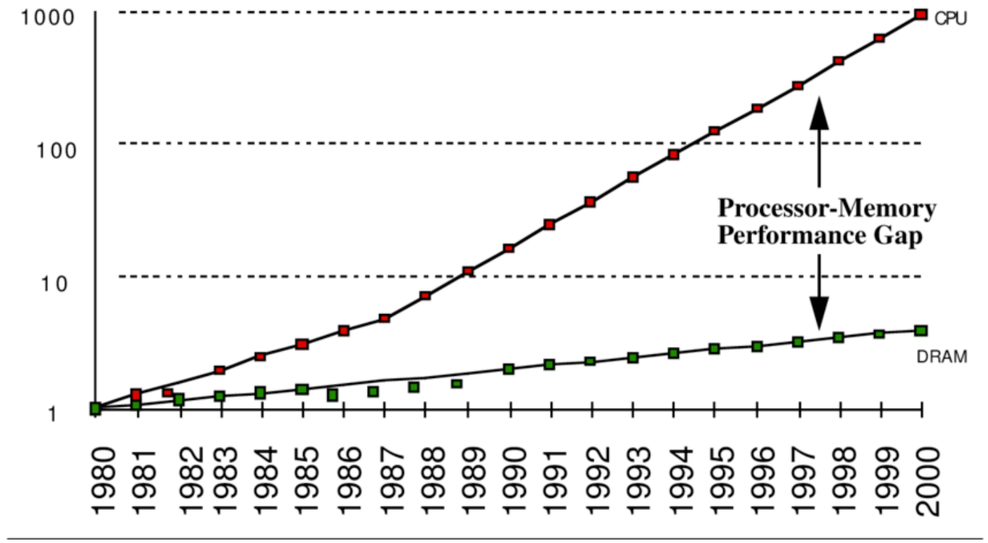
In-Memory Data Parallel Processor (ASPLOS'18)

Author: Daichi Fujiki(dfujiki@umich.edu)
Presenter: Choi, Jinyoung

PIM (Processing in Memory)

- Memory access is the main bottleneck in modern Von-Neumann architecture.
  - CPU-memory performance gap
  - Off-chip access is expensive!



- People think about implanting computing logic into the memory unit.

PRIME (ISCA'16)

