ECE 598NSG/498NSU
Deep Learning in Hardware
Fall 2020

DNN Accelerators – Overview
Roofline, Floorline plots, Reuse

Naresh Shanbhag

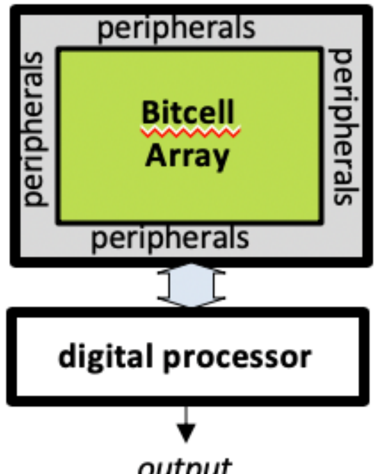
Department of Electrical and Computer Engineering

University of Illinois at Urbana-Champaign

http://shanbhag.ece.uiuc.edu

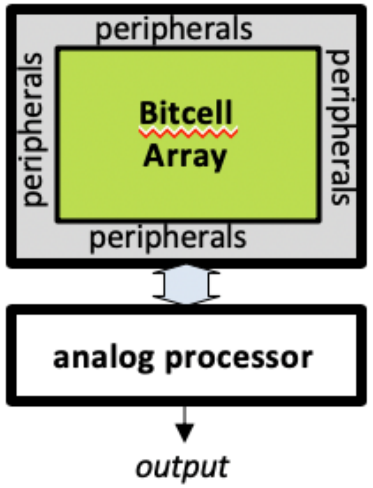
Architectures for Inference

digital



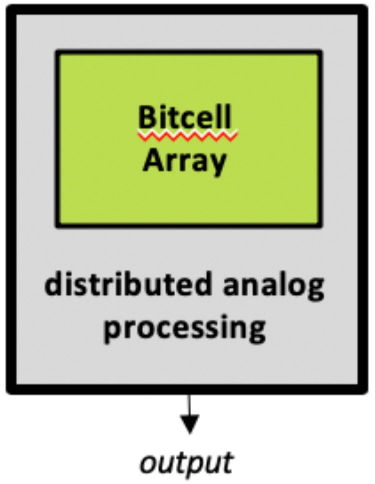
- memory accesses dominates energy & latency costs
- data reuse & comp. alleviate costs

analog near memory



- compute energy (↓)
- memory accesses still dominate energy & latency costs

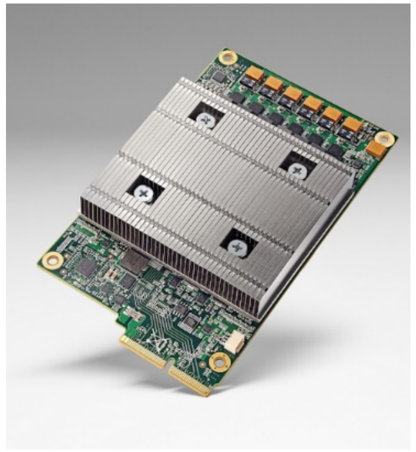
in-memory



- merges memory access and compute
- energy & latency (↓)
- compute density (↑)

DNN Accelerators

TPU V1

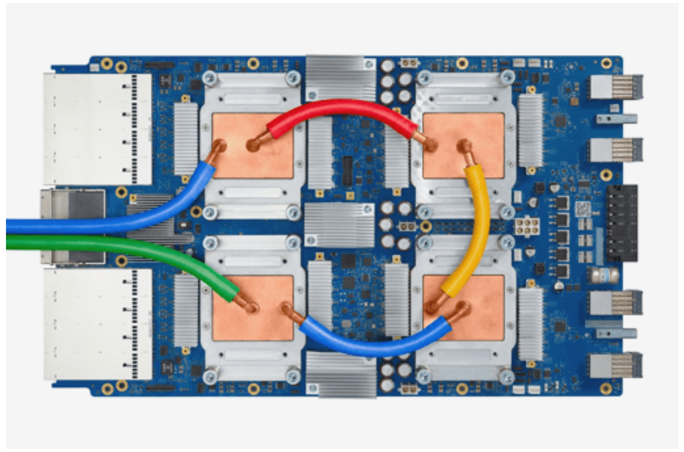


inference only
8b fixed-point

TPU V2

inference & training
fixed-point & floating-point

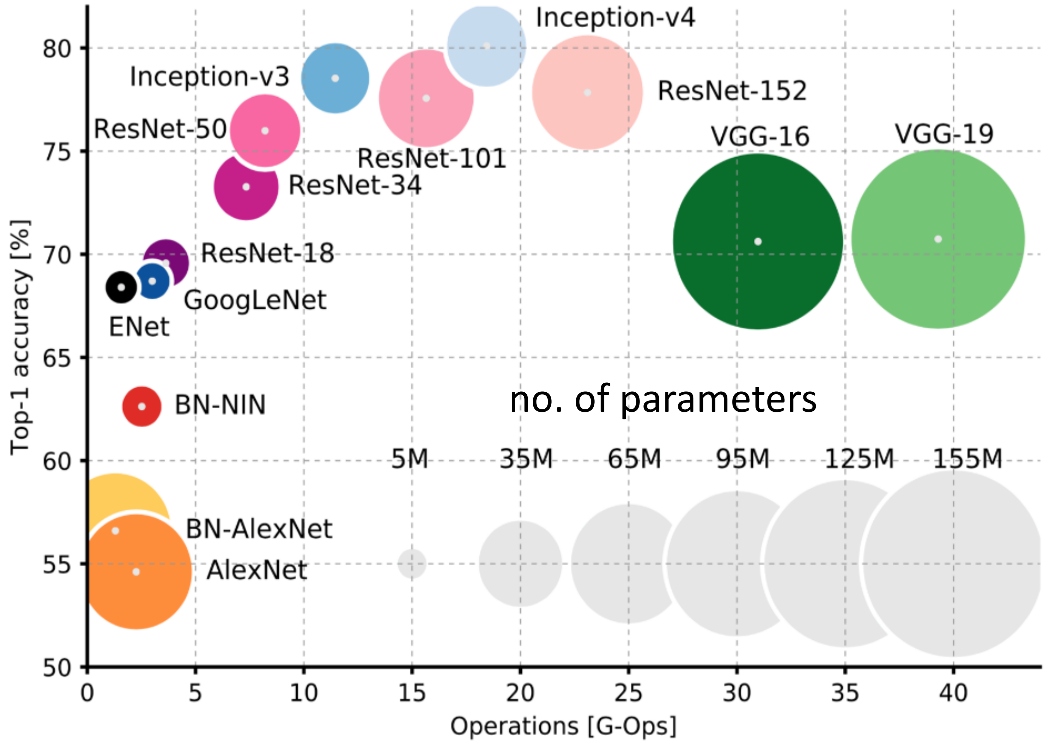
TPU V3



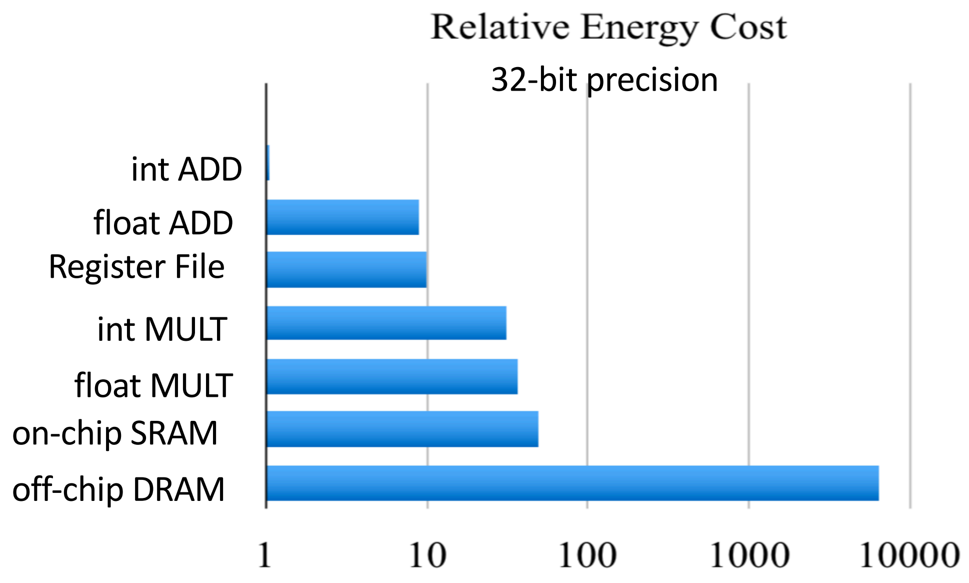
inference & training
fixed-point & floating-point

- going beyond GPUs
- accelerating inference & training in the cloud
- intersection of workload, architecture, circuits, software

Challenge: Memory Access Energy in DNNs



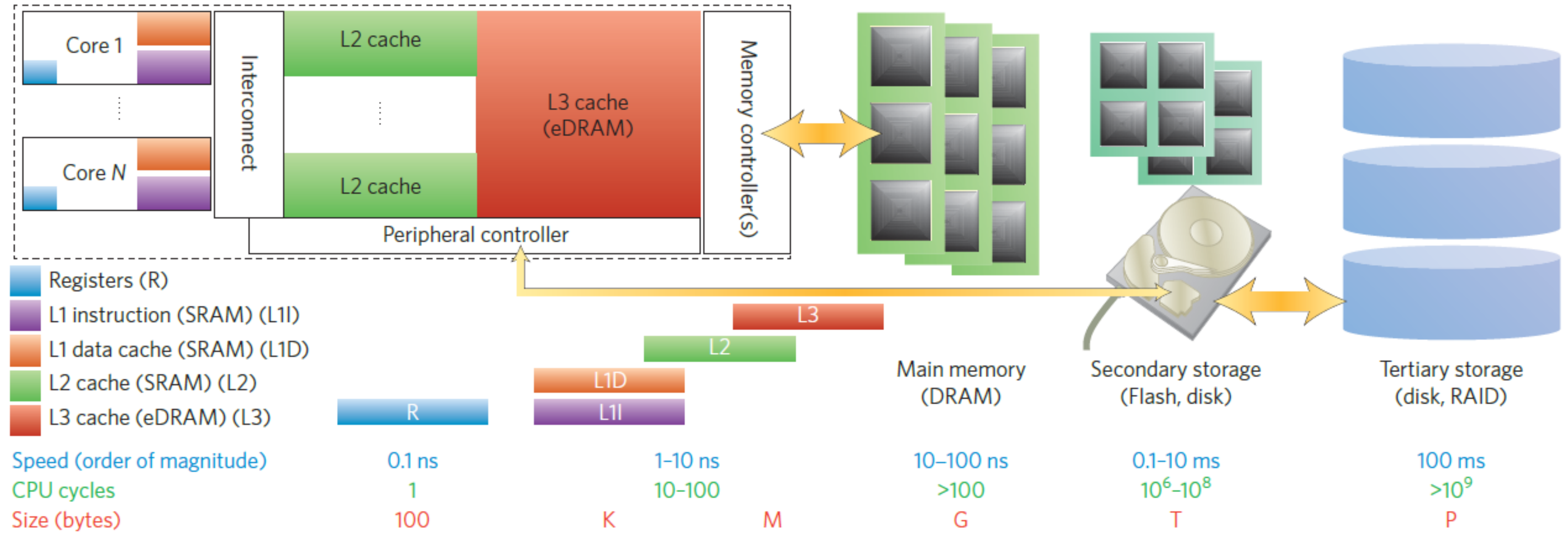
[Canziani-arXiv-2017]



[Horowitz-ISSCC-2014]

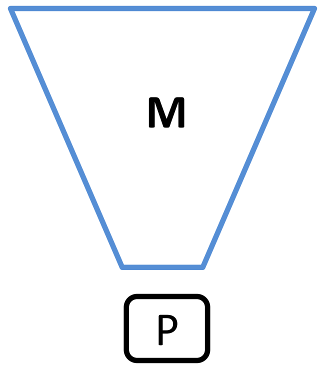
$$\boxed{E_R = \frac{energy/access}{energy/OP}}$$

- memory access-to-compute energy consumption ratio
- SRAM accesses: $E_R = 100\times$; DRAM accesses: $E_R = 1000\times$



[Wong, Salahuddin, Nature Nanotechnology, 2015]

the memory funnel

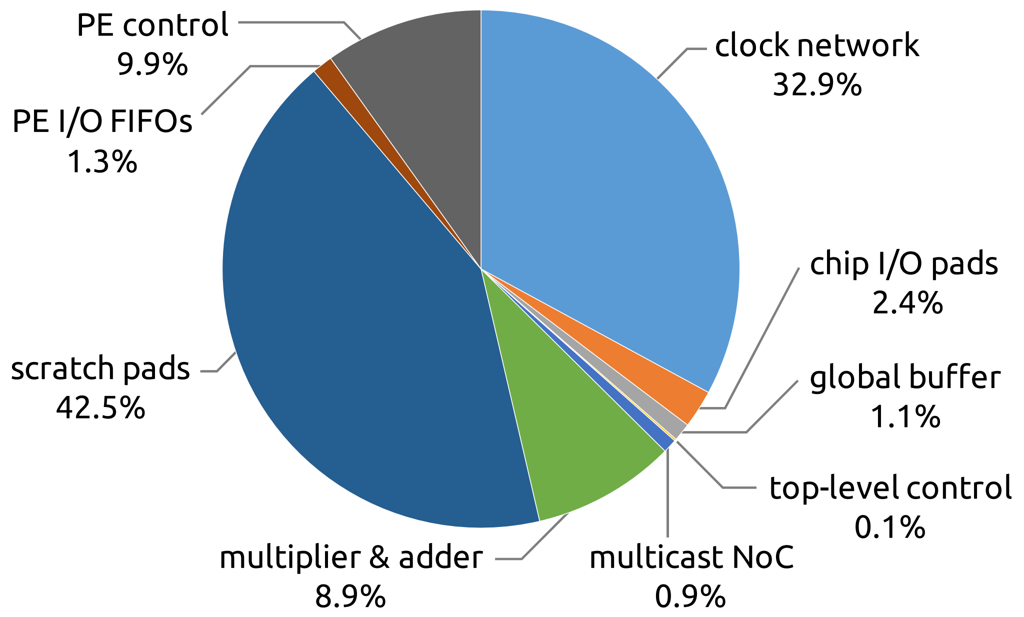


- deep memory hierarchy
- high density → more latency, energy
- data movement → dominant cause of energy dissipation in machine learning applications

