Debunking the CUDA Myth

Towards GPU-based AI Systems

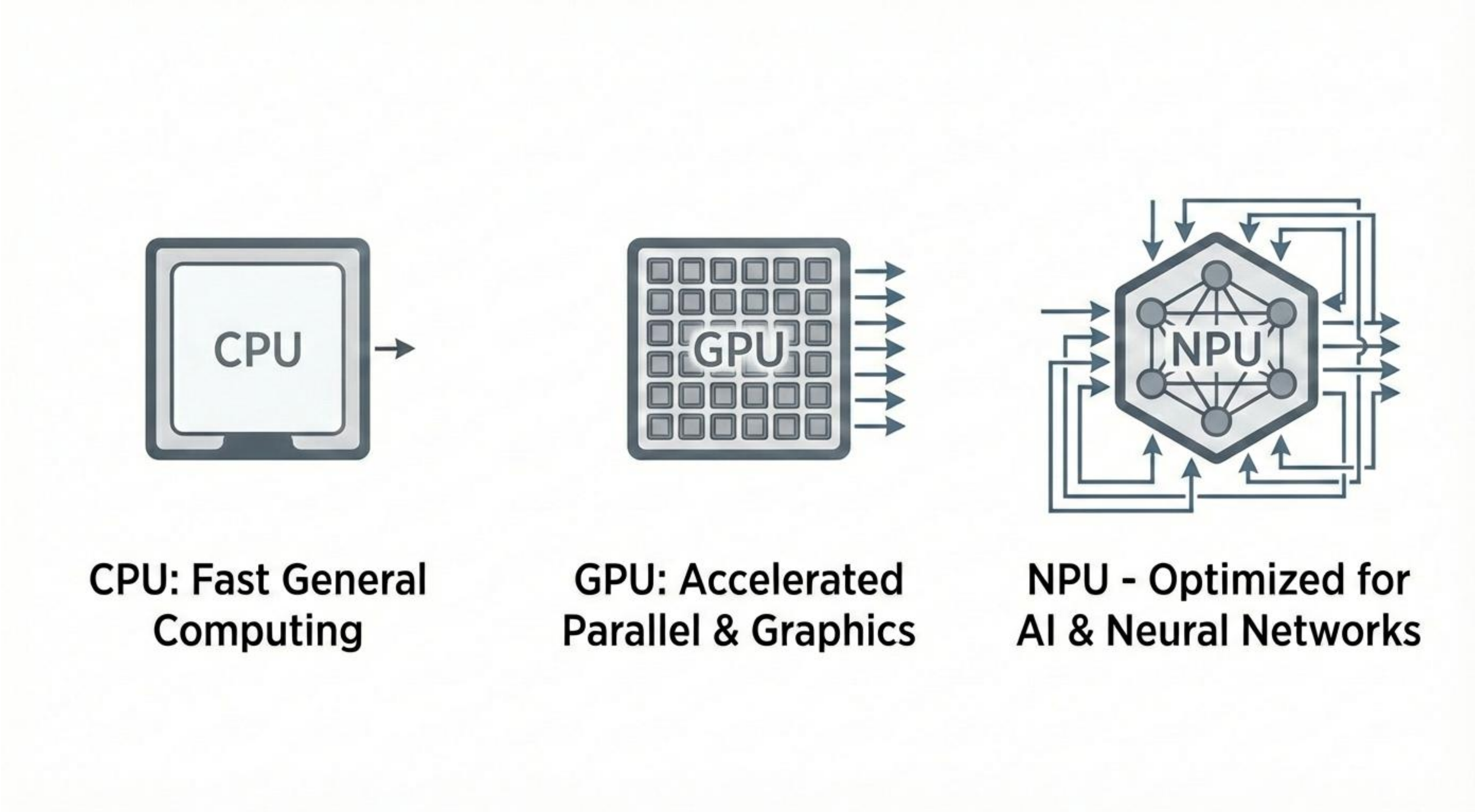
Original Paper by: Yunjae Lee et al. (Prof. Minsoo Rhu's Group, ISCA '25)

Presented by Jongyun Hur

Introduction

Introduction

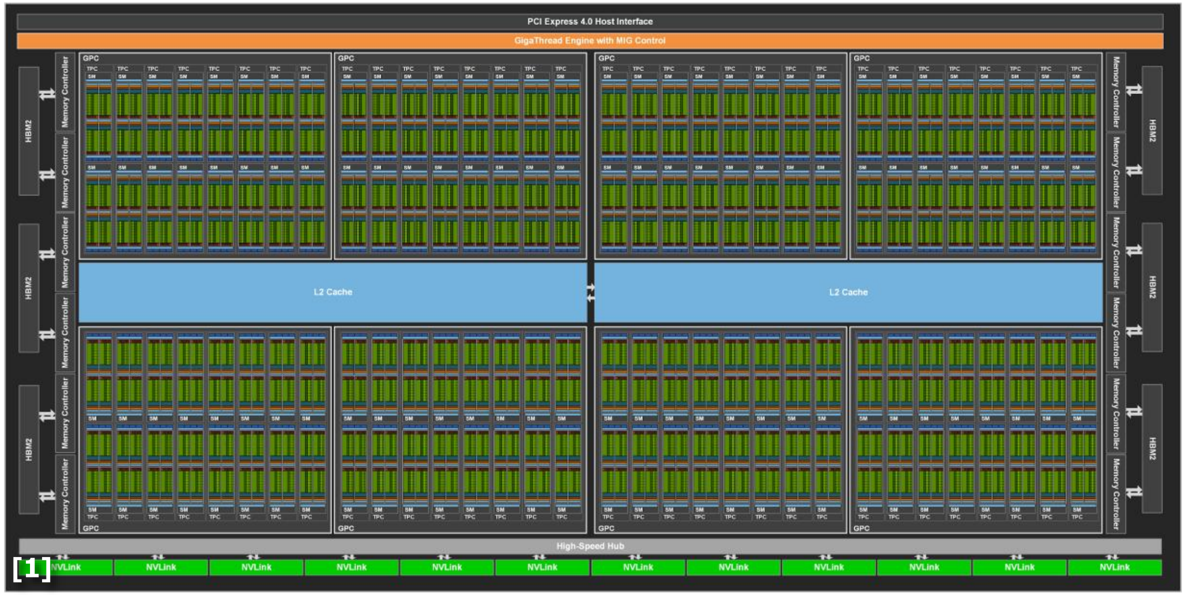
Hardware Evolution: From CPU Dominance to the Rise of NPUs



