YONSEI UNIVERSITY

GradPIM: A Practical Processing-in-DRAM Architecture for Gradient Descent

Authors: Heesu Kim, Hanmin Park, Taehyun Kim, Kwanheum Cho, Eojin Lee, Soojung Ryu, Hyuk-Jae Lee, Kiyoung Choi, and Jinho Lee

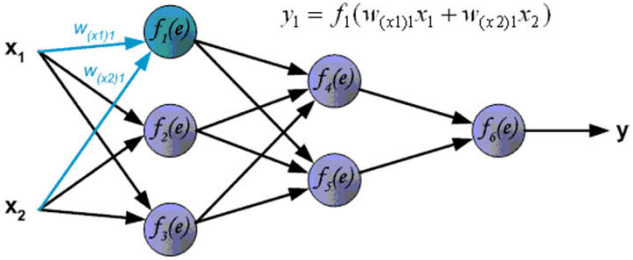
Presenter: Heesu Kim

What is GradPIM?

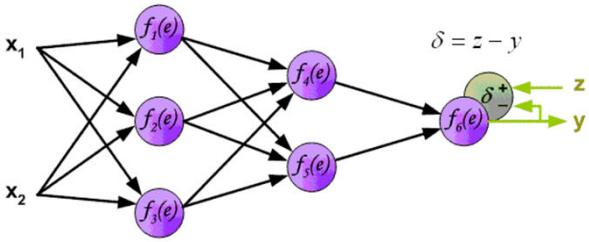
- **GradPIM** = **Grad**ient descent with processing-in-memory (**PIM).**
- **What**: Accelerating gradient descent (GD) algorithm for deep neural network training.
- **How**: Isolating memory traffics of GD from I/O between host and memory.

Background - Gradient Descent (GD)

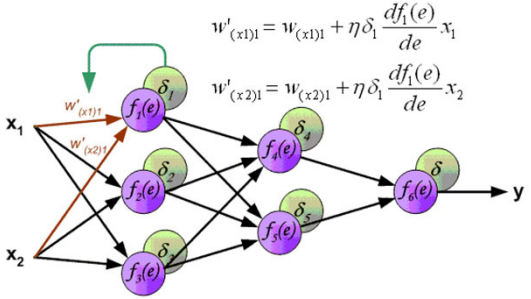
- Algorithm for training parameters (weights) of deep neural network.
- Three steps: Forward (Fwd) → Backward (Bwd) → Weight Update (Wup).
  - Fwd: compute an error.
  - Bwd: propagate error-gradients.
  - Wup: update weights with gradients.



Fwd



Bwd



Wup

Figures from https://medium.com/datadriveninvestor/what-is-gradient-descent-intuitively-42f10dfb293f

Background – Weight update (Wup) in GD

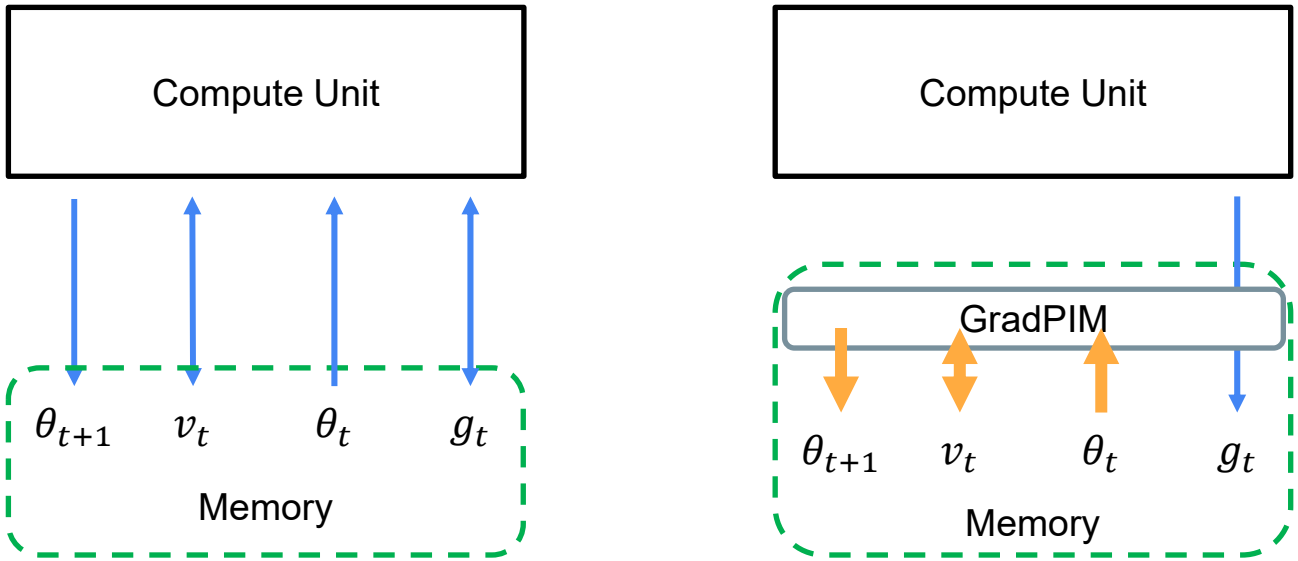
- Wup is **memory-intensive** (**Low Ops/byte**).
  - RD/WR: multiple values, Ops: a few MAC operations.
- Wup is appropriate target for PIM.
  - PIM provides high bandwidth with relatively low computing power.
- Note: Wup can be processed in vectorized form (**element-wise**).