Power Breakdown

[Eyeriss, Y.-H. Chen, JSSC 2017]

post-layout simulations with actual workloads

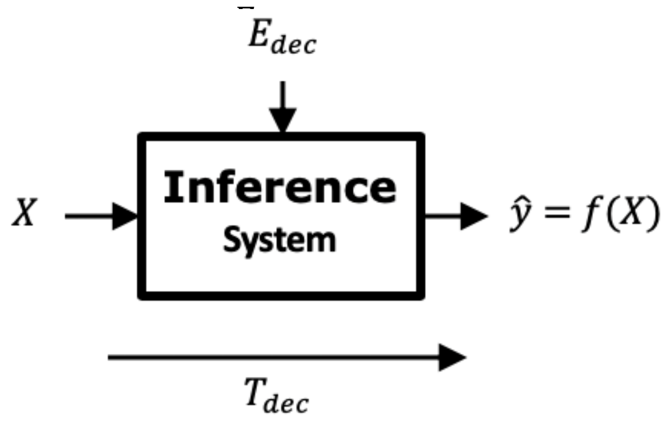


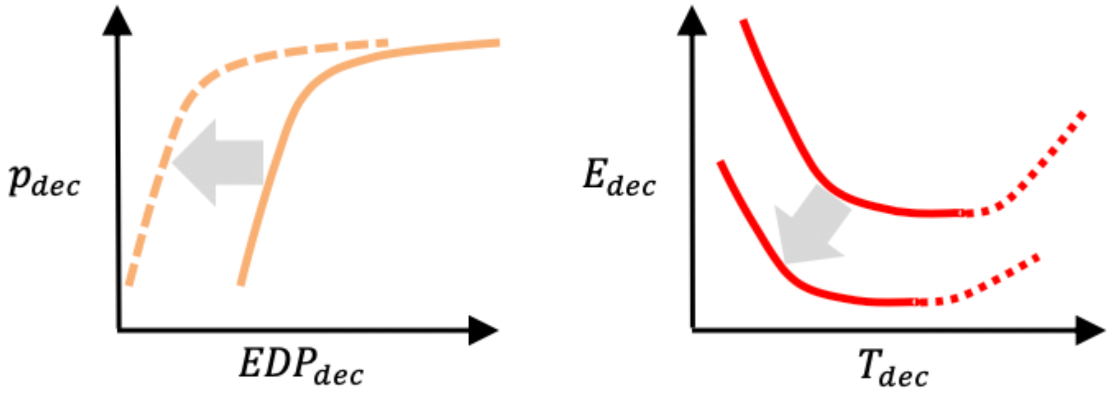
CONV1 layer of AlexNet

- Data movement costs 45% of total power
- Computation costs about 10%
- Clock network = 33% (hidden cost of digital realizations)

System Metrics for Inference

system-level energy vs. latency vs. accuracy trade-off





- ❑ T_{dec}: decision latency
- ❑ p_{dec}: decision accuracy
- ❑ $EDP_{dec} = E_{dec} \times T_{dec}$: decision EDP
- ❑ f_{dec}: decision throughput

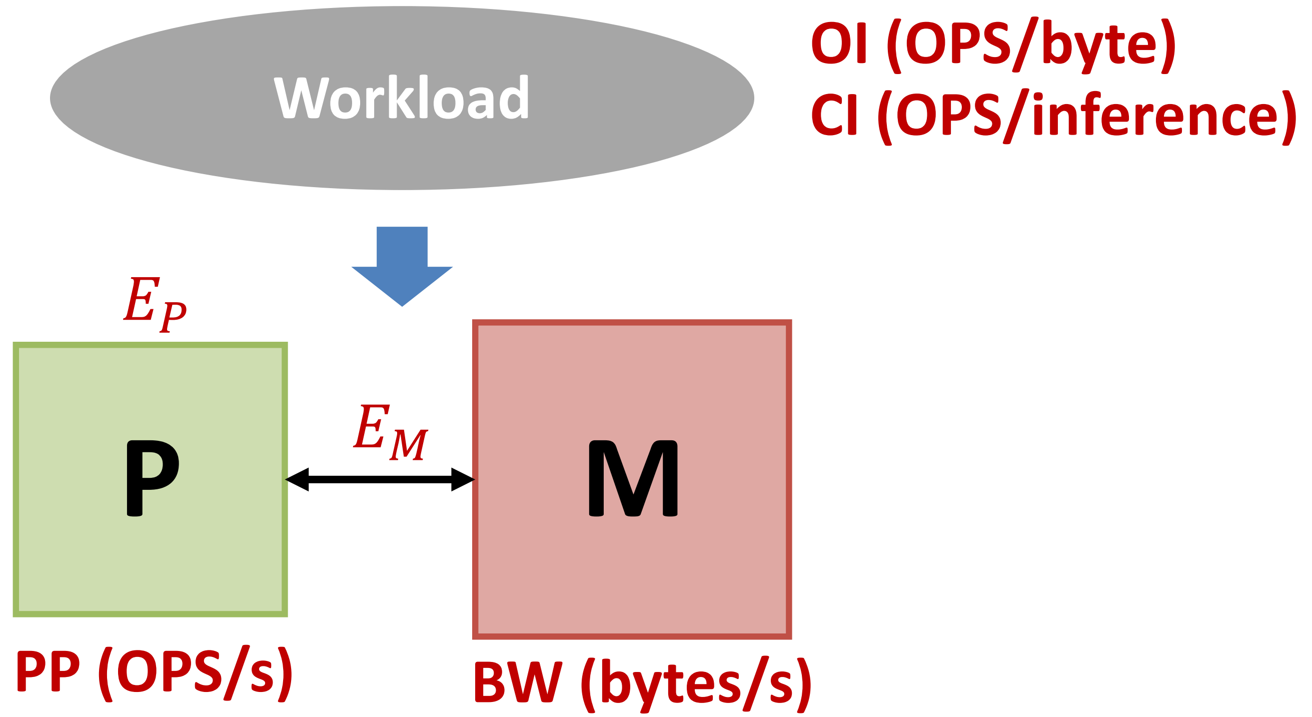
- ❑ $E_{dec} \propto 1/T_{dec} \propto$ data & network size
- ❑ $p_{dec} \propto EDP_{dec}$

Outline

- DNN architecture abstraction
- visualization via the roofline & floorline plots
- data reuse

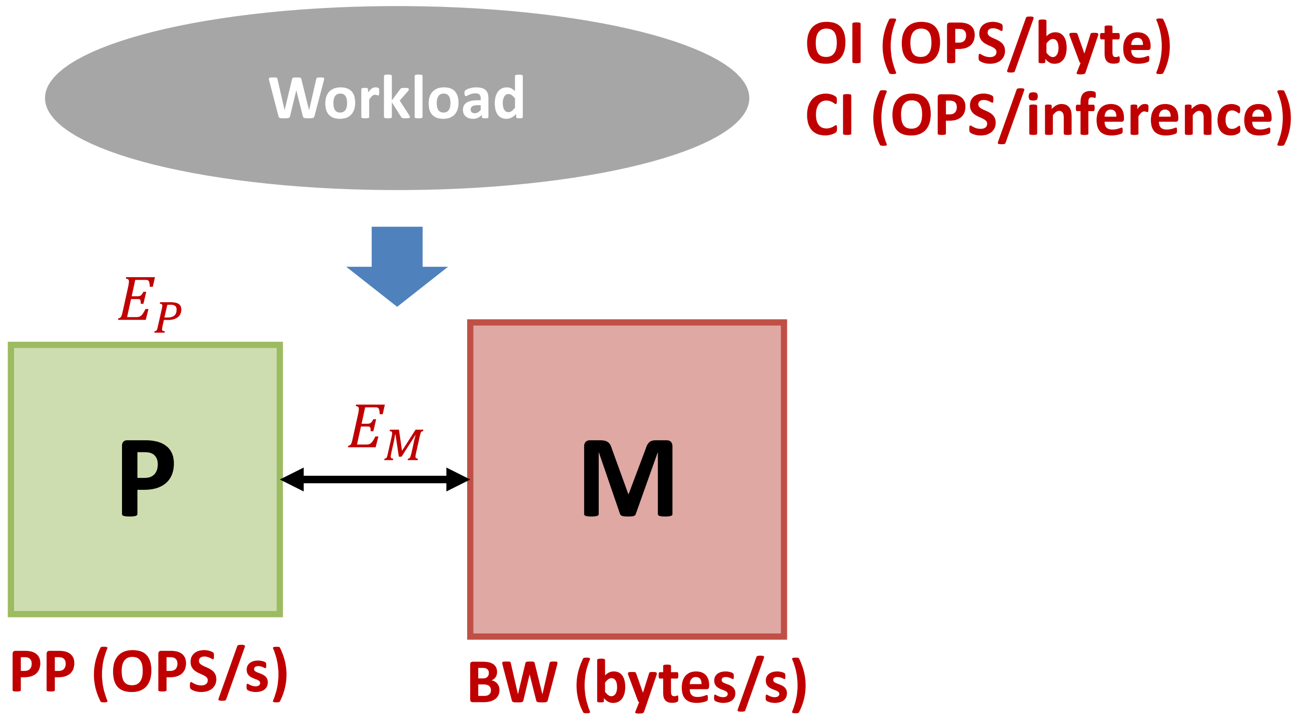
DNN Architectures – An Abstract View

DNN Architecture - Abstract View



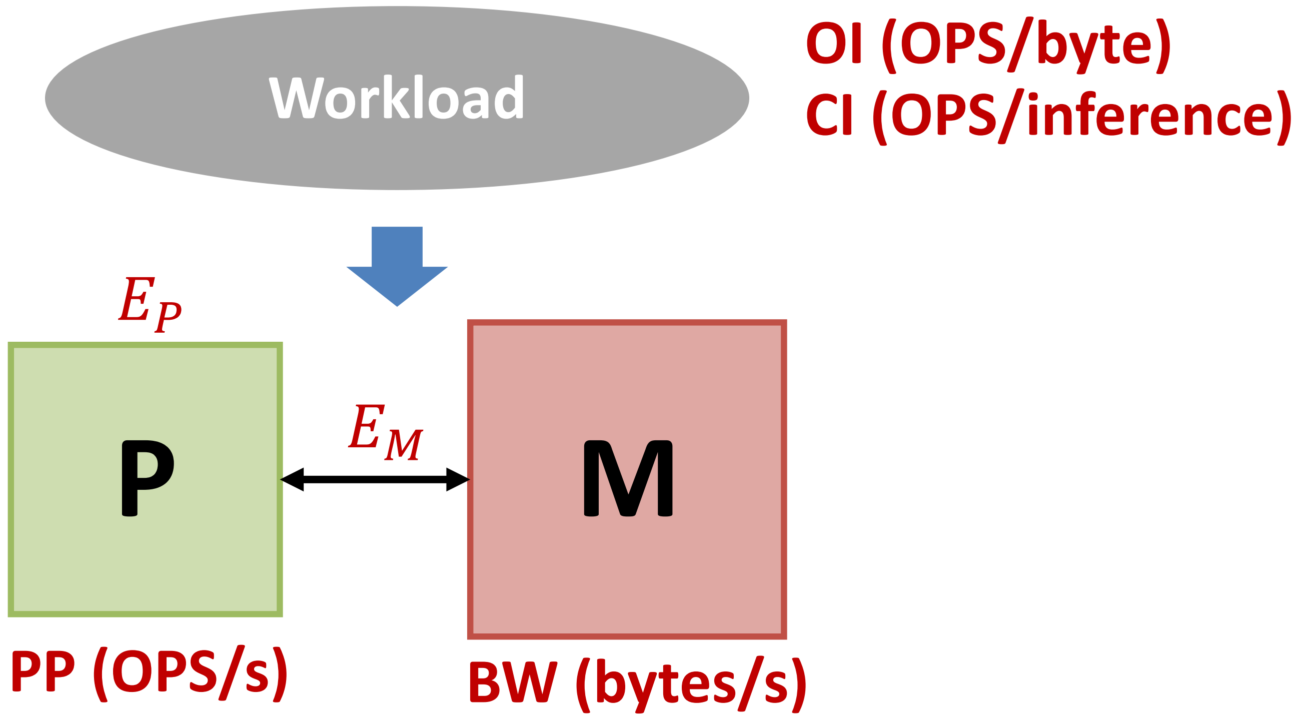
- Workload – DNNs, CNNs, LSTMs,….
- Processor **P** – GPU, accelerators, many-core architectures,….
- Memory **M** – register, SRAM, DRAM, flash,…..