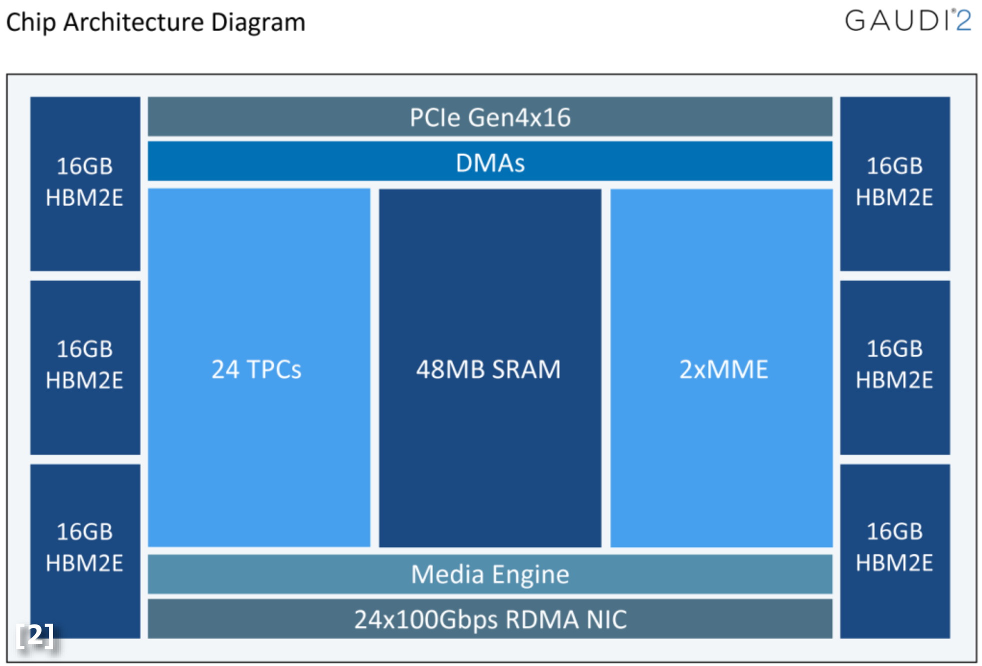
Introduction

Architecture Overview: GPU vs. NPU

NVIDIA GPU A100



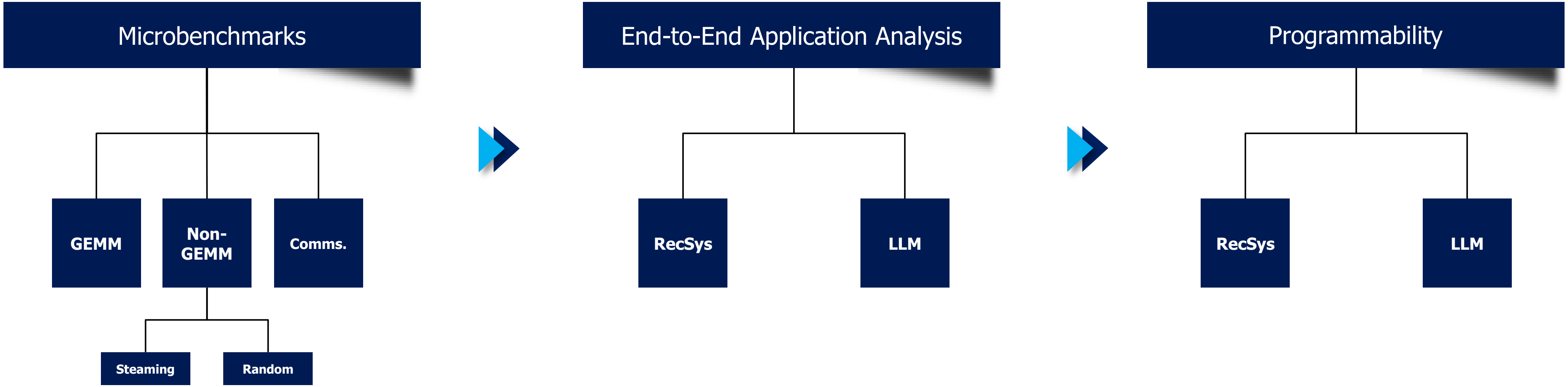
Intel NPU Gaudi-2



[1] NVIDIA Developer Blog, “NVIDIA Ampere Architecture In-Depth”
[2] Intel Gaudi Documentation, “Gaudi Architecture”

Introduction

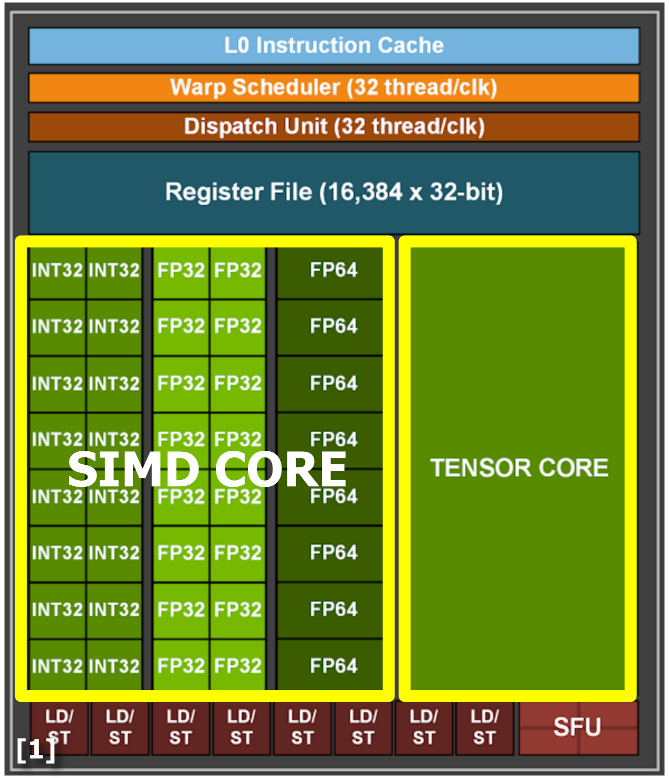
Evaluation Strategy: 3-Step Verification

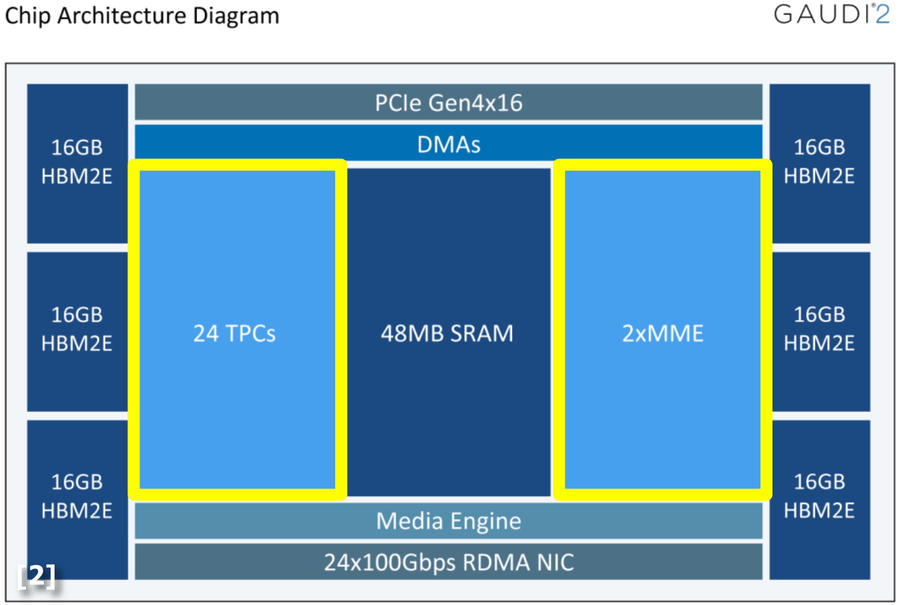


Architecture

Architecture

Chip Layout





[1] NVIDIA Developer Blog, "NVIDIA Ampere Architecture In-Depth"
[2] Intel Gaudi Documentation, "Gaudi Architecture"

Architecture

Compute Hierarchy: GEMM vs. Non-GEMM

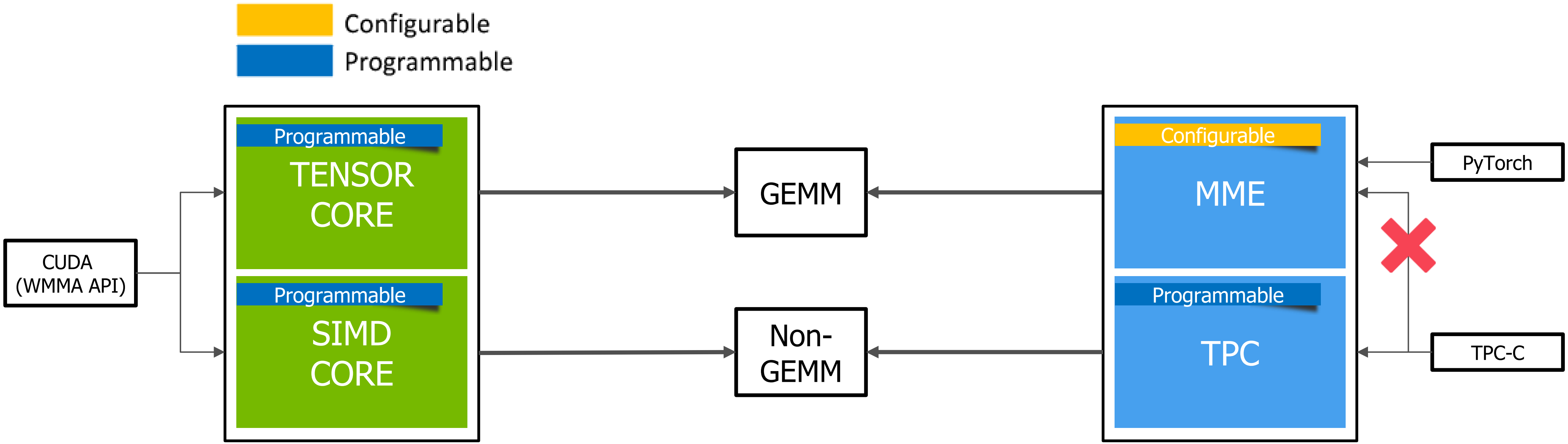
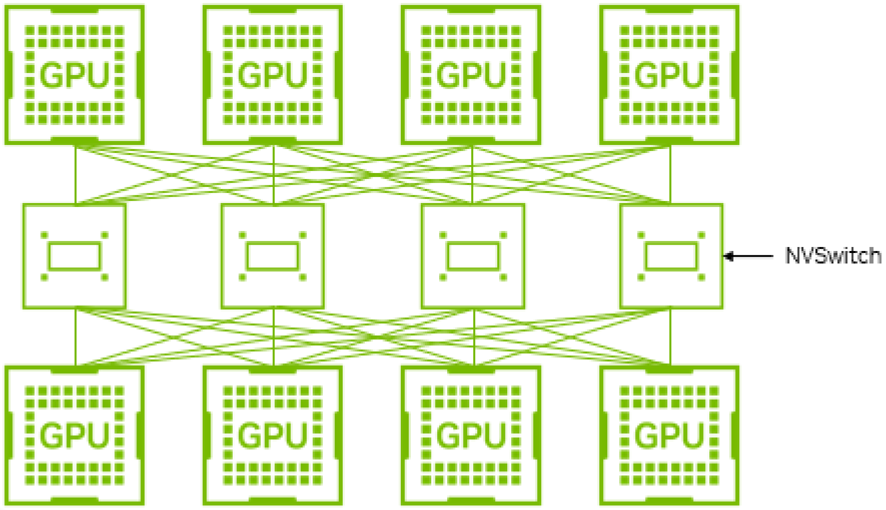


