JIT-Q: Just-in-time Quantization with Processing-in-Memory for Efficient ML Training

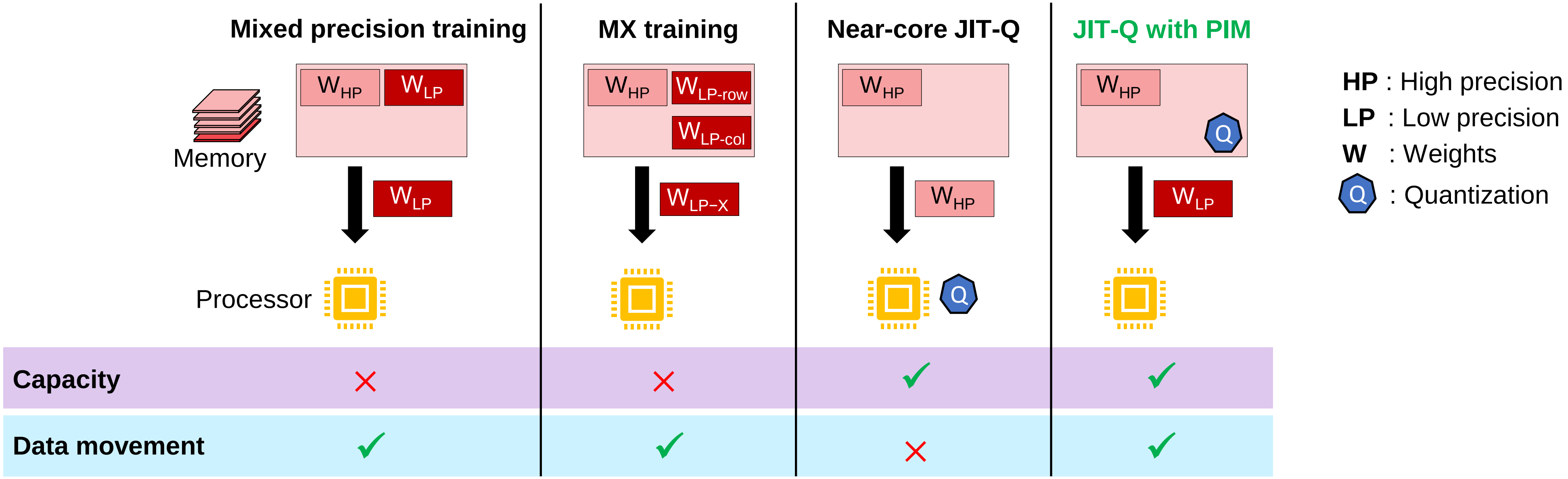
Mohamed Assem Ibrahim, Shaizeen Aga, Ada Li, Suchita Pati, and Mahzabeen Islam

MLSys 2024

AMD
together we advance_

Executive Summary

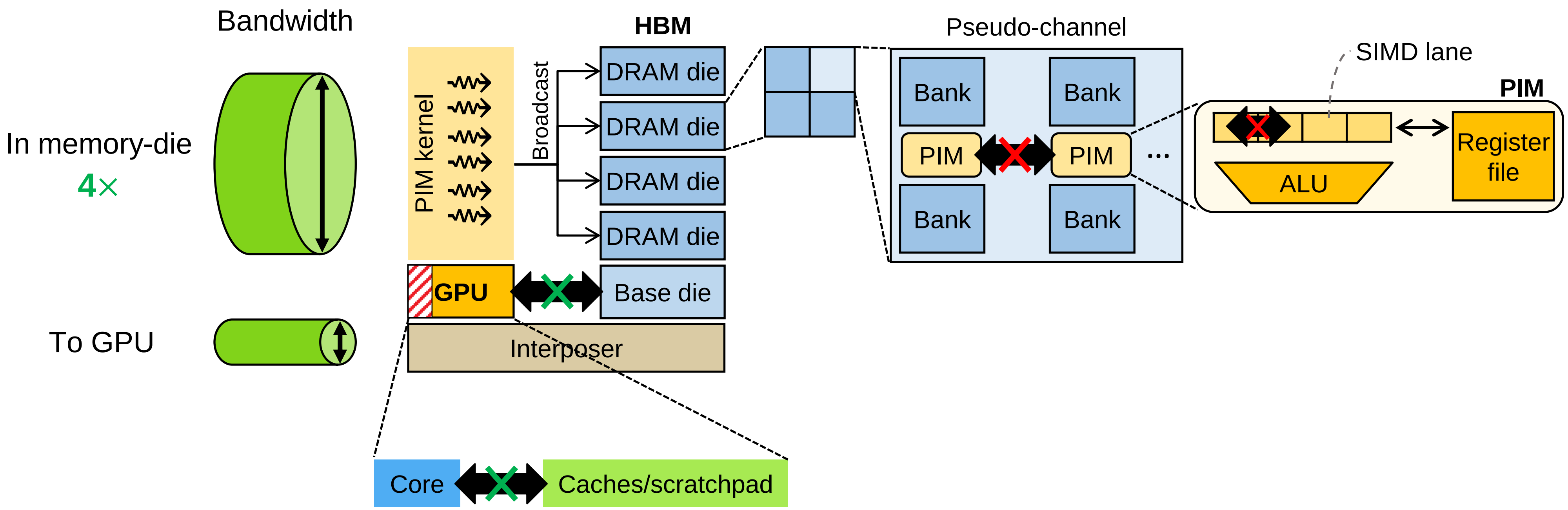
- **Problem:** Weights redundancy in mixed precision training
  - Memory capacity pressure



