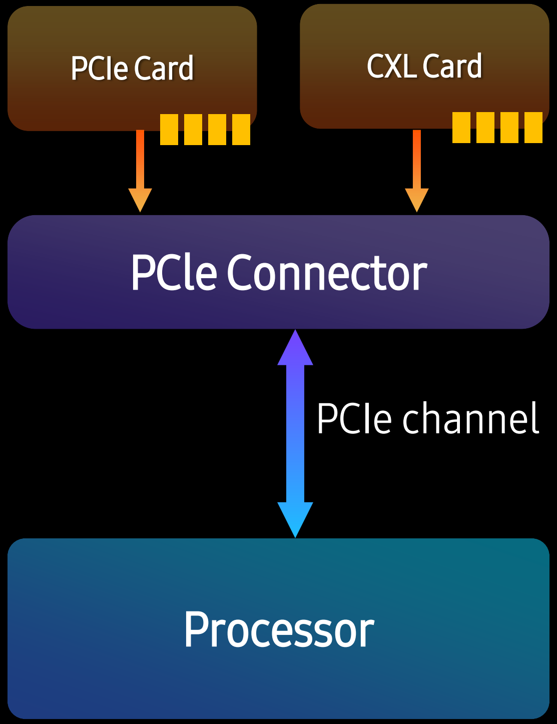


[source] Samsung

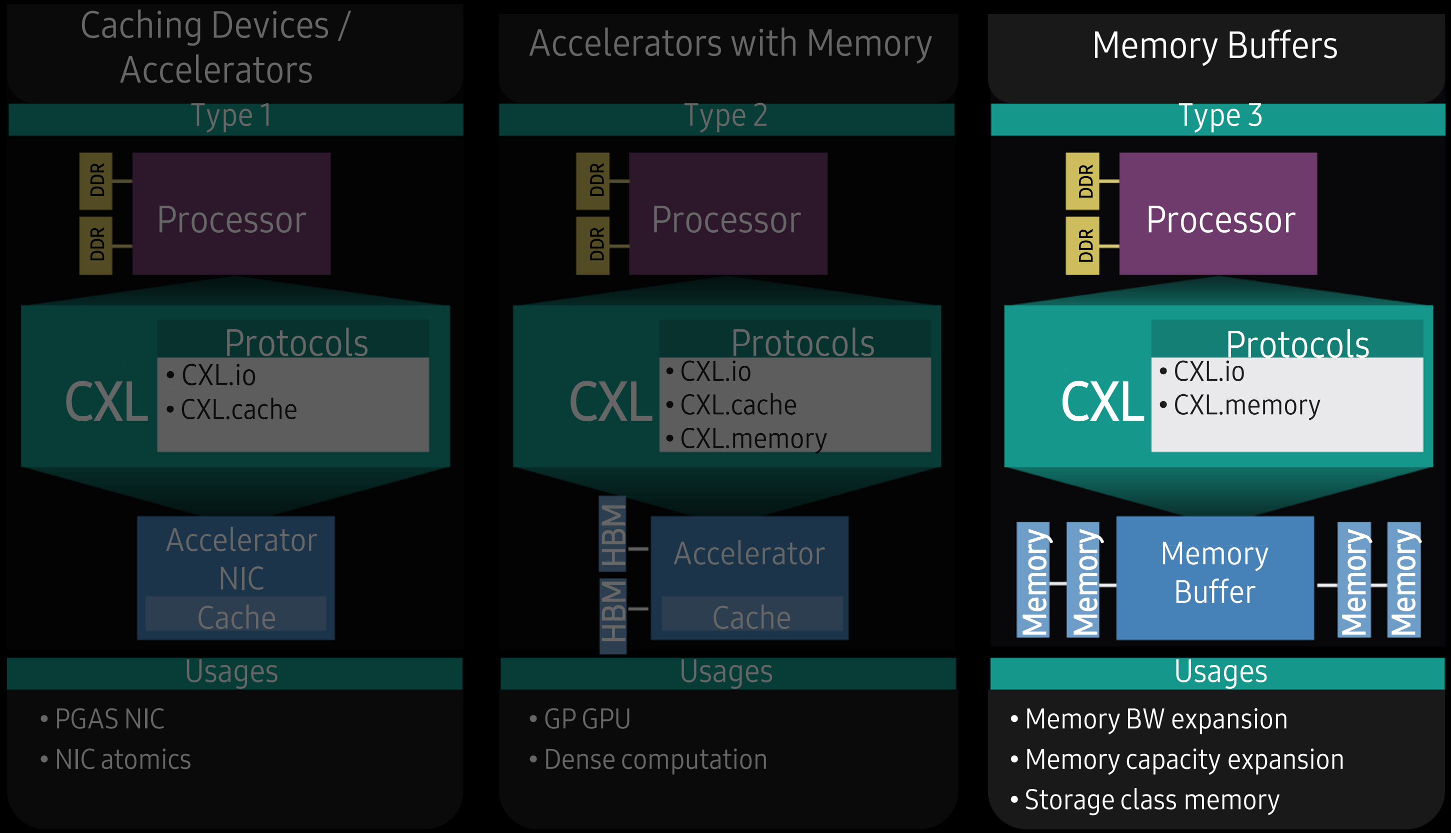
CXL solution, Trend to Pay Attention to

- CXL is strong candidate for memory hierarchy to address performance and density
- Successful power-on of memory expander, SSD/pooling solutions are next big-thing

CXL is a high performance, low latency protocol that leverages PCIe physical layer

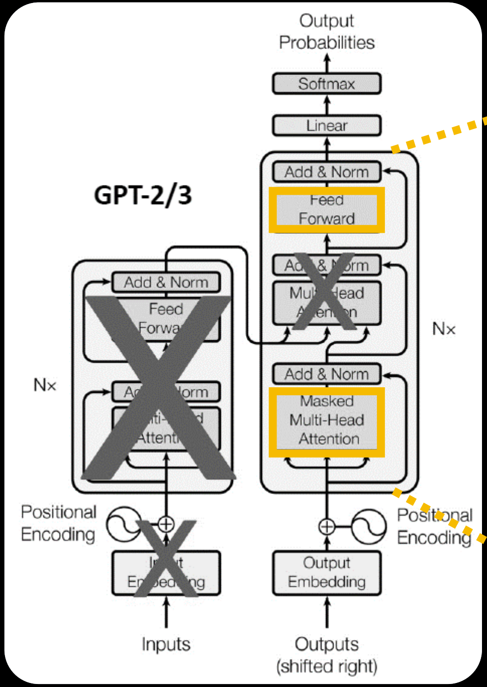


CXL is an open industry standard with broad industry support



Bottleneck in GPT: Linear Layers in Generation Stage

- Main target: transformer decoders used in ChatGPT, GPT-3
  - Linear layers in multi-head attention(MHA) and feed-forward networks(FFN)
- Focus on memory-bottleneck in Generation stage
  - Generation Stage shows poor performance with GPU due to its memory-bound & sequential characteristic



<Transformer>

[source] Google Transformer

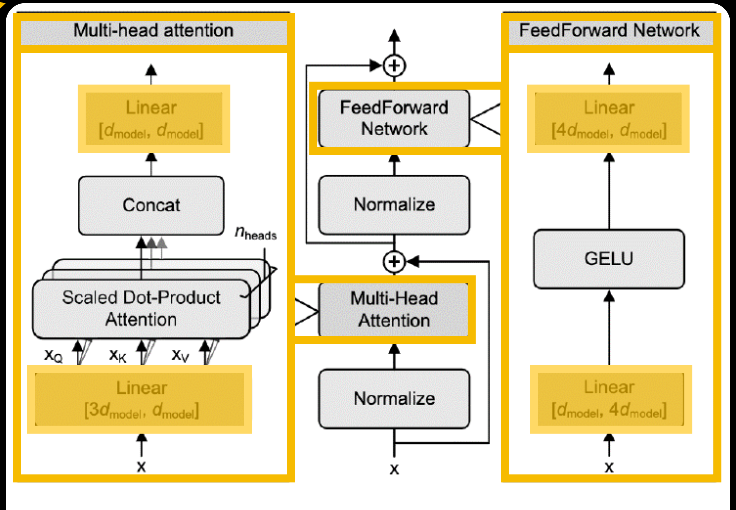
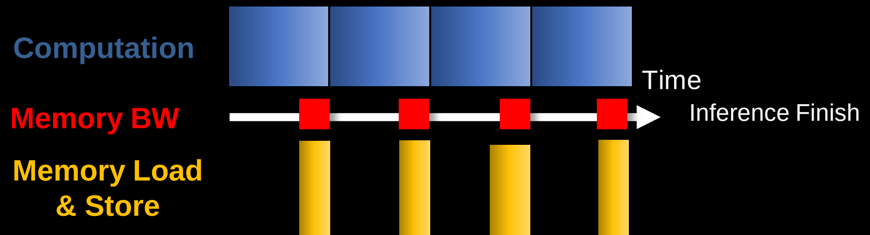
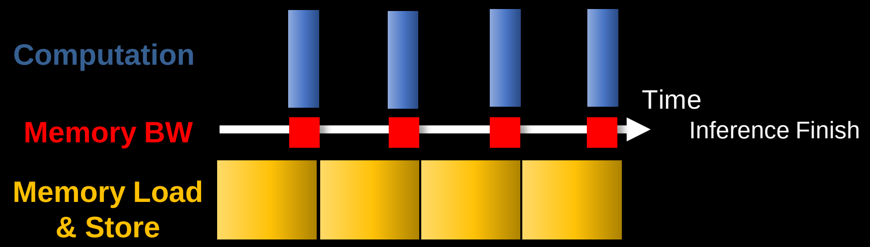


Fig. 10. Transformer layer incorporating multi-head attention and feed-forward network performing 4 major matrix multiplications.

<Four major matrix multiplication block>



<Summarization>

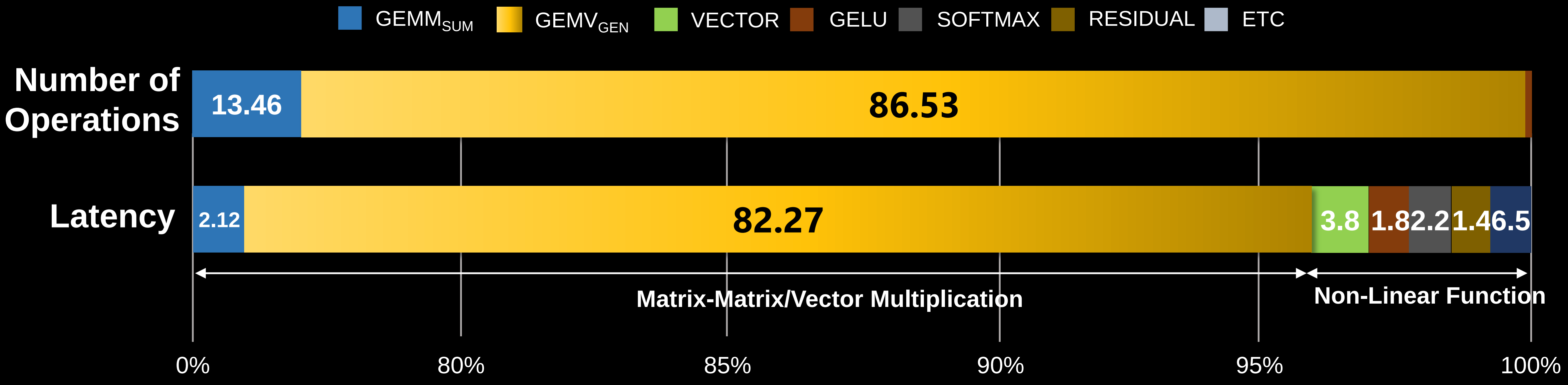


< Generation >

[source] Naver Clova

GPT Profiling Result

- GPT workload consists of Summarization(computing-bound) and Generation(memory–bound)
- GEMV portion can be 60–80% of total generation latency, which are the target of PIM/PNM



*Profiling result is measured in A100 System (DeepSpeed + GPT-J 6B, FP16, Input/Output token:7/46)
GPT-j: Google JAX framework

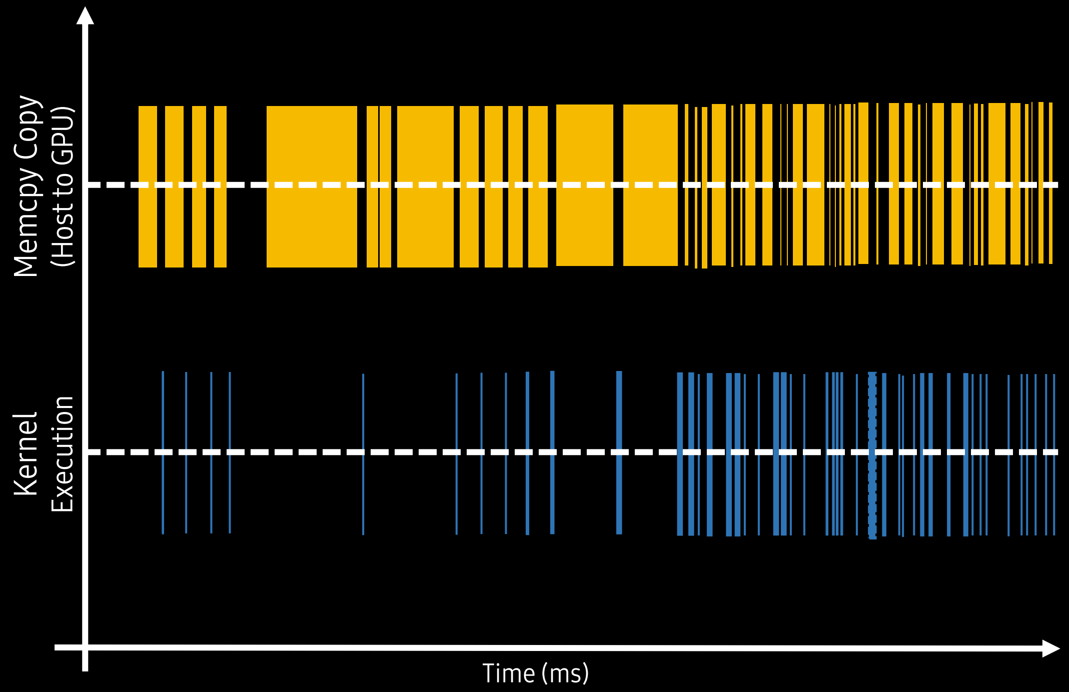
https://community.openai.com/t/how-does-chatgpt-have-such-massive-token-limit/25738/6