Table 1: Comparison of NVIDIA A100 and Intel Gaudi-2.

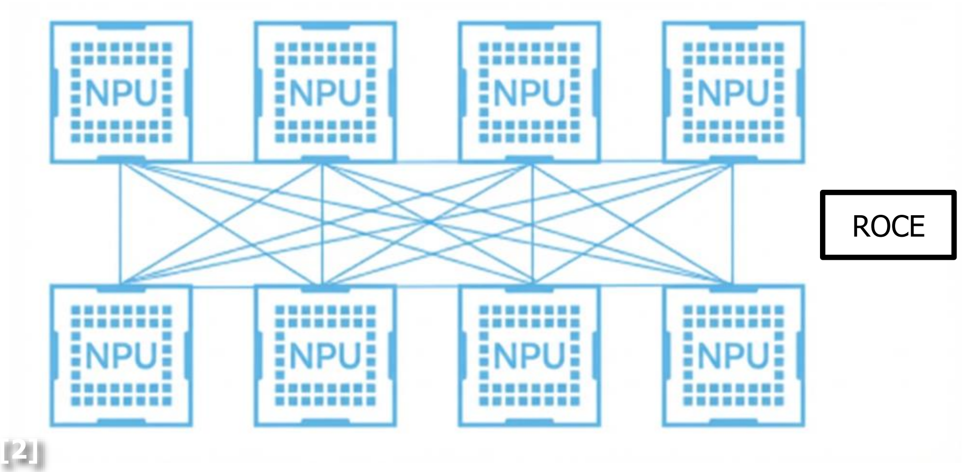
		NVIDIA A100	Intel Gaudi-2	Ratio
Compute	TFLOPS (BF16)	312 (Tensor Cores)	432 (MME)	1.4×
		39 (SIMD Cores)	11 (TPC)	0.3×

Architecture

Chip Communication



[1]



[2]

Communication	600 GB/sec bidirectional (NVLink)	600 GB/sec bidirectional (RoCE)	1.0×

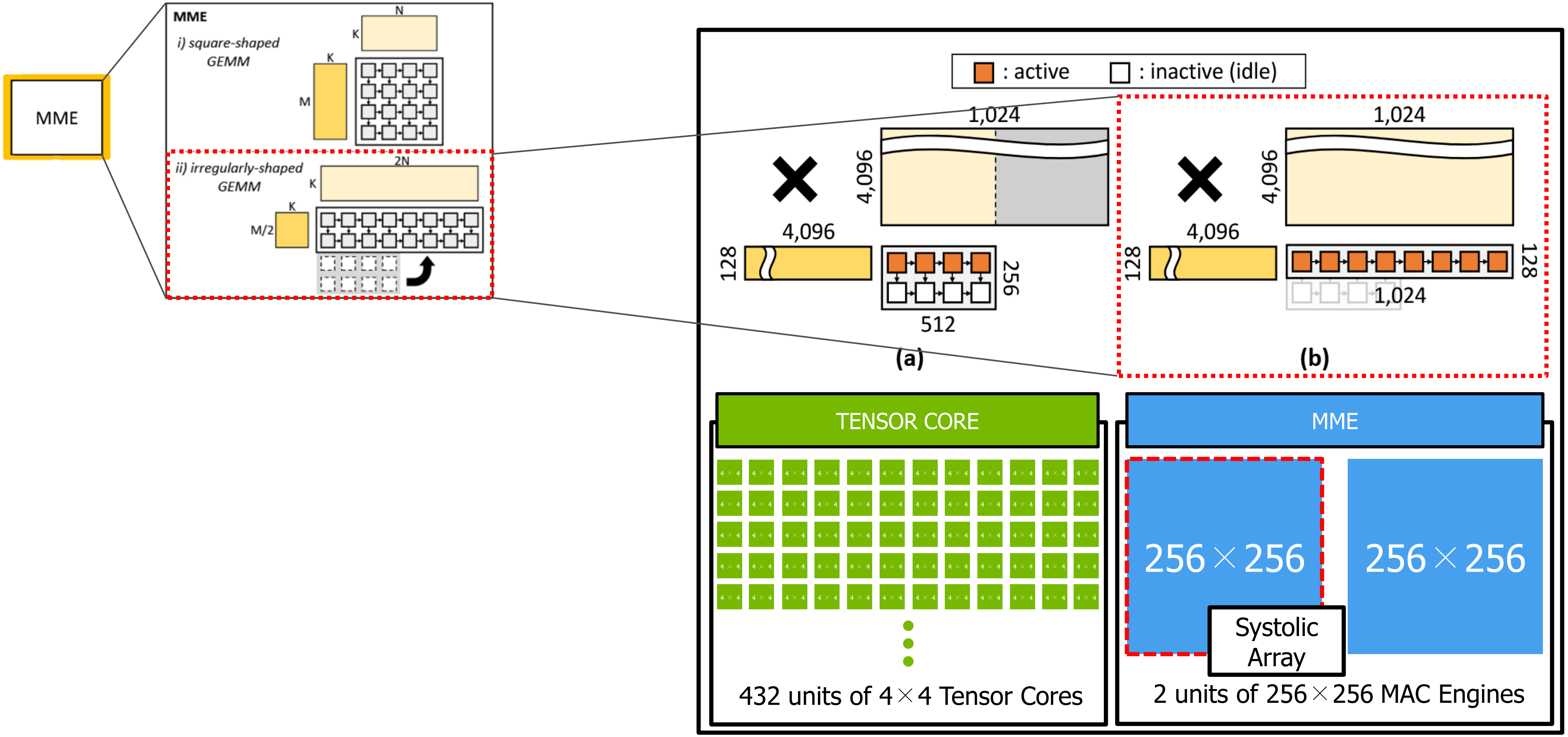
Condition: Single device

[1] NVIDIA Developer Blog, "NVIDIA NVLink and NVIDIA NVSwitch Supercharge Large Language Model Inference"
[2] Concept image modified from [1] to illustrate NPU interconnects