- **Proposal:** Just-in-time quantization (JIT-Q) with PIM
  - Memory capacity savings of up to 24% → Larger models, larger batch-sizes, lower model parallelism, etc.

AMD
together we advance_

Processing-in-Memory (PIM)



1. Harness higher memory bandwidth
2. Save data movement energy

1. No inter-bank communication
2. No cross-SIMD compute
3. Interference between concurrent PIM and GPU execution

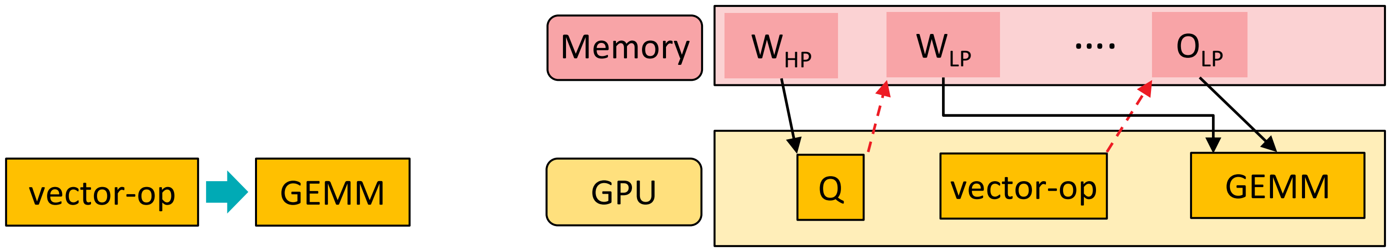
Opportunity for Capacity Savings

- Weights maintained in both high and low precision during training
  - Multiple low precision copies with directional numeric formats

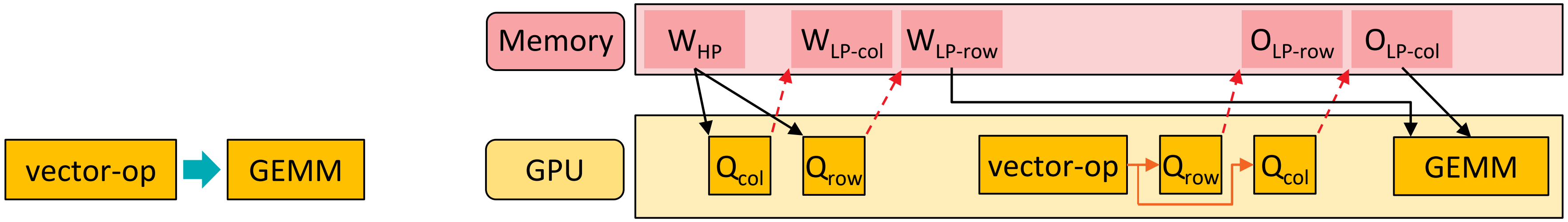
Legend:

Tensor in memory | GPU Compute | **HP**: High precision **LP**: Low precision **W**: Weights **O**: Output **Q**: Quantization

Mixed precision training (FP32/BF16)