Stage	Computation	Latency
SUM	78.95 GFLOP	7.62 ms
GEN	11.28 GFLOP	6.58 ms

Utilization and Execution Time Breakdown

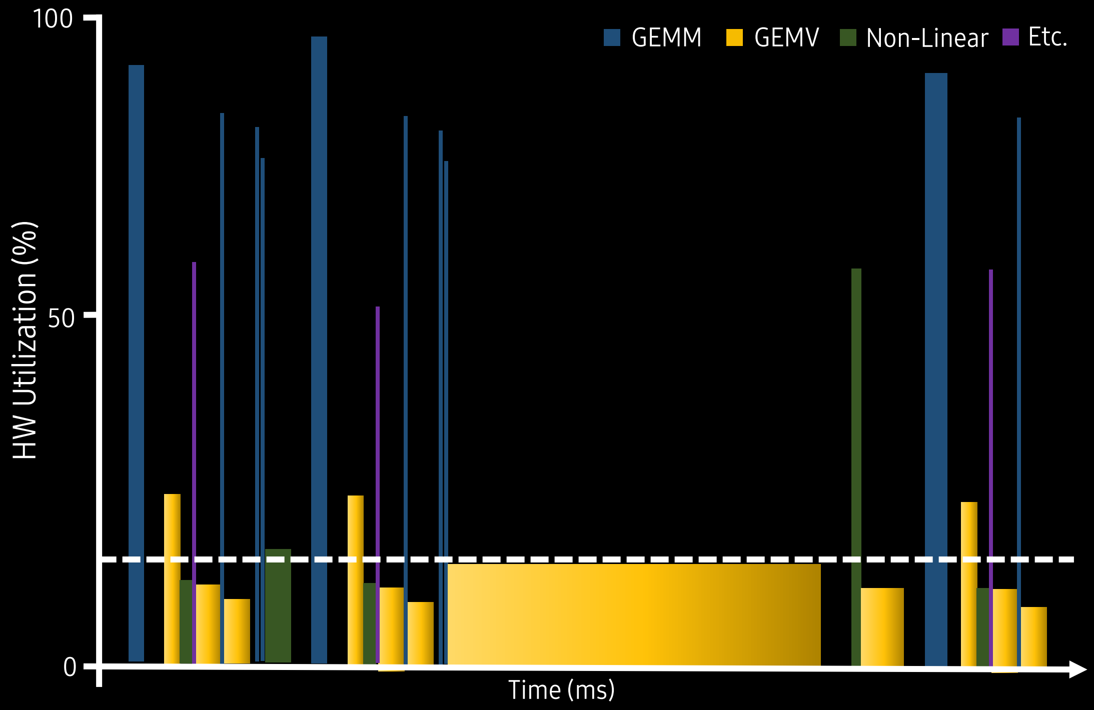
- Most of the execution time is spent for the memory copy from the host CPU memory to the GPU memory
- Utilization for performing GEMV operations (Generation stage) is seriously low, compared to GEMM
- As # of output tokens increases, GEMV operations dominate the inference time

Kernel Execution vs Memory Copy



*OPT-30B inference on Single A100 GPU (DeepSpeed, FP16 precision)

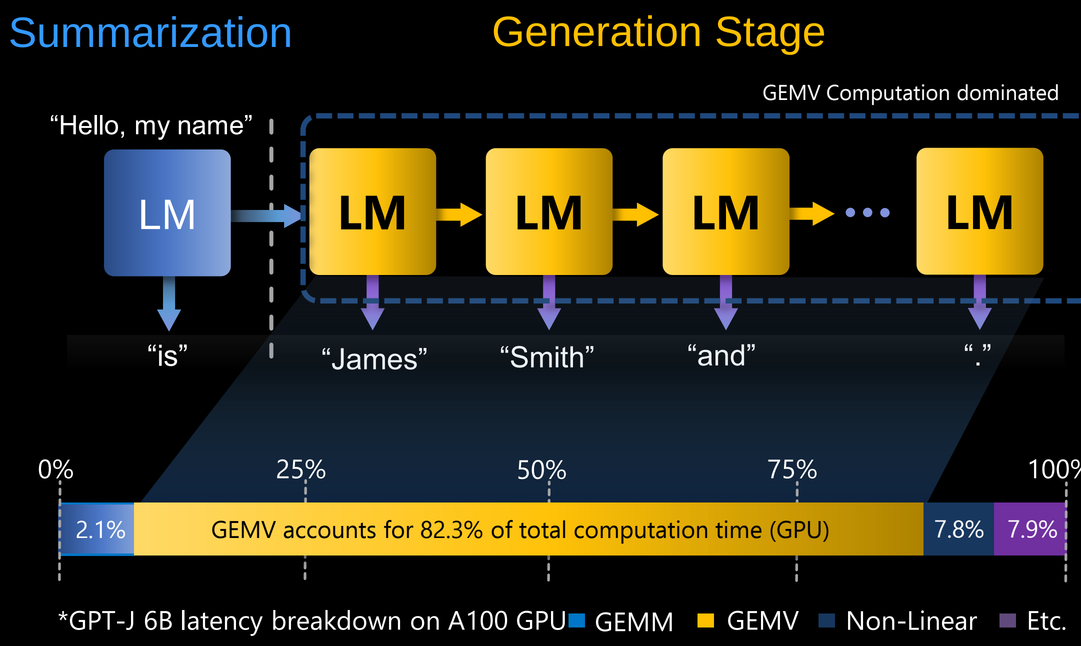
GPU HW Utilization



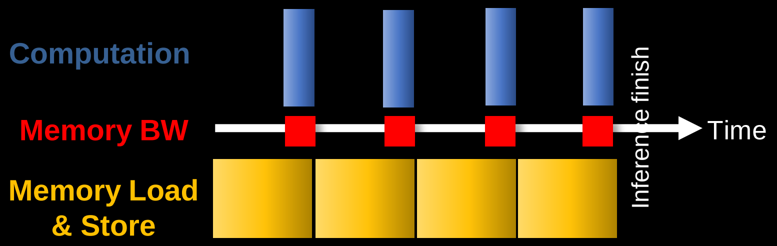
*GPT-J 6B profiling result (NVIDIA nsight system and compute)

Acceleration by PIM/PNM on Generation stage

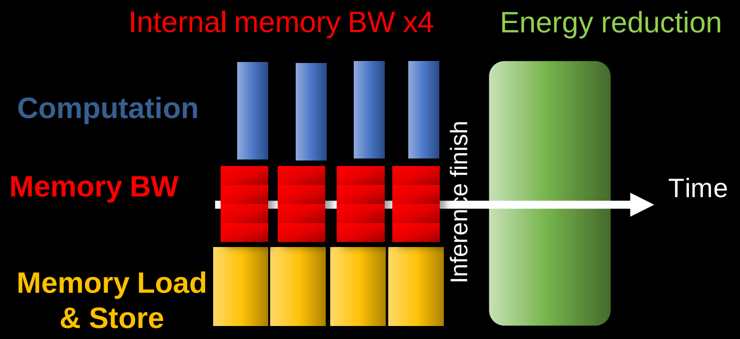
- Generation stage on GPT requires high capacity and bandwidth memory
- MHA and FFN can be fully offloaded to PIM/PNM, exploiting full bandwidth provided by PIM/PNM
- As a result, PIM/PNM can significantly reduce the time and energy spent on inference



*GPT-J 6B latency breakdown on A100 GPU



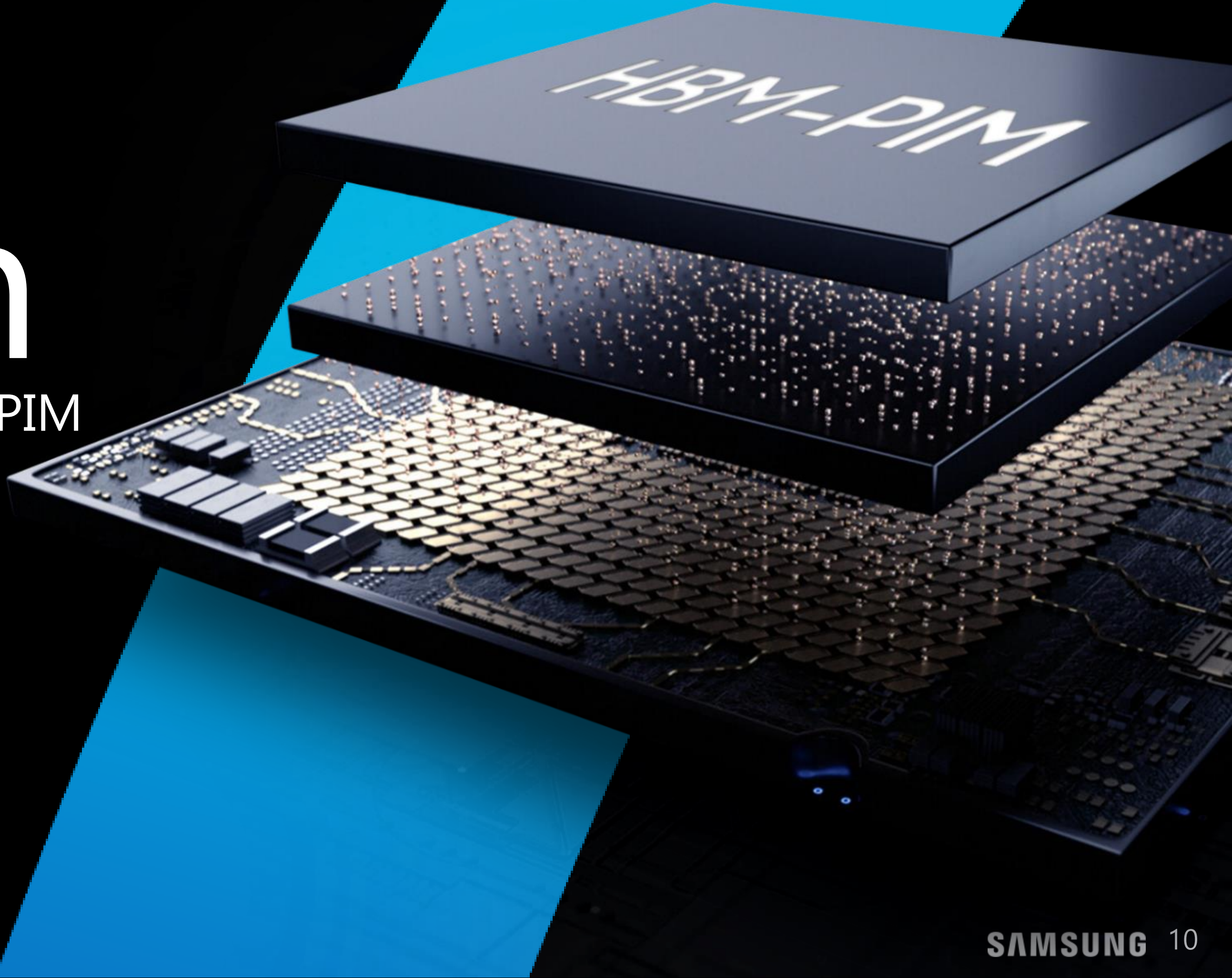
< Generation >



<Generation with PIM/PNM>

PIM Solution

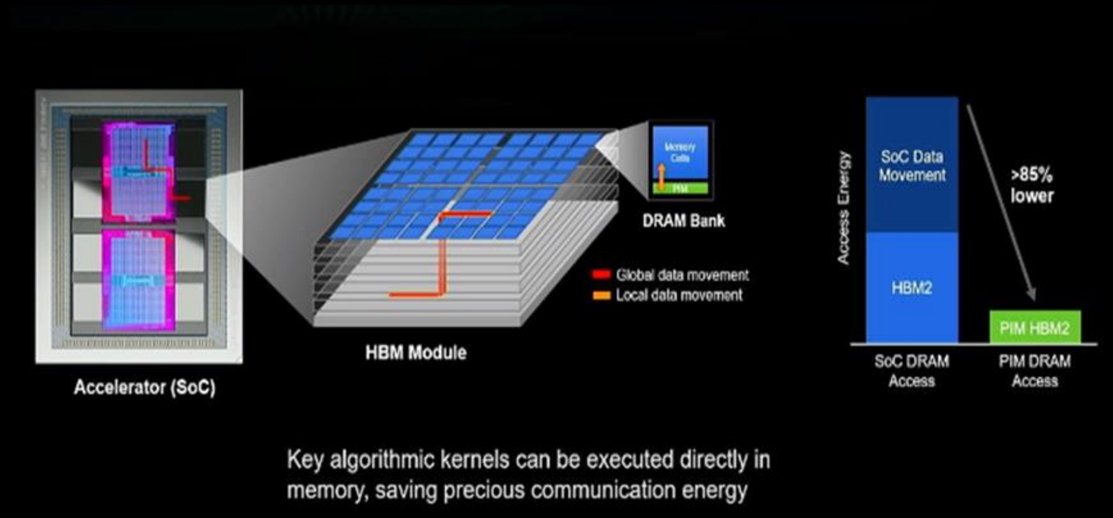
Redesigned to Advance AI : HBM-PIM / LPDDR-PIM