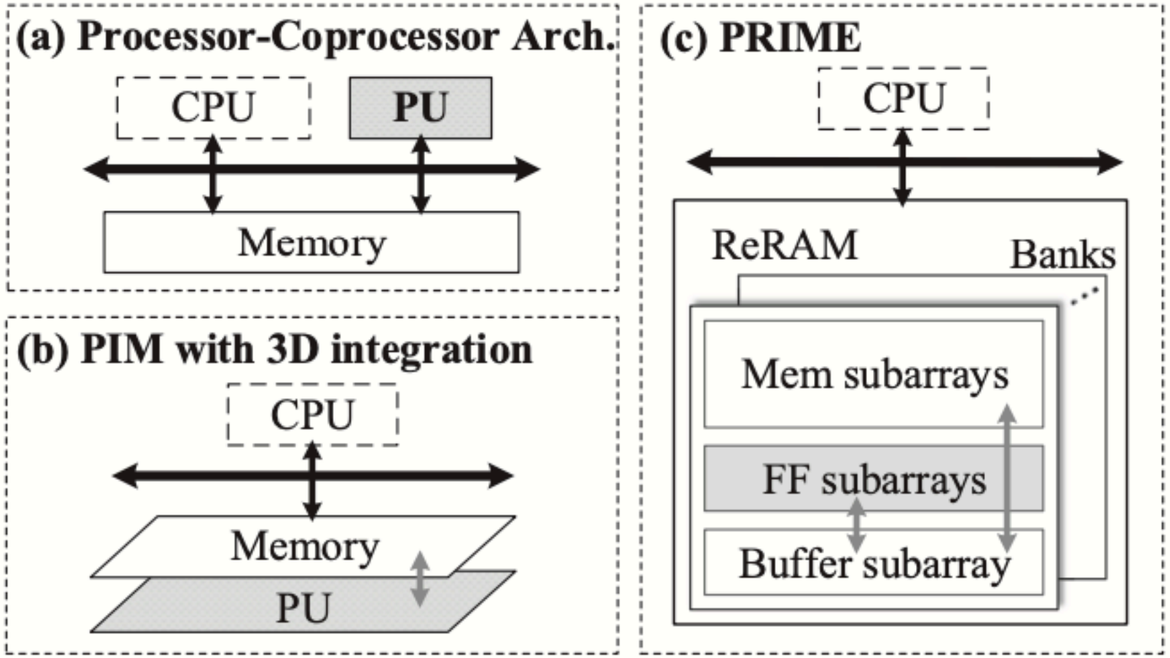


Figure 3. (a) Traditional shared memory based processor-coprocessor architecture, (b) PIM approach using 3D integration technologies, (c) PRIME design.

Dot operation using Re-RAM

$$a \cdot b = \underset{(1 \times n)}{\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{bmatrix}} \underset{(n \times 1)}{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}} = \{a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4 + a_5b_5\}$$

Dot Product

Basic Laws of Circuits

Ohm's Law: **About Resistors:**

The unit of resistance is ohms(Ω).

So, the current equals to the voltage divided by the resistance

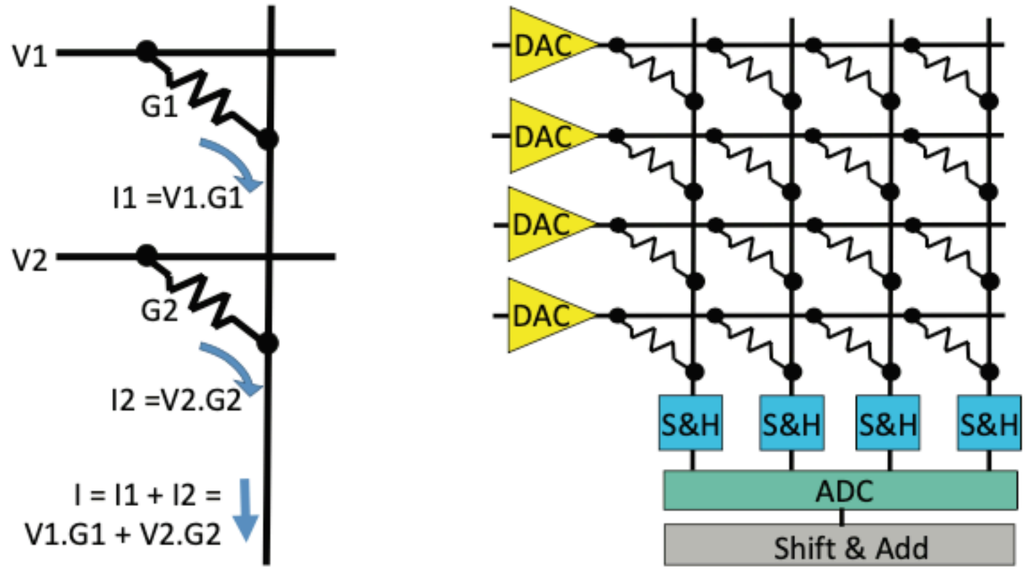
$$i = \frac{v}{R}$$

The reciprocal of the resistance is referred to as conductance

$$G = \frac{1}{R}$$ **S (Siemens)**

$$\boldsymbol{i = v\,G}$$

10



(a) Multiply-Accumulate operation (b) Vector-Matrix Multiplier

Fig. 1. (a) Using a bitline to perform an analog sum of products operation. (b) A memristor crossbar used as a vector-matrix multiplier.

(ISAAC, ISCA'16)

Why this paper?