Training with MX quantization



Opportunity for Capacity Savings → JIT-Q with PIM

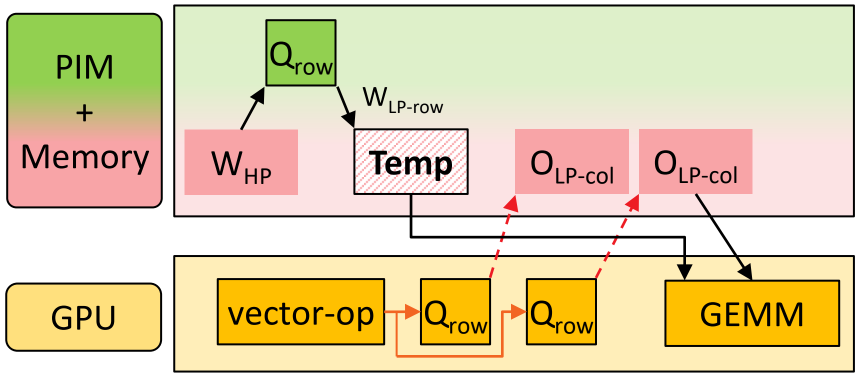
- Avoid storing low precision weights via **just-in-time quantization (JIT-Q) with PIM**
  - Overlap quantization on PIM with preceding GPU operation
  - Advantage: Capacity savings
    - Train larger models, enable larger batch-size, reduce model parallelism, etc.

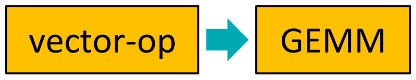
Legend:

Tensor in memory | **Temporary tensor in memory** | GPU Compute | PIM Compute

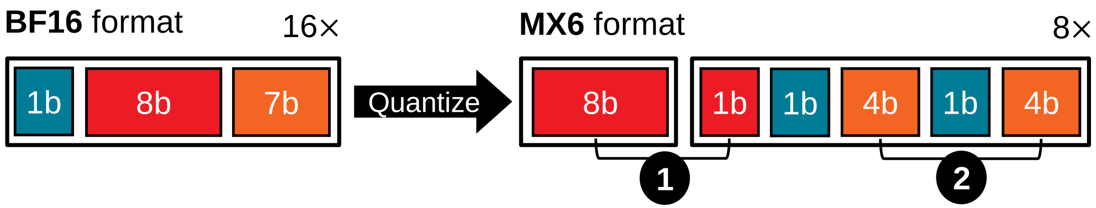
HP: High precision **LP**: Low precision **W**: Weights **O**: Output **Q**: Quantization

Training with JIT-Q on PIM



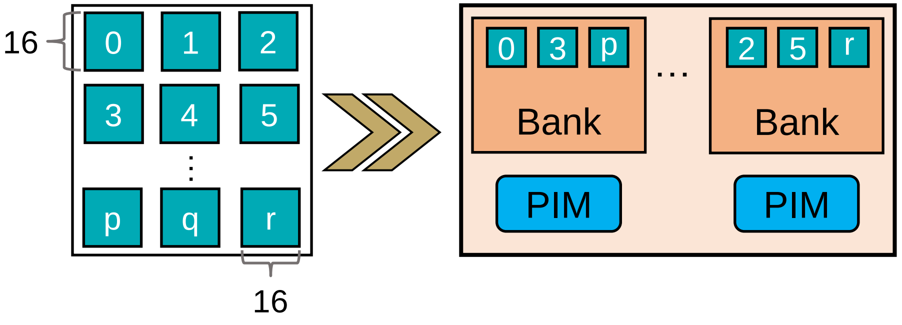


PIM Quantization Kernel Considerations



Data-mapping

- **Tiled data-mapping** to support row and column quantization
  - Avoid inter-bank compute → Map tile of input weight tensor to a single bank
  - Avoid cross-SIMD compute → Map each element of a given tile to the same SIMD lane



PIM Compute

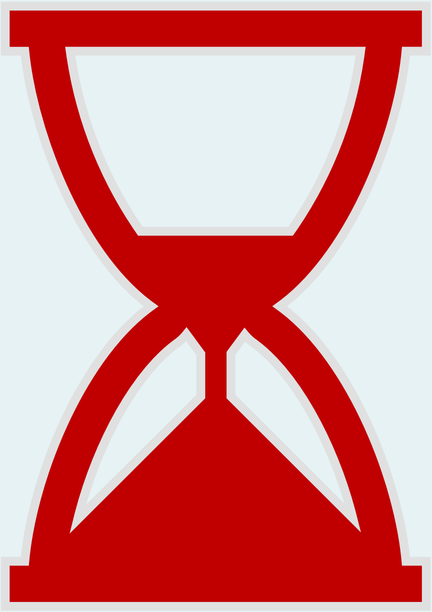
- **PIM ALU augmentations** to realize quantization
  - Support for lane-specific shifts and conditional execution (e.g., with mask)