Energy Advantage of PIM on Generative AI

- Since OpenAI focuses on developing new AI technologies and pushing the boundaries of what can be done with AI, it is likely that they will explore the use of PIM technology in the future.
- In ISSCC 2023, AMD mentioned
  - Key algorithmic kernels can be executed directly in memory, saving precious communication energy
  - PIM can reduce energy by 85% compared with conventional HBMs

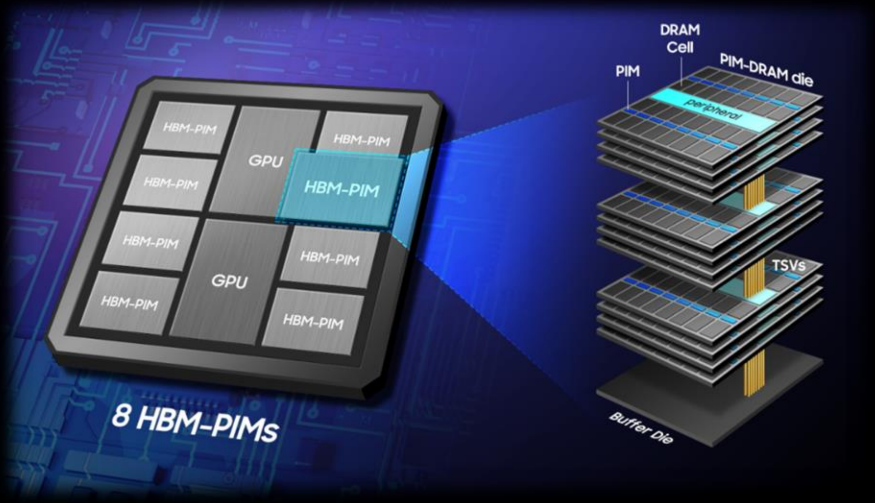
Processing-in-Memory



Key algorithmic kernels can be executed directly in memory, saving precious communication energy

[source] AMD ISSCC

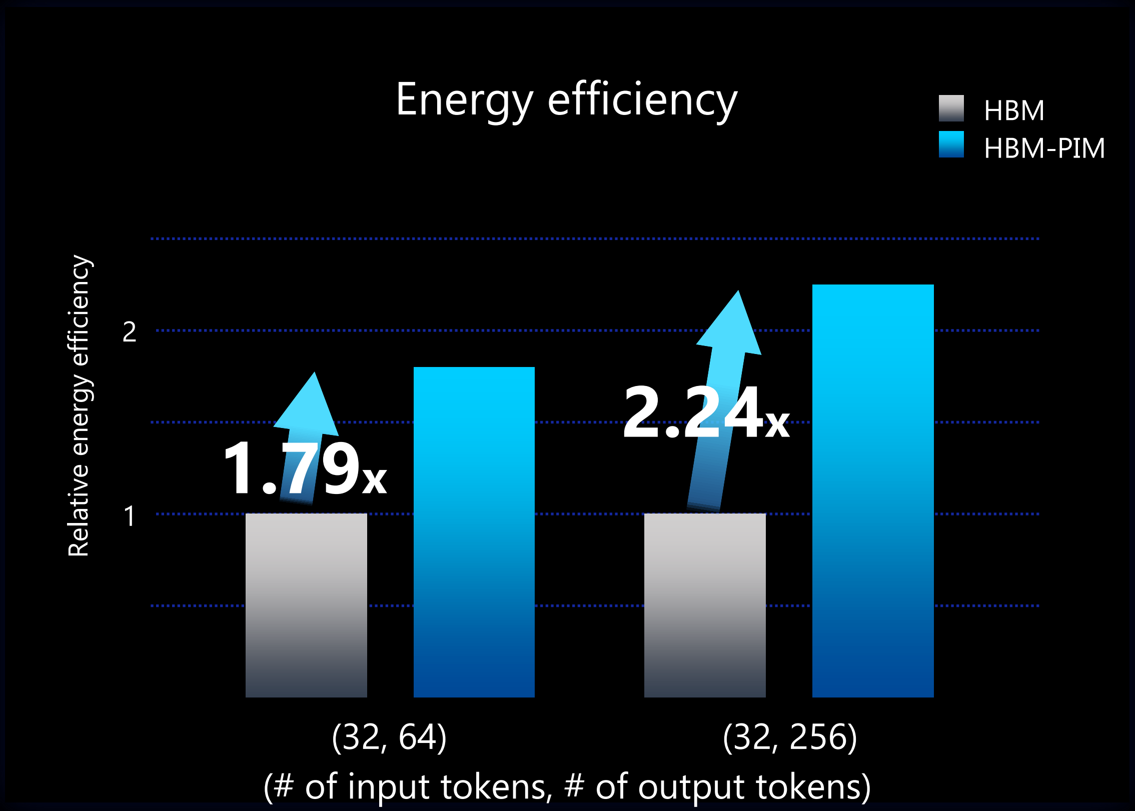
Future HBM-PIM concept

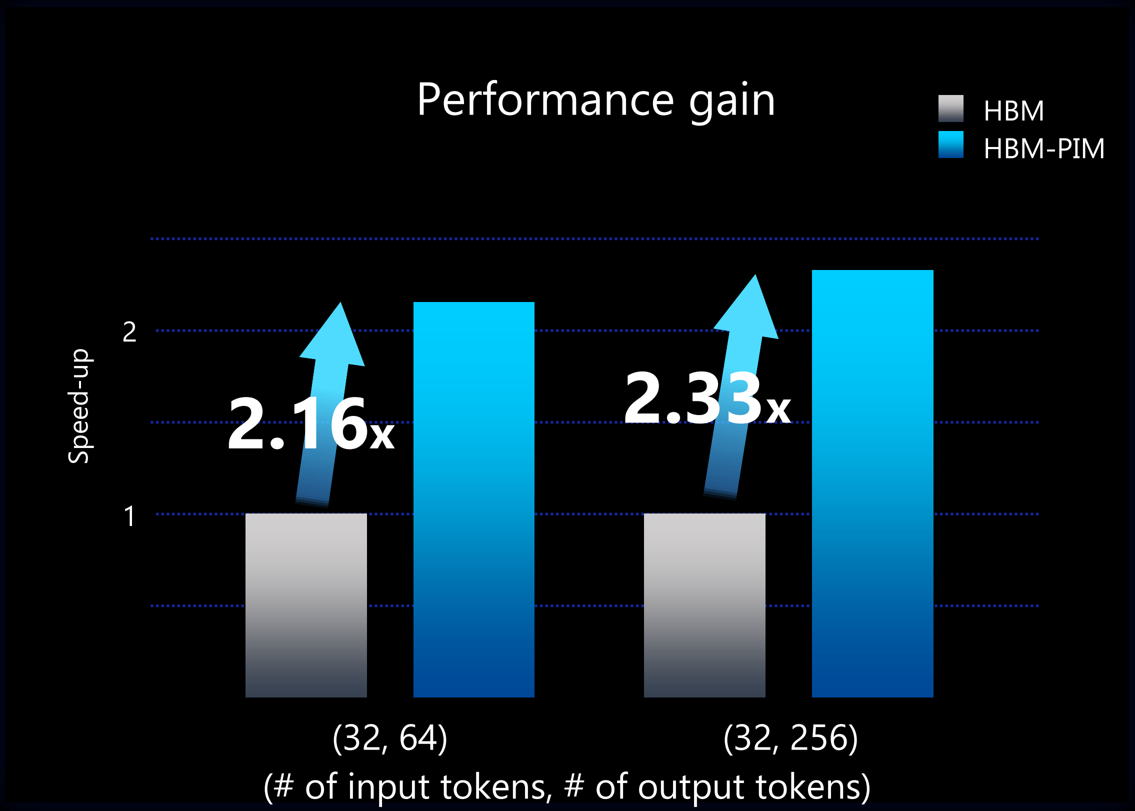


[source] Samsung, MemCon

Generative AI on HBM-PIM

- Experimental setup: GPT-J (6B, 32 input tokens), single AMD MI100-PIM GPU
- About 2x greater system energy efficiency compared to the GPU with a normal HBM
- GPT can be accelerated by more than 2x over baseline

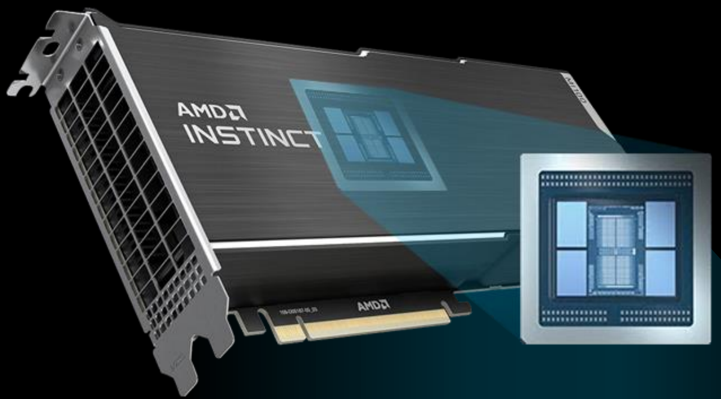




Architecture of HBM-PIM Cluster

- Installed 96 AMD MI100 GPUs fabricated with HBM-PIM
- Accelerate large-scale workloads with high energy efficiency and low latency

AMD MI100-PIM GPU



· Capacity : 24GB (4 cubes)
· PIM performance : 4.9 TFLOPS
. GPU performance : 184.6 TFLOPS (FP16)