$$\overrightarrow{\theta_{t+1}} = \alpha\overrightarrow{v_t} + \beta\overrightarrow{\theta_t} + \overrightarrow{g_t}$$

new weight momentum weight gradient

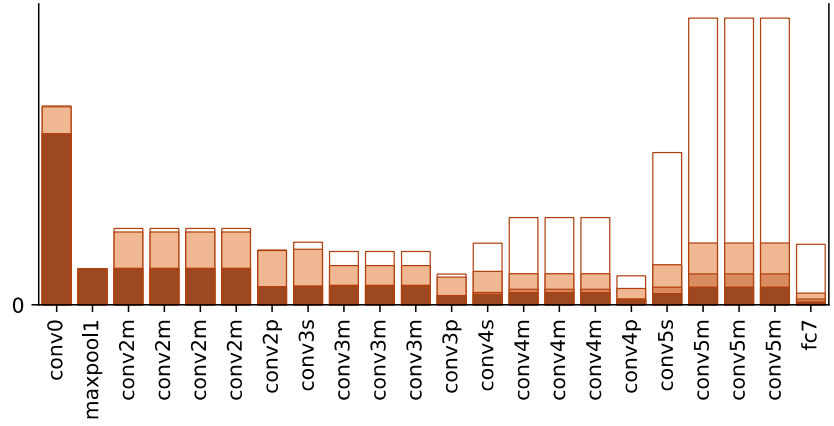
GradPIM accelerates Wup

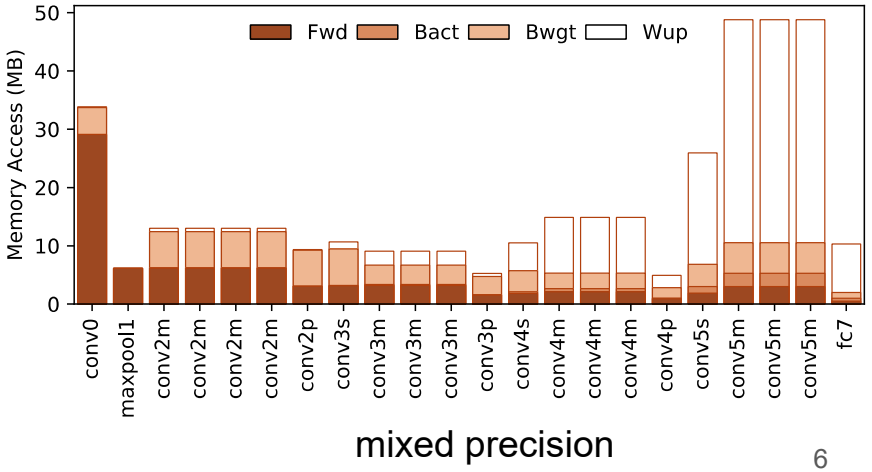
- GradPIM isolates memory traffics for Wup.
  - Use Internal bandwidth instead of limited external bandwidth.



Expected gain from GradPIM

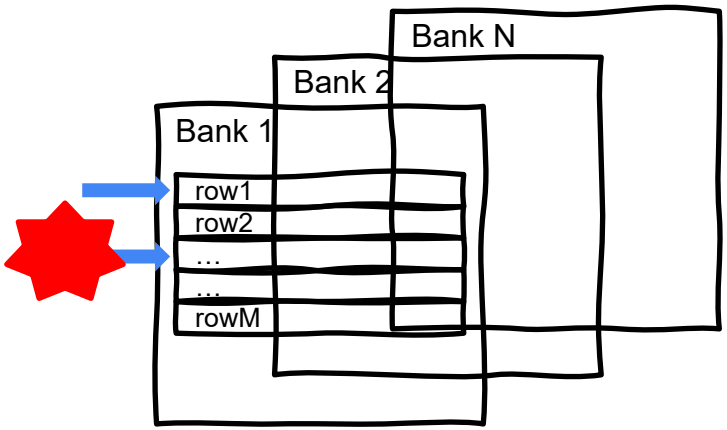
- GradPIM reduces the memory traffics in “Wup”.



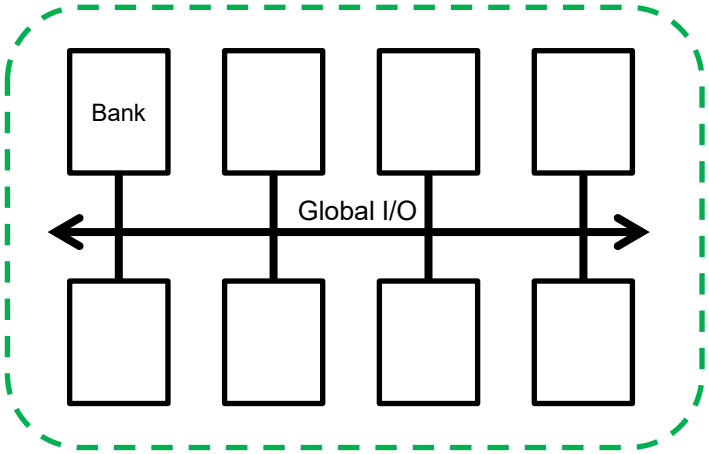


Background - DRAM Architecture

- Bank conflict: consecutive requests toward different rows in a bank.
  - Only a row can be activated at a time.
- Global I/O limits memory bandwidth.
  - Shared by all banks of a device.
  - Peak memory bandwidth < Σ bank's internal bandwidths.