- In previous Re-RAM accelerators,
  - they support only dot-product operation.
  - we need to manually map specialized kernel to the memory arrays.

latency. *In this paper, we seek to answer the question, to what extent is resistive memory useful for more general-purpose computation?*

Proposed architecture: In-Memory Data Parallel Processor

- In previous Re-RAM accelerators,
  - they support only dot-product operation.
    - ISA beyond dot-product
  - we need to manually map specialized kernel to the memory arrays.
    - Compiler

Architecture & ISA

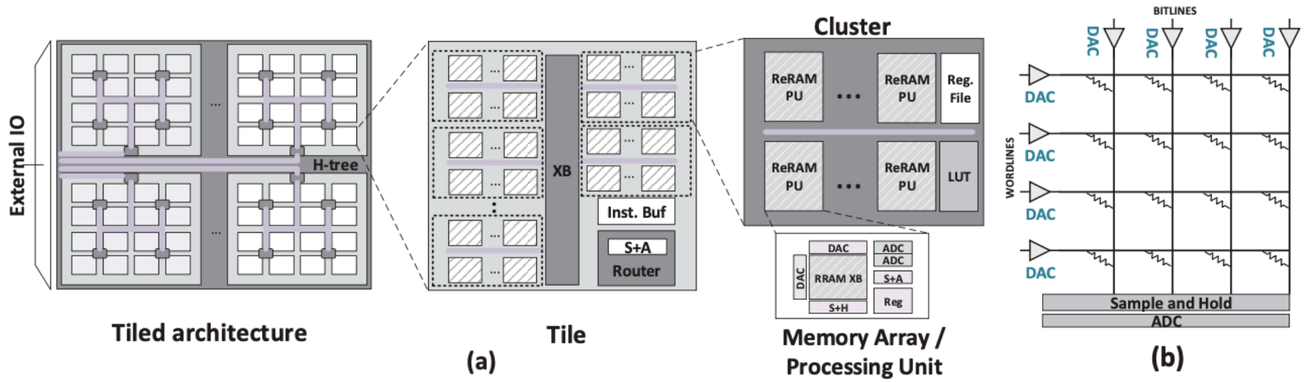


Figure 1. In-Memory Processor Architecture. (a) Hierarchical Tiled Structure (b) ReRAM array Structure

3-stage pipeline
- XB
- ADC
- S/A

Opcode	Format	Cycles
add	<mask><dst>	3
dot	<mask><reg_mask><dst>	18
mul	<src><src><dst>	18
sub	<mask><mask><dst>	3
shift{l|r}	<src><dst><imm>	3
mask	<src><dst><imm>	3
mov	<src><dst>	3
movs	<src><dst><mask>	3
movi	<dst><imm>	1
movg	<gaddr><gaddr>	Variable
lut	<src><dst>	4
reduce_sum	<src><gaddr>	Variable

Compiler

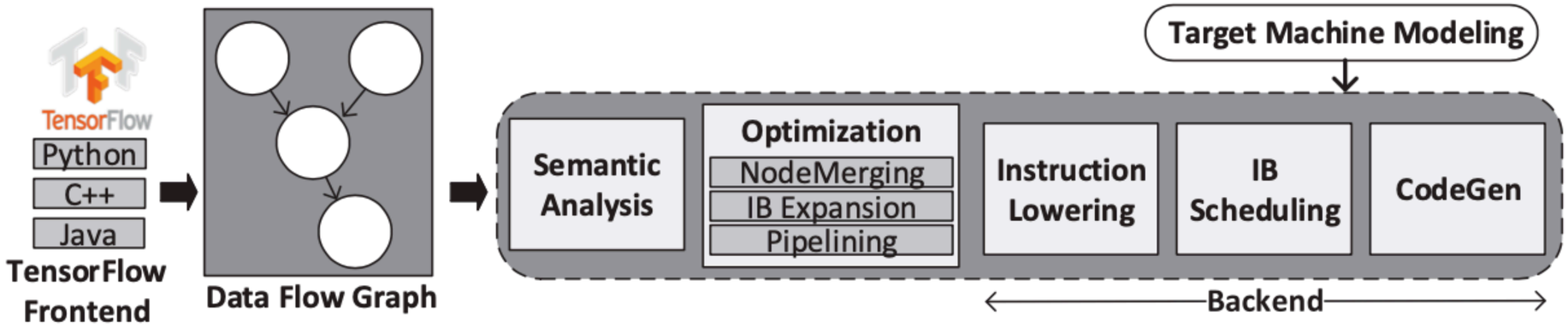


Figure 4. Compilation Flow.

