A WATER-FLOW ANALOGY FOR TEACHING DATA REUSE AND MEMORY HIERARCHIES

Tanvi Sharma

Kaushik Roy

PURDUE UNIVERSITY

Elmore Family School of Electrical and Computer Engineering

Workshop on Computer Architecture Education

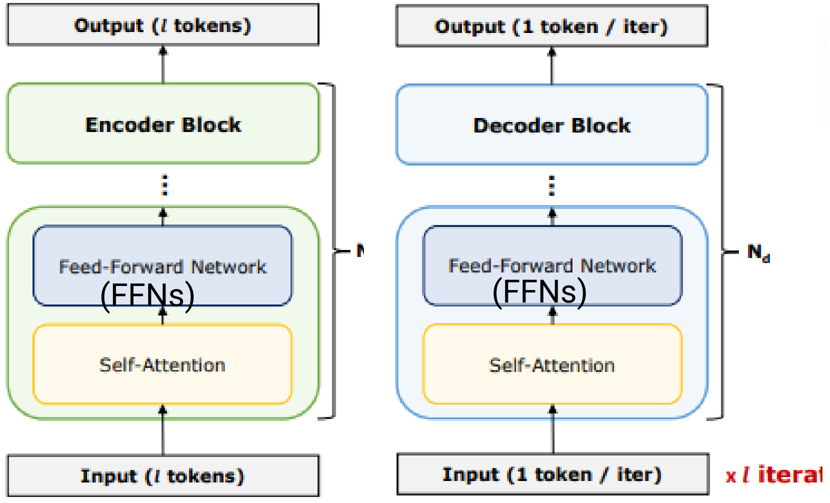
In conjunction with the 52nd International Symposium on Computer Architecture
21 June 2025
Waseda University, Tokyo, Japan
Okuma Auditorium-small

Overview

- Motivation
  - Lowering the barrier to entry for AI hardware in one lecture for undergraduates
  - Emphasis on the impact of data reuse on performance
- Outline
  - Examples of AI workloads
  - A simple memory-compute model for AI hardware
  - Introduction to water-flow analogy
  - Memory- and compute- bound scenarios to understand data reuse
  - Relation with roofline performance model
  - Latency-bound scenario and implications
  - Conclusion

Artificial Intelligence (AI) Workloads

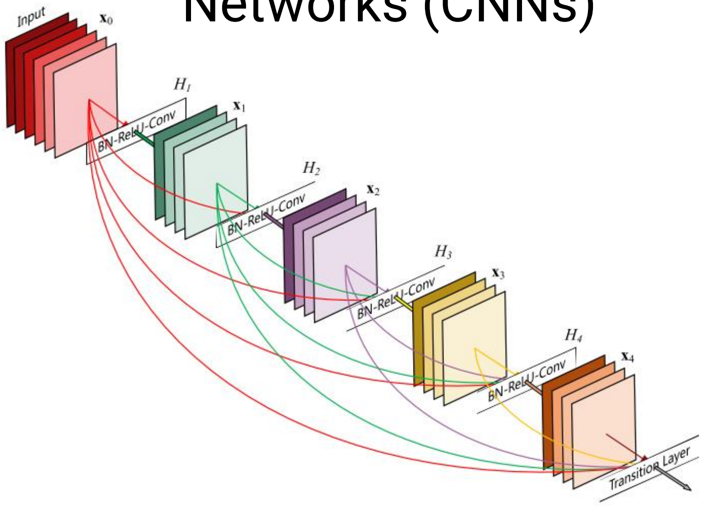
Transformer models



(a) Encoder-Only (b) Decoder-Only

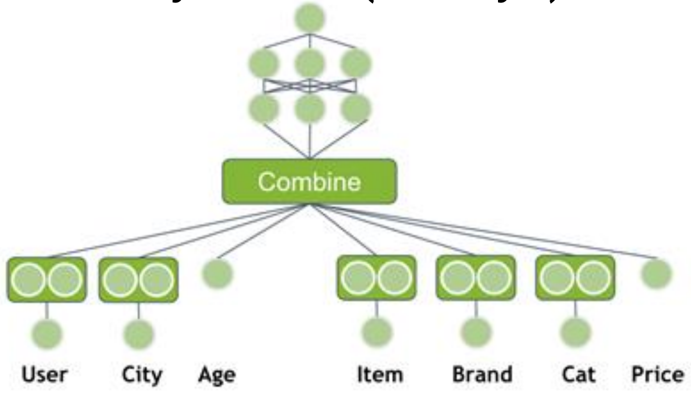
Full Stack Optimization of Transformer Inference, 2023