Workload Parameters



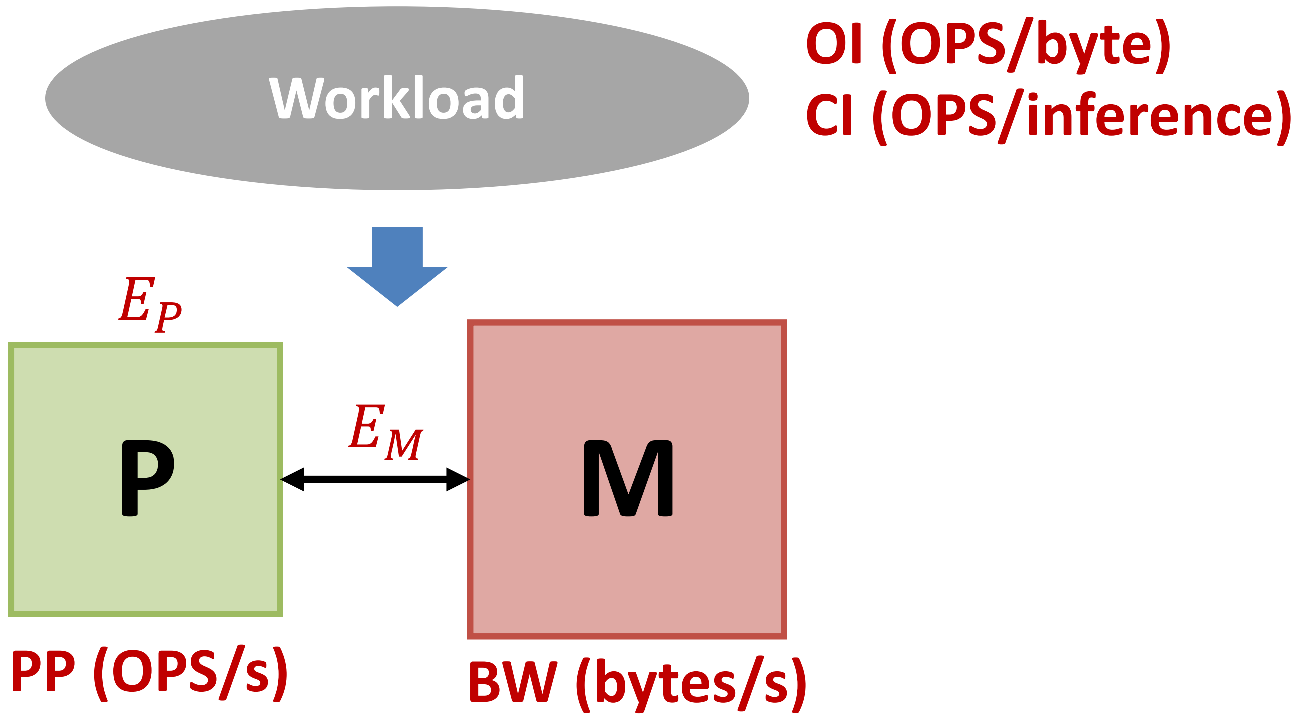
- operational intensity (OI): # of operations/byte (high is better) – reuse factor
- computational intensity (CI): # of operations per decision (small is better)

Processor Parameters



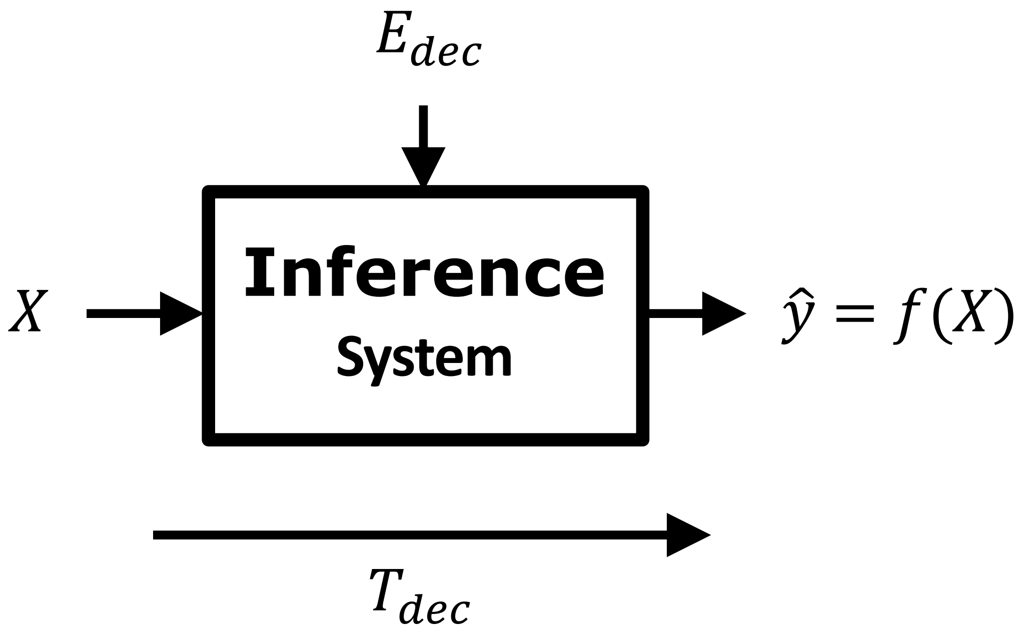
- peak performance (PP): # of operations/s (e.g., TOPS = 10^{12} OPS/s)
- arithmetic efficiency (E_P): energy per operation (e.g. $E_P = 0.2pJ$/op)

Memory Parameters



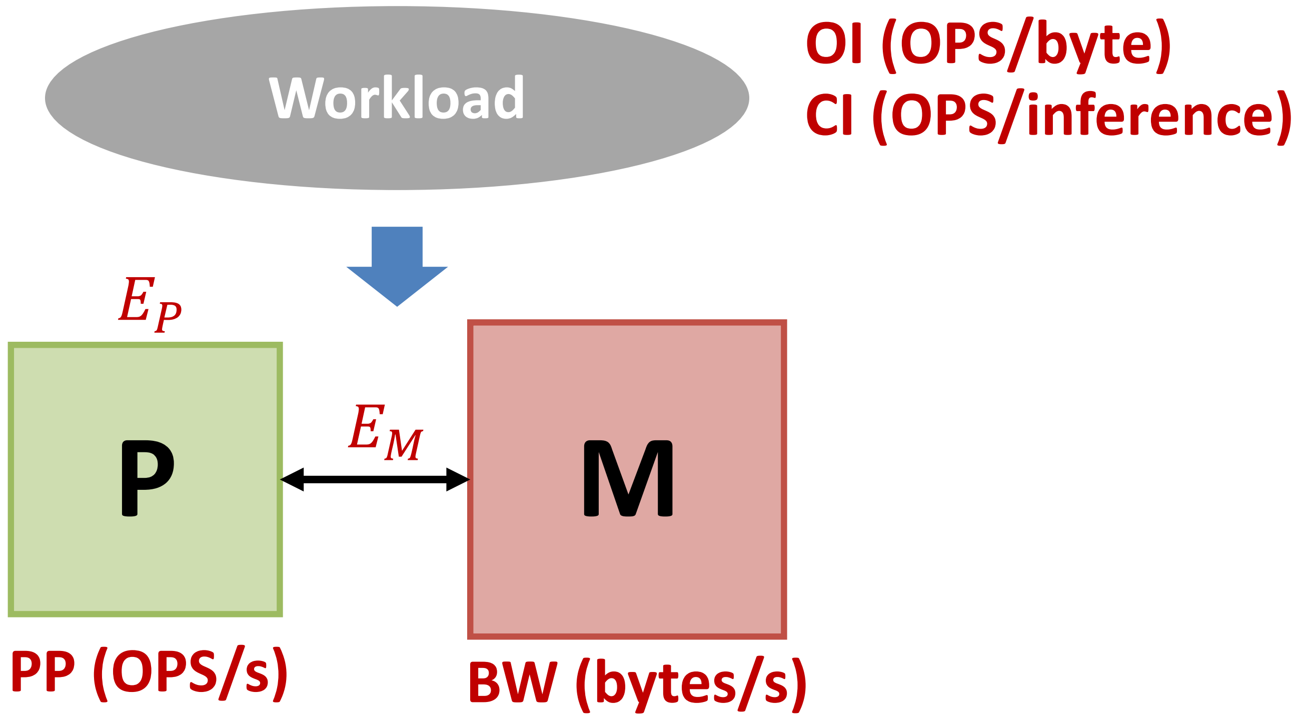
- read bandwidth (BW): # of bytes read/s (higher the better)
- energy/read (E_M): energy per byte (lower the better)
- $E_R = \frac{E_M}{E_P} = 100$ (SRAM); $= 500$ (DRAM); $= 1000$ (flash)

System Level Metrics



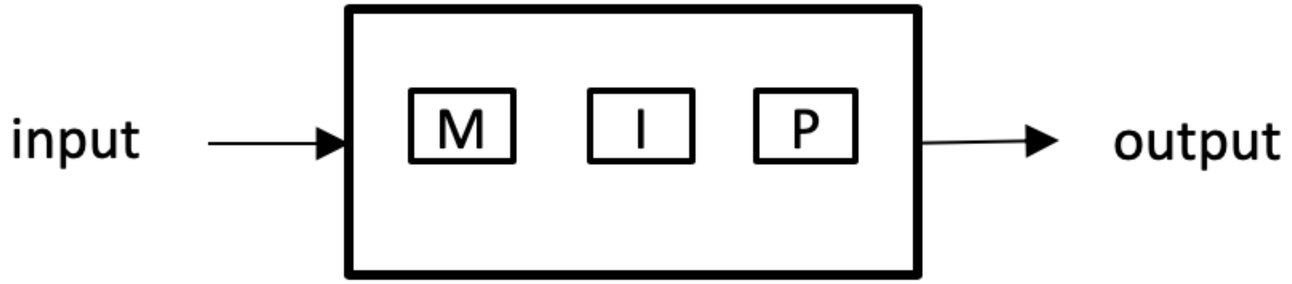
- useful for system-level benchmarking
- energy/decision (E_{dec})
- decision throughput ($f_{dec} = \frac{1}{T_{dec}}$)

Component Level Metrics



- used for hardware benchmarking
- energy/OP (E_P); peak OP/s (PP); bytes read/s (BW); energy/byte (E_M)

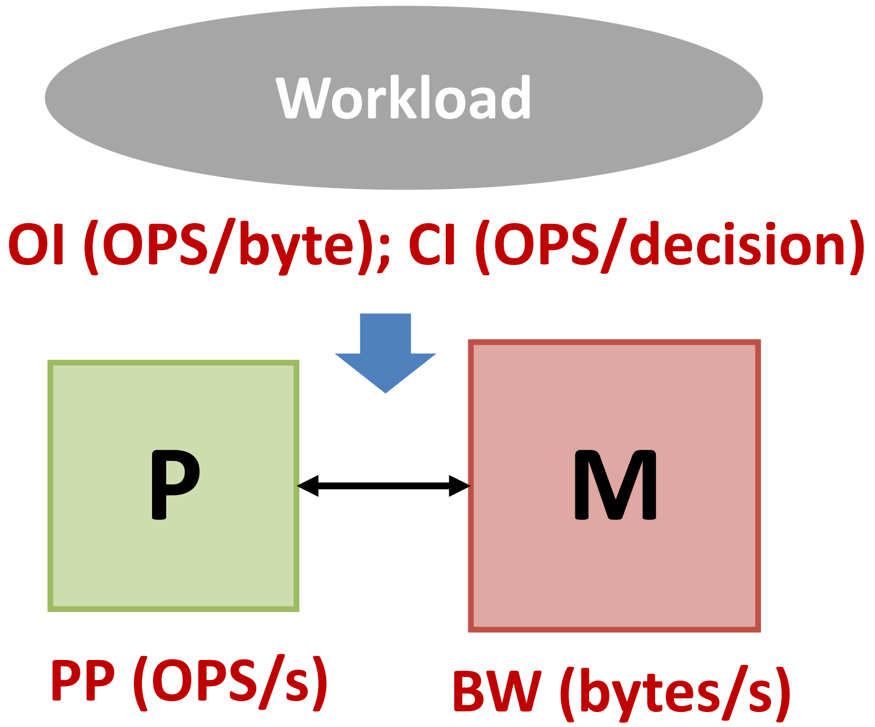
System vs. Component Level Metrics

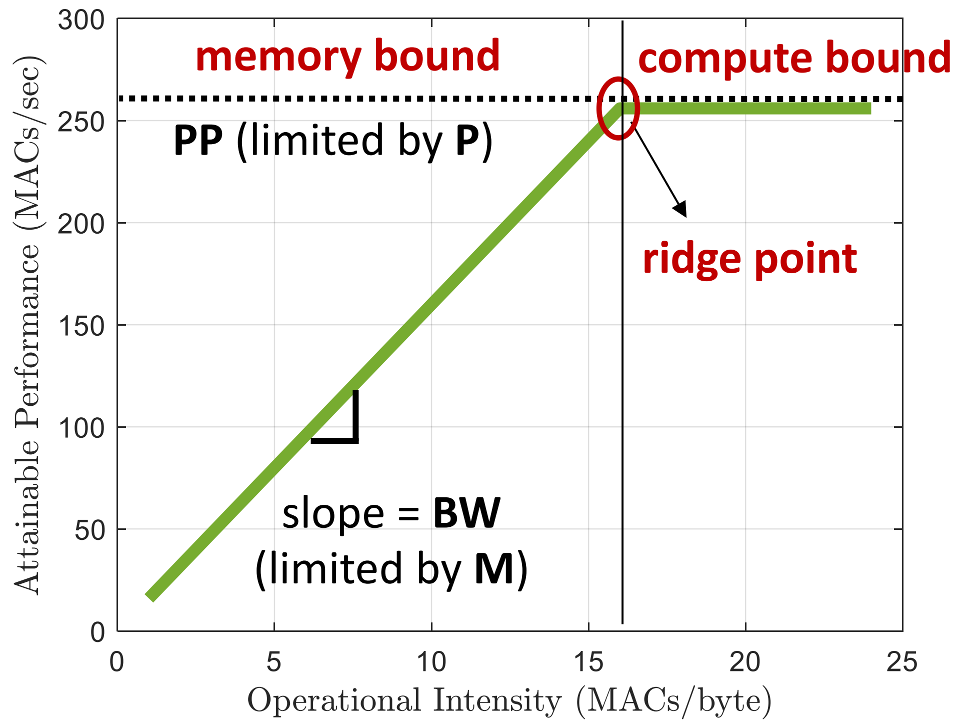


- system-level metrics are a function of component-level metrics, workload requirements, and workload mapping onto hardware
- system-level metrics are the true metrics – hard to measure and report
- alternative – measure @ the component-level and estimate @ the system-level