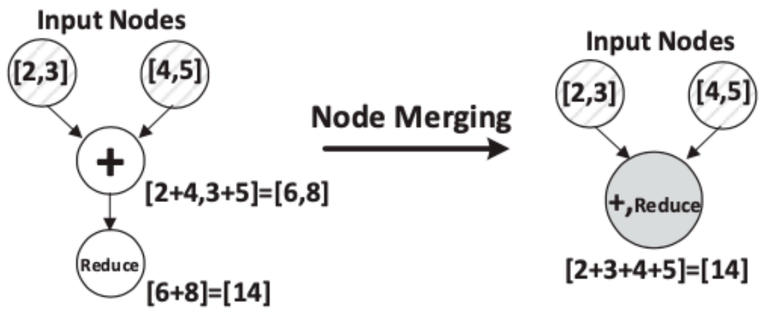


Figure 5. Node Merging.

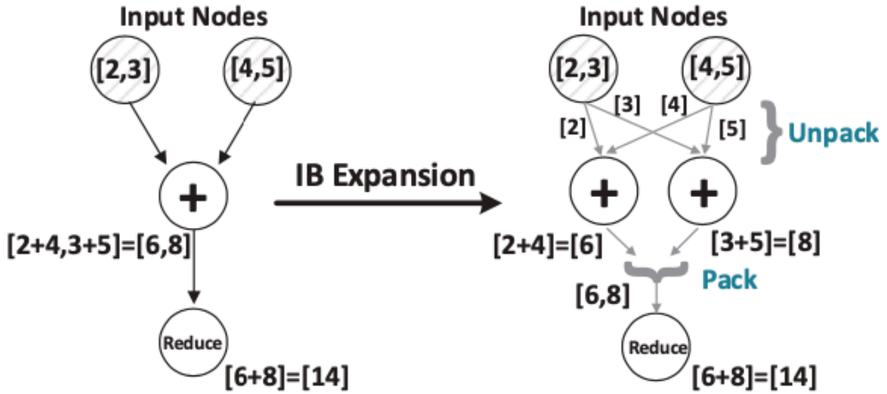


Figure 6. IB Expansion.