Evaluation

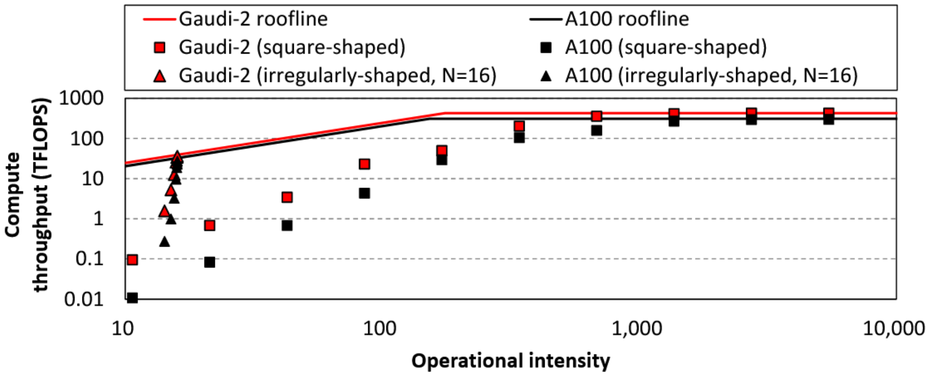
Evaluation: Microbenchmark

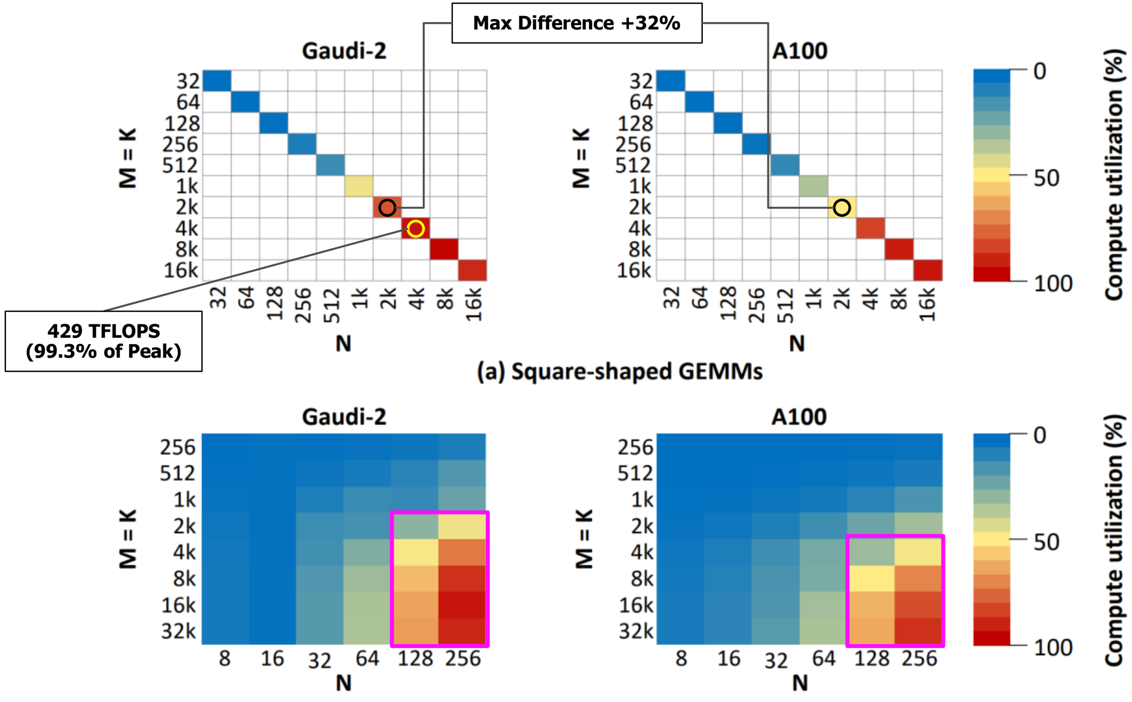
Micro-Architecture Optimization – MME



Evaluation: Microbenchmark

Roofline Analysis & Characterization across GEMM Geometries





(a) Square-shaped GEMMs

(b) Irregularly-shaped GEMMs

Evaluation: Microbenchmark

Reverse-Engineering MME: Geometry Configuration Map

- (M, N) of GEMM while K is fixed to 16,384

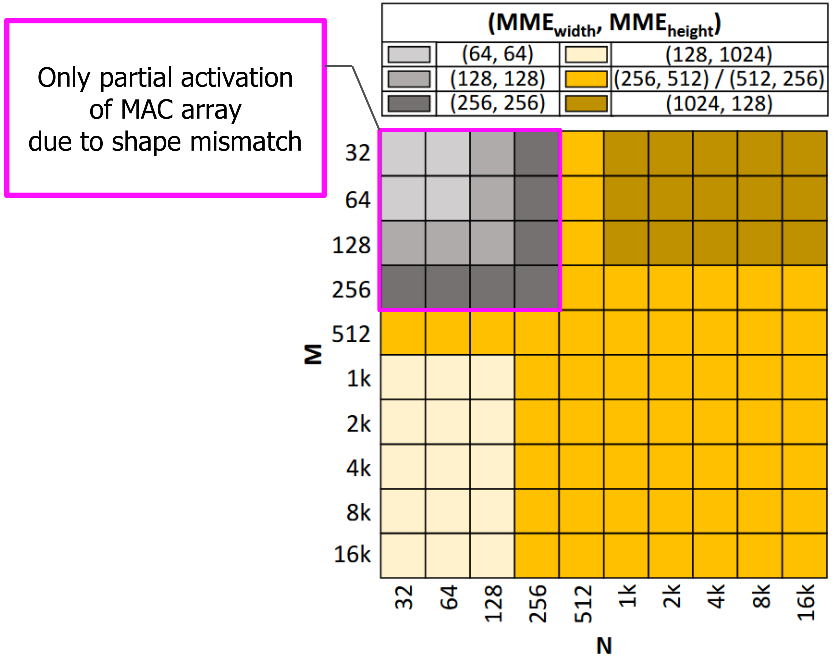


Figure 7(a): MME systolic array geometry configuration based on GEMM shapes (*M*,*N*)

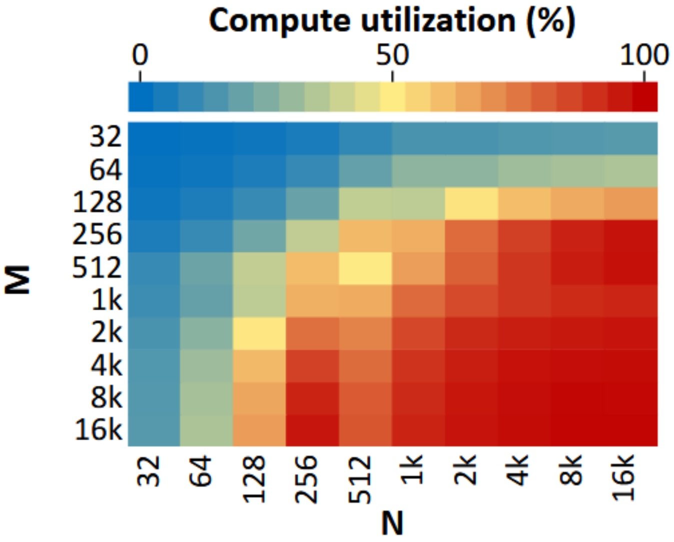
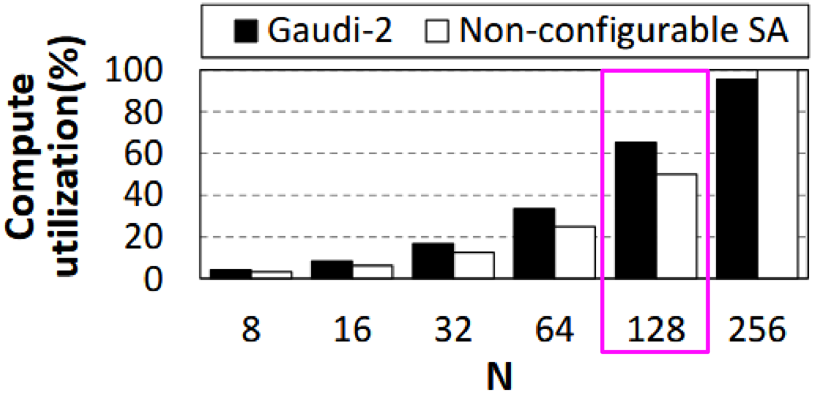


Figure 7(b): Compute utilization of Gaudi-2 MME for various GEMM shapes

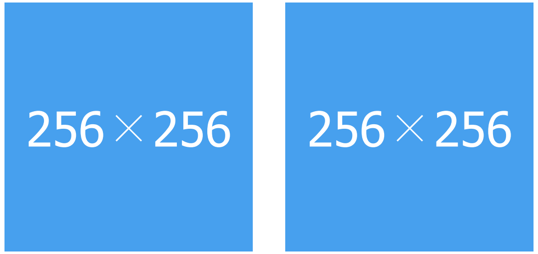
Evaluation: Microbenchmark

Impact of MME Reconfigurability on Compute Efficiency

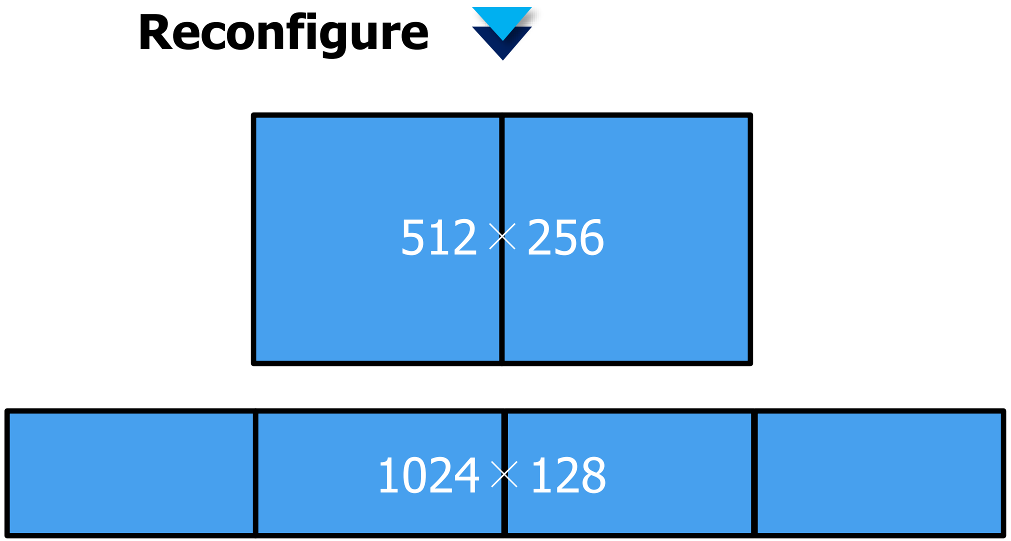
- M, K is fixed to 16,384



(c)