Convolutional Neural Networks (CNNs)



ResNet Depiction

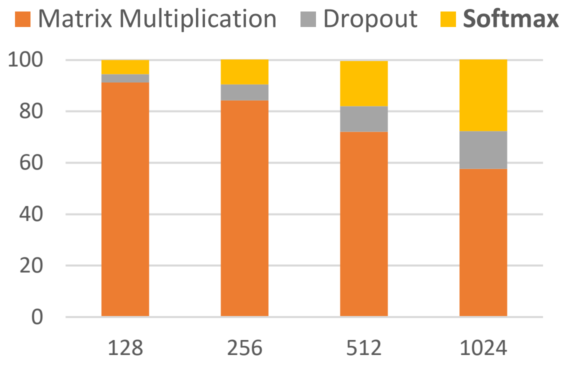
Recommendation Systems (RecSys)



RecSys, Nvidia Blog

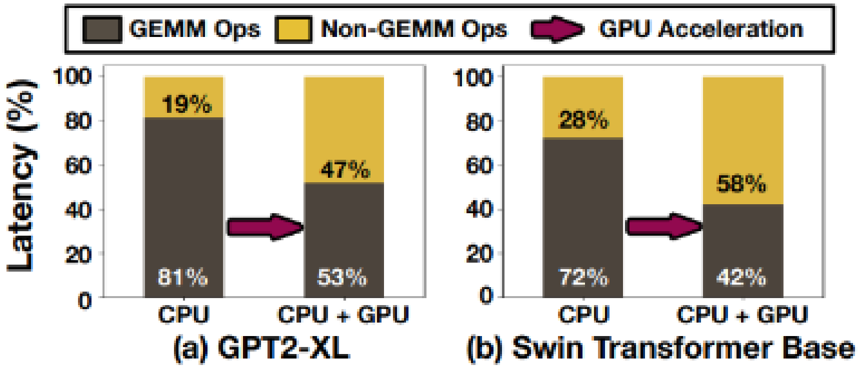
- Artificial Intelligence (AI) is used everywhere in day-to-day applications such as chat-bots, text autocompletion, ad recommendations, summarization tasks and image recognition.
- AI model architectures – Transformers (most common these days), CNNs, FFNs, RecSys, etc.

Dominant Computation in AI workloads

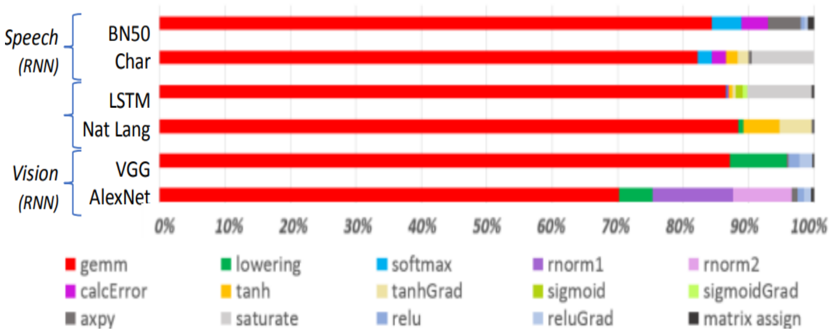


BERT-Large Latency % with sequence length

Softermax. DAC 2021.