Execution model

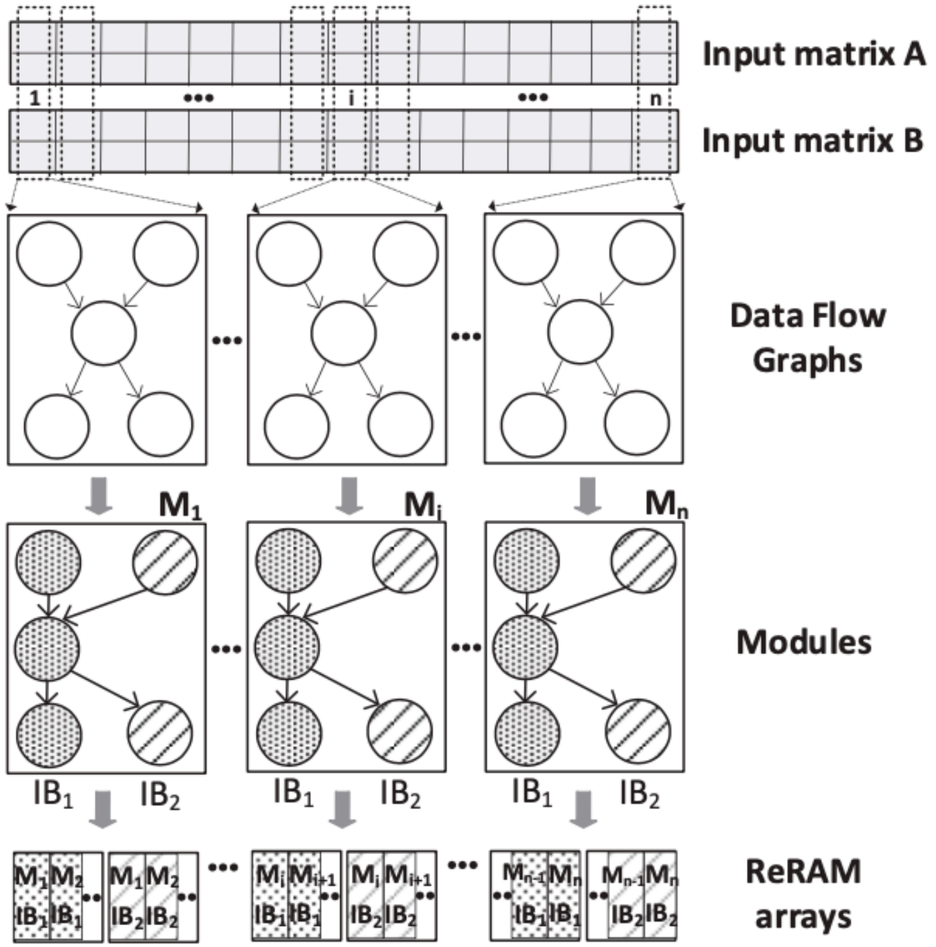


Figure 3. Execution Model.

Testbed

- Baseline 1: Intel Xeon E5-2697 v3 multi-socket server
- Baseline 2: Nvidia Titan XP
- In-Memory Processor(IMP)

Parameter	CPU (2-sockets)	GPU (1-card)	IMP
SIMD slots	448	3840	2097152
Frequency	3.6 GHz	1.58 GHz	20 MHz
Area	912.24 mm^2	471 mm^2	494 mm^2
TDP	290 W	250 W	416 W
Memory	7MB L2; 70MB L3 64GB DRAM	3MB L2 12GB DRAM	1GB RRAM

Table 5. Comparison of CPU, GPU, and IMP Parameters

Component	Params	Spec	Power	Area(mm^2)
ADC	resolution frequency number	5 bits 1.2 GSps 64 × 2	64 mW	0.0753
DAC	resolution number	2 bits 64 × 256	0.82 mW	0.0026
S+H	number	64 × 128	0.16 mW	0.00025
ReRAM Array	number	64	19.2 mW	0.0016
S+A	number	64	1.4 mW	0.0015
IR OR Register	size size size	2KB 2KB 3KB	1.09 mW 1.09 mW 1.63 mW	0.0016 0.0016 0.0024
XB	bus width size	16B 10 × 10	1.51 mW	0.0105
LUT	number	8	6.8 mW	0.0056
Inst. Buf	size	8 × 2KB	5.83 mW	0.0129
Router	flit size num_port	16 9	0.82 mW	0.00434
S+A	number	1	0.05 mW	0.000004
1 Tile Total			101 mW	0.12
Inter-Tile Routers	number	584	0.81 W	2.50
Chip total			416 W	494 mm^2

Table 4. In-Memory Processor Parameters

Operations

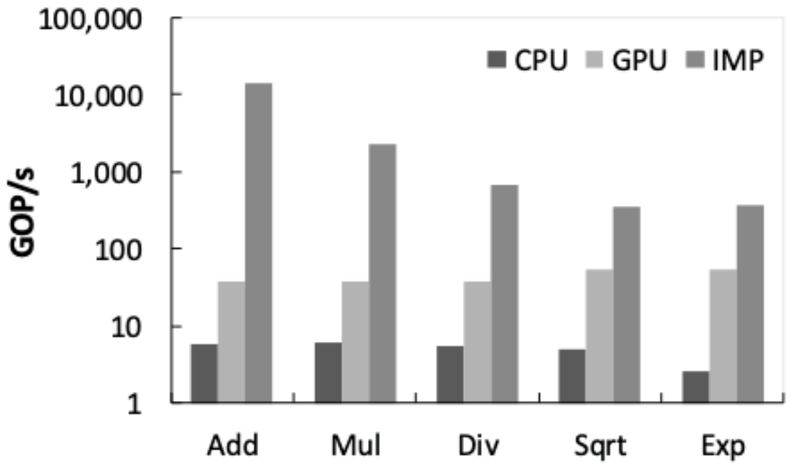


Figure 7. Operation throughput (log scale).