2 units of 256 × 256 MAC Engines

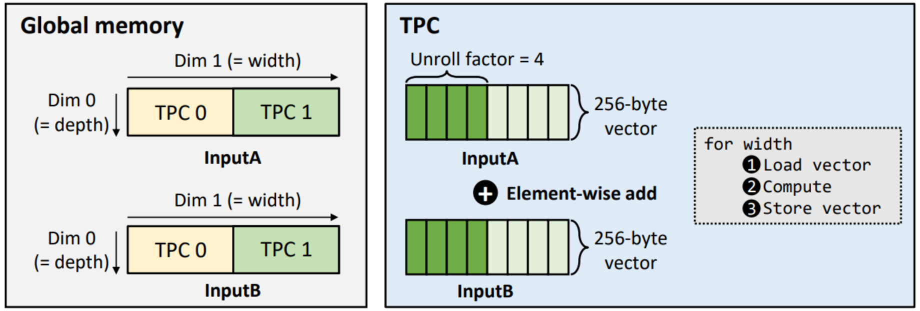


Evaluation: Microbenchmark

Micro-Architecture Optimization – TPC

Table 2: Evaluated microbenchmarks.

Microbenchmark		System	Implementation
Compute	GEMM	Gaudi-2	PyTorch API
		A100	PyTorch API
	non-GEMM	Gaudi-2	TPC-C
		A100	CUDA
Memory	Vector gather-scatter	Gaudi-2	TPC-C
		A100	CUDA
Communication	Collective communication	Gaudi-2	Intel HCCL [28]
		A100	NVIDIA NCCL [59]




```
void add_tpc(tensor inputA, tensor inputB, tensor outputC) {
  // Get index space information
  int5 InputCoord, OutputCoord;
  int  depthStart, depthEnd, widthStart, widthEnd = get_index_space_information();

  // A single step in the depth dimension is 256B / 4B (FP32) = 64
  int depthStep = 64; int depth_dimension = 0; int width _dimension = 1;

  // Declare input/output for 256-byte FP32 vectors (=float64)
  float64 x, y, result;

  for (int d = depthStart; d < depthEnd; d += depthStep) {
    InputCoord[depth_dimension] = d; OutputCoord[depth_dimension] = d;

    // Unroll factor is set as 4
    #pragma unroll(4)
    for (int w = widthStart; w < widthEnd; w += 1) {
      InputCoord[width_dimension] = w; OutputCoord[width_dimension] = w;

      // Fetches 256-byte vector from global memory
      x = v_f32_ld_tnsr(InputCoord, inputA); y = v_f32_ld_tnsr(InputCoord, inputB);

      // Element-wise vector add operation
      result = v_f32_add_b(x, y);

      // Stores the 256-byte vector to global memory
      v_f32_st_tnsr(OutputCoord, outputC, result);
      …
}
```

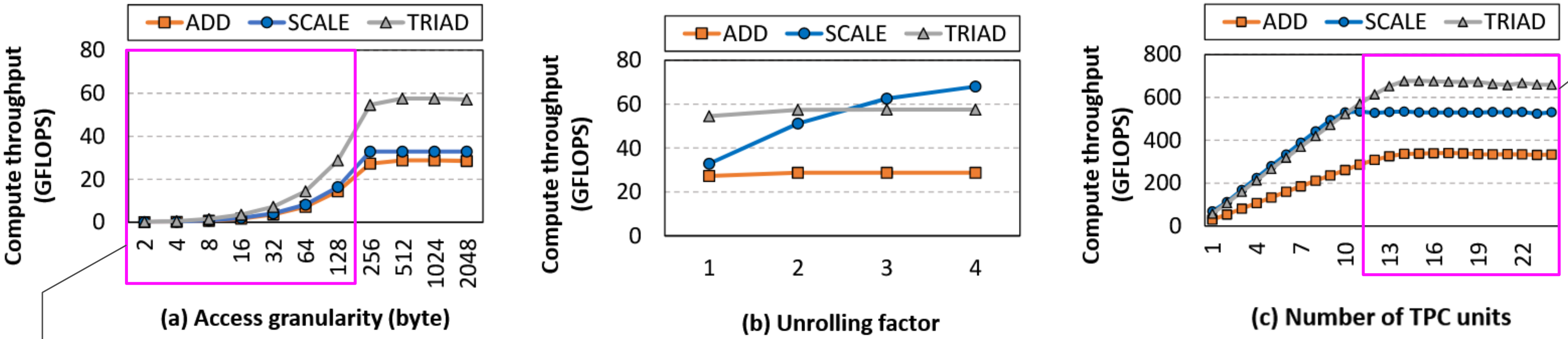
Figure 2 (c): add_tpc function using TPC-C

Evaluation: Microbenchmark

STREAM Benchmark & TPC Scalability Analysis

	Operation	Formula
STREAM	ADD	C = A + B
	SCALE	B = α • A
	TRIAD	C = A + α • B

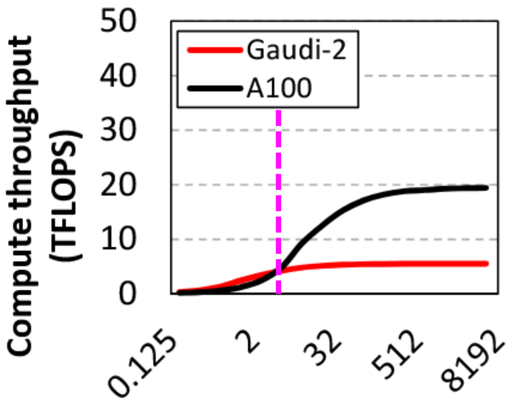
Performance saturates at 11–15 TPCs due to memory limits



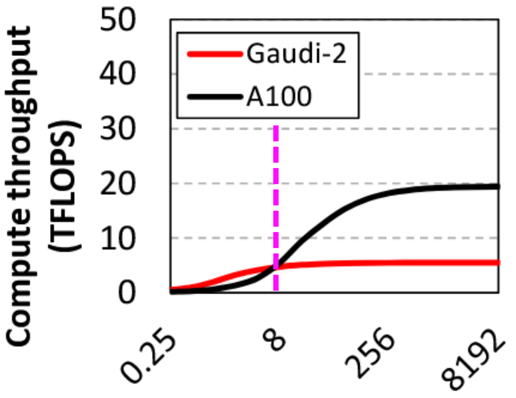
Throughput drops sharply below 256-byte granularity

Evaluation: Microbenchmark

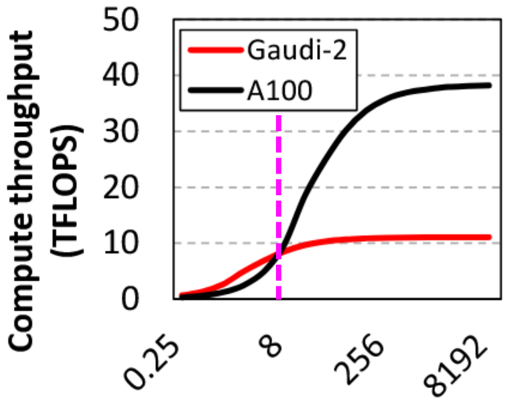
Compute Throughput Analysis via STREAM-based Operational Intensity



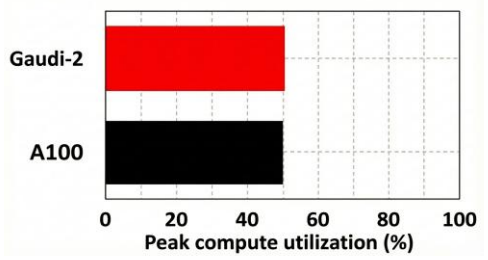
(d) Operational intensity (ADD)