1. Deduce shared exponents (e.g., max)
   Augmentation: Masked compare
2. Adjust mantissa bits (e.g., conditional shift)
   Augmentation: Intra-lane conditional shift

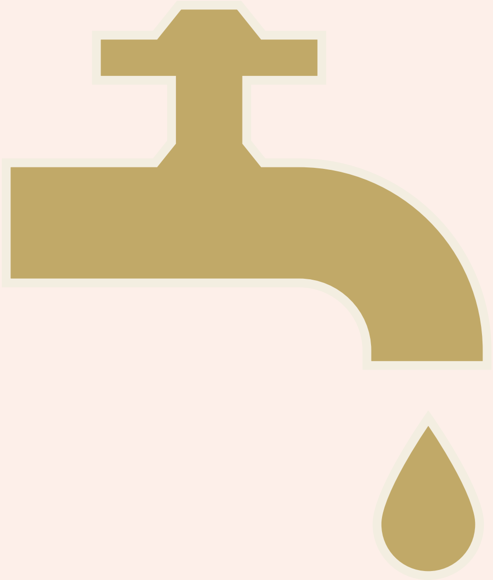
AMD together we advance_

Key Evaluation Questions

Is there *slack* for PIM to quantize the weights JIT?



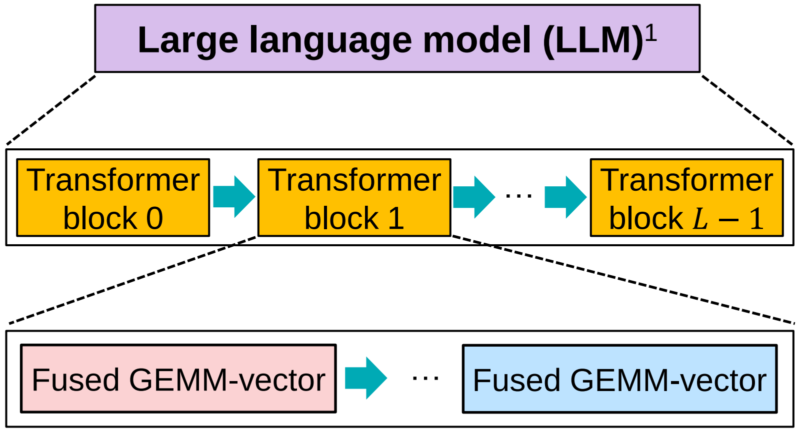
What is the effect of JIT-Q on *training throughput*?

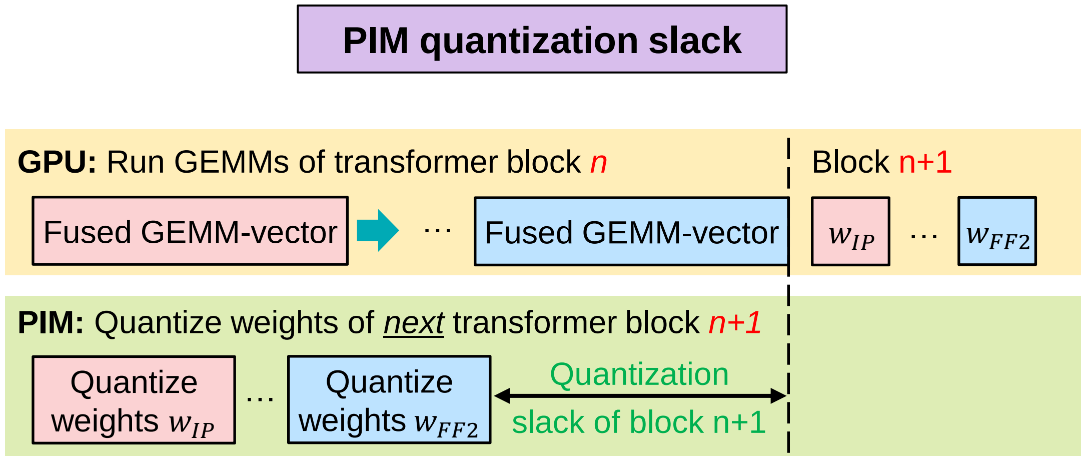


What are the *capacity savings* of JIT-Q with PIM?