Roofline & Floorline Plots – Visual Performance & Efficiency Models

Architectural-level Performance - Roofline Plot



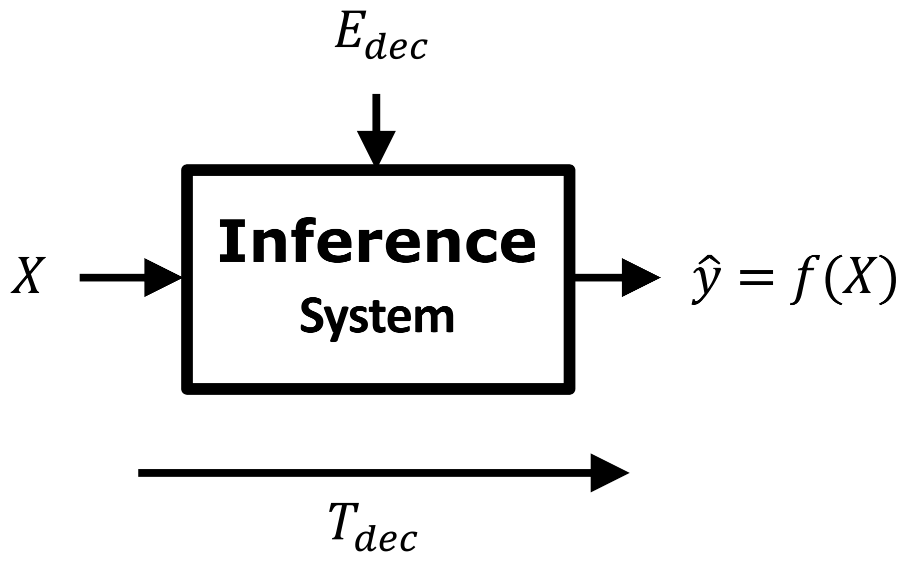


$$AP = \min(\mathrm{PP}, \mathrm{OI} \times \mathrm{BW})$$

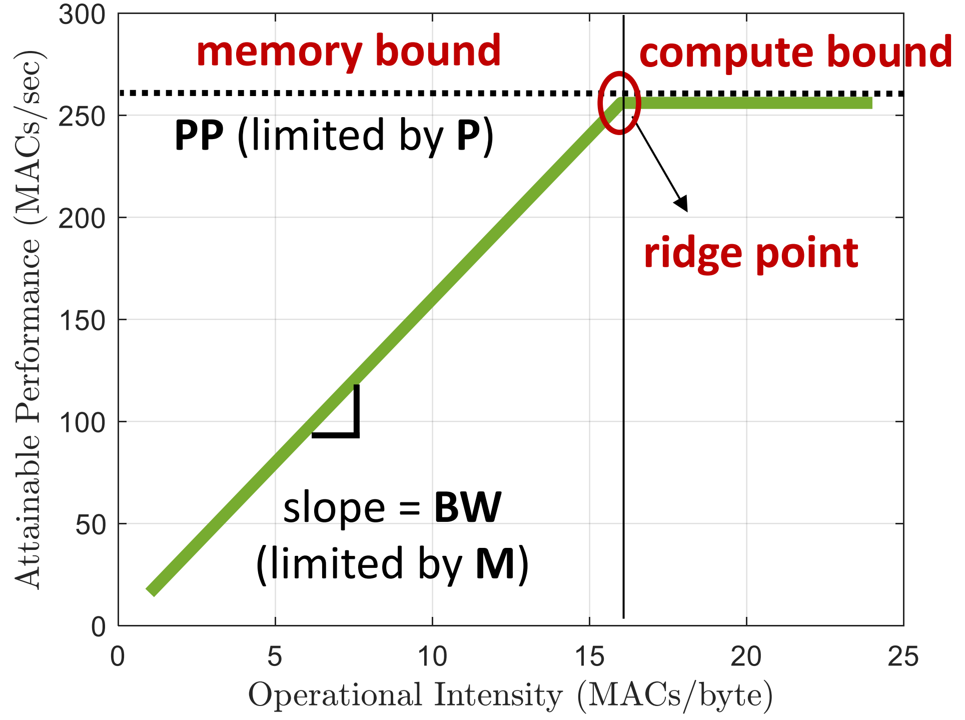
- AP: attainable performance of the architecture for the workload(s)
- Note: $AP \leq PP$

(Williams, Waterman, Patterson, Comm. of ACM '09)

System-level Performance - Bytes/Decision - M_{dec}



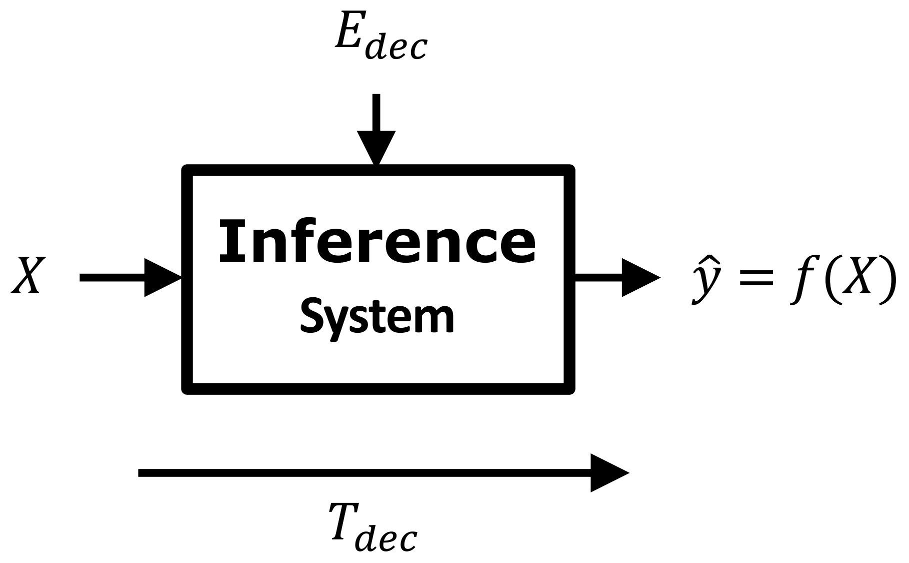
Roofline plot



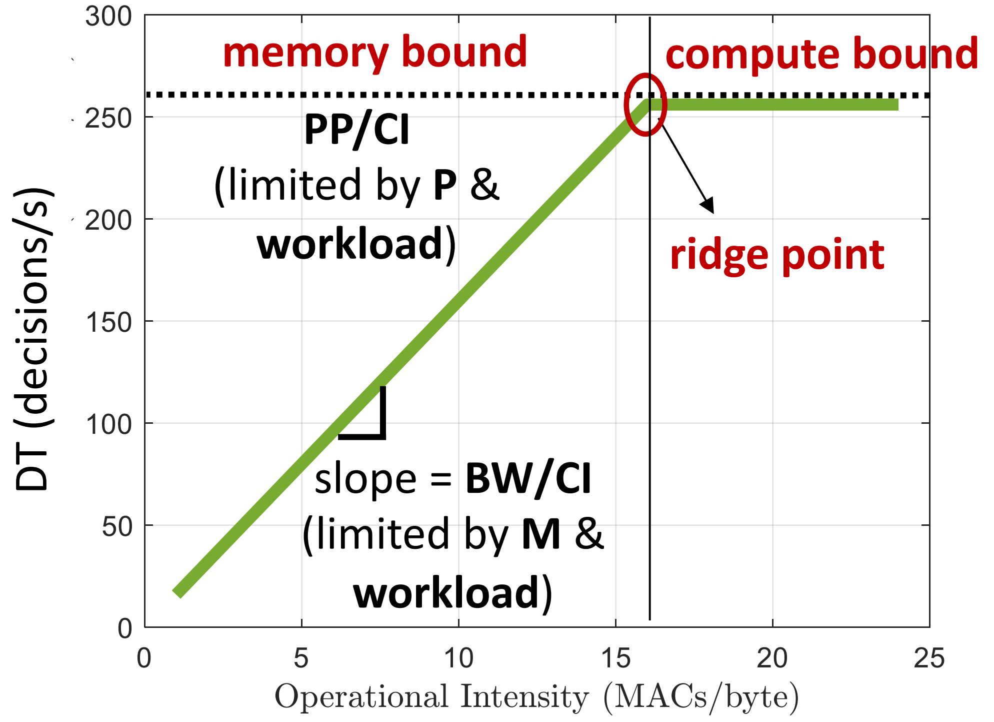
- Memory bandwidth (BW) $\neq$ bytes/decision (M_{dec})
- Given a workload OI and CI, bytes/decision :

$$M_{dec} = \frac{\text{bytes}}{\text{decision}} = \frac{\text{number of OPs/dec}}{\text{number of OPs/byte}} = \frac{CI}{OI}$$

System-level Performance - Decision Throughput (DT)



Roofline plot



- Attainable Performance (AP) $\neq$ Decision throughput (DT)
- Given a workload OI and CI, decision throughput (dec/s):

$$DT = \frac{\text{decisions}}{s} = \frac{\text{achievable OPs/s}}{\text{\# of OPs/dec}} = \frac{AP}{CI} = \min\left[\frac{PP}{CI}, OI \times \left(\frac{BW}{CI}\right)\right] = \min\left[\frac{PP}{CI}, \frac{BW}{M_{dec}}\right]$$

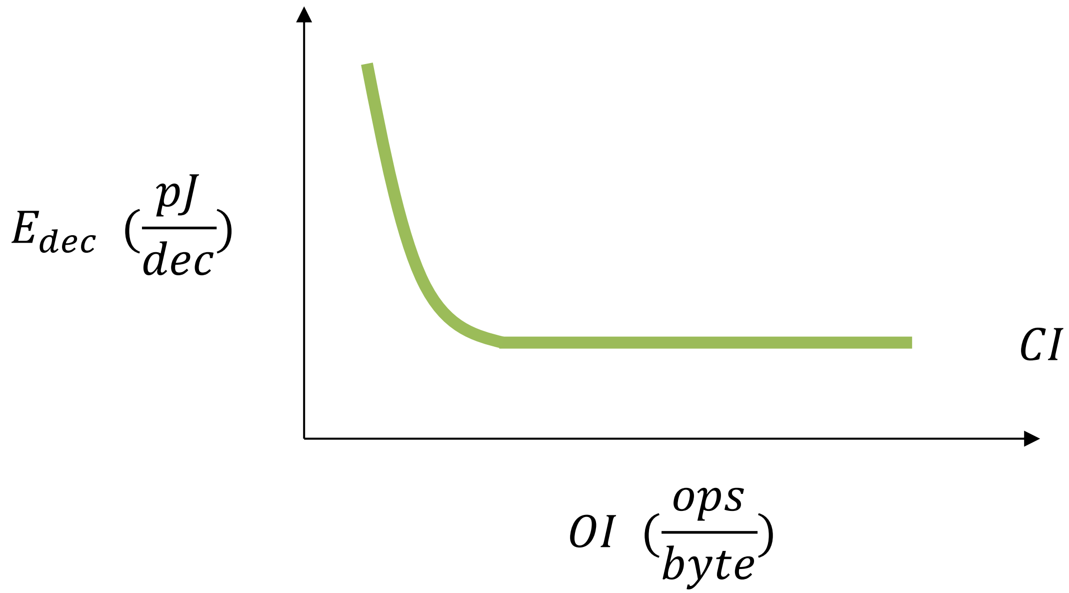
System Energy Efficiency – Floorline Plot

(Shanbhag, Dbouk, Sakr, ECE 498NSU/ECE 598NSG 2019)

- Decision energy (pJ/dec):

$$E_{dec} = CI \times E_P + M_{dec} \times E_M = CI \times E_P + \frac{CI}{OI} \times E_M = CI\left(E_P + \frac{E_M}{OI}\right) = CI \times E_P\left(1 + \frac{E_R}{OI}\right)$$