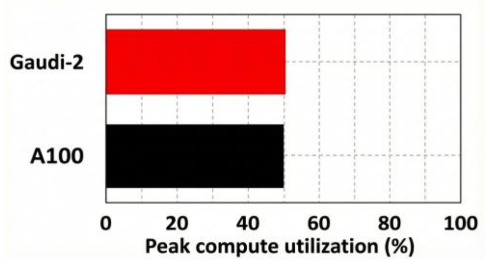


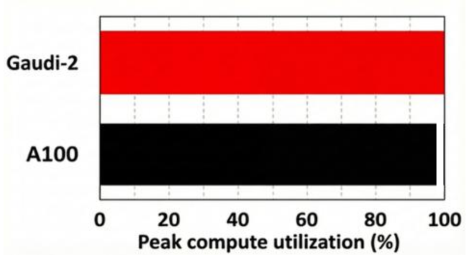
(e) Operational intensity (SCALE)



(f) Operational intensity (TRIAD)

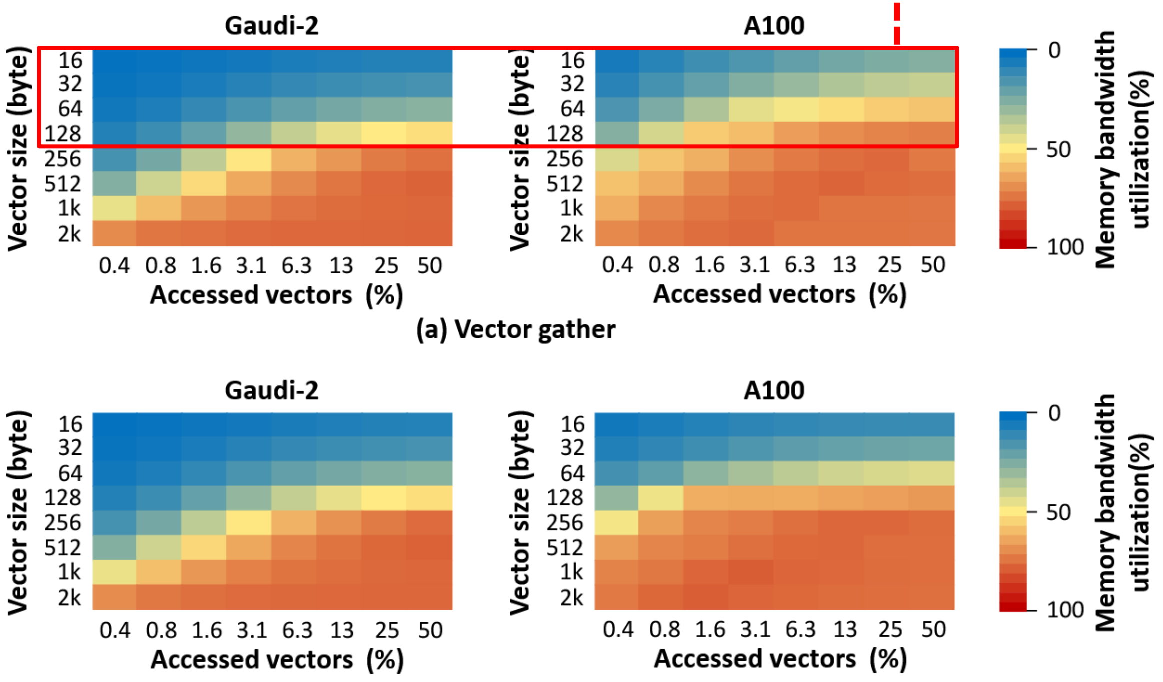






Evaluation: Microbenchmark

Random Memory Access Efficiency (Gather/Scatter)



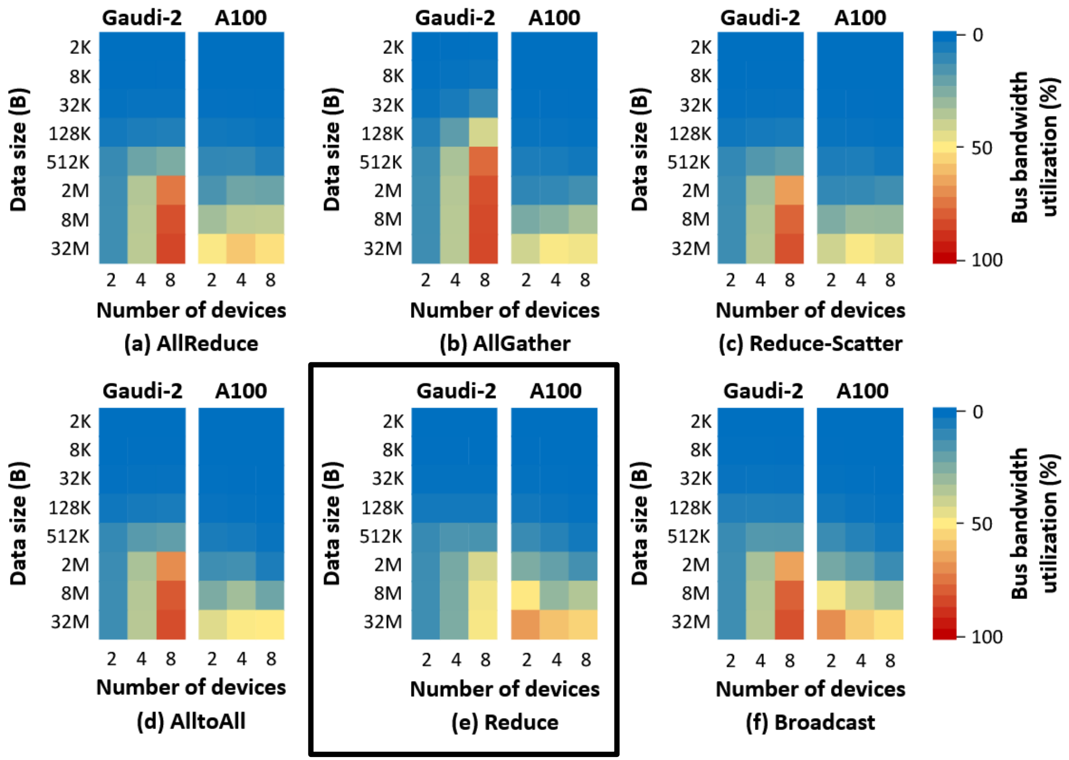
(a) Vector gather

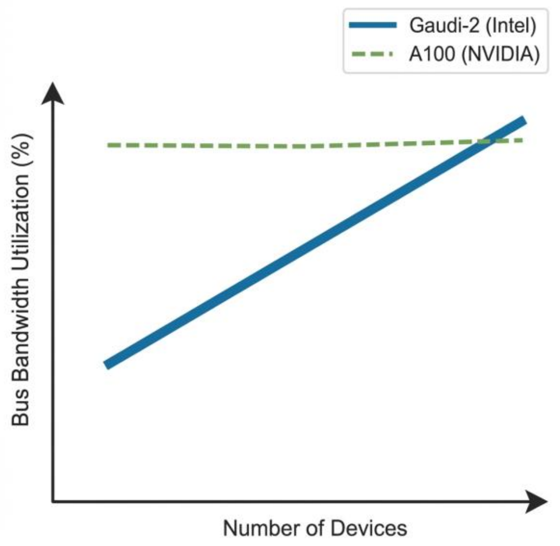
(b) Vector scatter

	Large Vectors (≥256 bytes)	Small Vectors (≤128 bytes)
Gaudi-2	64%	15%
A100	72%	36%
Gap	1.1×	**2.4×**

Evaluation: Microbenchmark

Inter-chip Communication





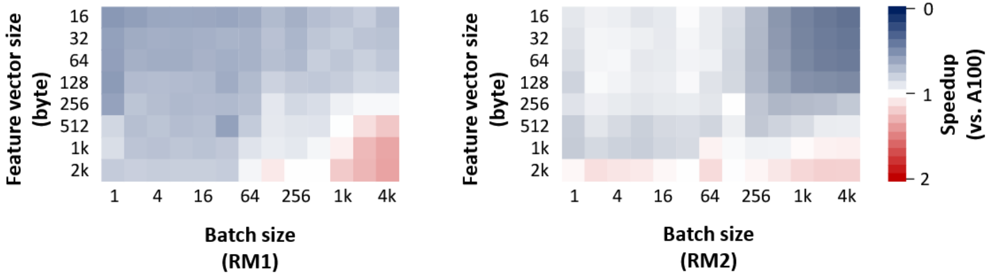
Evaluation: End-to-end Analysis

RecSys Workload: Performance & Energy-Efficiency Analysis

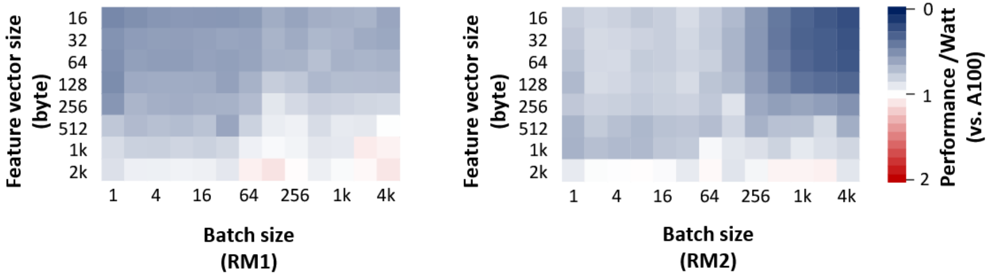
- Single-chip comparison only: Gaudi SDK lacks multi-device RecSys support.