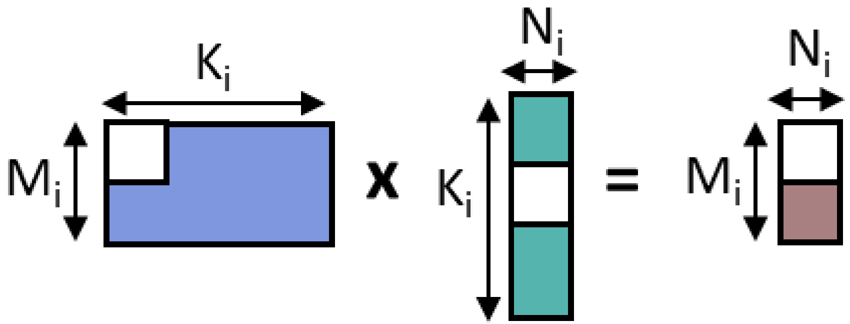
NonGEMMBench. ISPASS 2025



RNN Runtime Distribution

Approximate Computing AI Acceleration. IBM Blog 2018.

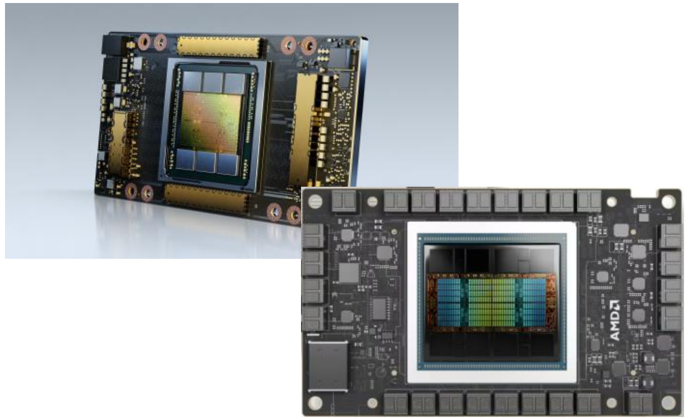
- Most AI workloads are dominated by GEMMs (GEneral Matrix Multiplications).
- GEMMs are represented as GEMM(M, N, K).
- AI hardware is designed to maximize performance of GEMMs.

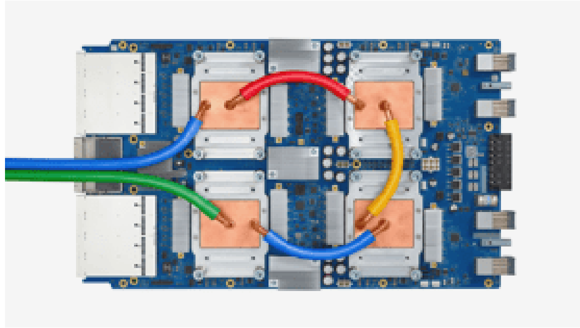


GEMM (M_i, Ni, Ki)

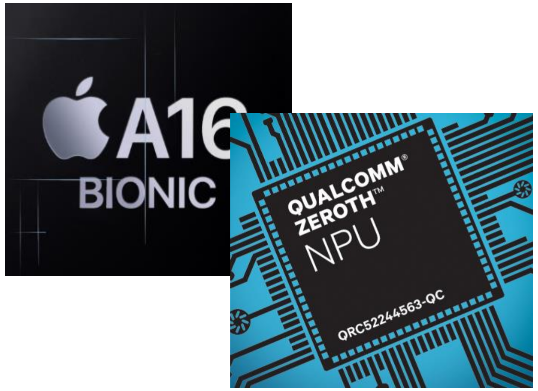
Common Examples of AI Hardware

- Graphics Processing Units (GPUs) – started as domain specific ICs for accelerating graphics, now general-purpose programmable IC.
- Tensor Processing Units (TPUs) – Domain specific IC specially designed for AI processing.
- Neural Processing Units (NPUs) – Application Specific IC designed for one end system.