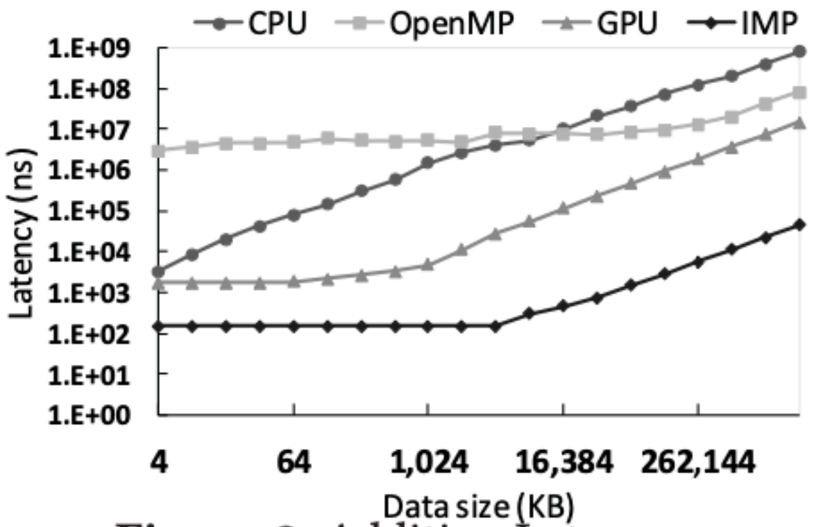


Figure 8. Addition Latency.

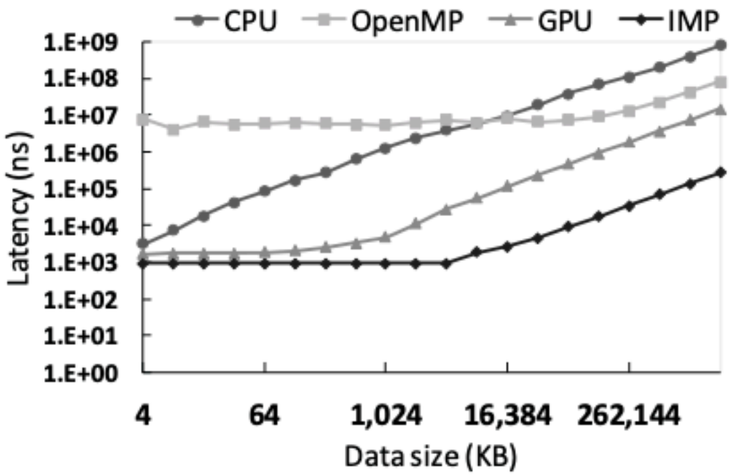


Figure 9. Multiplication Latency.

Speedup

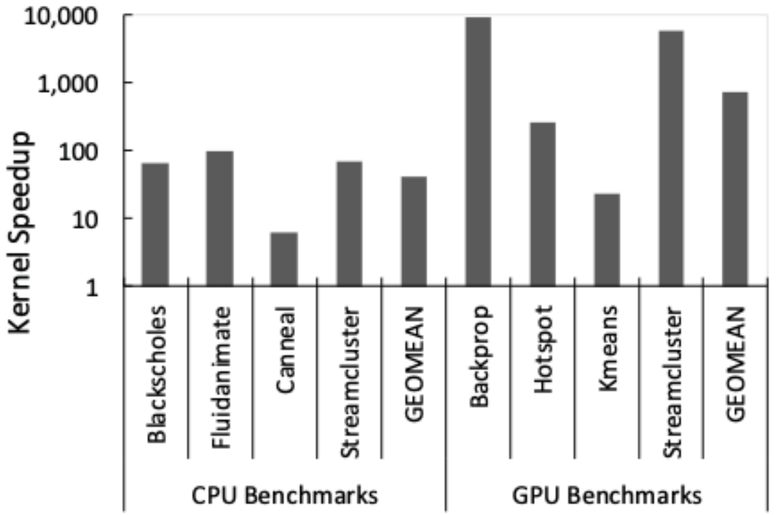


Figure 11. Kernel speedup.