Table 3: Evaluated end-to-end AI workloads.

Model		Embedding layer	MLP layer	Interaction layer
DLRM-DCNv2 [52]	RM1	# tables: 10 # embeddings: 1M # gathers: 10	Bottom: 512-256-64 Top: 1024-1024-512-256-1	Low rank dim: 512 # layers: 3
	RM2	# tables: 20 # embeddings: 1M # gathers: 100	Bottom: 256-64-64 Top: 128-64-1	Low rank dim: 64 # layers: 2



(a) Performance



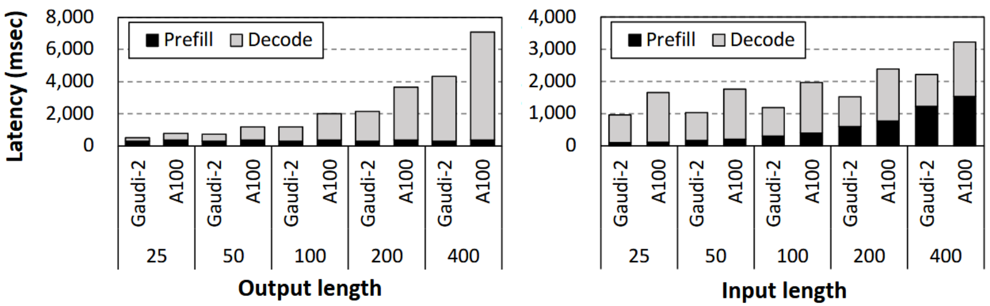
(b) Energy-efficiency

Performance (vs A100)	**-20%**
Energy-efficiency (vs A100)	**-28%**

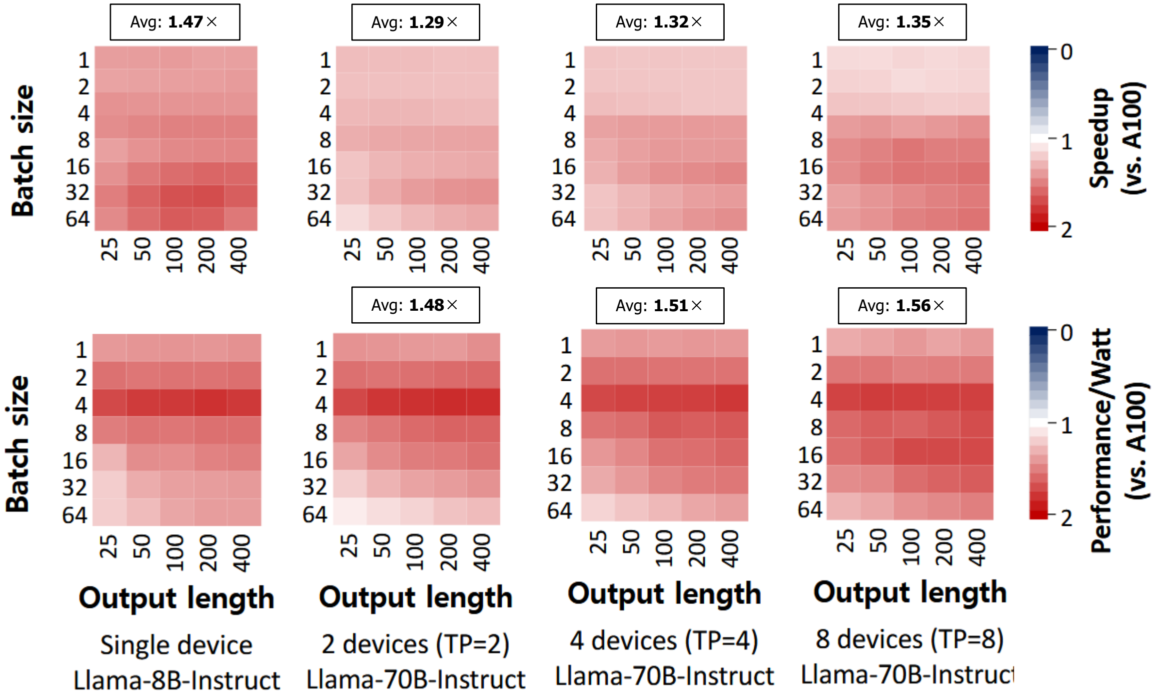
Evaluation: End-to-end Analysis

LLM Inference: Single & Multi-device Scalability & Latency Breakdown

Model		Embedding layer	Decoder layer
Llama-3.1 [12]	8B	# vocabularies: 128,256	# layers: 32 # heads for query: 32 # heads for key, value: 8 hidden/intermediate size: 4,096/14,336
	70B	# vocabularies: 128,256	# layers: 80 # heads for query: 64 # heads for key, value: 8 hidden/intermediate size: 8,192/28,672

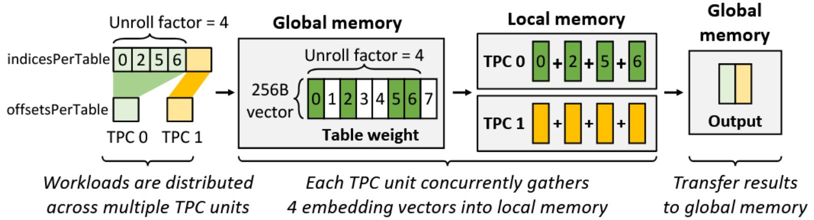


(b) Latency breakdown

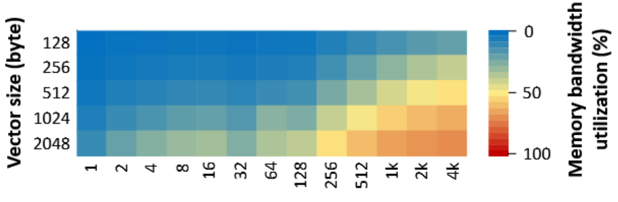


Evaluation: Programmability

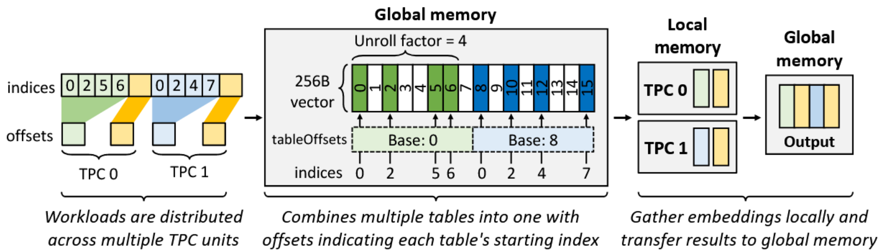
RecSys: Batched Embedding Lookup



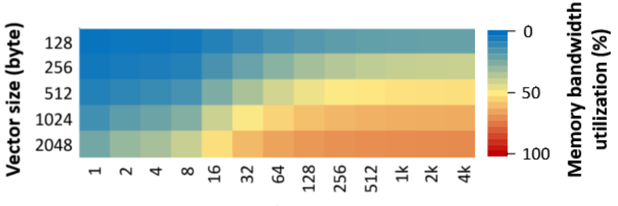
(a) `SingleTable` embedding lookup operation



SingleTable (Gaudi-2)
(b)