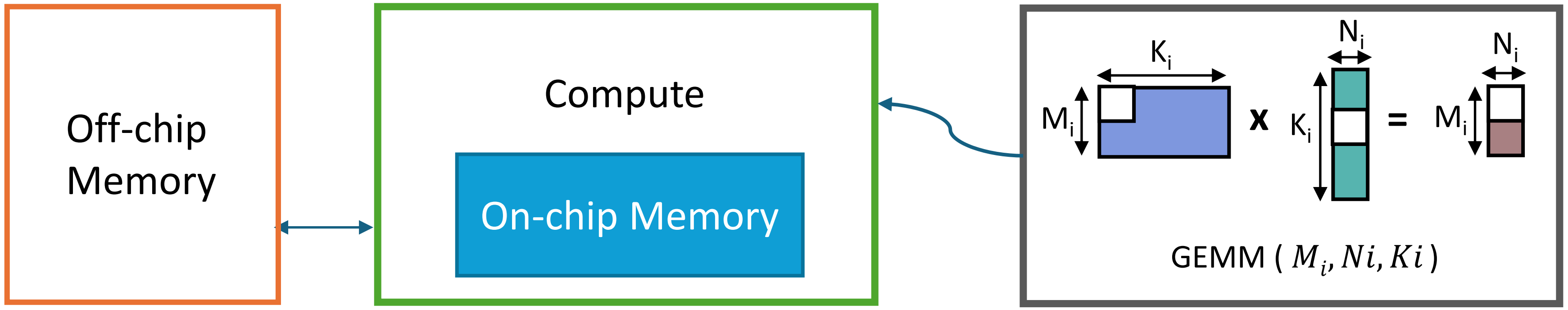




What do they have in common?