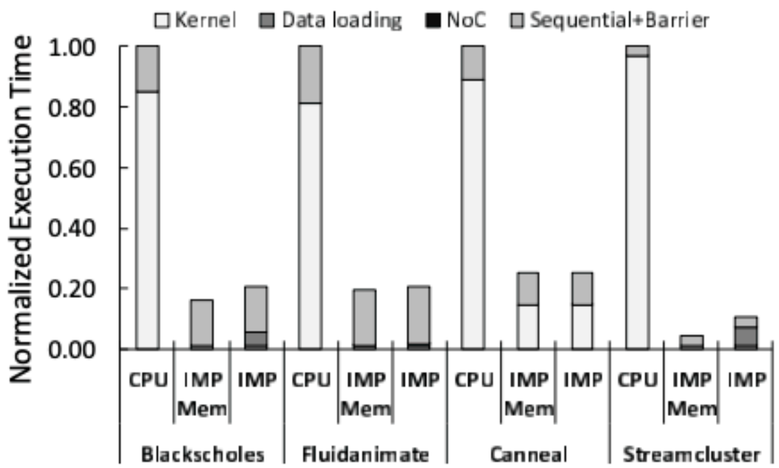


Figure 12. CPU Application performance.

before, in-memory accelerator is better coupled with the existing memory hierarchy to avoid data loading overhead.

Power & Energy

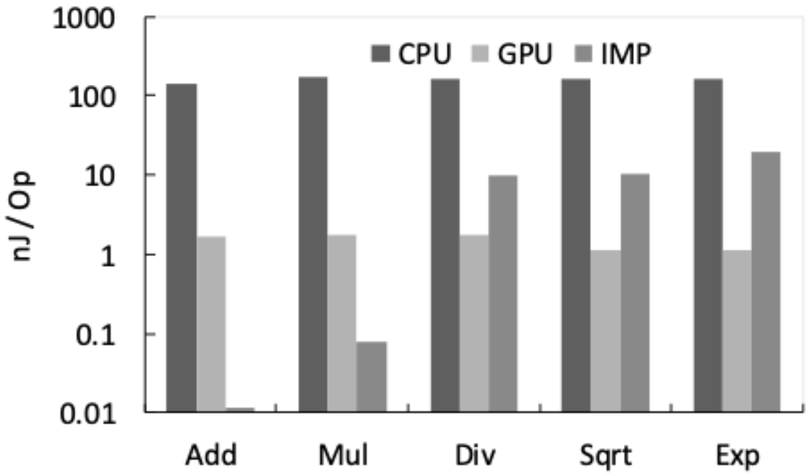


Figure 10. Operation energy.

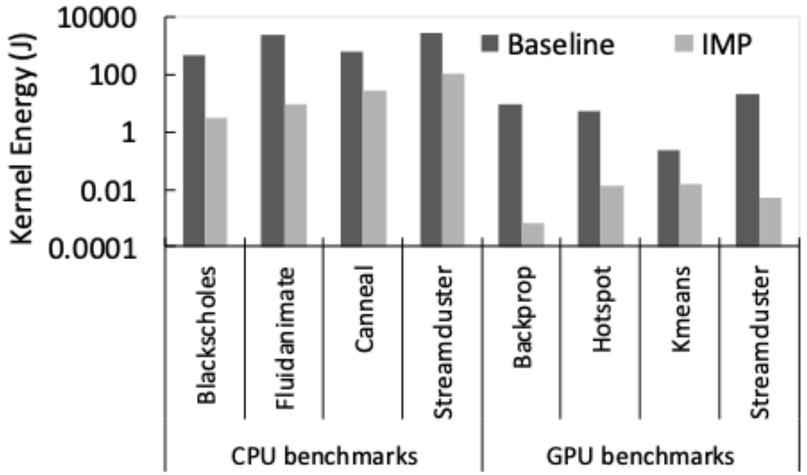


Figure 13. Energy consumption.