Modeling JIT-Q Slack

- ***Slack*** = Computation in transformer block – PIM quantization for weights of next transformer block
  - **GPU performance model** = max (compute time, memory time)
    - Compute time = GEMM ops at peak compute throughput
    - Memory time = Time to read quantized GEMM inputs at peak memory bandwidth
  - **PIM performance model** = Detailed DRAM commands to realize quantization
  - Model next transformer block quantization for simplicity

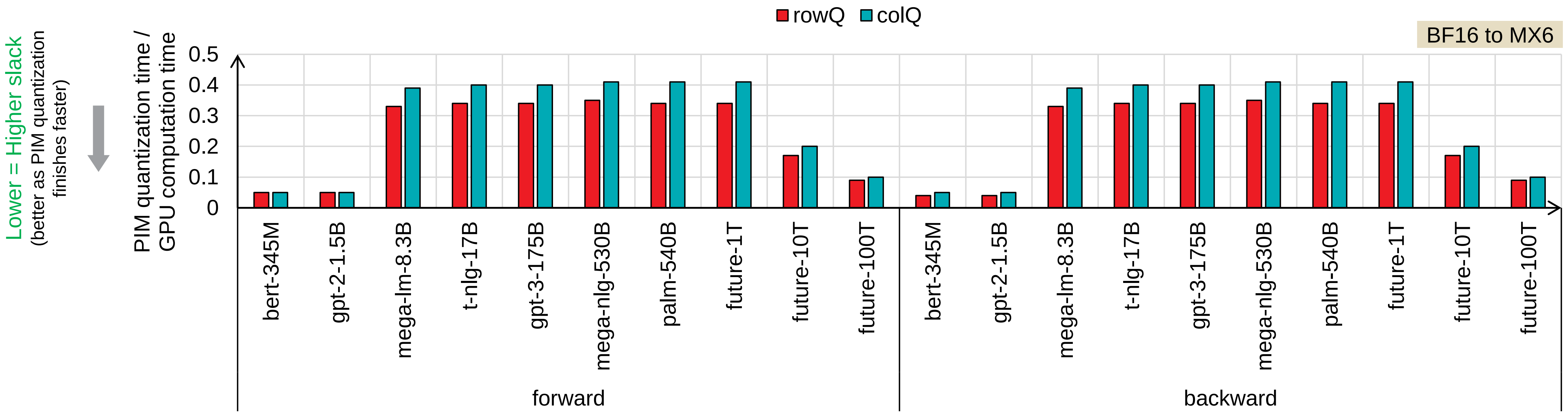




[1] We model computations that are representative of BERT, Megatron-LM, GPT3, PaLM, etc.

Quantization with PIM exhibits sufficient slack to be just-in-time

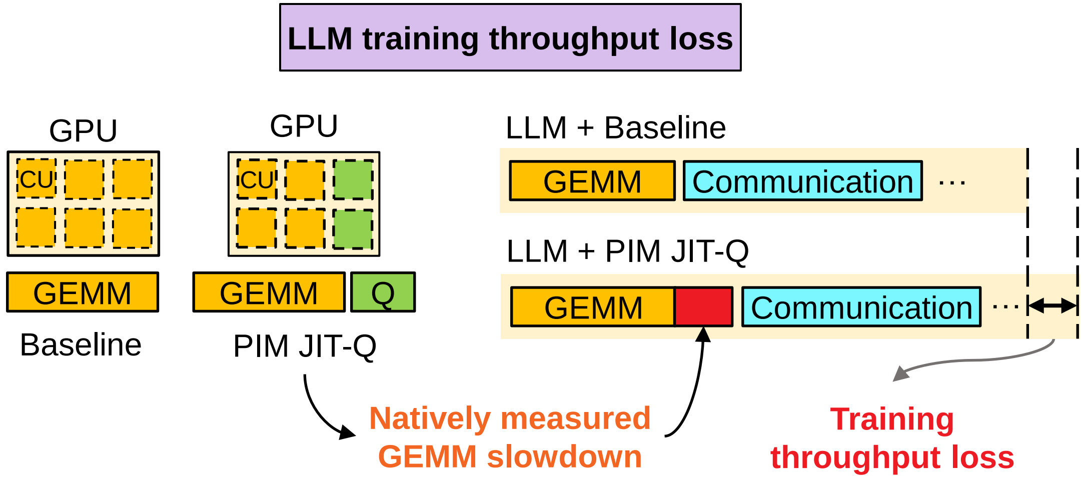
- Both forward and backpropagation have enough slack for PIM to complete quantization
- Column quantization has lower slack vs. row quantization due to additional DRAM row opens
- Pushing precision lowers PIM slack **BUT** enough slack still available for PIM JIT-Q