Server node

· 8 MI100-PIM GPUs per node

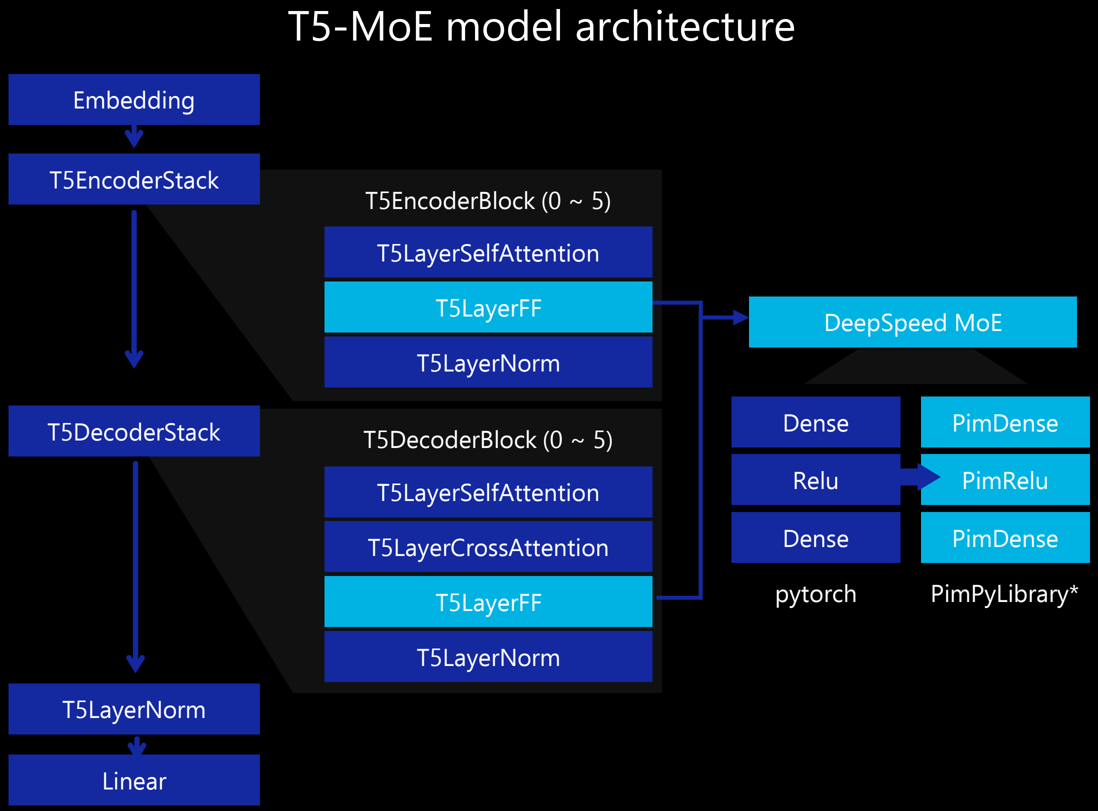
HBM-PIM cluster

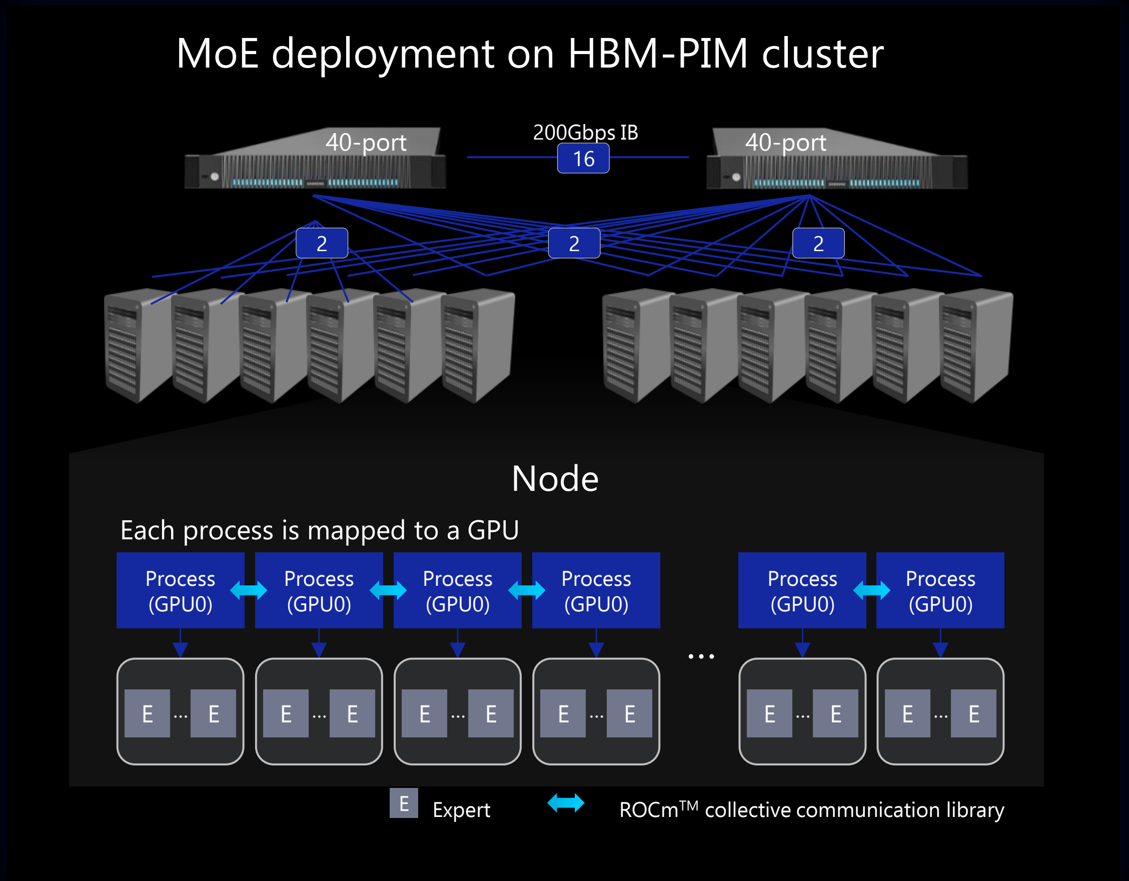
· Total 96 MI100-PIM GPUs in a cluster
. 12 nodes interconnected
through 200G InfiniBand network
(bi-section bandwidth : 1.2TB/s)
. Total memory capacity : 2.25TB
. Total PIM performance : 471.9 TFLOPS
. Total GPU performance : 17.7 PFLOPS (FP16)

Workload : T5(Transformer)-Based MoE Model

(Mixture of Experts)

- DeepSpeed MoE replaces "T5LayerFF" layers to accelerate T5-large model with PIM
- The MoE layers are updated to use our PimPyLibrary* APIs

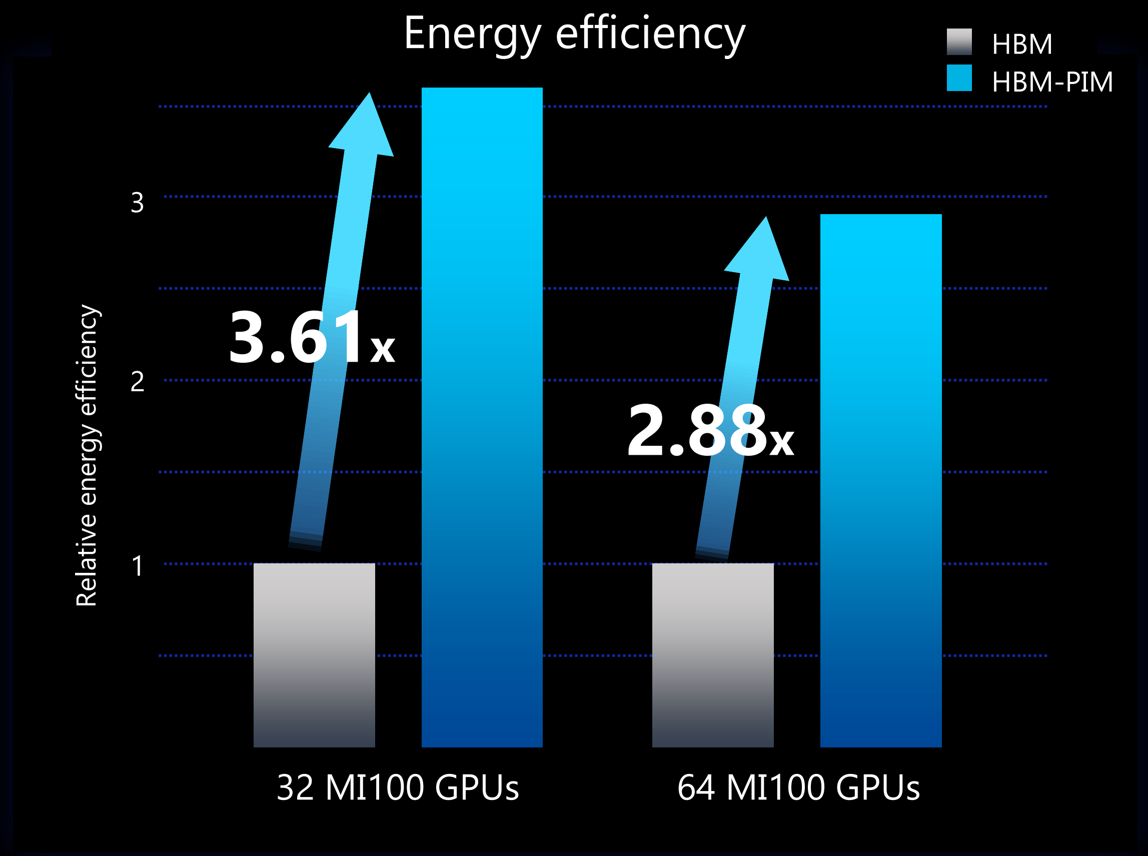


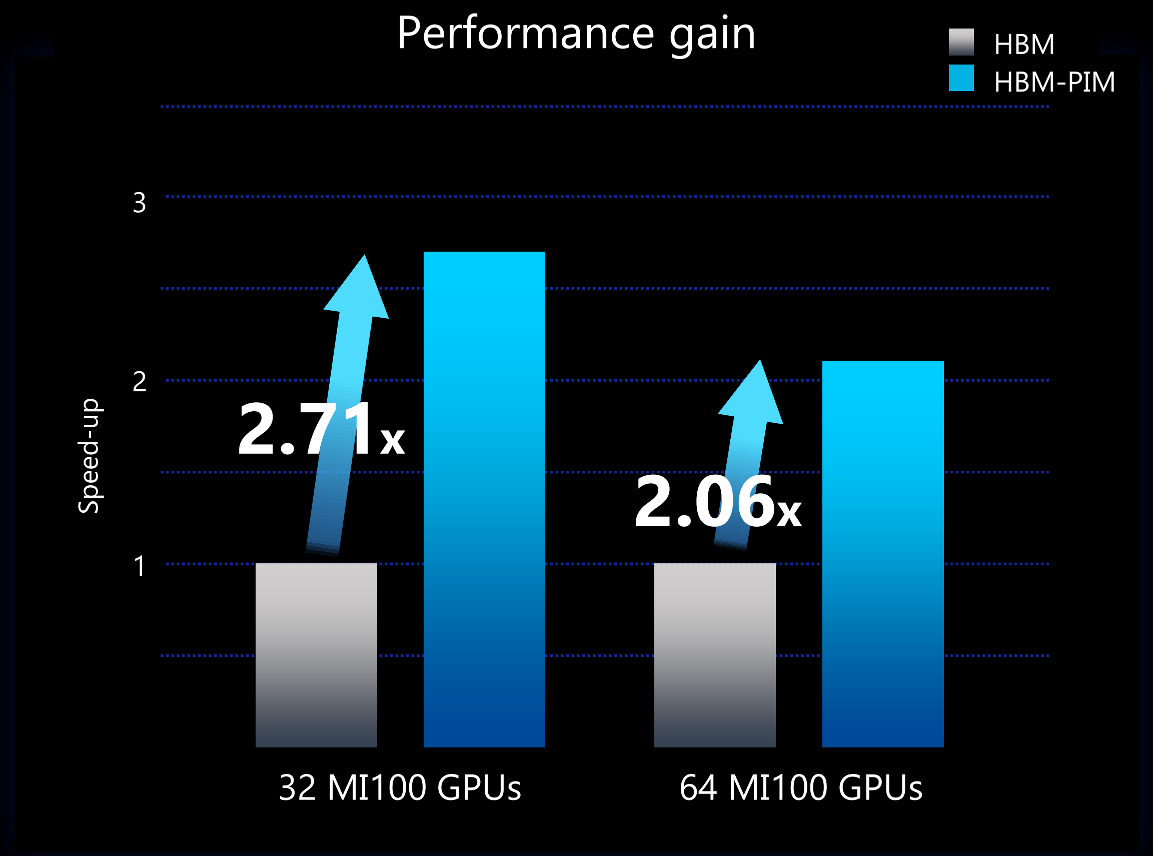


* PimPyLibrary is a python library for providing PIM-enabled AI operators. PIM SDK provides not only PimPyLibrary but also full SW stack for utilizing PIM

Energy Efficiency and Performance on MoE Model

- More than 3x greater system energy efficiency compared to normal GPU clusters
- Increases performance by more than 2x over baseline



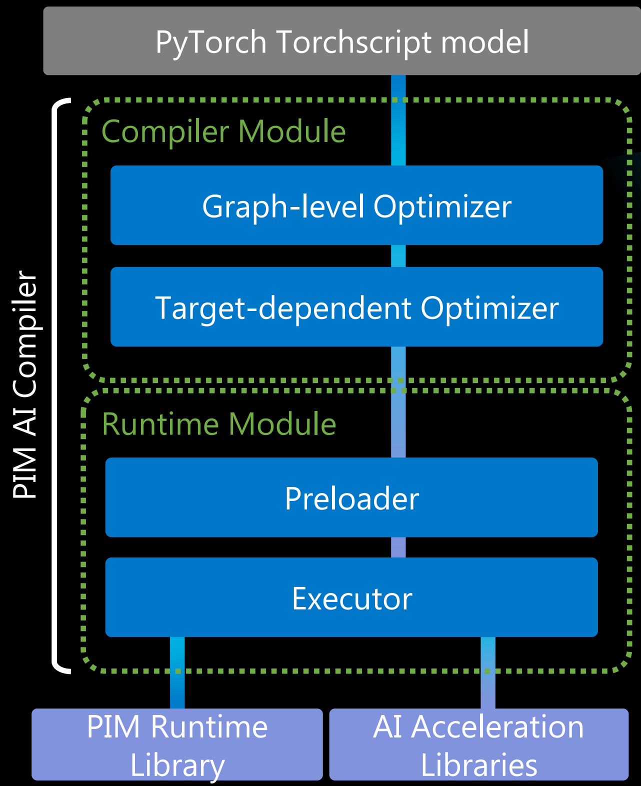


* Acknowledgement: Jaeyoung Heo and professor Sungjoo Yoo (Seoul National University) provided the idea of PIM acceleration for this workload

PIM S/W Stack for AI

- Support existing AI frameworks (e.g., PyTorch and TF) for users to utilize PIM functions
- PIM Runtime Library: Apply PIM and provide operator-level optimizations during PIM operation
- PIM AI Compiler: Provide graph-level optimizations during end-to-end execution

Execution Pipeline



PyTorch Torchscript model

PIM AI Compiler

Compiler Module

Graph-level Optimizer

Target-dependent Optimizer

Runtime Module

Preloader

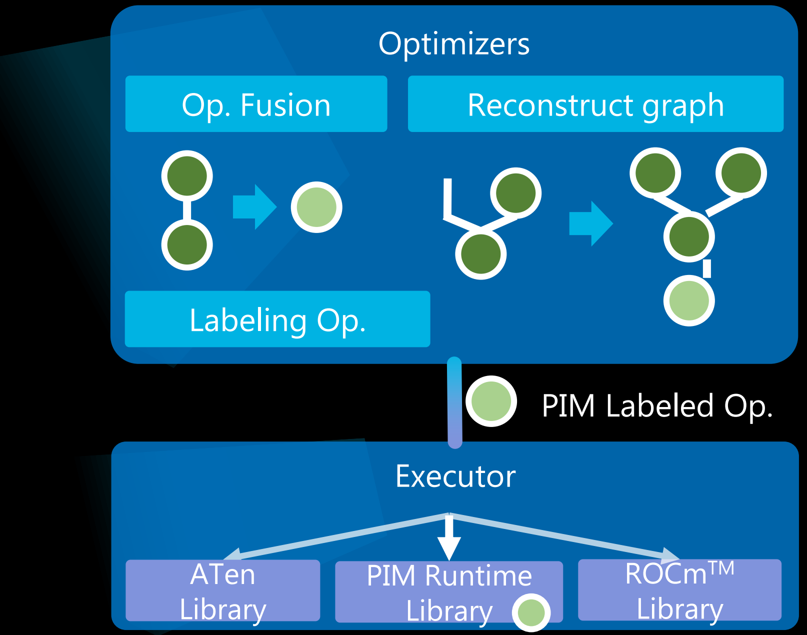
Executor

PIM Runtime Library

AI Acceleration Libraries

Optimization Examples

- Operator Fusion is to mitigate kernel load overhead
- Reconstruct graph is to make possible to run on PIM



Optimizers

Op. Fusion

Reconstruct graph

Labeling Op.