Bank conflict



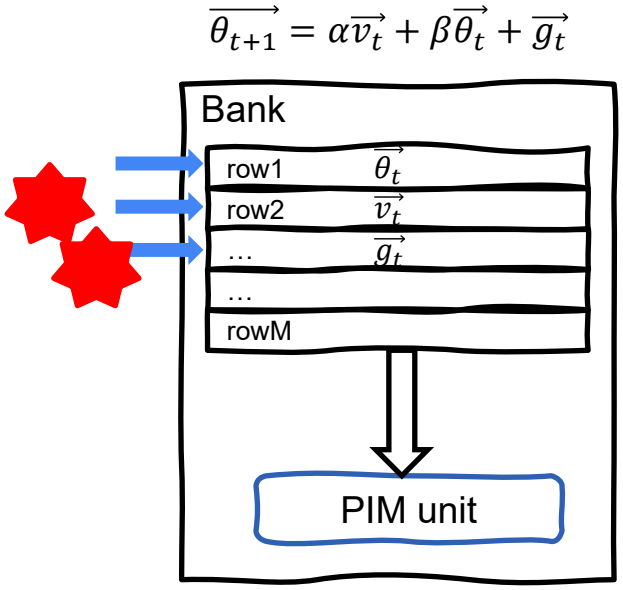
Sharing a global I/O among banks

Design rules for GradPIM

- (Compatibility) Compatible with DDR protocol.
  - Memory controller has a complete control over GradPIM (passive device).
  - Use the reserved command (RFU) of DDR protocol.
- (Compatibility) Non-invasive to memory cell-arrays.
  - Modifying a cell-array is too radical in view of process.
- (Performance) Utilizes local I/O while being decoupled to global I/O.

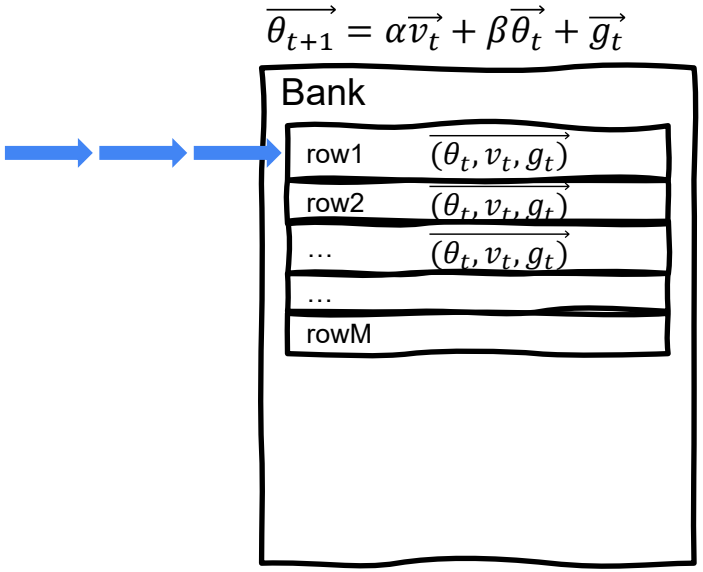
Naïve PIM design – Bank-level parallelism

- Place GradPIM near each bank I/O.
  - To use internal bandwidth of each bank.
- Put data-arrays on different rows of a bank.
  - Accessing data-arrays for weight update → **Lots of bank conflicts**.



Naïve PIM design – Bank + Array of Structure

- Change the array layout to “array of structure” format.
  - Interleaving multiple data-arrays $\overrightarrow{\theta_t}$, $\overrightarrow{v_t}$, $\overrightarrow{g_t}$, to a single data-array $\overrightarrow{(\theta_t, v_t, g_t)}$.
- No bank conflicts.
  - Accessing consecutive columns in a row.
- However, degraded bandwidth in forward and backward steps.