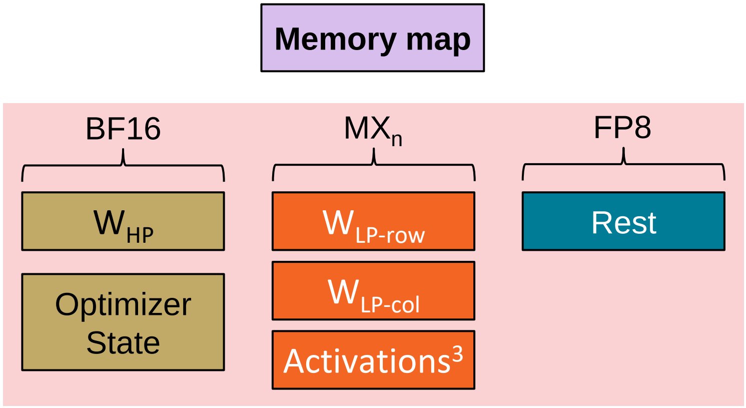


Modeling Throughput Effect and Capacity Savings

- **Throughput:** JIT-Q necessitates concurrent GPU/PIM execution
  - GPU compute units to orchestrate PIM computation cause GEMM slowdown
    - Offloading PIM orchestration away from GPU can prevent this slowdown
  - Assess training throughput loss[1] via GEMM slowdown measured natively

- **Capacity:** FP8 mixed precision training setup[2]