Simple Memory-Compute Model for AI Hardware

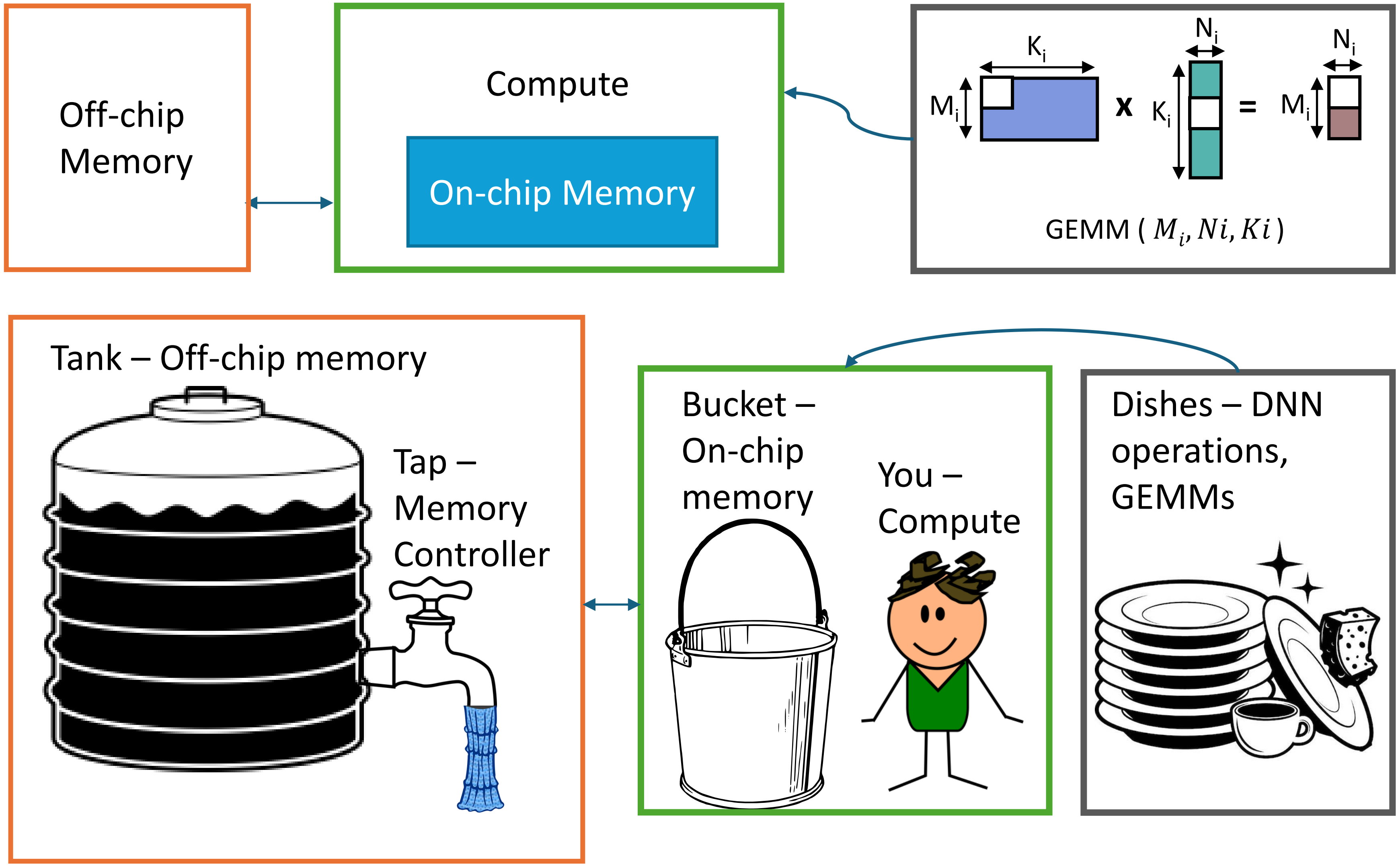


A simple analytical model of AI hardware consists of

- off-chip memory (say DRAM or HBM)
- on-chip memory (SRAM)
- some compute units (ALUs or SIMD or Systolic Arrays)

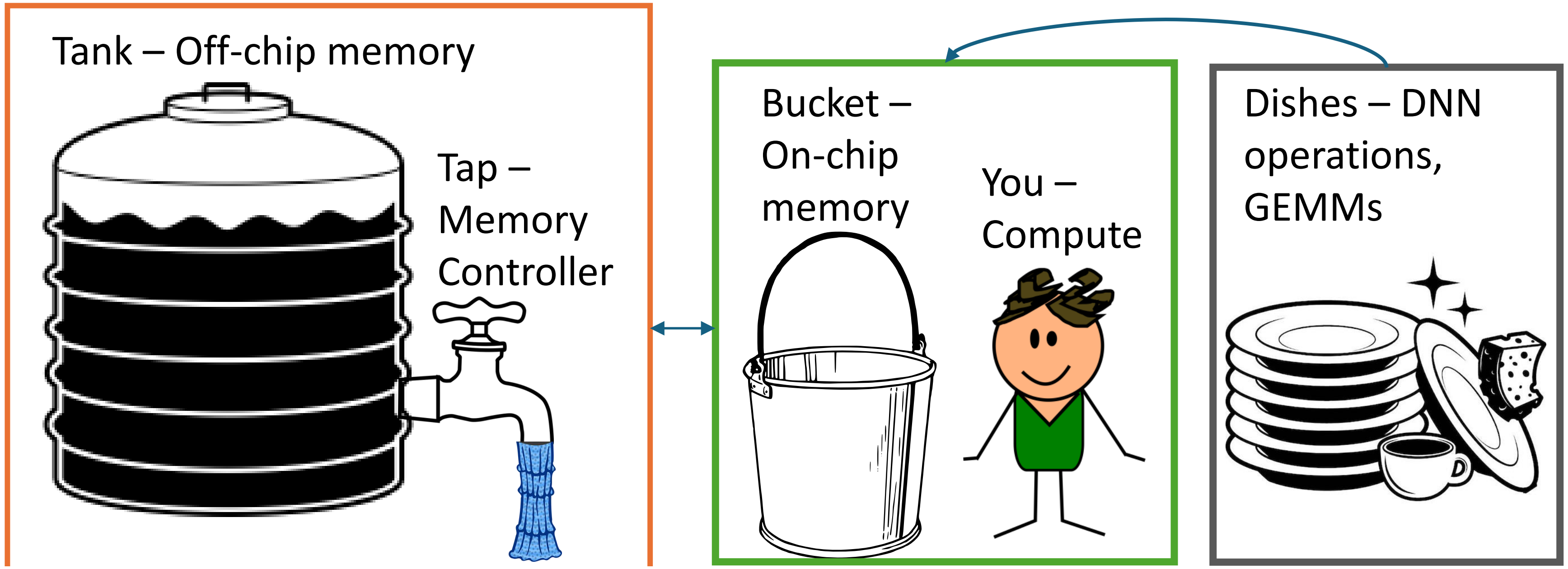
Focus of this lecture - understand how data reuse in GEMMs impacts the hardware performance through a water-analogy.