$\overrightarrow{(\theta_t, \cancel{v_t}, \cancel{g_t})}$

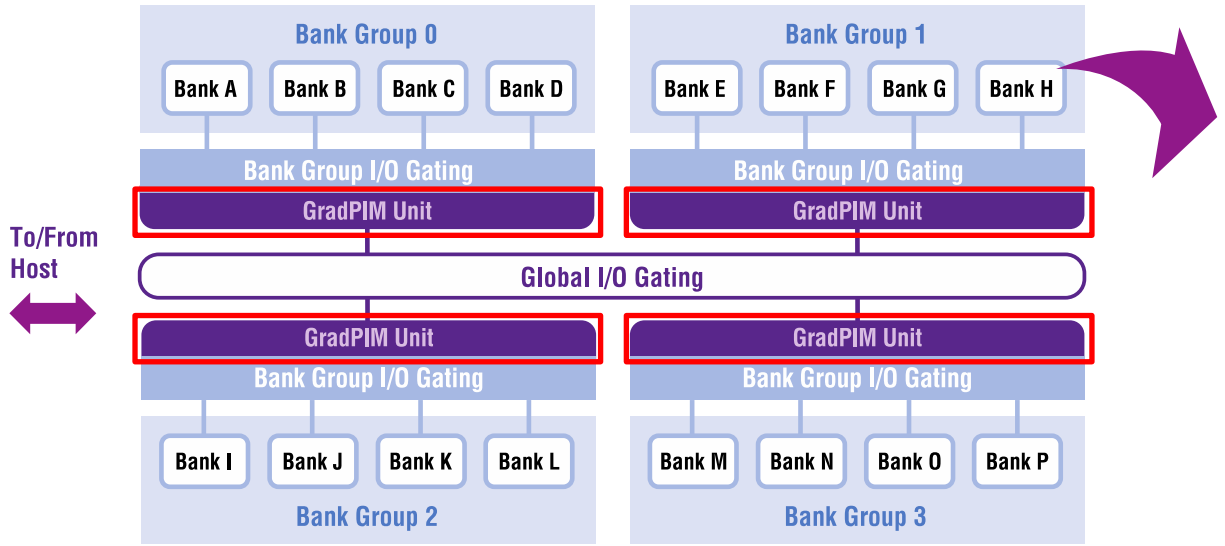
Required data in forward.

$\overrightarrow{(\cancel{\theta_t}, \cancel{v_t}, g_t)}$

Required data in backward.

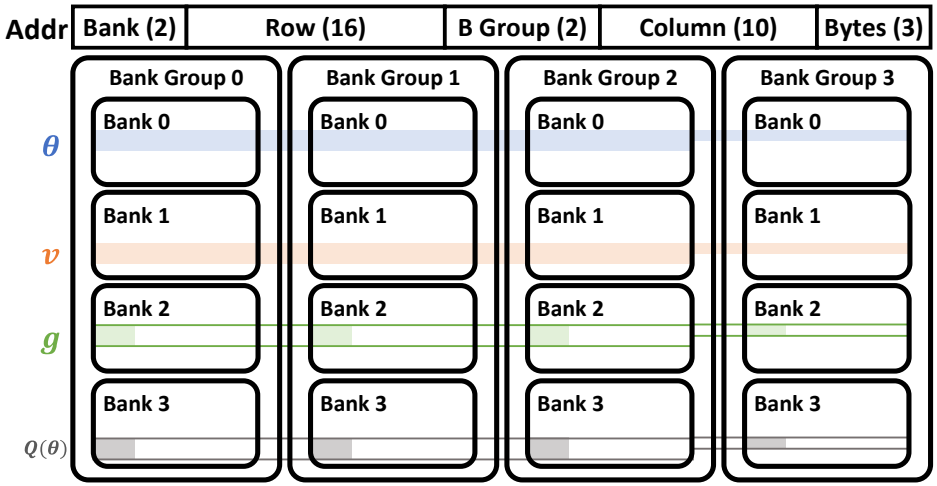
GradPIM – Bank-group level parallelism 1.

- Place GradPIM units near each bank-group I/O gating.
  - To use internal bandwidth of each bank-group.
  - Note: a bank-group has four banks in DDR4.



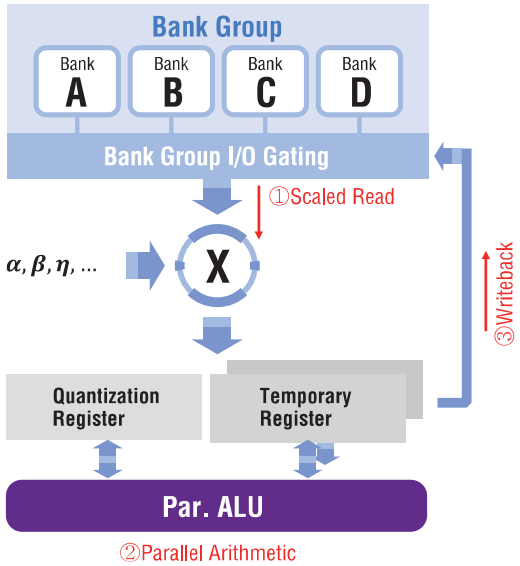
GradPIM – Bank-group level parallelism 2.

- Put data-arrays on different banks within a bank-group.
  - Different banks: **No bank conflict.**
- Address Mapping: Bank-ID is in MSB.
  - Each value spreads over same bank-ID.



GradPIM – Unit Architecture

- Components: Registers (2 for temp, 1 for quant), Scalers, and Parallel arithmetic units.
- Supported operations:
  - Scaled read: read through scaler. (Scale a value by $2^m \pm 2^n$)
  - Parallel operations: element-wise add/sub and quant/dequant.
  - Writeback: Store the value from registers into memory cells.