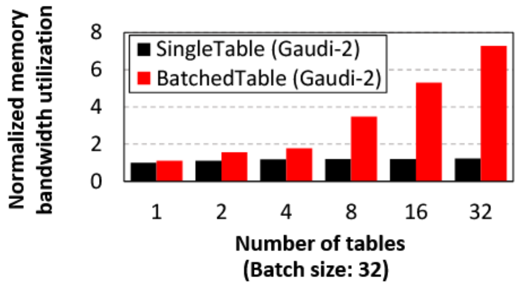


(b) `BatchedTable` embedding lookup operation



BatchedTable (Gaudi-2)
(c)

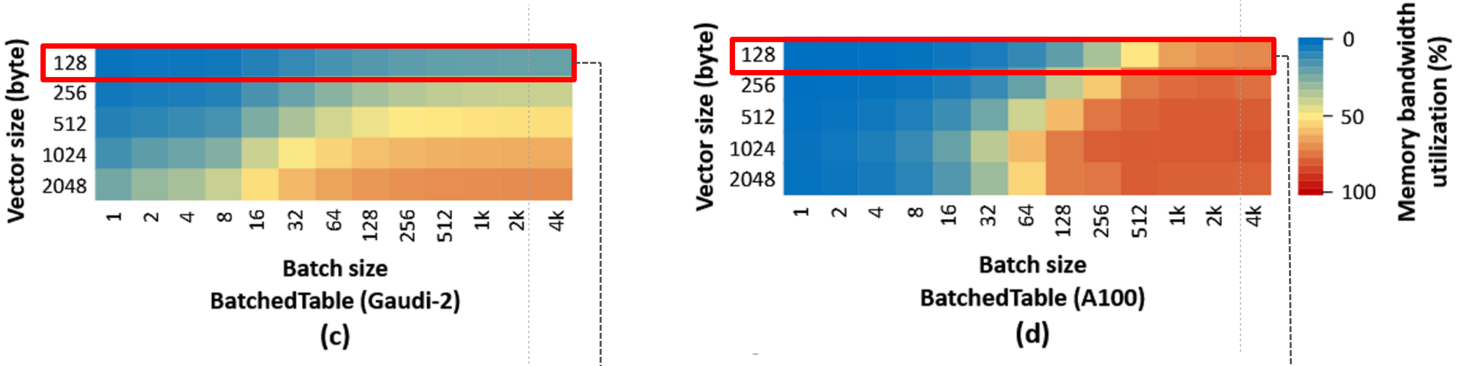
37% → 60% (vs. A100)



(a)

Evaluation: Programmability

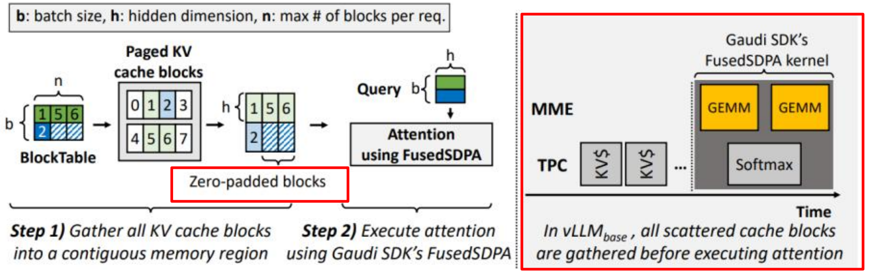
RecSys: Bandwidth Efficiency



	Gaudi-2	A100
Small Vectors (<128 bytes)	**12%**	**25.3%**
Avg	34.2%	38.7%
Peak	**70.5%**	**81.8%**

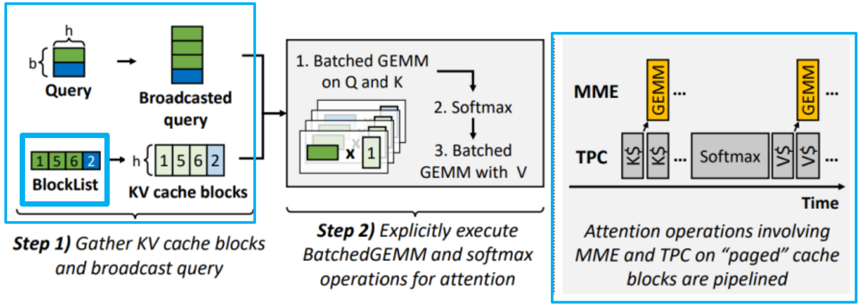
Evaluation: Programmability

LLM: Maximizing Gaudi Utilization via vLLM_opt



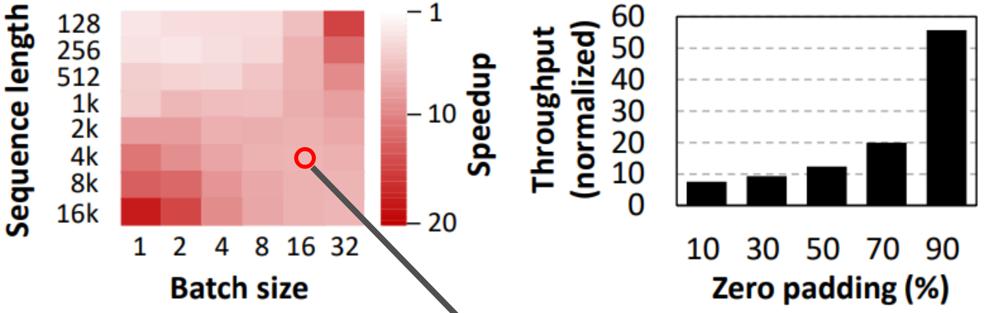
(a) $vLLM_{base}$ **Sequential**

PagedAttention Transformation



(b) $vLLM_{opt}$ **Pipelined**

normalized to vLLMbase



(a) **(b)**

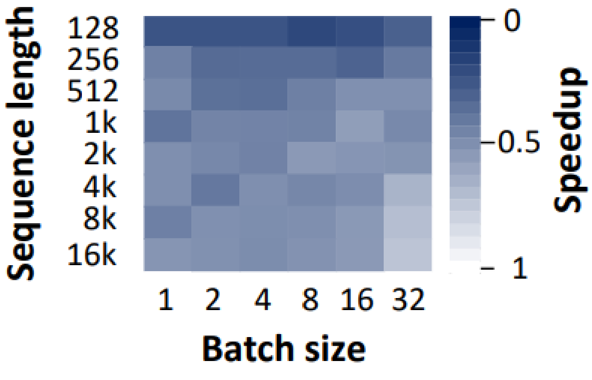
Avg. Speedup(vs. $vLLM_{base}$): 7.4x

Condition:
Batch size=32, Sequence length=4K

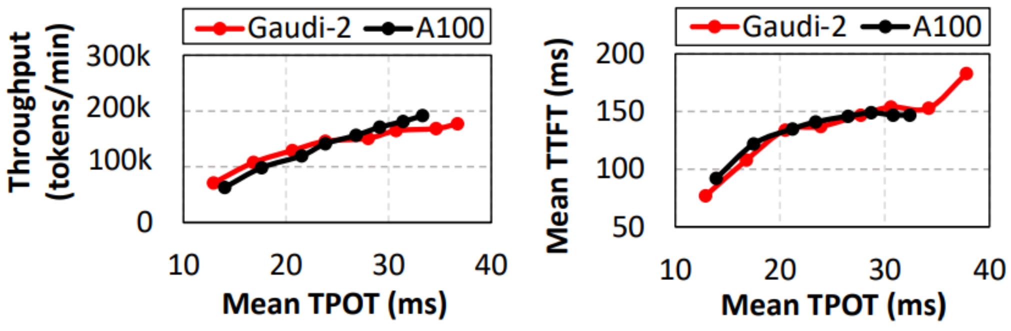
Throughput Improvement:
21x (Avg) ~ 55.7x (Max)

Evaluation: Programmability

LLM: Bridging the Gap Between Kernel and System Performance



(c)



(d) (e)

Relative PagedAttention Throughput 45% of A100

Similar End-to-End Performance

Conclusion

Conclusion

Performance Characterization: Gaudi-2 vs. A100

Category	Evaluation Item	Outcome (Gaudi-2 vs. A100)	Key Insights / Root Causes
Microbenchmark	GEMM (Matrix)	Gaudi-2 Superior	High utilization via flexible **MME architecture**
	Non-GEMM (Vector)	A100 Superior	Limited vector perf. Gaudi-2 (11 TFLOPS) vs. A100 (39 TFLOPS)
	Memory / Comms.	A100 Superior	Bottleneck due to **256B access granularity** constraints
Workload	LLM (Inference)	Gaudi-2 Dominant	**GEMM efficiency offsets vector weakness.** Achieves **1.47x speedup** & **1.5x efficiency**
	RecSys (DLRM)	A100 Superior	Vulnerable to fine-grained memory access patterns