PIM Labeled Op.

Executor

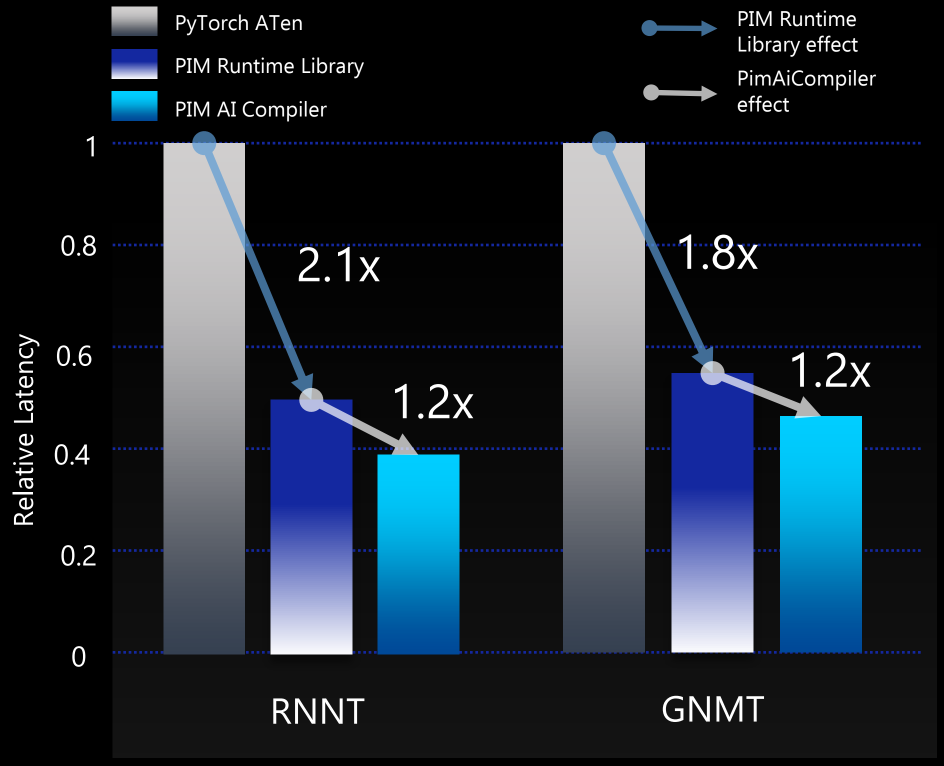
ATen Library

PIM Runtime Library

ROCm™ Library

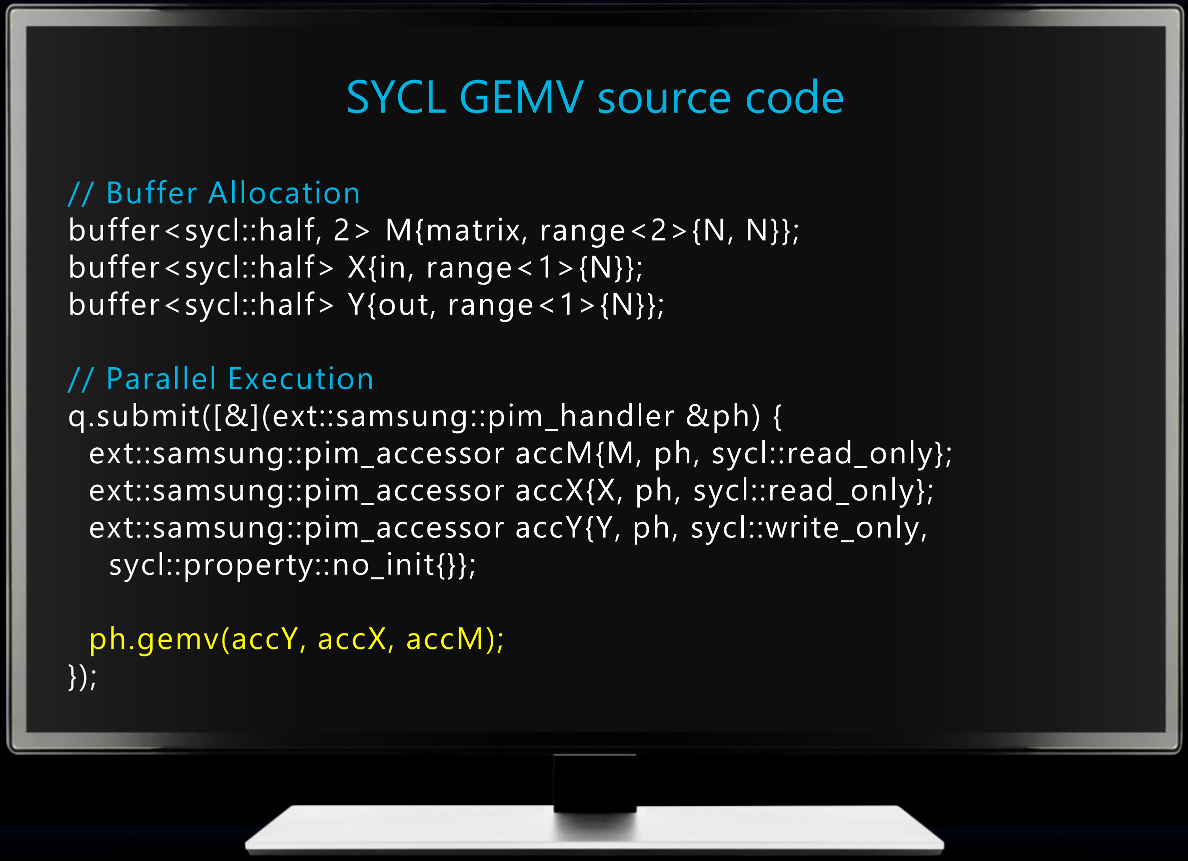
- Runtime choose appropriate library per given tensor

Effect of PIM with S/W optimizations



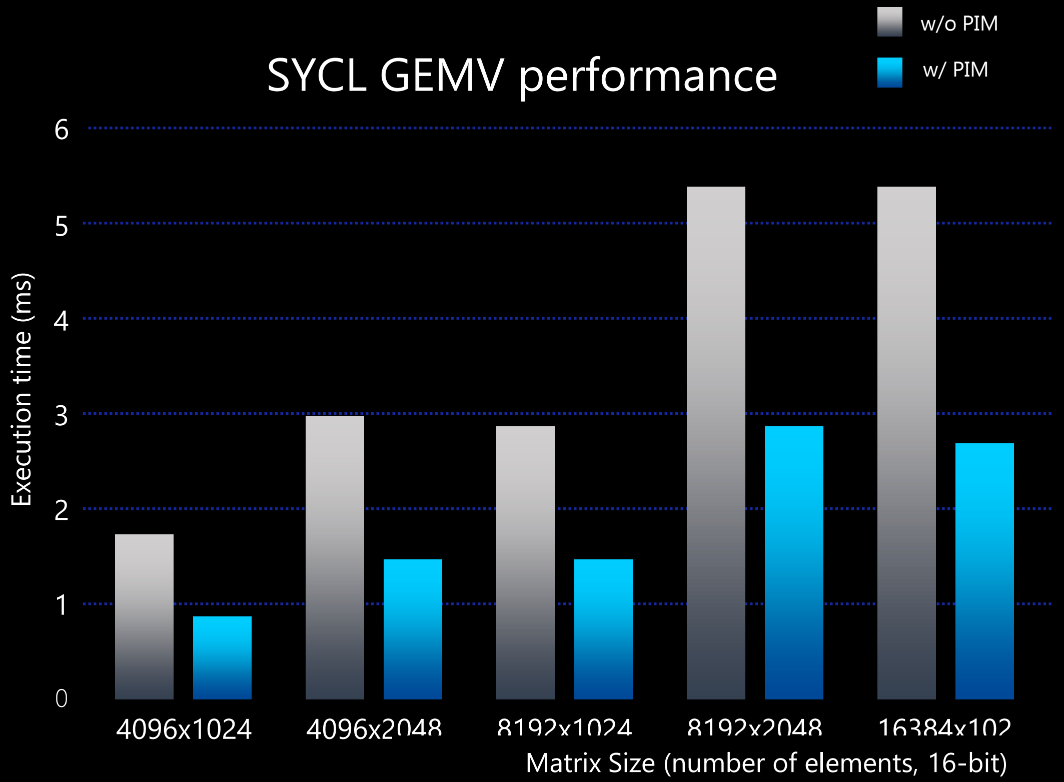
Promising Standard Programming Models

- PIM-SYCL accelerates upcoming HPC/AI applications on heterogeneous platform
  - SYCL supports CPU/GPU/NPU/FPGA in modern C++ template
- PIM-OpenACC is under development for legacy scientific applications
  - OpenACC enables incremental parallelization from C/Fortran serial code



SYCL GEMV source code

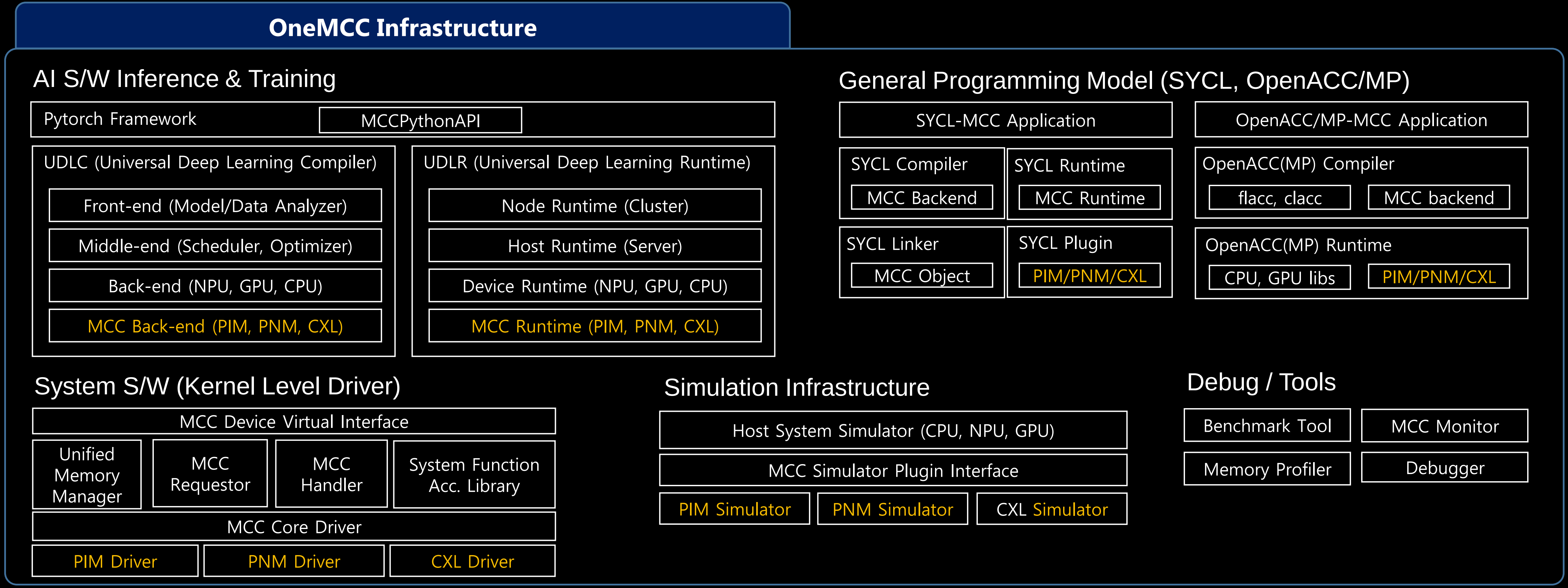
```
// Buffer Allocation
buffer<sycl::half, 2> M{matrix, range<2>{N, N}};
buffer<sycl::half> X{in, range<1>{N}};
buffer<sycl::half> Y{out, range<1>{N}};

// Parallel Execution
q.submit([&](ext::samsung::pim_handler &ph) {
  ext::samsung::pim_accessor accM{M, ph, sycl::read_only};
  ext::samsung::pim_accessor accX{X, ph, sycl::read_only};
  ext::samsung::pim_accessor accY{Y, ph, sycl::write_only,
    sycl::property::no_init{}};

  ph.gemv(accY, accX, accM);
});
```



SYCL GEMV performance

OneMCC S/W Standardization (To be)

- OneMCC (Memory Coupled-Computing) is an open & standard S/W for PIM, PNM, CXL solutions
- Plan to provide standard programming model to support multi-architecture and domain
- Boost AI and HPC workloads with a variety of accelerators like CPUs, GPUs, and NPUs



OneMCC Infrastructure

AI S/W Inference & Training

Pytorch Framework | MCCPythonAPI

UDLC (Universal Deep Learning Compiler)
- Front-end (Model/Data Analyzer)
- Middle-end (Scheduler, Optimizer)
- Back-end (NPU, GPU, CPU)
- MCC Back-end (PIM, PNM, CXL)

UDLR (Universal Deep Learning Runtime)
- Node Runtime (Cluster)
- Host Runtime (Server)
- Device Runtime (NPU, GPU, CPU)
- MCC Runtime (PIM, PNM, CXL)

General Programming Model (SYCL, OpenACC/MP)

SYCL-MCC Application
- SYCL Compiler: MCC Backend
- SYCL Runtime: MCC Runtime
- SYCL Linker: MCC Object
- SYCL Plugin: PIM/PNM/CXL

OpenACC/MP-MCC Application
- OpenACC(MP) Compiler: flacc, clacc | MCC backend
- OpenACC(MP) Runtime: CPU, GPU libs | PIM/PNM/CXL