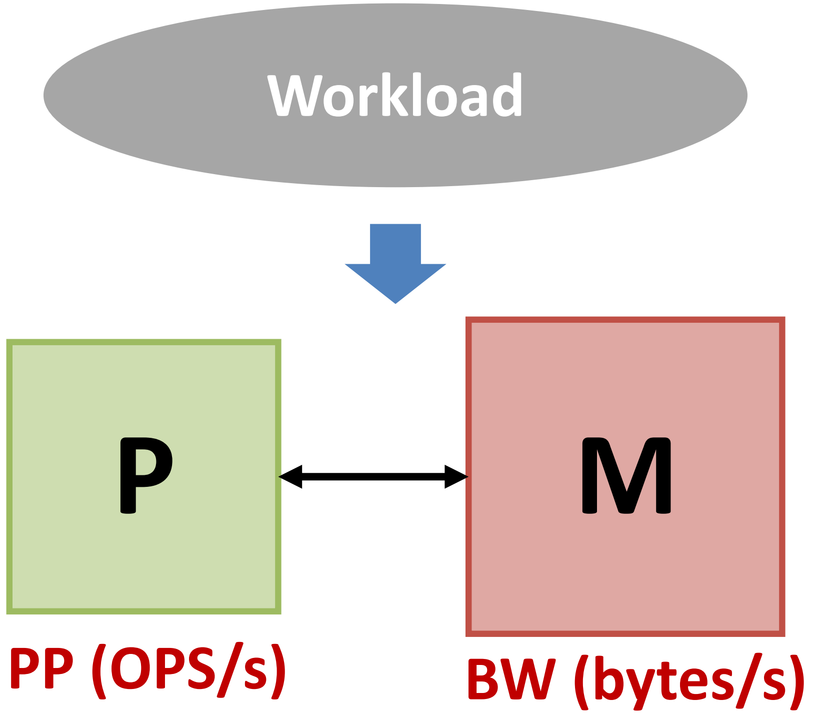
Floorline plot



$CI \times E_P$ (memory access cost = 0)

$$E_R = \frac{energy/access}{energy/OP} \sim 10^2 \text{ (SRAM)}$$

Maximizing Inference Throughput

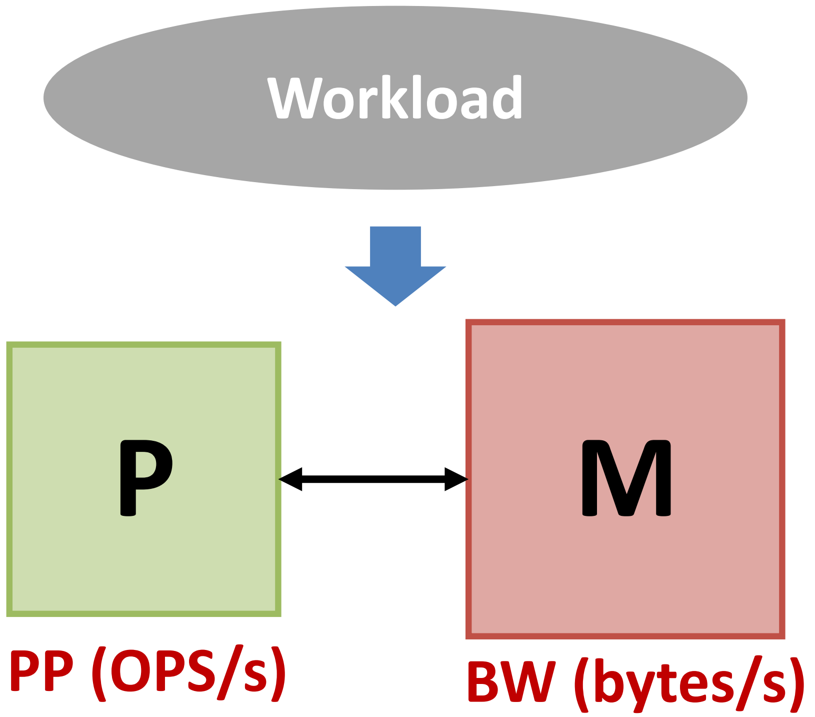


OI (OPS/byte)
CI (OPS/inference)

$$DT = \frac{AP}{CI} = \frac{\min(PP, OI \times BW)}{CI}$$

- Increase OI: data reuse & HW mapping → helps reduce decision energy
- Decrease CI: low-complexity DNNs, fast algorithms (FFT) → helps reduce decision energy
- Increase PP: systolic architectures & algorithm transforms
- Increase BW: new memory technologies + in-memory computing

Maximizing Energy Efficiency



OI (OPS/byte)
CI (OPS/inference)

$$E_{dec} = CI\left(E_P + \frac{E_M}{OI}\right)$$

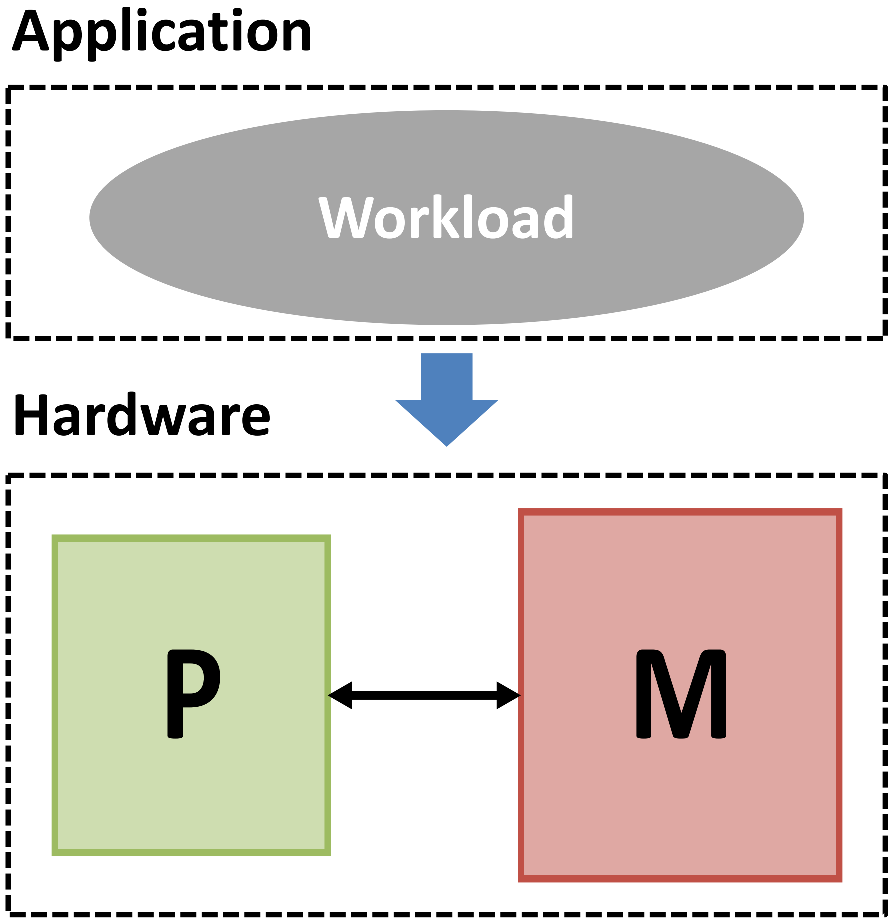
- Increase OI: data reuse & HW mapping → helps increase decision throughput
- Decrease CI: low-complexity DNNs → helps increase decision throughput
- Decrease E_P: energy-efficient computing (e.g. near-V_t computing), reduce precision
- Decrease E_M: new memory technologies + in-memory computing

Example

Given:

- an inference workload:
  - $CI = 1000\ \text{OPs/dec}$
  - $OI = 10\ \text{OPs/byte}$

- hardware platform with:
  - $PP = 256\ \text{OPs/s}$
  - $BW = 16\ \text{bytes/s}$
  - $E_P = 0.23\ \text{pJ/OP}$
  - $E_M = 2.5\ \text{pJ/byte}$



determine decision energy and latency

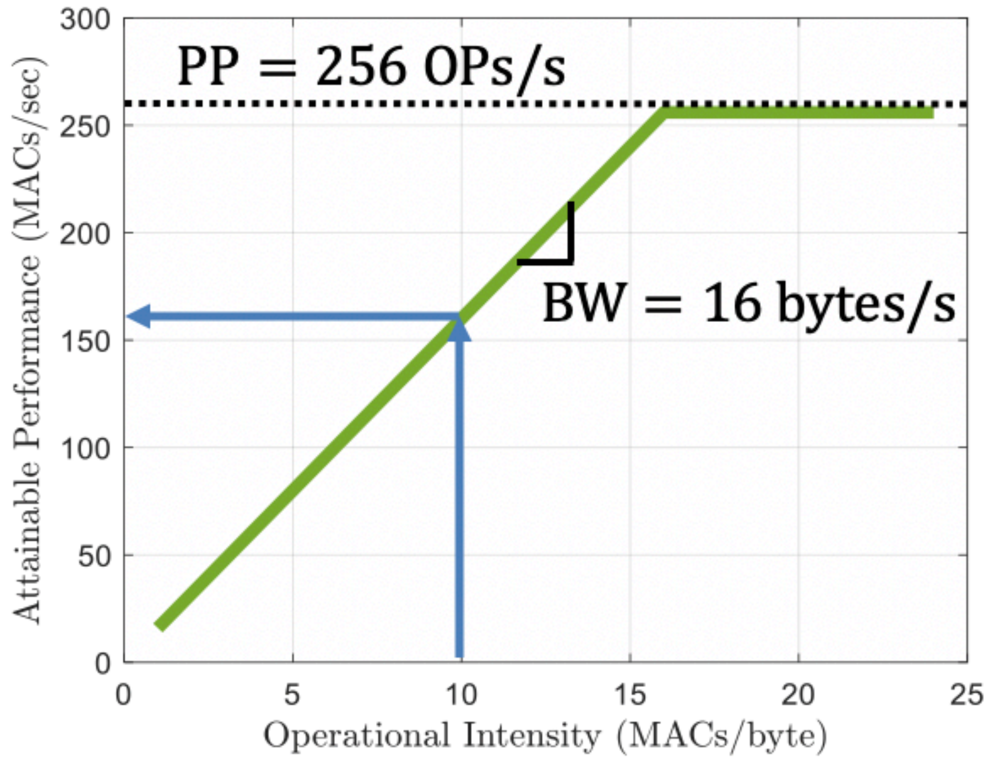
Throughput - Roofline plot

- Attainable performance:

$$AP = \min(PP, OI \times BW) = 160\,\mathrm{OPs/s}$$

- Decision throughput:

$$DT = \frac{AP}{CI} = \frac{160}{1000} = 0.16\ \mathrm{dec/s}$$



Energy Efficiency - Floorline plot

- Decision energy:

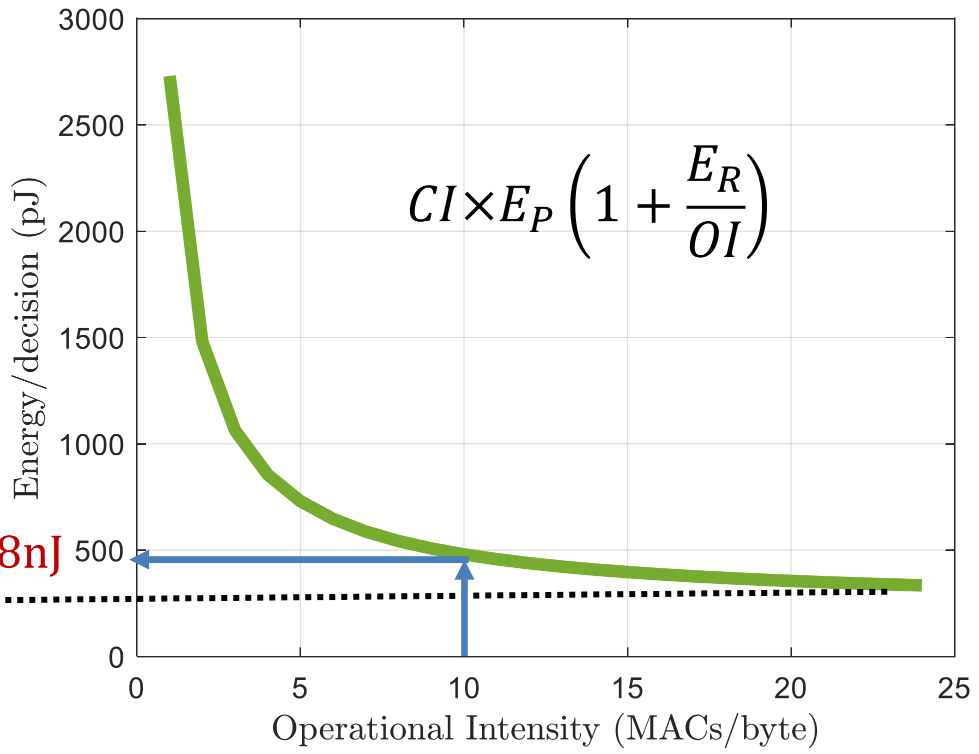
$$CI\left(E_P + \frac{E_M}{OI}\right) = 1000\left(0.23 + \frac{2.5}{10}\right)$$
$$= 0.48 \text{ nJ/dec}$$

- Floorline plot:

(lower bound) $CI \times E_P = 0.23$ nJ

$$E_R = \frac{E_M}{E_P} = 10.87$$
$$\mathrm{E_{dec}(nJ)} = 0.23\left(1 + \frac{10.87}{\mathrm{OI}}\right)$$

$E_{dec} = 0.48$nJ



Component-level Energy-Delay Trade-offs

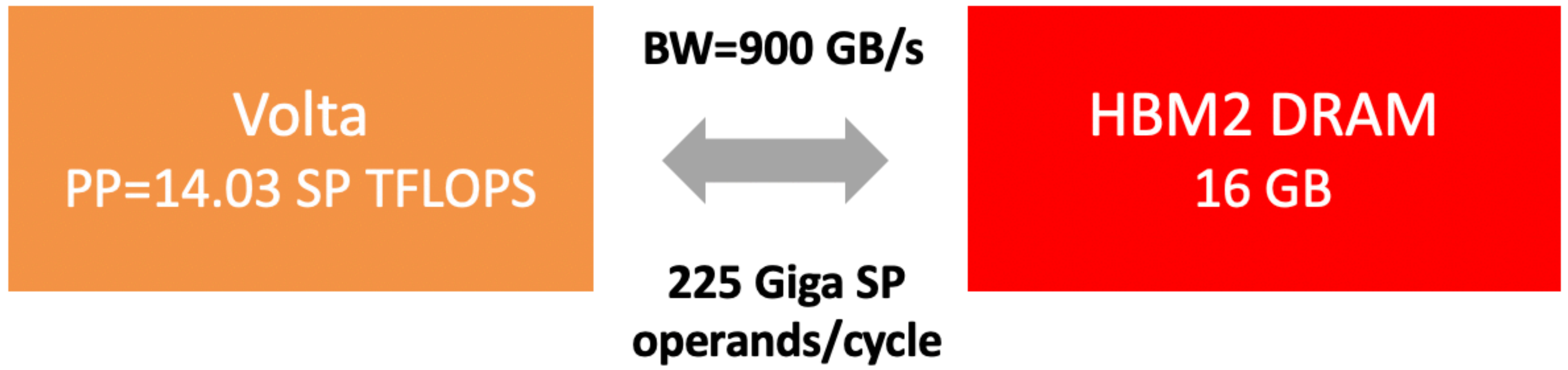
- Energy/OP (E_P) & Peak OP/s (PP) – low-power and high-throughput digital architectures and circuits (high-ceiling home)
- Bytes read/s (BW) & Energy/byte (E_M) – energy-efficient memory and interface design
- some of these are mature topics but keep evolving to meet the needs of machine learning applications

- energy consumption vs. delay (vs. robustness) trade-off
- energy vs. delay trade-off captured via **energy-delay product** EDP
- energy/delay vs. robustness trade-off is more subtle (future lecture)