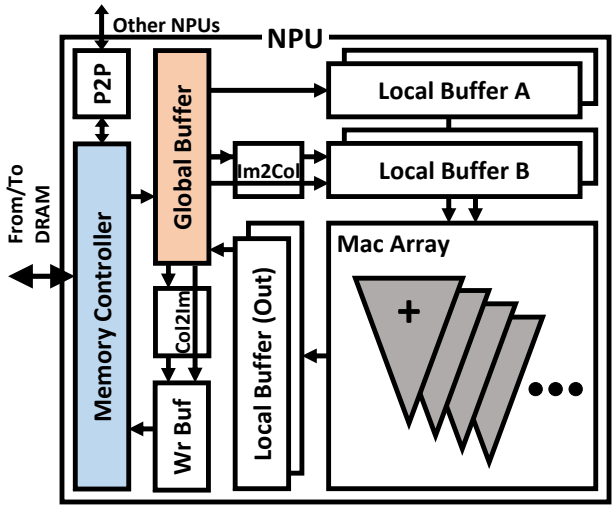
GradPIM – Commanding

- Use RFU (Reserved for Future Use) command of DDR4 protocol.
  - Only **5 bits** are left us to control GradPIM.
- Truth table
  - Scale ID: Scaler's hyper-param.
  - Src/Dst Position: offset in a register.
  - Src/Dst: A bit is enough to identify a register out of **two registers**.

Signal / Func.	Op0	Op1	Param0	Param1	Src/Dst
Scaled Read	L	L	Scale ID		Dst
DeQuant	H	L	Src Position		Dst
Quant	H	H	Dst Position		Src
Writeback	L	H	L	L	Src
Q. Reg	L	H	H	L	RD/WR
Add	L	H	H	H	Dst
Sub	L	H	L	H	Dst

Experiment setup – Hardware

- Neural Processing Unit (NPU): 256x256 multiplier-adder trees.
  - 128 TOPS @ 1GHz, Nangate 45nm.
- Memory: DDR4-2133 with 4 ranks (x8 8Gb DDR4-SDRAM).
- GradPIM: Layout in 32nm (scaled from 45nm) with 3-metal layers.
  - Area: Equivalent to 1Mb DRAM cells.



Module	Area (μm^2)	Power (mW)
Adder	320.1	0.058
Quantize	275.4	0.056
Dequantize	244.8	0.041
Scaler	606.1	0.159
Registers ($\times 3$)	206.7	0.04
Total	8267.8	1.74

Experiment setup – Simulation & Networks

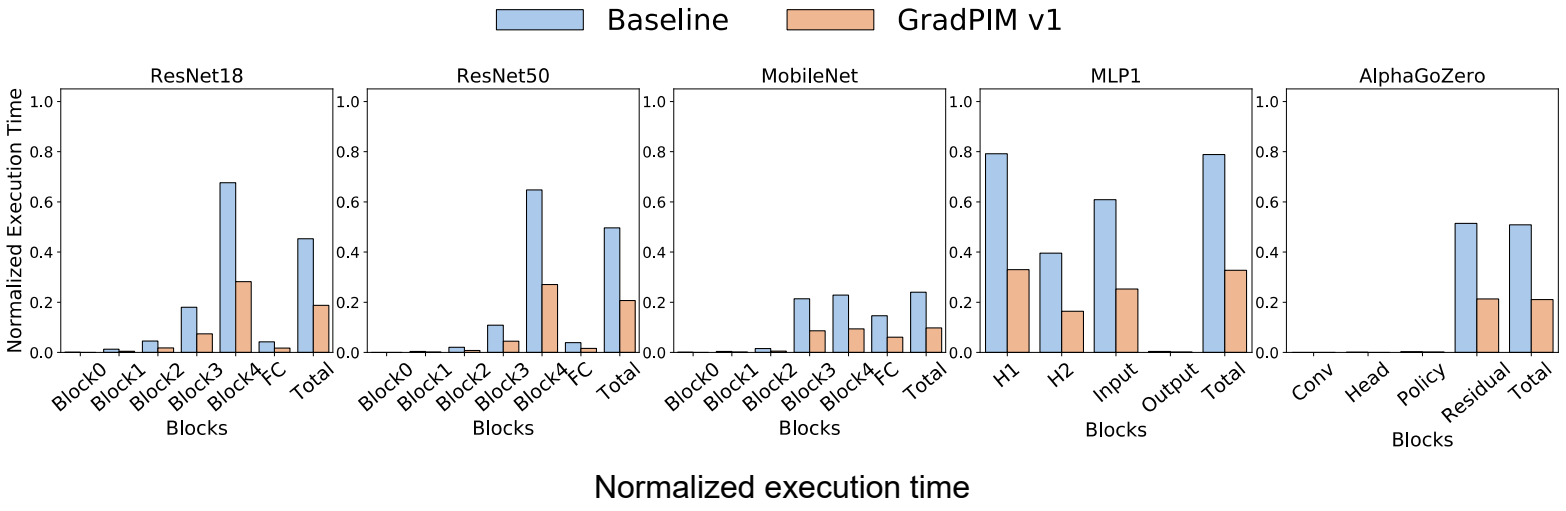
- In-house simulator with DRAMSim3*.
  - SystemC based cycle-accurate simulator.
  - Timing and energy of GradPIM are modeled in DRAMSim3.

- Target deep neural networks.
  - ResNet18 and ResNet50 – Convolutional layers.
  - MobileNet – Light-weight network for mobile devices.
  - MLP1 – Fully-connected layers.
  - AlphaGoZero – Very deep convolutional layers for GO game.

*S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, “DRAMsim3: a cycle-accurate, thermal-capable DRAM simulator,” *Computer Architecture Letters*, 2020.