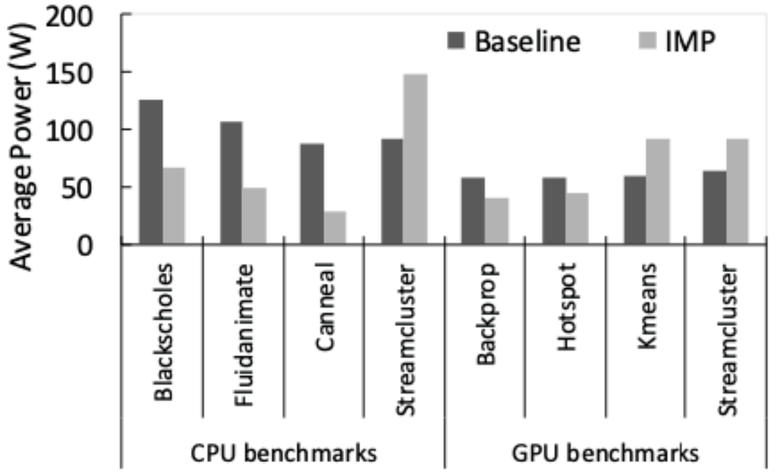
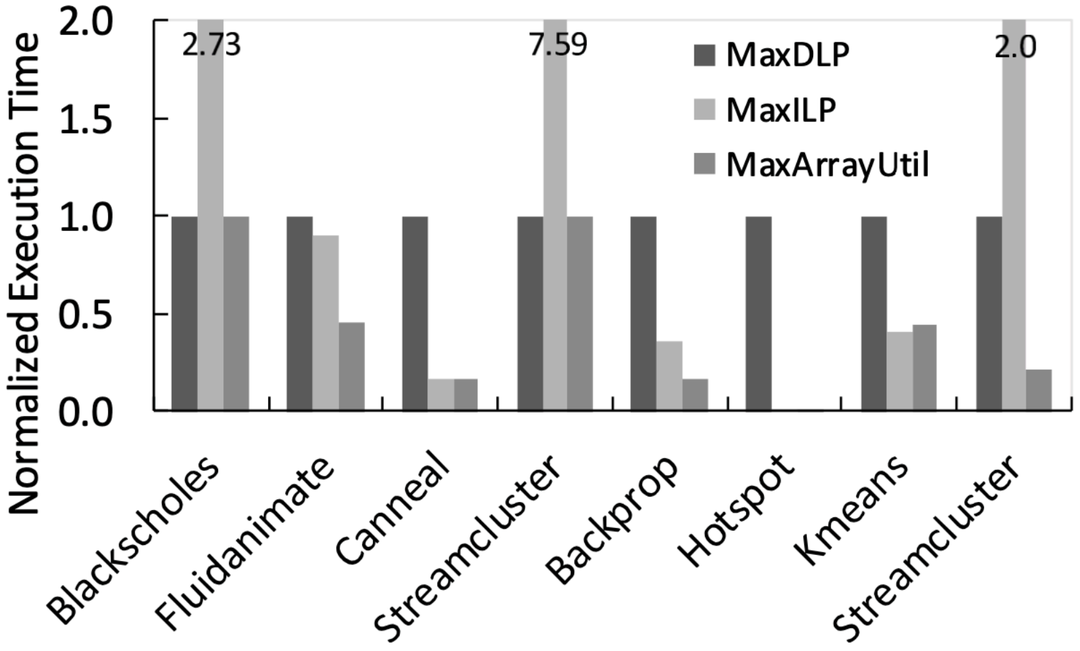


Figure 14. Average power.

Compiler optimization



MaxDLP: one IB for a module
- cannot exploit ILP
- under utilization with small small size

MaxILP: maximum IBs for a module
- serialization overhead
- need to invoke kernel multiple time (SIMD slot < data)

MaxArrayUtil: considering #of SIMD slots needed by input data

Thanks

- QnA

Re-RAM array operations