Roofline Plot – A Visual Performance Model

Example – Memory Bound Scenario

[adapted from Hwu – IBM AI Compute Symposium, 2019]



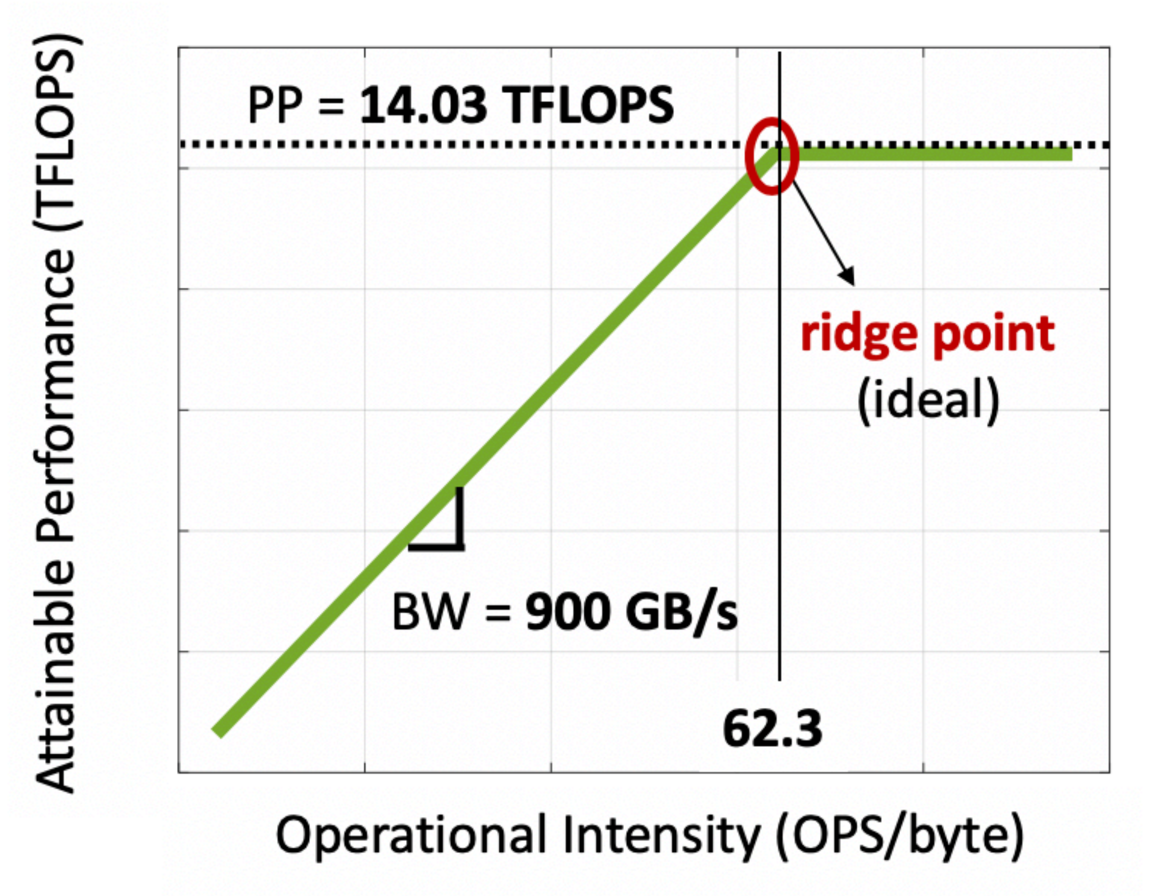
SP: single-precision (32-bit)

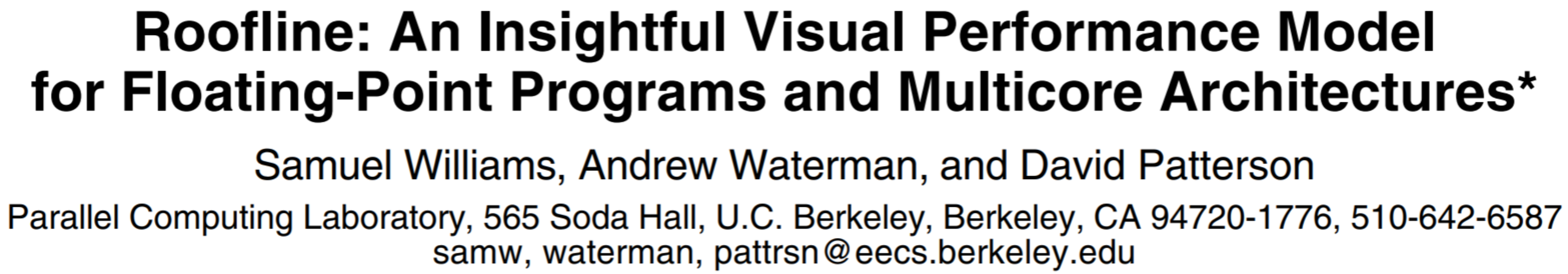
Each operand must be used **62.3 times** (reuse factor) once fetched to achieve peak FLOPS rate (ridge point)

or

Sustain < **1.6%** of peak without data reuse

$$\text{Achievable Performance } (AP) = \min(\mathrm{PP}, \mathrm{OI} \times \mathrm{BW})$$





Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures*

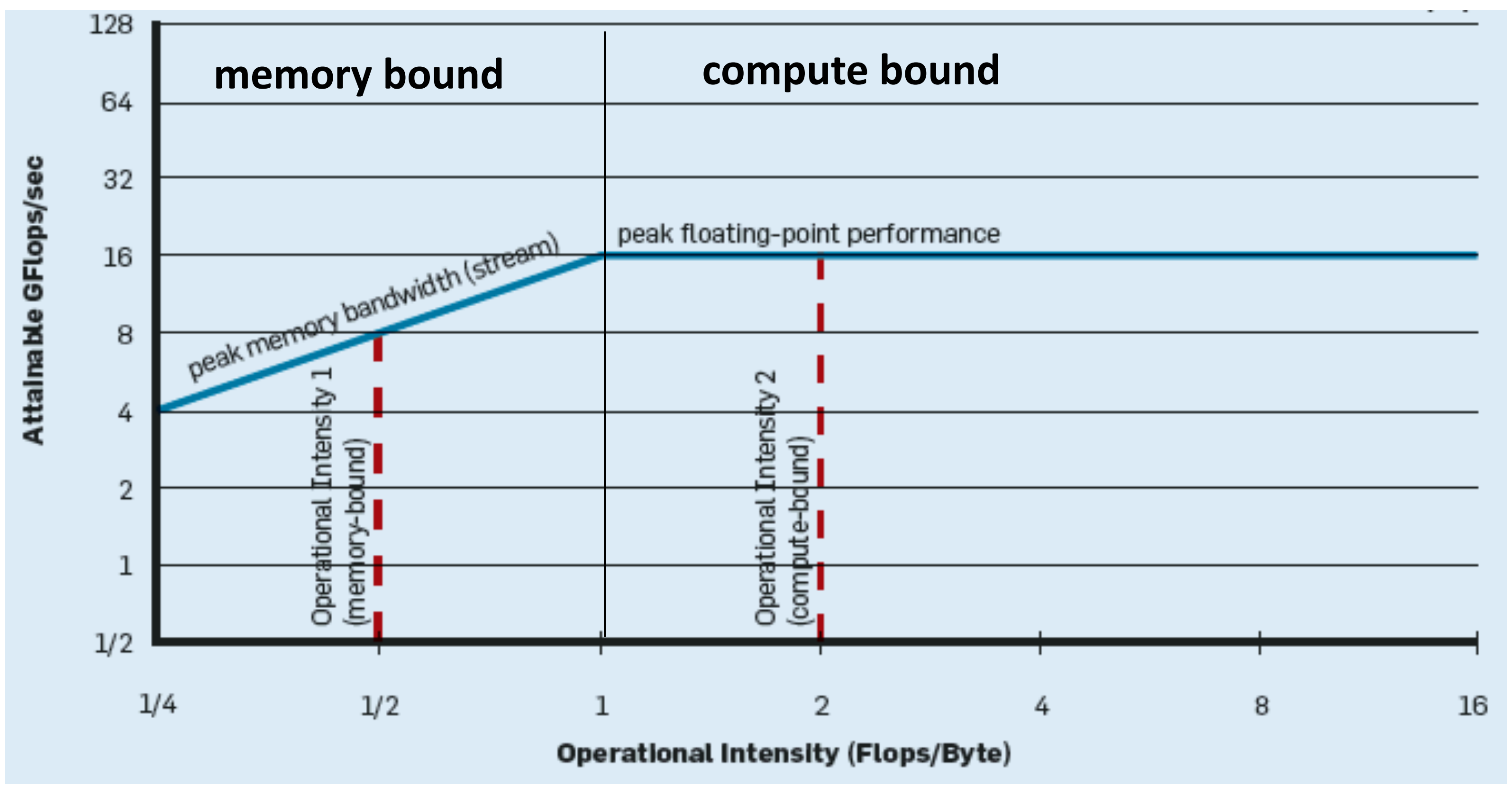
Samuel Williams, Andrew Waterman, and David Patterson

Parallel Computing Laboratory, 565 Soda Hall, U.C. Berkeley, Berkeley, CA 94720-1776, 510-642-6587
samw, waterman, pattrsn@eecs.berkeley.edu

"A model need not be perfect, just insightful."

- Motivation: off-chip memory bandwidth will limit the performance
- Need a simple model relates processor performance to off-chip memory traffic similar to Amdhal's Law
- **Amdhal's Law**: performance gain of a parallel computer is limited by the serial portion of the workload

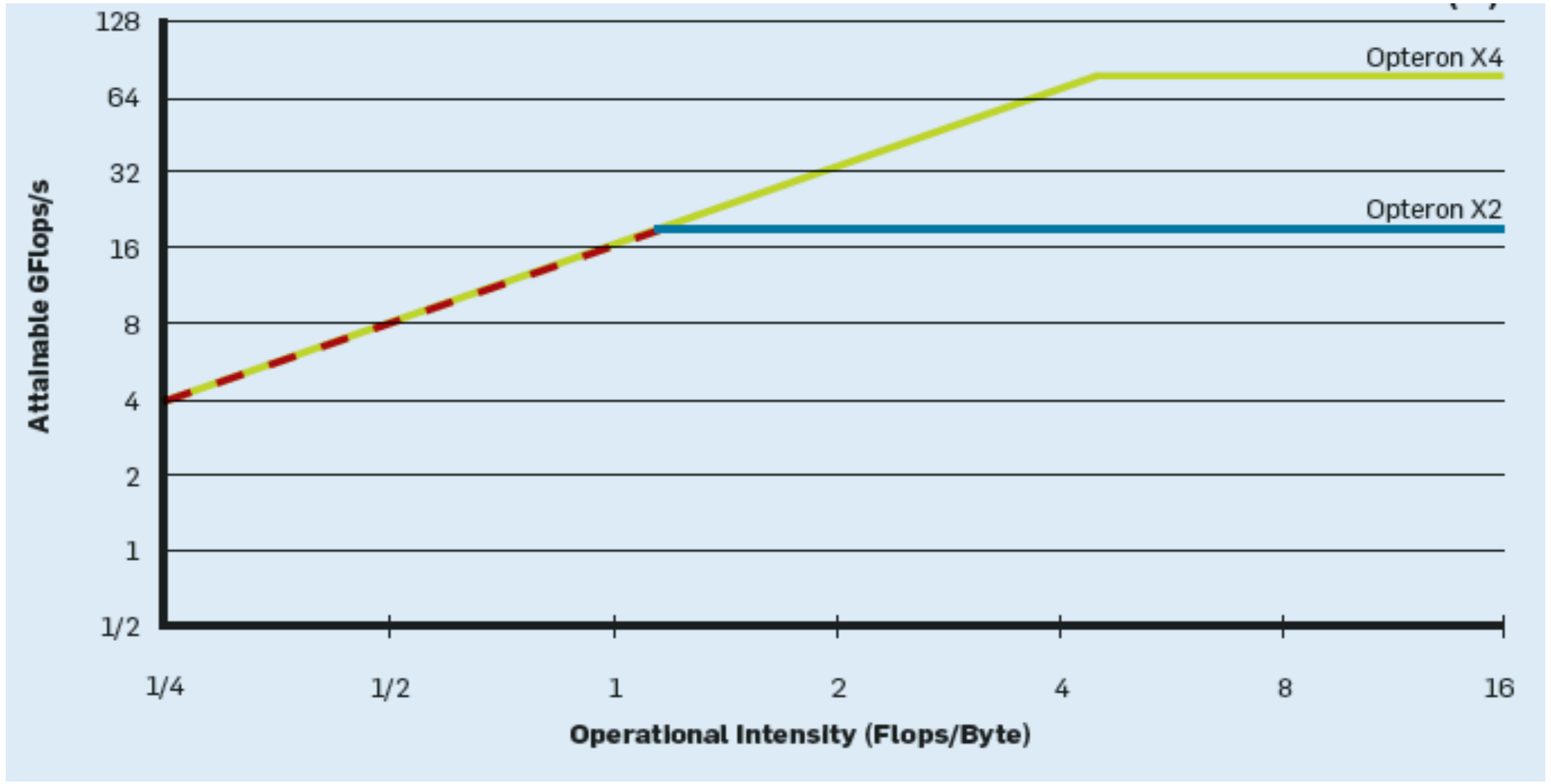
Roofline for AMD Opteron X2



[Williams, ACM Communications, 2019]