System S/W (Kernel Level Driver)

MCC Device Virtual Interface

Unified Memory Manager | MCC Requestor | MCC Handler | System Function Acc. Library

MCC Core Driver

PIM Driver | PNM Driver | CXL Driver

Simulation Infrastructure

Host System Simulator (CPU, NPU, GPU)

MCC Simulator Plugin Interface

PIM Simulator | PNM Simulator | CXL Simulator

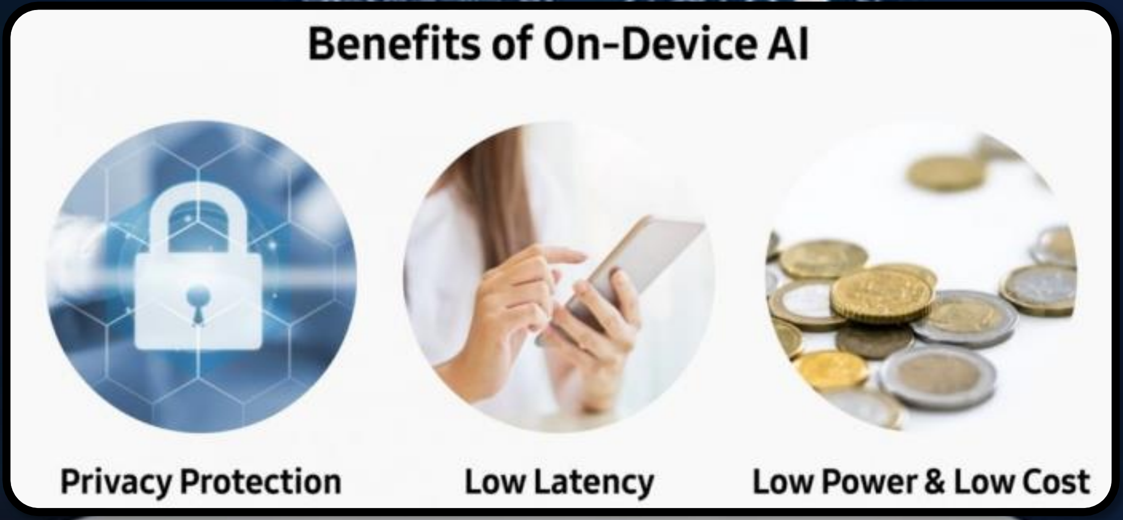
Debug / Tools

Benchmark Tool | MCC Monitor

Memory Profiler | Debugger

Processing-in-Memory for On-device Generative AI

- Expanding On-device AI Necessity:
  - Data center costs and power consumption are increasing due to the growing demand for cloud AI
  - Privacy concerns are rising as sensitive data is transmitted to the cloud for processing
  - Network connectivity is not always reliable or available, particularly in remote areas
- LPDDR-PIM improves battery life by preventing memory over-provisioning just for bandwidth

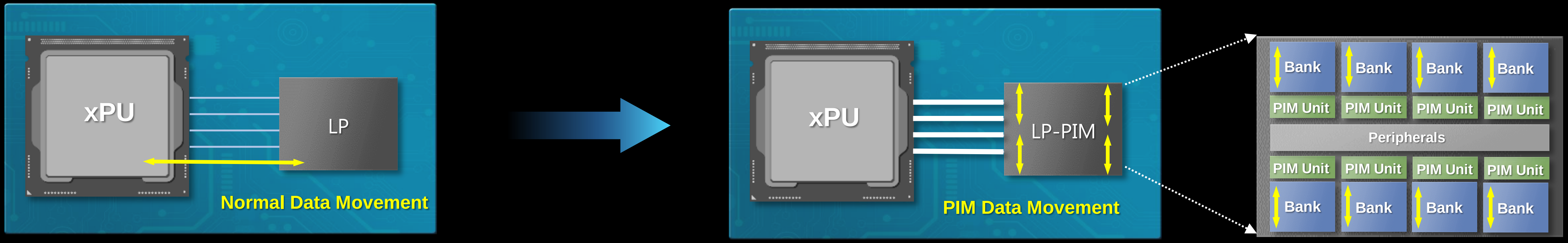


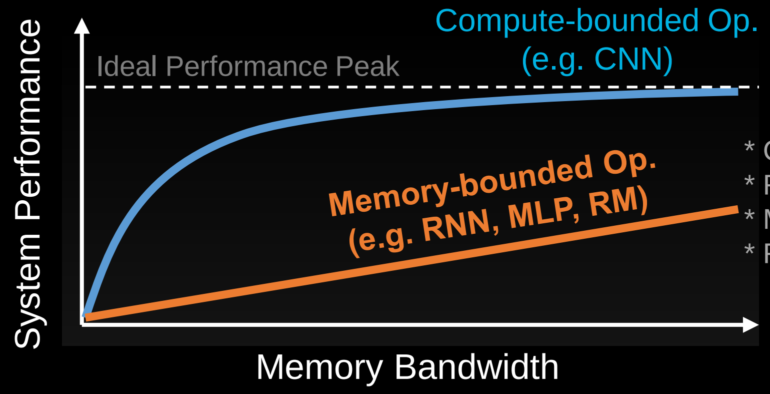
LPDDR-PIM Concept

- Improve the 4.5x performance and save 72% of energy in the system with in-DRAM processing
  - Performance: Utilize up to 8× higher in-DRAM bandwidth by multi-bank parallel operation
  - Energy Efficiency: Reduce data movement energy by utilizing PIM unit

LP5-xPU System

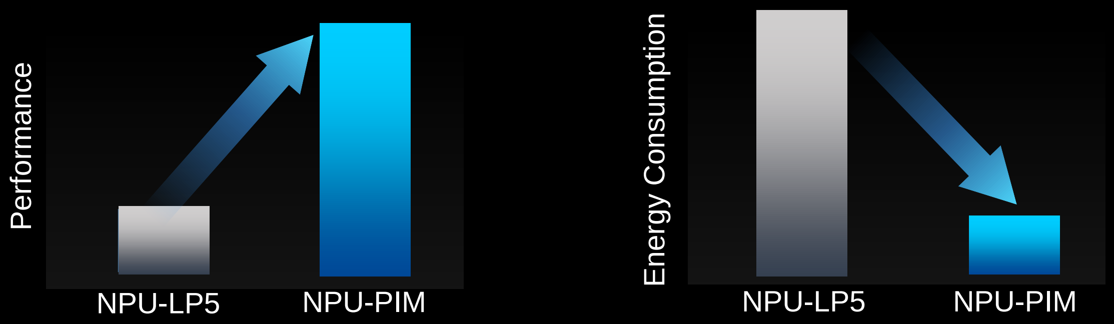
PIM-xPU System





* CNN: Convolutional Neural Network
* RNN: Recurrent Neural Network
* MLP: Multi-layer Perceptron
* RM: Recommendation Model

System Performance and Energy Comparison

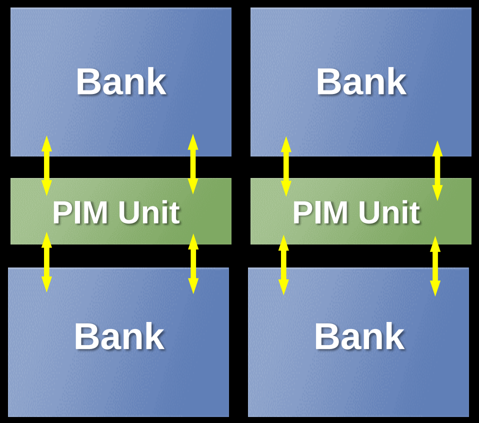


LPDDR-PIM Features

- Peak internal bandwidth: 102.4 GB/s
  - Using Bank-level parallelism, 8x bandwidth of the base LPDDR product
- Supporting native integer/floating point arithmetic and logical (and/or/…) operations
- Peak performance: 102.4 GFLOPS/s (FP16), 204.8 GOPS/s (INT8)
- Acceleration target: memory-bounded operations such as BLAS1 and BLAS2
- Samsung can support LPDDR-PIM simulator package to measure performance gain & energy reduction

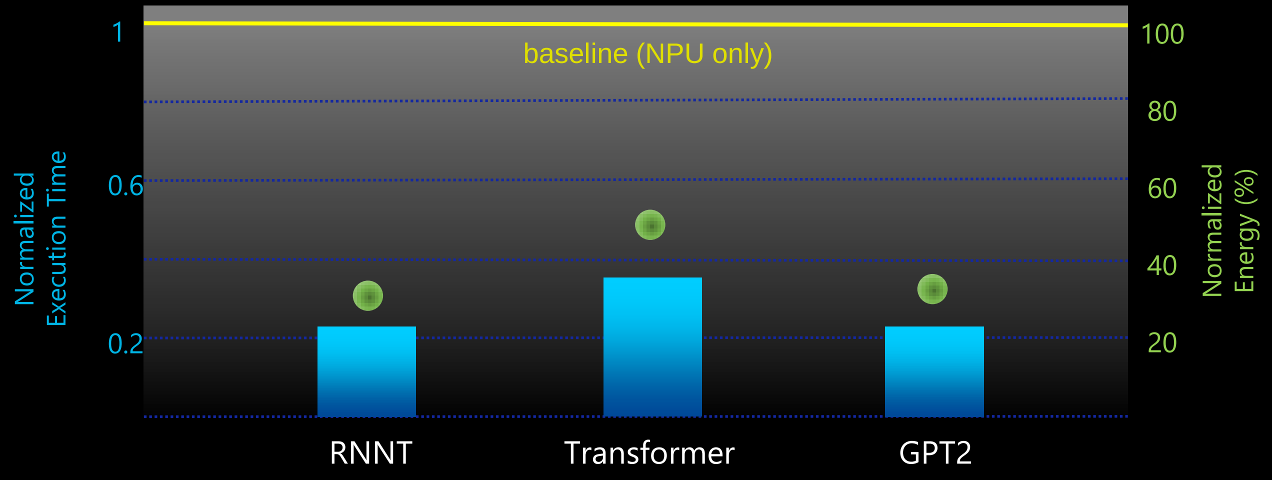
BLAS1: Element-wise addition/multiplication
or layer normalization
BLAS2: vector-matrix multiplications

Bank Parallelism



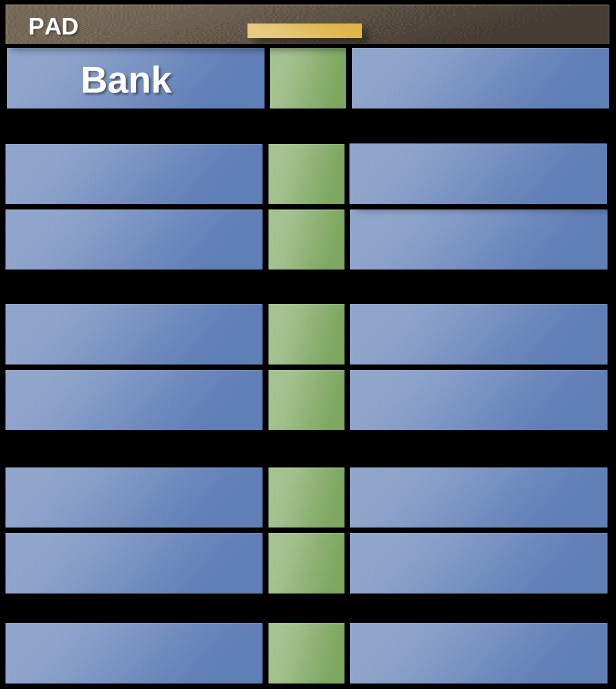
End-to-end inference on AI application (simulation-based)

	[RNNT]	[Transformer]	[GPT2]
Perf. gain:	4.51x	2.85x	4.47x
Energy reduction:	-72.5%	-58.5%	-70.6%



LPDDR-PIM Architecture

- PIM Unit is placed between every 1 bank (maximum performance) or 2 banks (moderate area overhead)
- PIM Unit: 256-bit SIMD FPU and registers (~640 bytes per PIM block)
  - Supporting operations: FP16 multiplication, FP32 accumulation, int8 arithmetic, etc.
  - PIM registers : Instruction (IRF), Vector (VRF), and Scalar (SRF)

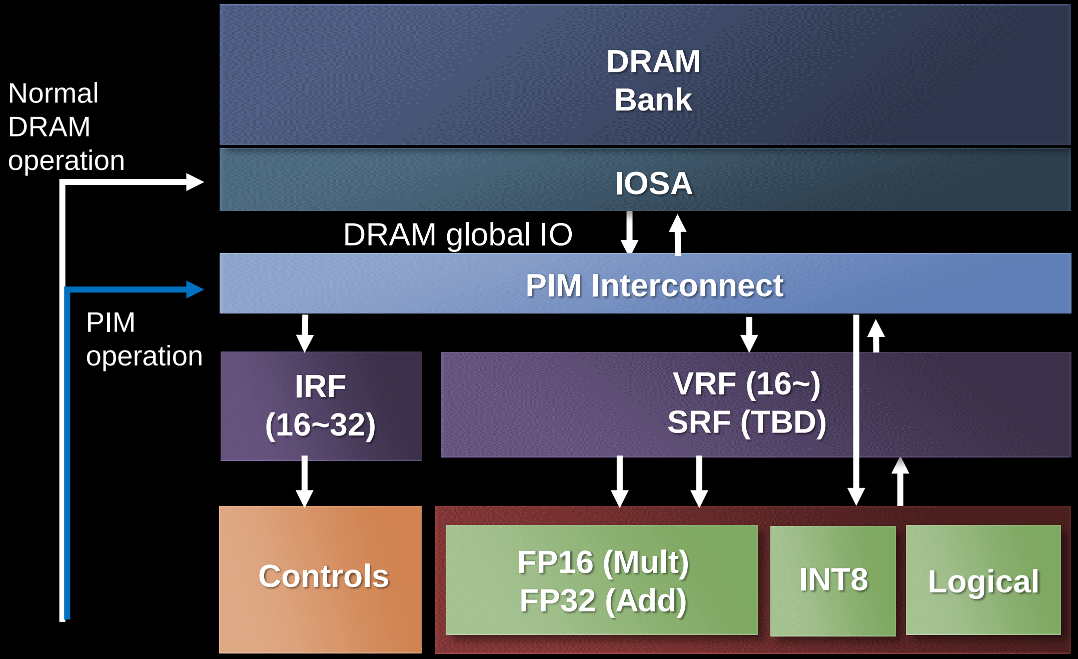


🟦 DRAM Bank (Cell)