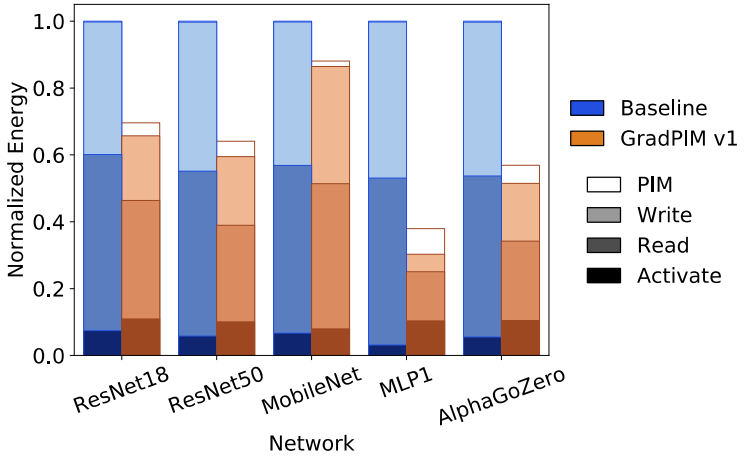
Experimental result – Execution time

- Baseline performs Wup in NPU.
- GradPIM.v1 achieves **2.25X** speedup in Wup, thus **1.38X** overall speedup.
  - “Filled” part for Wup and “grayed part” for the others (Fwd and Bwd).
  - Bigger portion of Wup → Higher overall gain.



Experimental result – Energy consumption

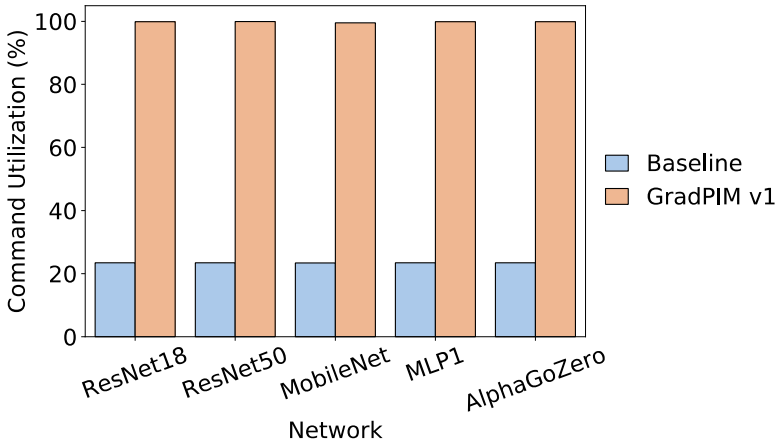
- GradPIM.v1 consumes extra energy for PIM.
- GradPIM.v1 shows less energy consumption with reduced global I/O.

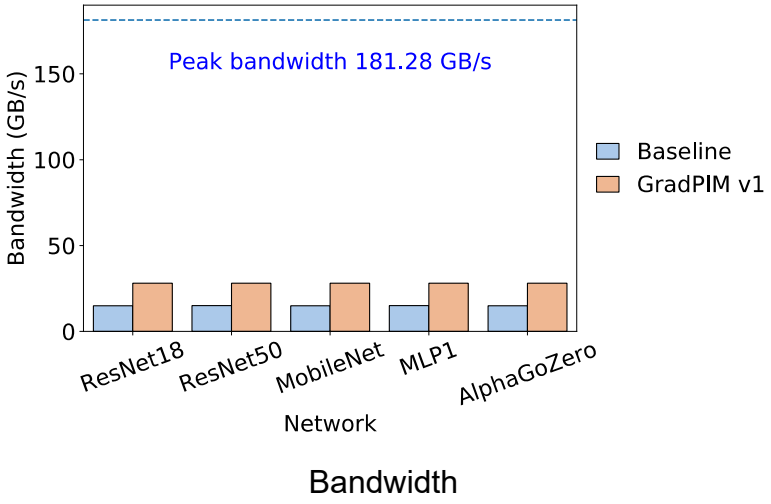


Energy breakdown of memory

Bottleneck analysis – Command-bus utilization

- Command-bus restricts the internal bandwidth of GradPIM.v1.
  - All GradPIM units share a single command-bus.
- GradPIM.v1 reaches 28GBps internal bandwidth.
  - Much lower than 181.28GBps maximum.