- 2 core processor
- BW = 15 GB/s; PP = 17.6 GFlops/s

Opteron X2 vs. X4



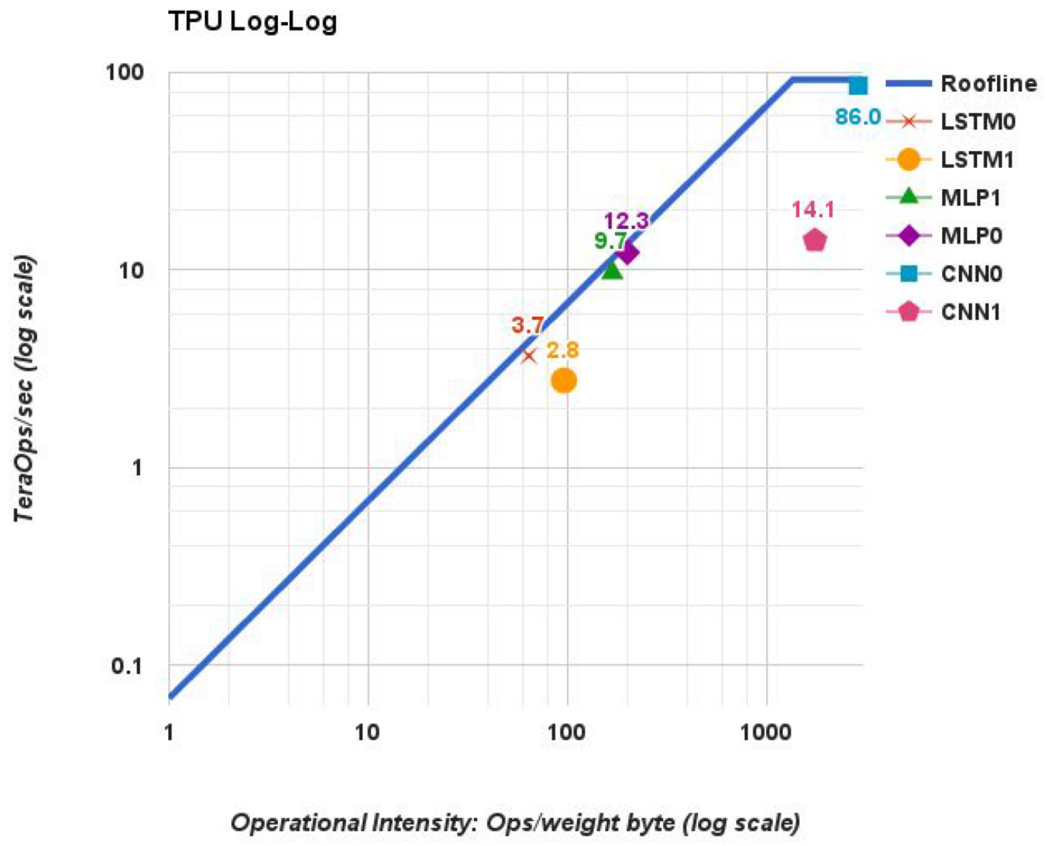
- X2 = 2 core (2.2 GHz) X4 = 4 core (2.3 GHz)
- workloads with > 1 Flop/Byte OI needed for X4

Roofline Plots – The Reality

- Roofline plot provides a ceiling on the achievable performance

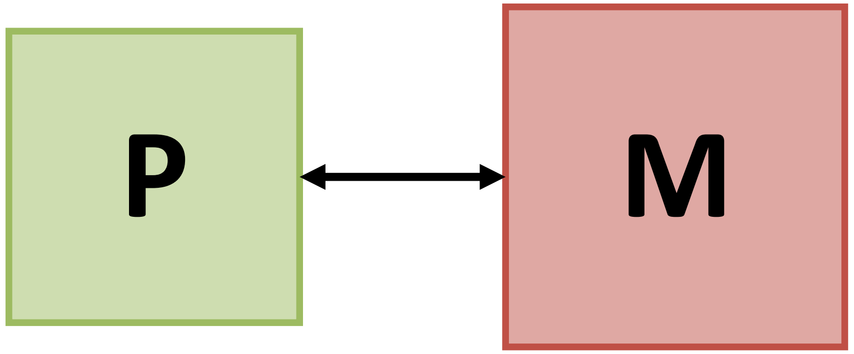
- They ignore intricate data movement required by applications

- Different layer shapes might cause idle PEs => unutilized resources



[TPU – ISCA'17]

Dot Product Example



- Processor **P** with a peak performance (PP) of 256 operations/sec
- Memory **M** with a bandwidth (BW) of 16 bytes/sec
- Workload: perform a 1D "convolution" (correlator):

$$y_i = \sum_{j=1}^{M} w_j x_{i+j-1}$$

where $x \in \mathbb{R}^N, w \in \mathbb{R}^M \Rightarrow y \in \mathbb{R}^{N-M+1}$ (with zero-padding), and $M \in [N]$

assume 32b floating point operations; $N = 128$;

Maximum AP

- Assume we only have to fetch w from **M**.

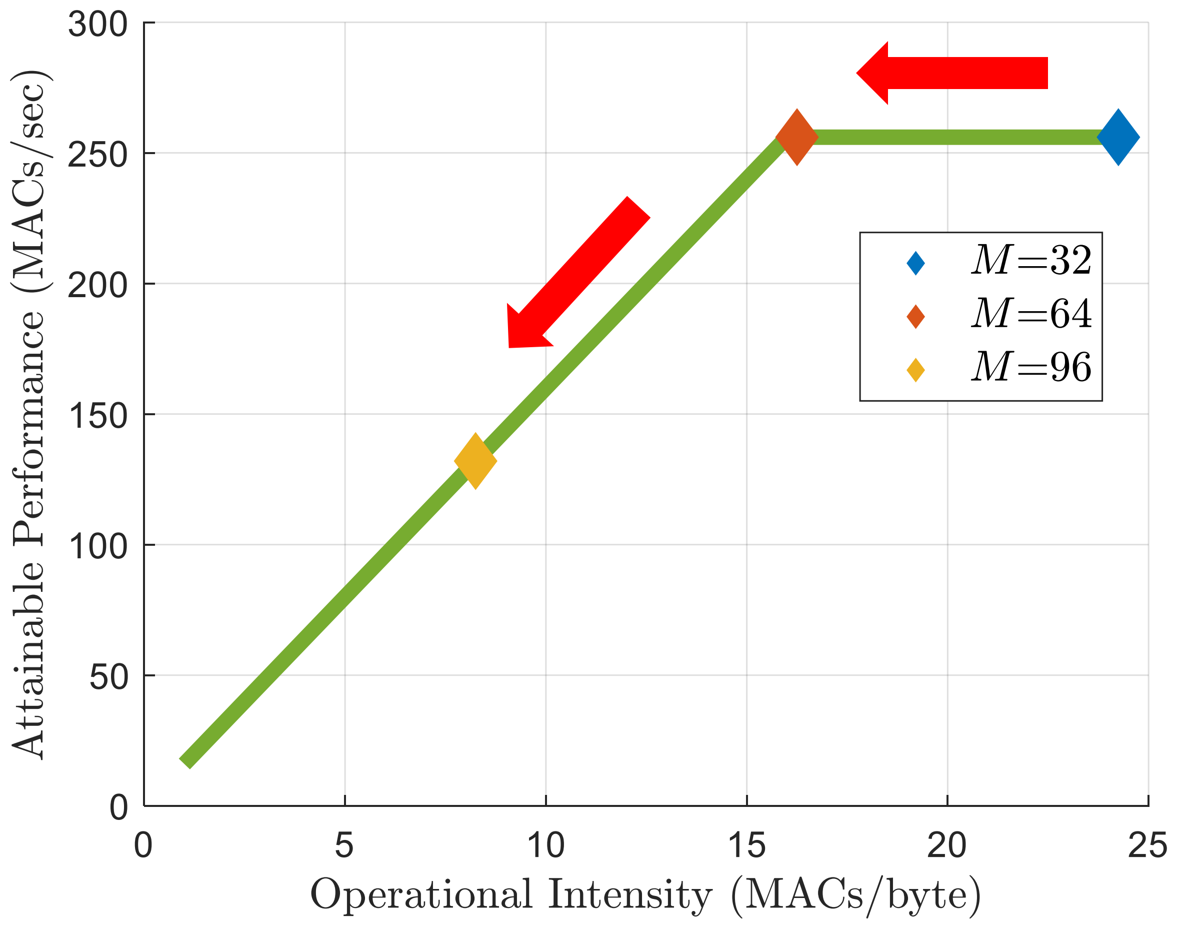
- OI for 1D convolution (DP):

$$OI = \frac{\text{number of MACs}}{\text{number of bytes read}} = \frac{\overbrace{(N-M+1)}^{\text{number of dot products}}\times \underset{\text{\# MACs/DP}}{M}}{\underset{\text{each weight value (32b) needs 4 memory reads}}{4}\times M}$$

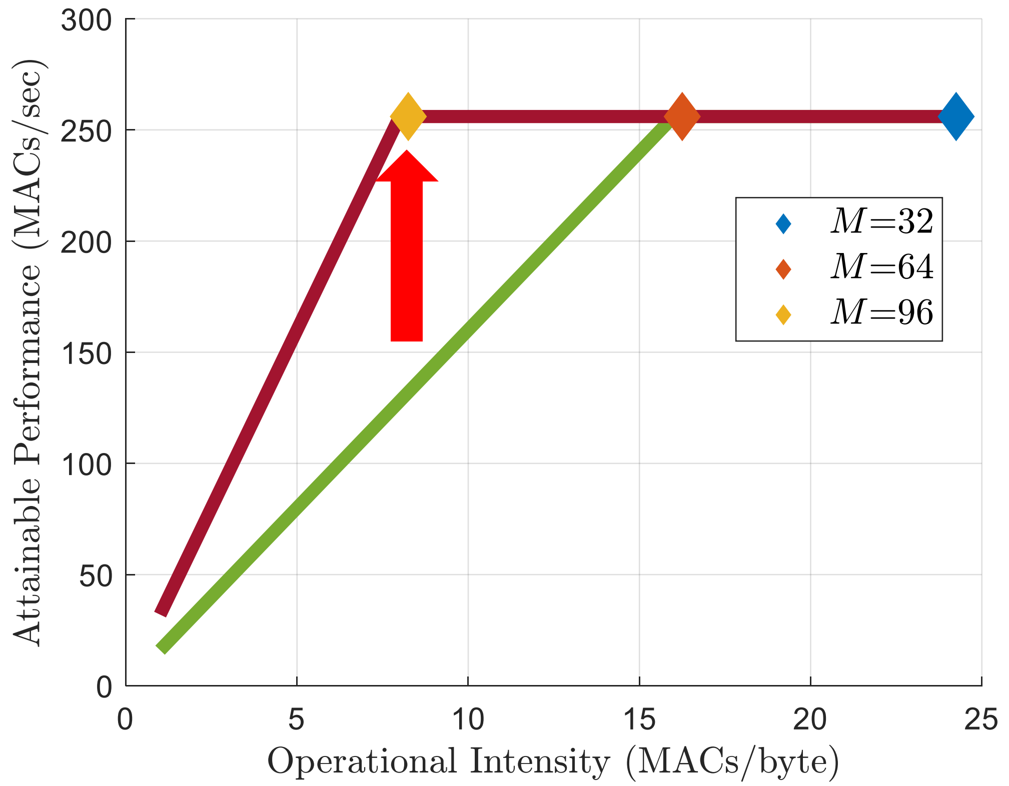
$$OI = \frac{(N-M+1)}{4}$$

- OI **reduces** with kernel size M
- OI **increases** with input data size N

Varying Kernel Length (M)