🟩 PIM Block

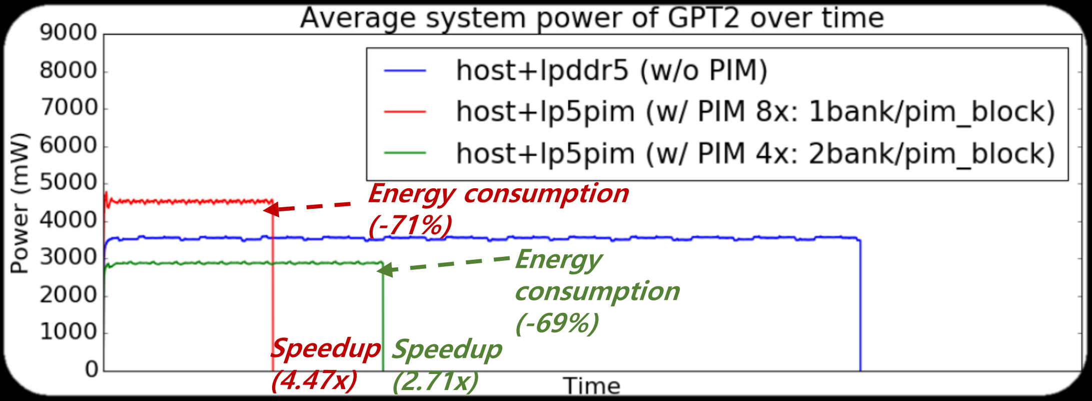
🟨 PIM Support Modules

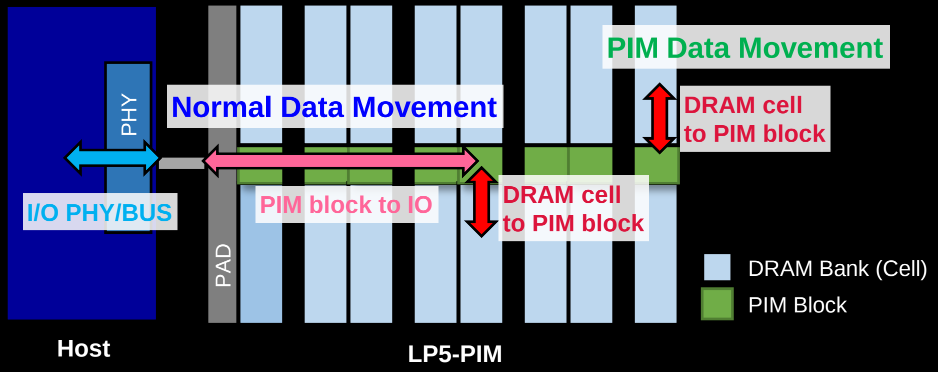
*Not scaled to
actual physical silicon area



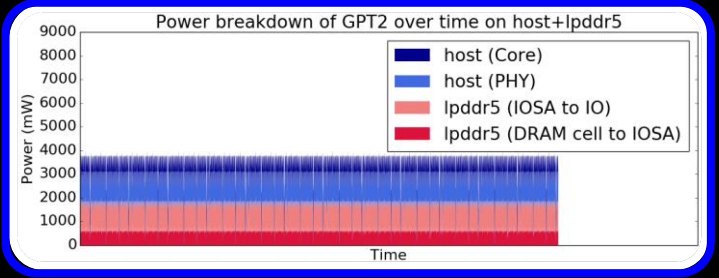
LPDDR-PIM System Performance/Power Analysis

- Evaluate the performance and power consumption of GPT2
  - LP5-PIM improves energy efficiency by shorter execution time

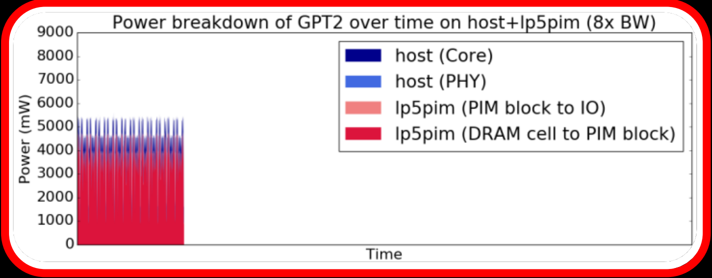




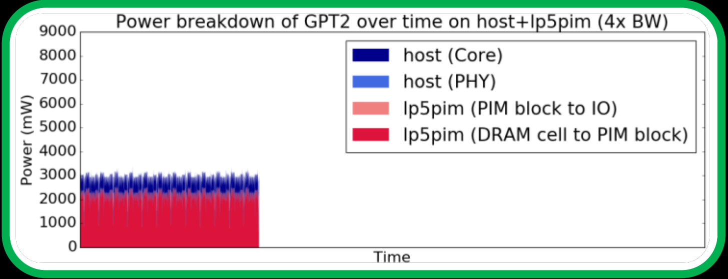
- Power consumption of DRAM internal component (red) increases proportionally
- Power consumption of global I/O bus (light red) and I/O PHYs (light blue) considerably decreases



[host + lpddr5]



[host + lp5pim (8x: 1 bank/PIM block)]



[host + lp5pim (4x: 2 bank/PIM block)]

* The power ratio is 17.9% (DRAM cell to IOSA), 31.3% (IOSA to IO), 31.1% (PHY), and 19.7% (core) for baseline.
* The power ratios are 85.3% (DRAM cell to PIM block), 14.6% (core), and 0.1% (etc) for lp5-pim(8x), and 76.9% (DRAM cell to PIM block), 23% (core), and 0.1% (etc) for lp5-pim(8x), respectively.

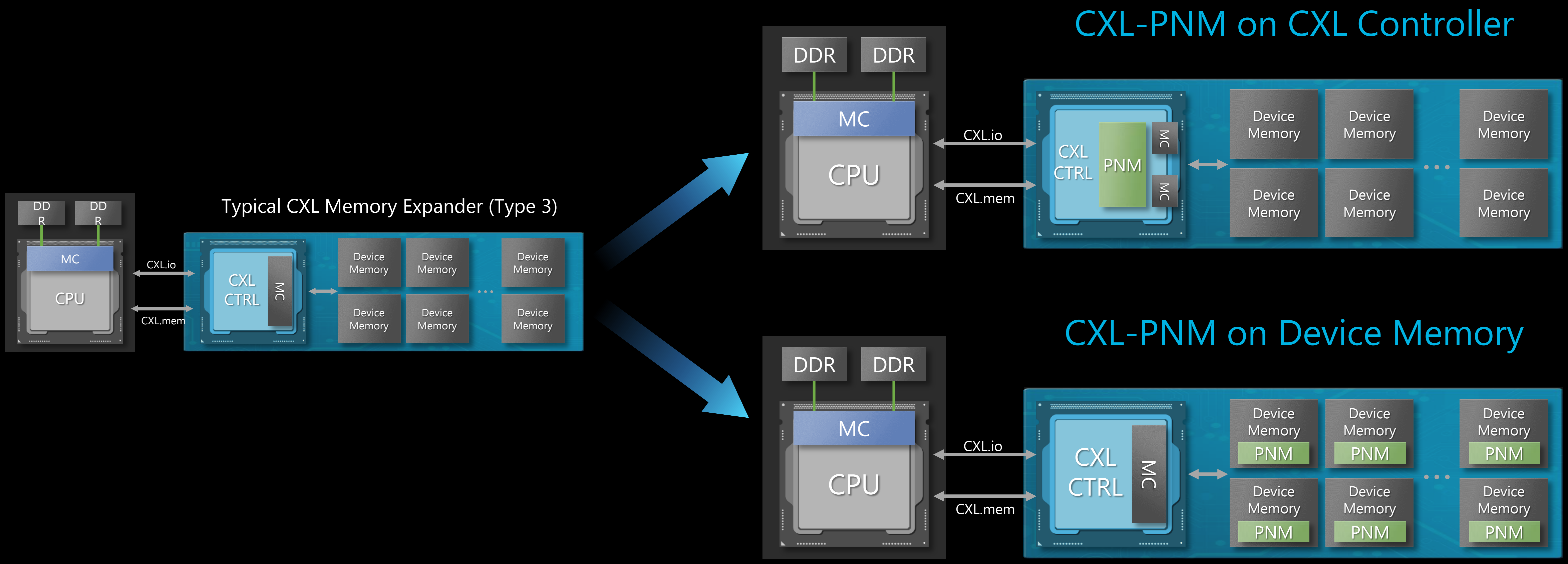
* Simulation experiment uses 4 memory (lp5/pim) channels
* lp5pim (DRAM cell to PIM block) includes ACT, PRE, IDLE, and REF
* The result of simulation is that more than 99% of memory traffic decreases by PIM
* host (core) includes processing and IDLE power

CXL-PNM

Industry's 1st CXL-PNM (Processing-near-Memory)

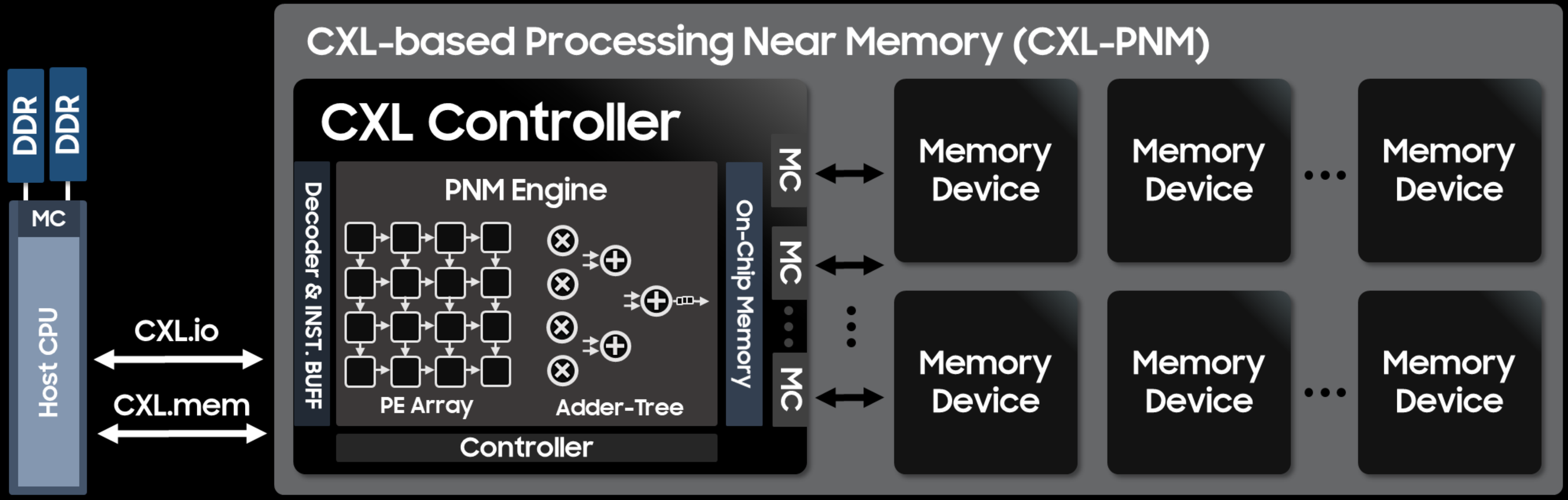
CXL-PNM Architecture

- A CXL-based Processing-near-Memory (PNM) Solution
- Two types of CXL-PNM: on CXL controller and on device memory