Waterflow-Analogy for Simple Memory-Compute Model



- Water – Data (bytes)
- Rate of doing dishes – Performance (operations/sec)

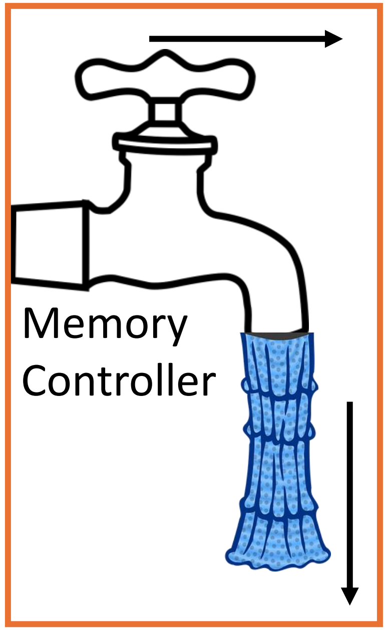
Waterflow-Analogy for Simple Memory-Compute Model



Given the number of DNN operations and limited BW constraint, how to maximize R_{obs}?

Metric	Architecture Definition	Water analogy definition
Latency (sec)	Time bw initiating a data request until it is retrieved	Time bw turning the tap on to receiving water
Bandwidth (BW)	Rate of data transfer	Rate of water flow
Data (L)	Byte	Water (liter)
OPs	Compute Operations	Number of dishes
Data Reuse	Arithmetic Intensity (AI) or OPs per byte	Total number of dishes done per liter of water
Reuse Factor (F_{reuse})	New AI = $AI_{org} \times F_{reuse}$	No. of dishes reusing water
T_{data}	Total time to fetch data	Total time to fetch water required to do all dishes
T_{comp}	Ideal time to perform all OPs	Ideal time to do dishes with no water constraint
R_{obs}	Operations/second (OPS or OPs/sec); Throughput	Observed rate of washing dishes (dishes/s)

Waterflow-Analogy Model Illustration (1)



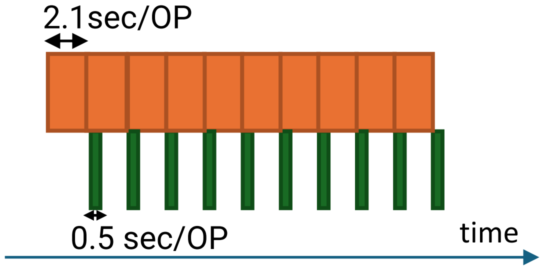
Latency
= 0.1 sec

BW
= 1L/sec

Assuming each dish requires 2L of water,

Peak rate (R_{peak})
= 2 OPS * 2 L/op
= **4 L/sec**

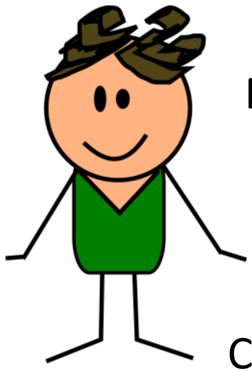
Case 1: Reuse factor (F_{reuse}) = 1
OPs = 10
Data (required)
= 2 *10 / F_{reuse}
= 20 L
T_{data} = L + Data/BW
= 0.1 + 20/1
= 20.1 sec
T_{comp} = Data/R_{peak}
= 20/4
= 5 sec
R_{obs} = 10/(20.1 + 0.5) OPS
= 0.485 OPS



Say, Ops = 100.