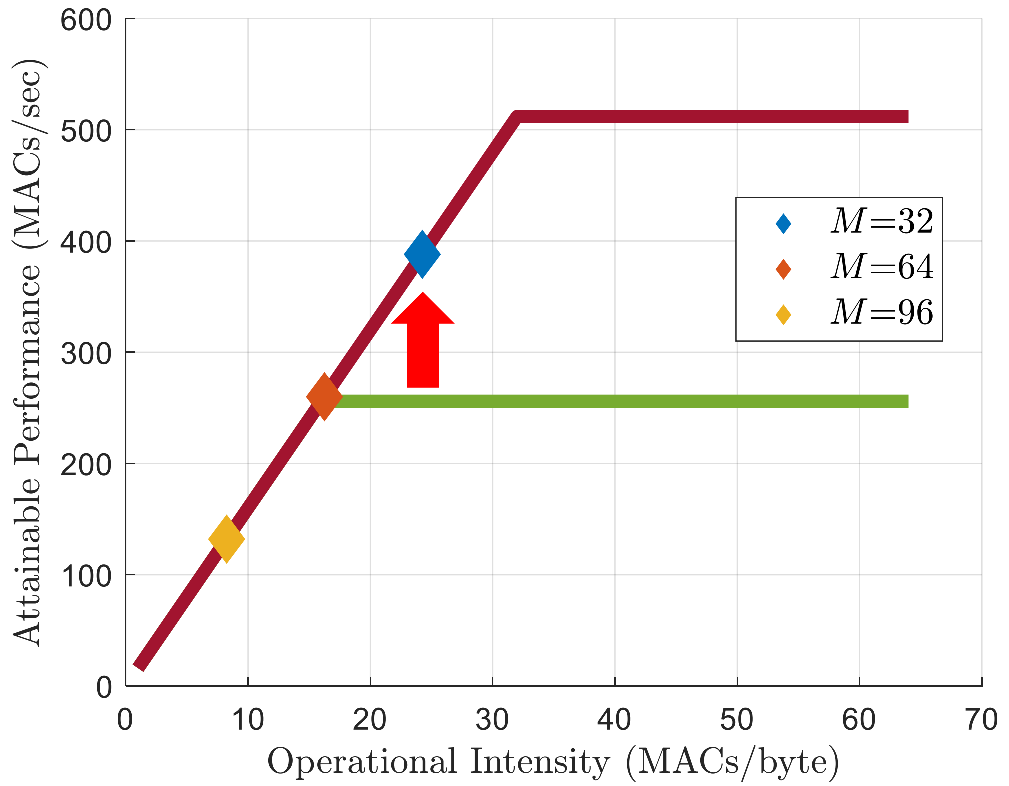


Doubling Memory BW



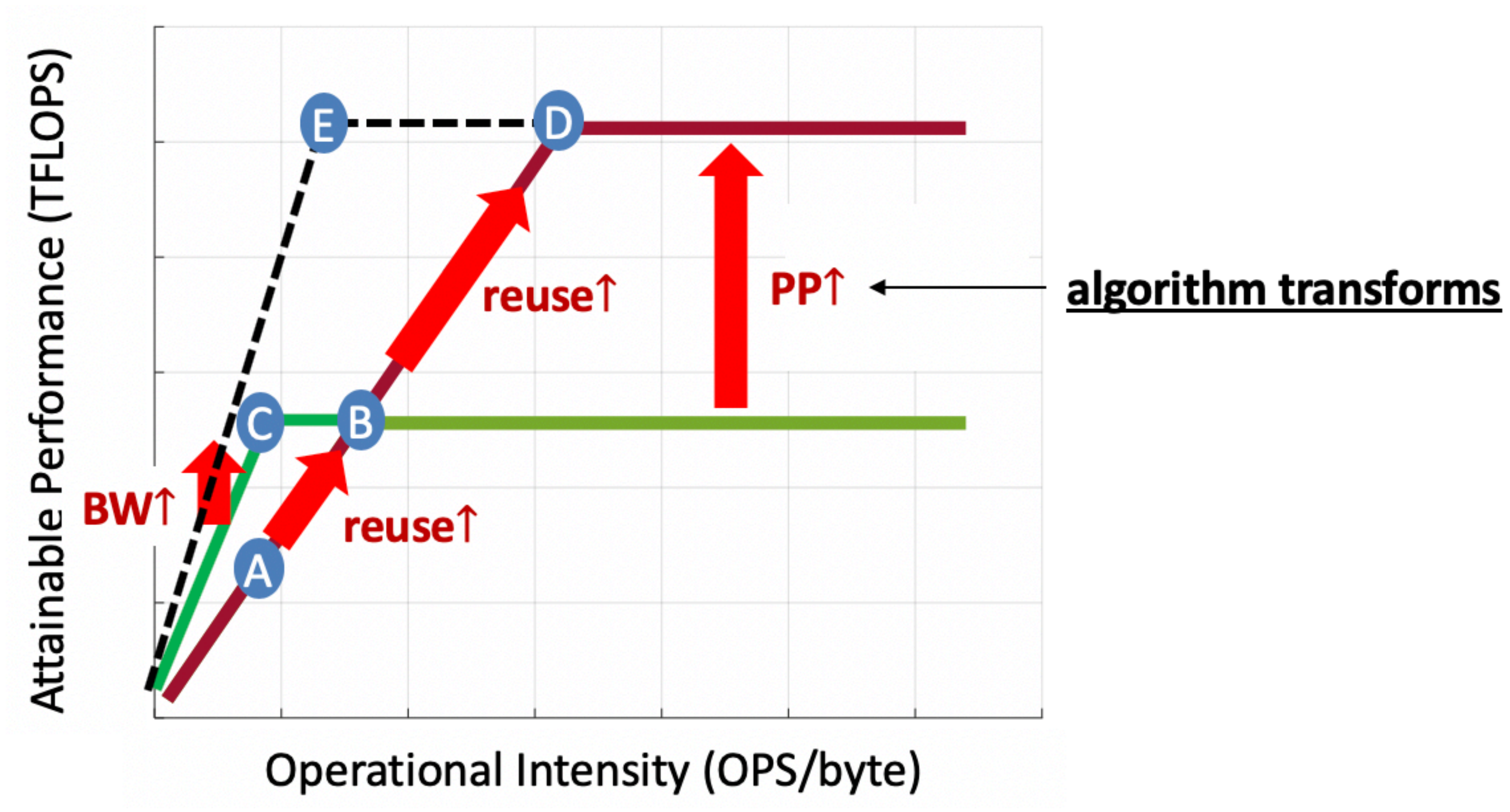
- $M = 32$: AP is *compute bound* → no benefit from doubling BW
- $M = 96$; AP is *memory bound* → benefits from doubling BW

Doubling Peak Performance (PP)



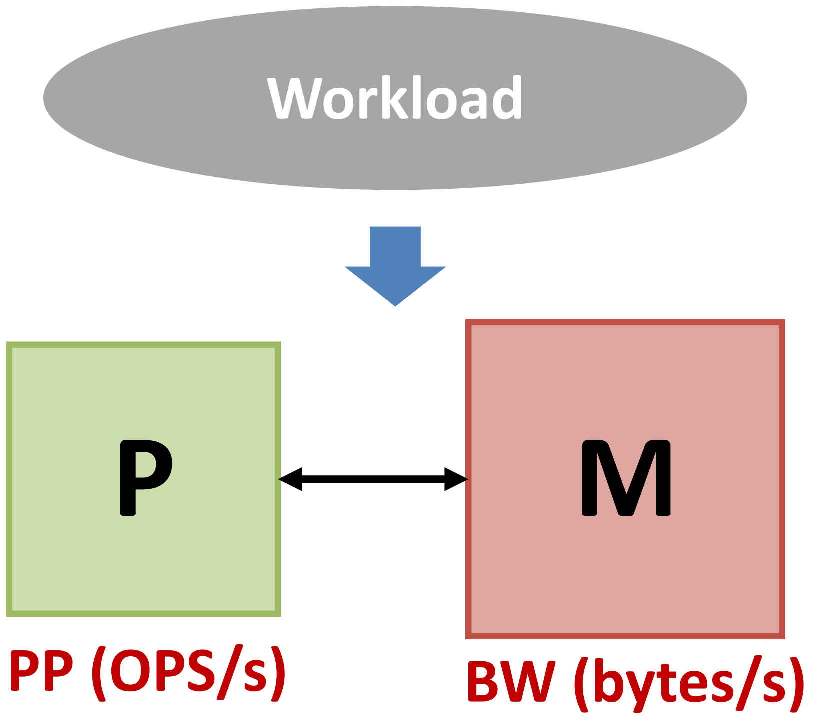
- $M = 32$: AP increases and it becomes *memory bound*
- $M = 96$: AP is already *memory bound,* so no change

Methods to Improve AP



- A → B: same architecture - **enhance reuse**
- A → C: **same reuse** & processor – enhance BW
- B → D: same interface – **enhance reuse** and PP
- A → E: tunable - **enhance reuse**, BW and PP

Data Reuse



OI (OPS/byte)
CI (OPS/inference)

$$DT = \frac{AP}{CI} = \min\left[\frac{PP}{CI}, OI \times \left(\frac{BW}{CI}\right)\right]$$

$$E_{dec} = CI\left(E_P + \frac{E_M}{OI}\right)$$

- **Increase OI: data reuse** & HW mapping
- Decrease CI: low-complexity DNNs
- Increase PP: systolic architectures & algorithm transforms
- Increase BW: new memory technologies + in-memory computing

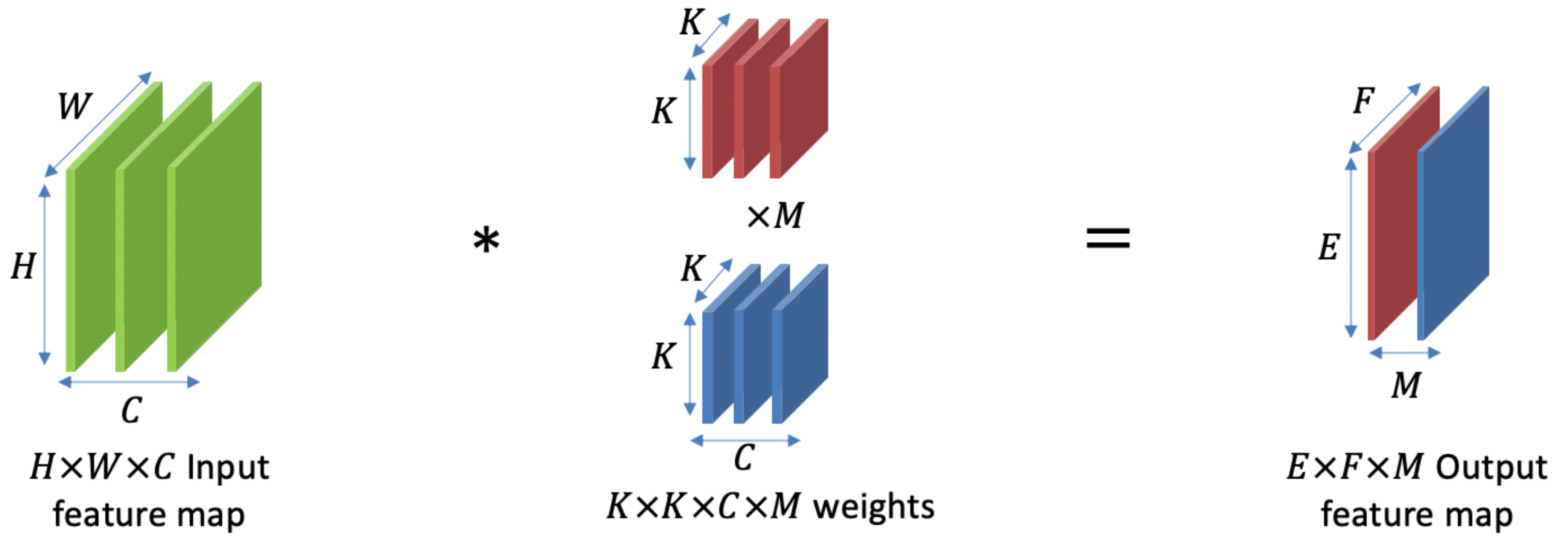
Terminology

- **Data reuse**: # of MACs per data value **read** (i.e., MACs/data).
- For every MAC operation involved in a convolution operation, there are 3 data types:

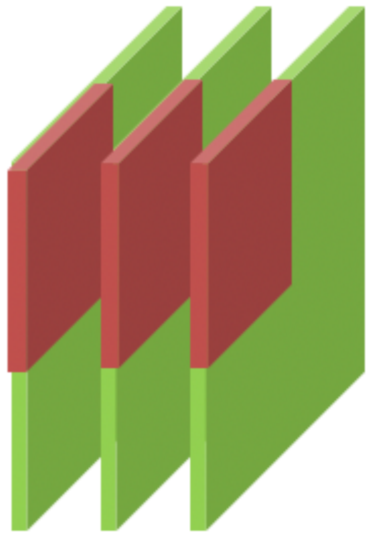
$$\begin{tikzcd} & \textbf{weight} & \\ \textbf{output partial sum} & z \leftarrow z + w \times x \arrow[l] \arrow[u] \arrow[r] & \textbf{input activation} \end{tikzcd}$$

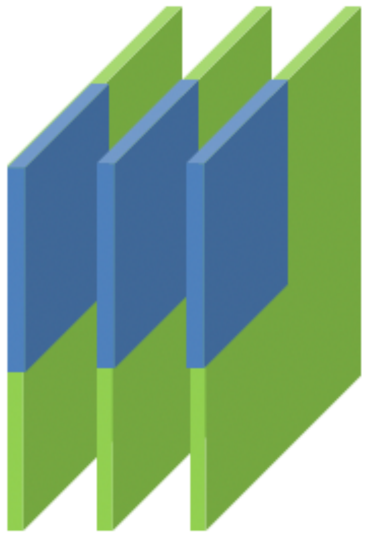
- Hence, three types of data reuse: input, weight, and partial sum
- Partial sum reuse = DP length = # of partial sums needed to compute one DP

2D Convolution


W
H
C
$H \times W \times C$ Input feature map
$*$
K
K
$\times M$
K
K
C
$K \times K \times C \times M$ weights
$=$
F
E
M
$E \times F \times M$ Output feature map

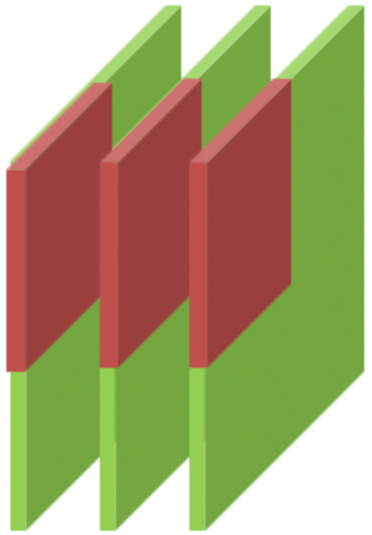
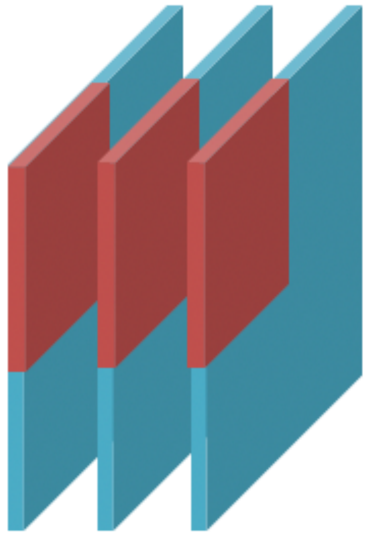
Input Reuse





- Each **input pixel** is **reused $\sim K \times K$ times** across the same weight filter
- Each **input pixel** is **reused M times** due to M filters processing the same input feature
- Total input reuse: $K^2 M$
- e.g., Layer 1 of VGG-16 on ImageNet – $3 \times 3 \times 64 = 576$ MACs/byte = OI

Weight Reuse

- Each **weight pixel** is **reused $E \times F$ times** across the same input feature map
- Each **weight pixel is being used N times** when multiple $(N > 1)$ input batches are processed
- Total weight reuse: $E \times F \times N$
- e.g., Layer 1 of VGG-16 on ImageNet – $224 \times 224 \times 1 = 50176$ MACs/byte = OI

Course Web Page

https://courses.grainger.illinois.edu/ece598nsg/fa2020/
https://courses.grainger.illinois.edu/ece498nsu/fa2020/

http://shanbhag.ece.uiuc.edu