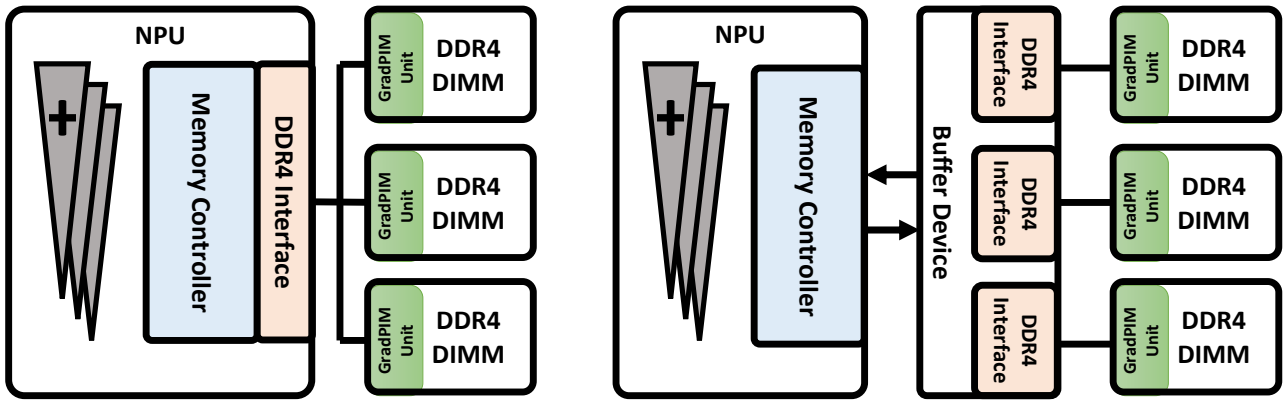




Bandwidth

Buffer device (BD) interface

- Use buffer devices to alleviate the command bus bottleneck.
  - Buffer devices function as fan-out expanders.
- Previous works utilizes buffer device in high-performance computing.
  - e.g., TensorDIMM*: places computing units on buffer devices.

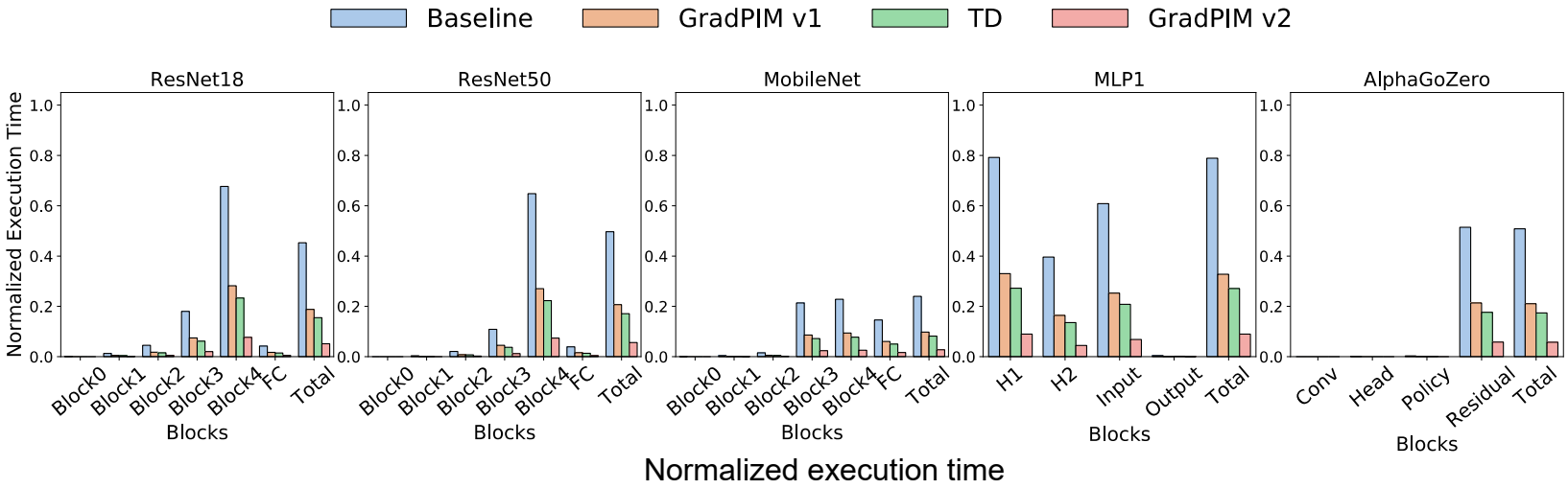


(a) GradPIM-Direct (GradPIM.v1) (b) GradPIM-Buffered (GradPIM.v2)

* Y. Kwon, Y. Lee, and M. Rhu, "TensorDIMM: A practical near-memory processing architecture for embeddings and tensor operations in deep learning," in *MICRO*, 2019.

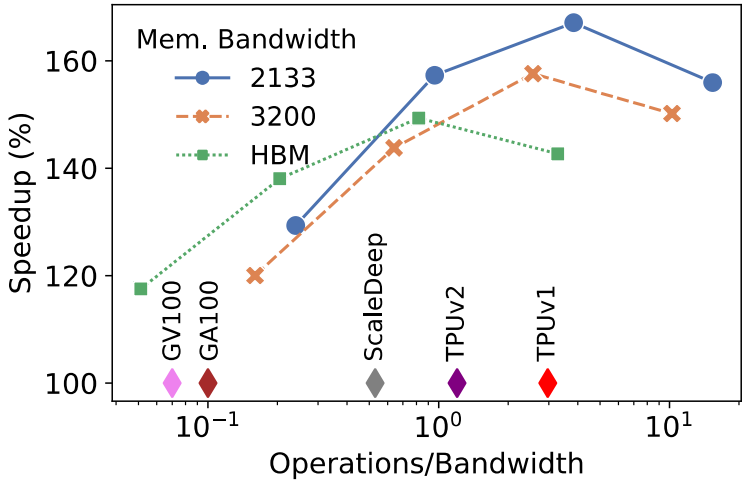
Experimental result – Execution time (w/ BD)

- GradPIM.v2 outperforms all baselines including TD.
  - TD: TensorDIMM-like gradient descent accelerator.
  - Achieves **8.23X** in Wup → **1.94X** overall.