How can we improve the performance?

Does rate-matching memory bandwidth and compute throughput help?

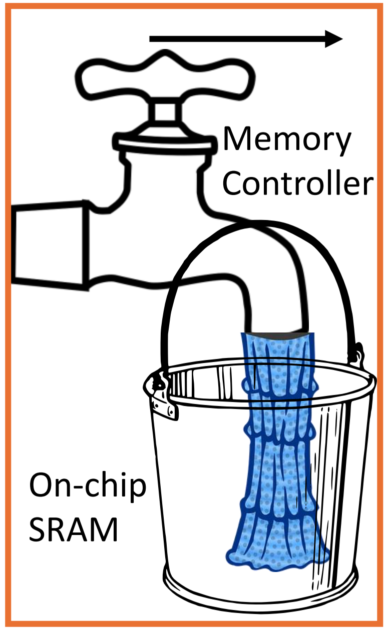


Peak rate
= 2 OPS

Compute

GEMMs

Waterflow-Analogy Model Illustration (2)



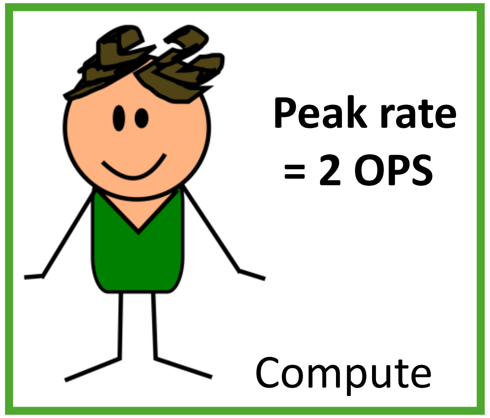
Latency
= 0.1 sec

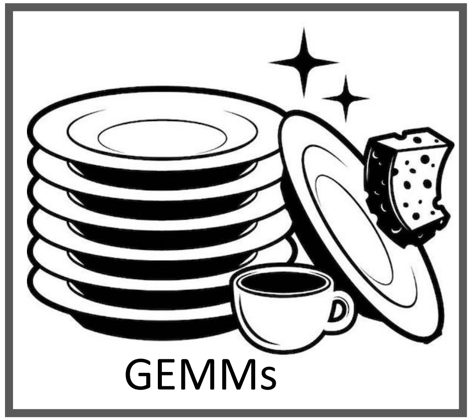
BW
= 1 L/sec

BW
= 4 L/sec

Assuming each dish requires 2L of water,

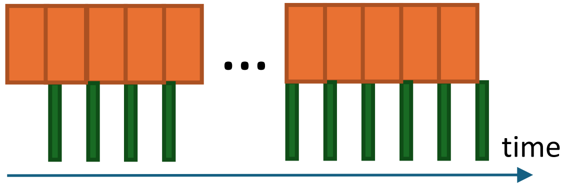
Peak rate
= 2 OPS * 2 L/op
= **4 L/sec**