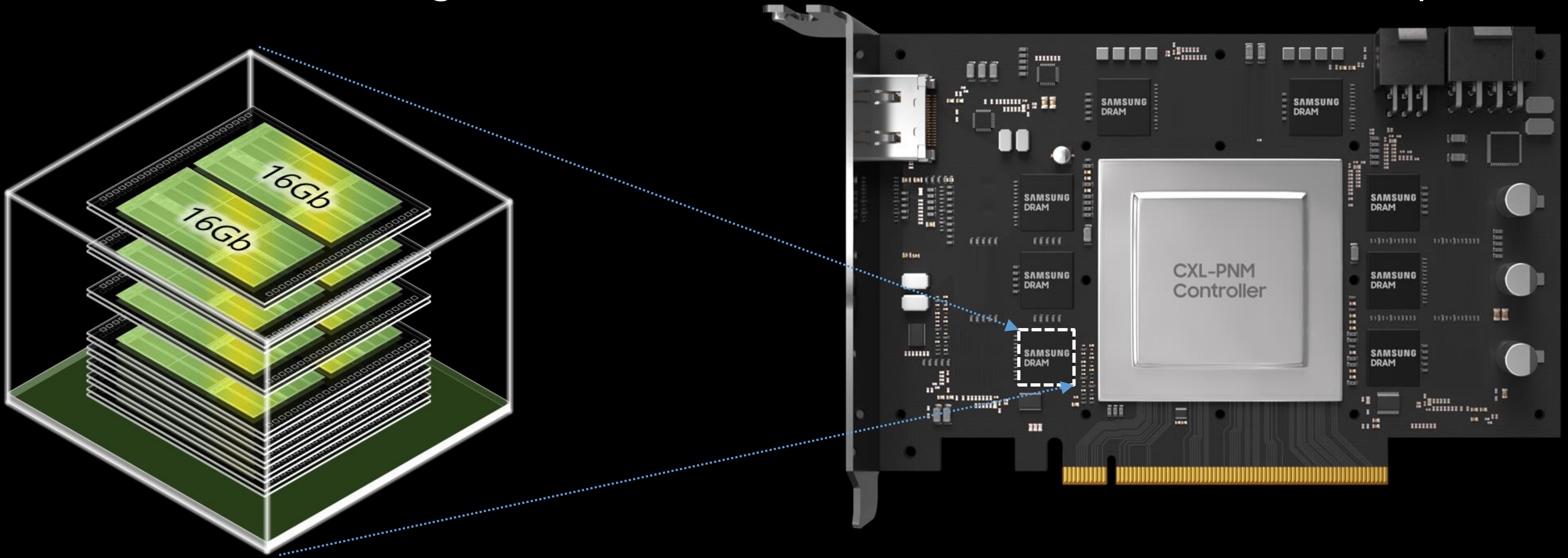
CXL-PNM Architecture for GPT

- Heterogeneous compute unit (PE Array and Adder-Tree) on PNM Engine
  - Adder-tree designed to perform GEMV operation (Generation stage)
  - PE Array for acceleration of GEMM operation (Summarization stage)



512GB 1.1TB/s CXL-PNM Concept

- CXL-PNM is able to be used for a wider range of systems including AI/ML accelerators
- Compared to other solutions, it can give unique trade-off among capacity, bandwidth, and power
- CXL-PNM can provide 512GB capacity and 1.1TB/s bandwidth

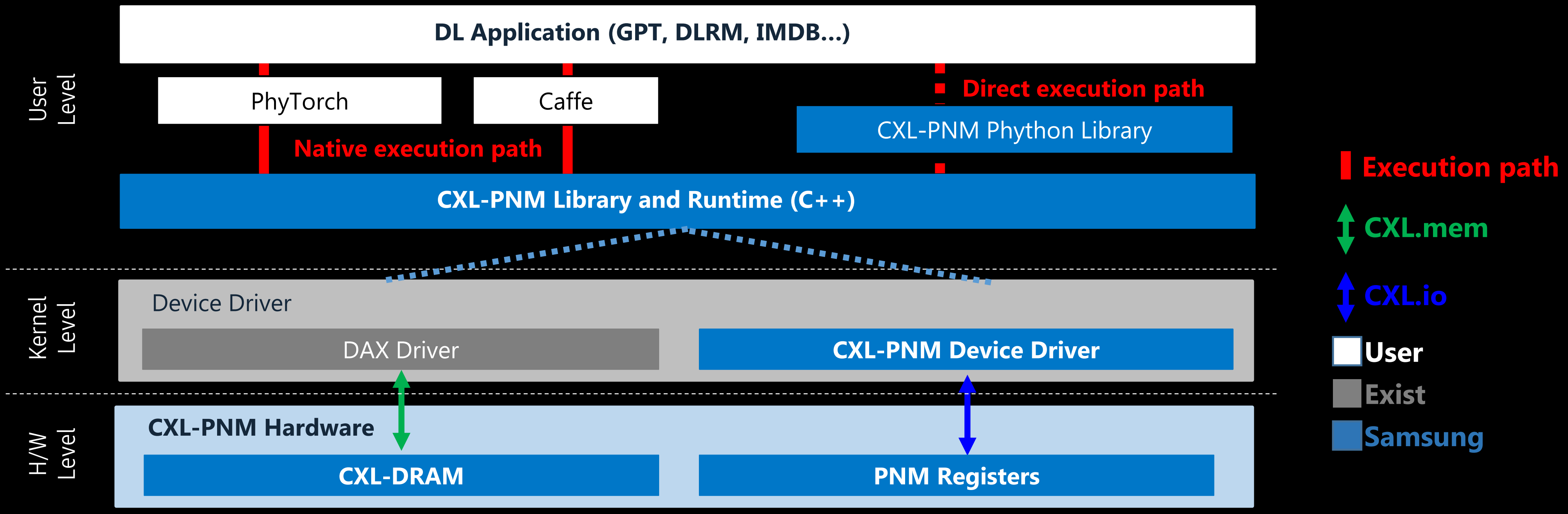


512Gb DRAM Package

CXL-PNM Module Concept

CXL-PNM Software Stack

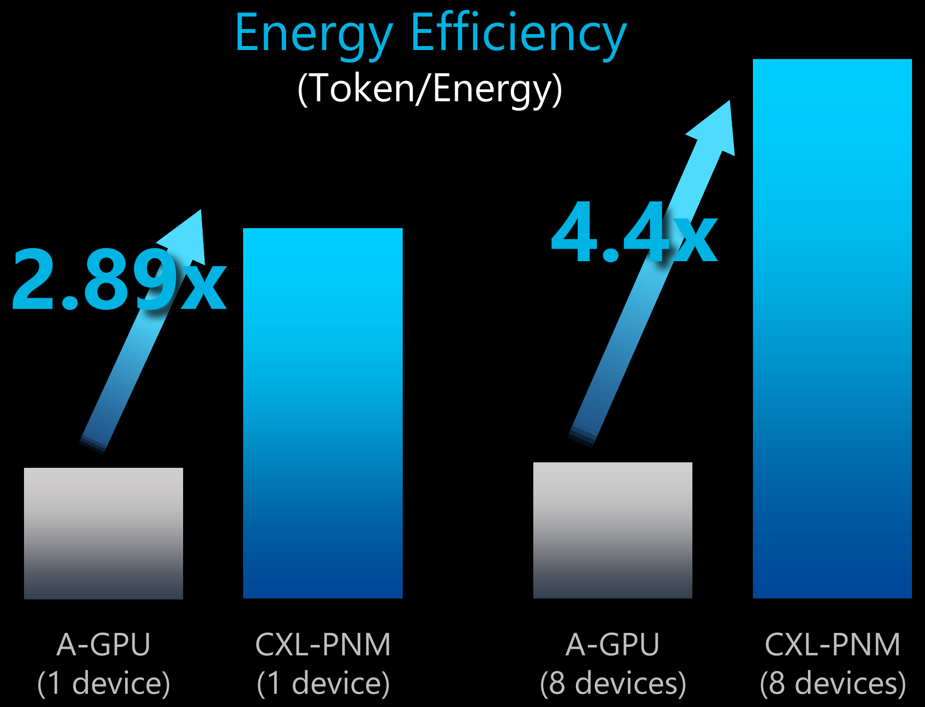
- CXL-PNM software stack for users to seamlessly and transparently utilizes the CXL-PNM platform
- CXL-PNM software stack includes user-level of library, runtime and kernel level of device driver
- CXL-PNM software stack supports two execution paths
  - Native execution path – Automatically offload PNM operation without modification of application source code
  - Direct execution path – Explicitly call PNM operations on the user application

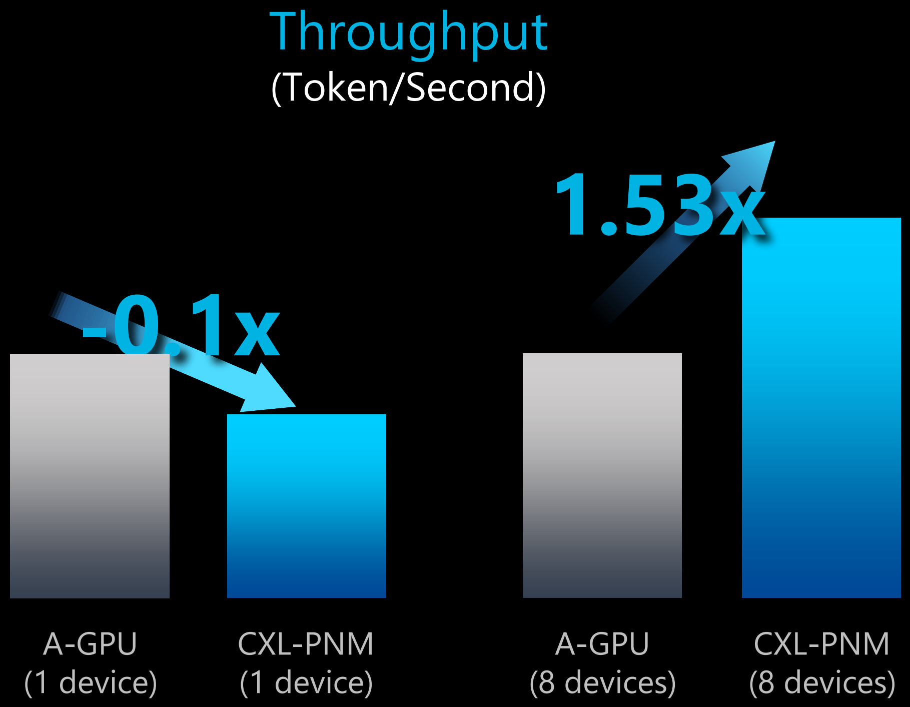


CXL-PNM Software Stack

Energy and Throughput Comparison for LLM

- End-to-end performance evaluation of GPU and CXL-PNM with the same number of devices
  - – CXL-PNM gives 2.9× higher energy efficiency, only 10.8% lower throughput, compared to A-GPU
  - – However, CXL-PNM with large capacity can accelerate large-scale LLMs w/o any communication overhead
  - – Multiple CXL-PNM can offer 4.4× higher energy efficiency and 53% higher throughput than multiple GPUs

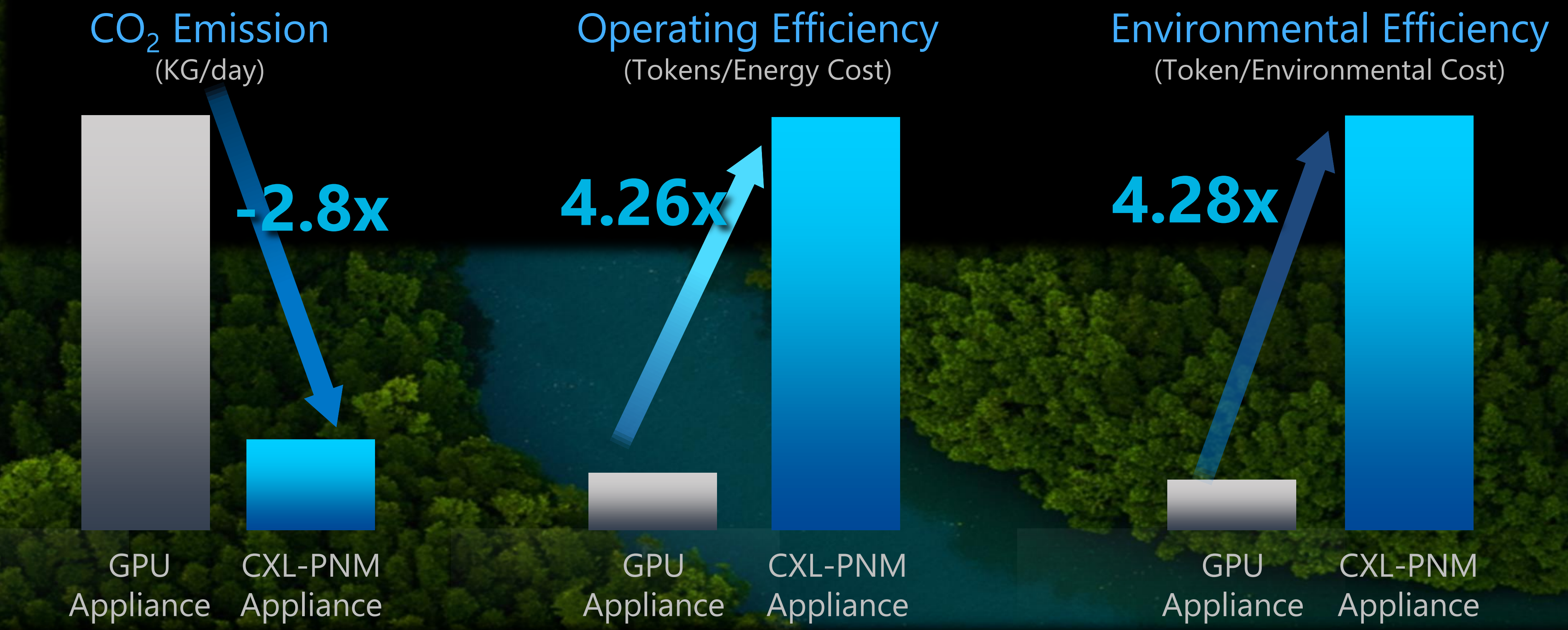




• Single Device (OPT-13B), 8 Device (OPT-66B), input token(64) output token(1024)

CO_2 Emission Reduction by CXL-PNM

- Operating/Environmental (Energy/CO_2) cost of GPU/CXL-PNM appliance with eight devices
  - GPU appliance is 2.8× more expensive than CXL-PNM appliance for the environmental cost
  - Operating efficiency of CXL-PNM appliance reduces the amount of CO_2 emission
  - CXL-PNM appliance is 4.3× more efficient than that of GPU appliance



- GPU Appliance (A-GPU server System with 8 A-GPU, FasterTransformer), CXL-PNM Appliance (8 CXL-PNM Card)

Samsung AI memory