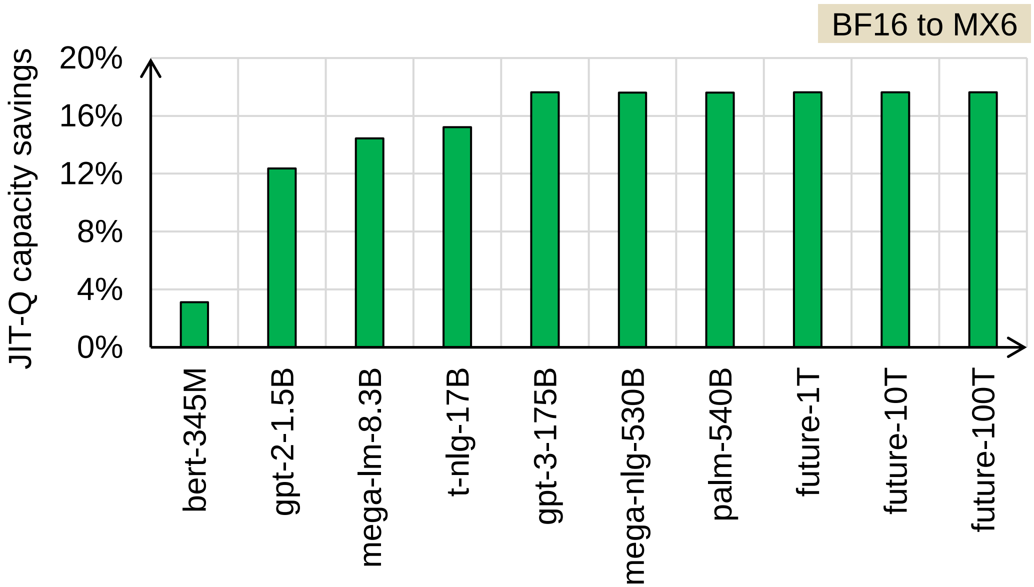


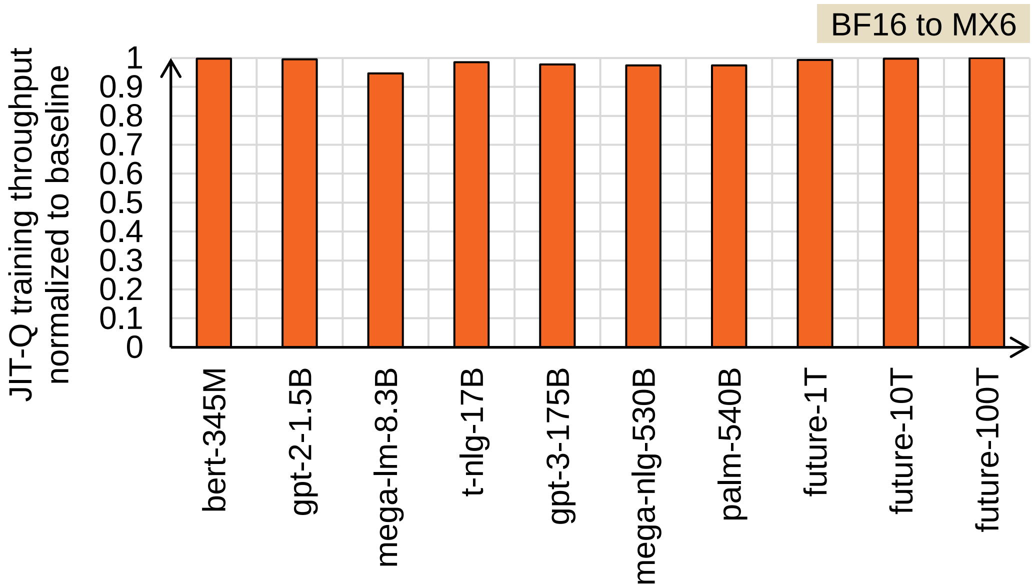
[1] https://arxiv.org/abs/2302.02825
[2] https://arxiv.org/abs/1905.12334
[3] https://arxiv.org/abs/2205.05198

PIM JIT-Q → Capacity savings at small training throughput loss

- MX6 → Capacity savings = 12.5% , Throughput loss = 1.6%
- Harnessing capacity savings – Example LLM GPT3-175B
  - Train 20% larger model or 12.5% lower tensor-slicing degree
- Capacity savings/throughput effects dependent on target MX format

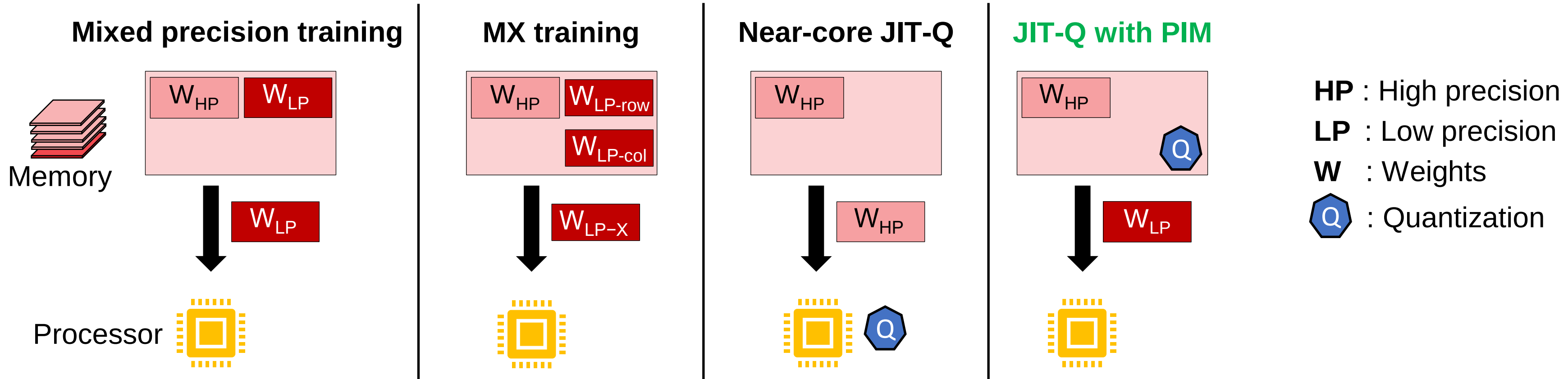
More results in the paper





AMD
together we advance_

Conclusion



	Mixed precision training	MX training	Near-core JIT-Q	JIT-Q with PIM
Capacity	✗	✗	✓	✓
Data movement	✓	✓	✗	✓

- JIT-Q – PIM is interesting for ML
  - Quantize weight tensors JIT with PIM → Avoid storing low-precision weight tensors in memory
  - JIT-Q with PIM has sufficient slack vis-à-vis GPU compute
  - PIM JIT-Q delivers capacity savings at marginal throughput loss

COPYRIGHT AND DISCLAIMER

AMD