Sensitivity – MAC size and Memory bandwidth

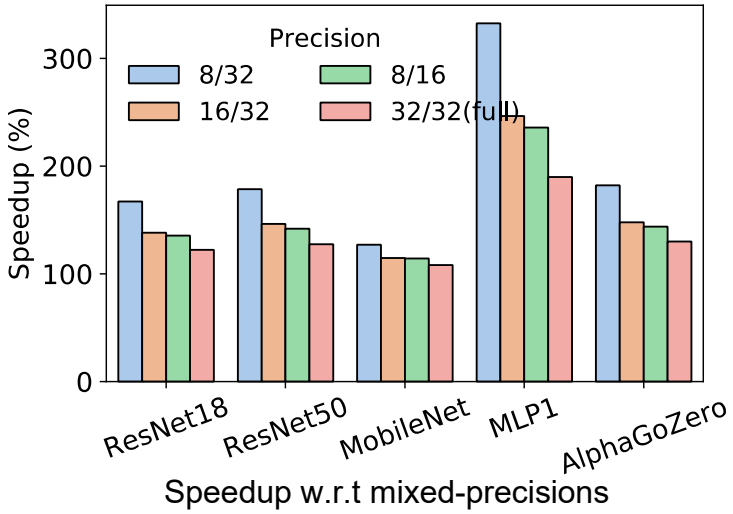
- Verify speedups on various “MAC size”/“Mem BW”.
  - Higher ratio → memory bottleneck → higher speedup from GradPIM.
  - Until MAC setup latency dominates execution time.
- We plot the ratio of famous NPUs and GPUs on X-axis.



Speedups w.r.t operations/bandwidth

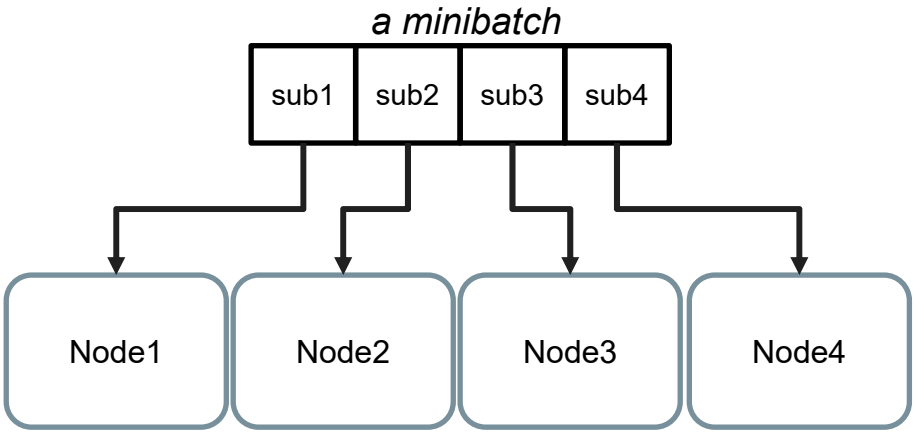
Sensitivity – Data precisions

- Mixed-precision: use both of high precision and low precision values.
- Bigger gap between precisions results in the higher speedups.
  - Bigger gap makes portion of Wup in execution time more dominant.
- However, even in 32/32, GradPIM still shows a substantial amount of speedup.

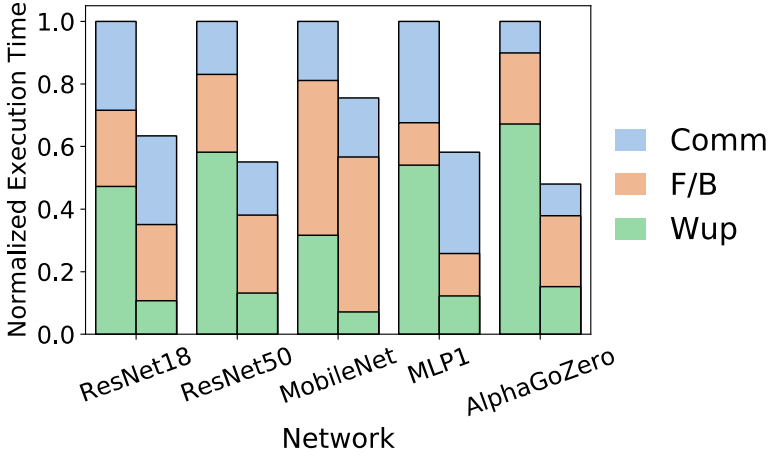


Future work – Distributed training

- Large-scale training → Distributed training
  - Workload of GD is distributed to multiple nodes.
- In data-parallelism approach, each node processes a subset of a minibatch.
  - In each node (total four nodes), forward and backward become faster.
  - But, Wup is not dependent on the batch size → Wup portion becomes larger.



Distributed learning (data-parallelism)



Excecution time of single/distributed training

Summary

- Gradient descent algorithm is suitable for PIM.
  - Memory intensive. (Low ops/byte ratio)
- GradPIM exploits bank-group level internal bandwidth of DRAM.
- GradPIM is practical.
  - Compatible with DDR protocol.
  - Non-invasive to cell array of DRAM.
- GradPIM achieves 1.94X speedup.
  - Works well for distributed training.