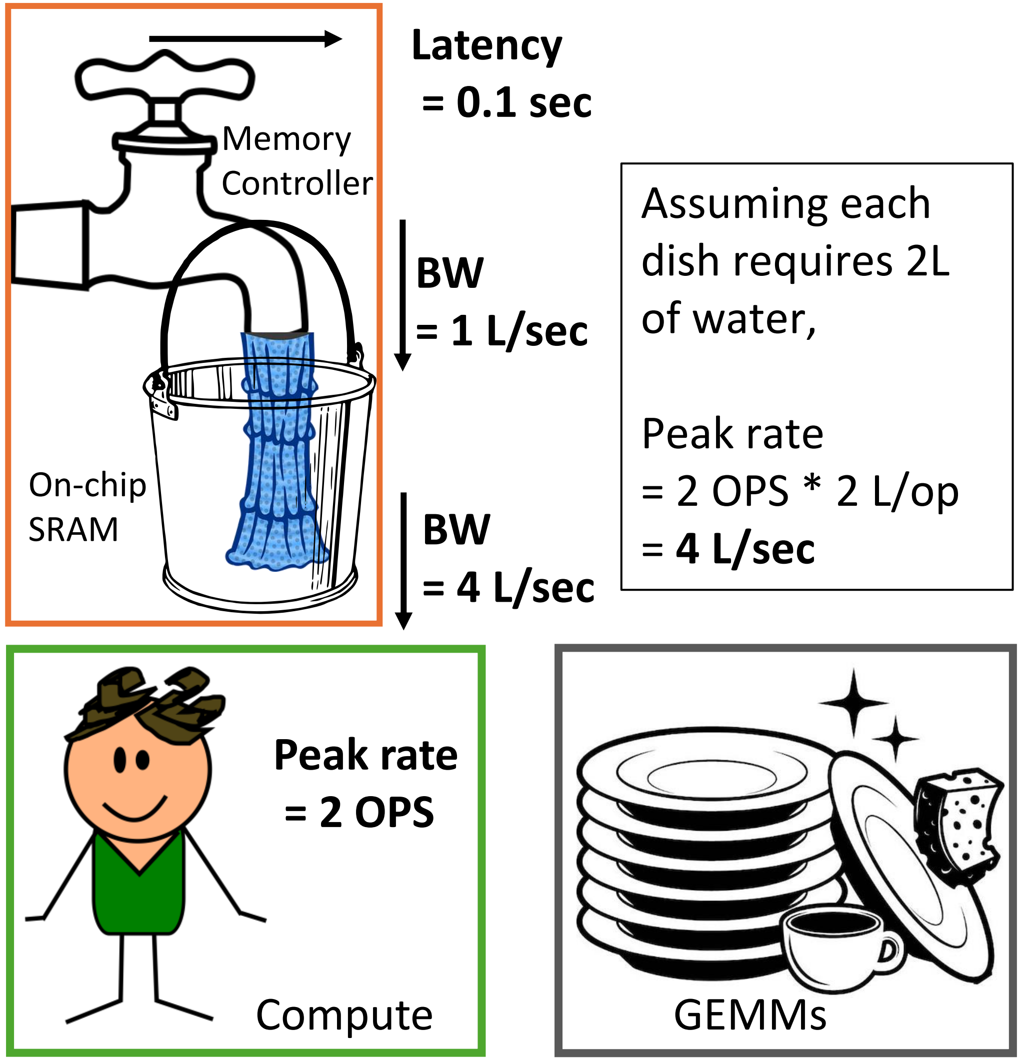
Water Analogy with caches/bucket

Case 2: Reuse factor (F_{reuse}) = 1
OPs = 100
Data (effective) = 200 L
Data (required) = 2 *100 / F_{reuse}
= 200 L
T_{data} = max(T_{tap} , T_{bucket})
= max(200.1, 200/4)
= 200.1 sec
T_{comp} = Data/R_{peak}
= 50 sec
R_{obs} = 100/200.6 OPS = 0.50 OPS
or dropping the tail latency,
R_{obs} = 100/200.1 OPS = 0.50 OPS



Caches are useful only if there are data reuse opportunities.

Impact of Data Reuse on Performance (1)



(a) Water Analogy with caches/bucket

Case 3: Reuse factor (F_{reuse}) = 2

OPs = 100

Data (effective) = 200L

Data (required)

= 2 *100 / F_{reuse}

= **100 L**

T_{data} = max(T_{tap} , T_{bucket})

= max(100.1, 200/4)

= 100.1 sec

T_{comp} = Data/R_{peak}

= 200/4

= 50 sec

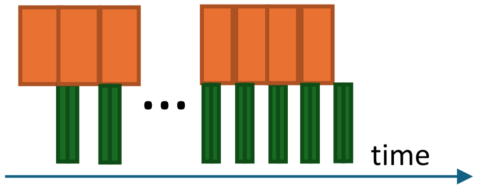
(T_{tap} >> T_{comp})

R_{obs} = 100/100.1 OPS

= 0.99 OPS

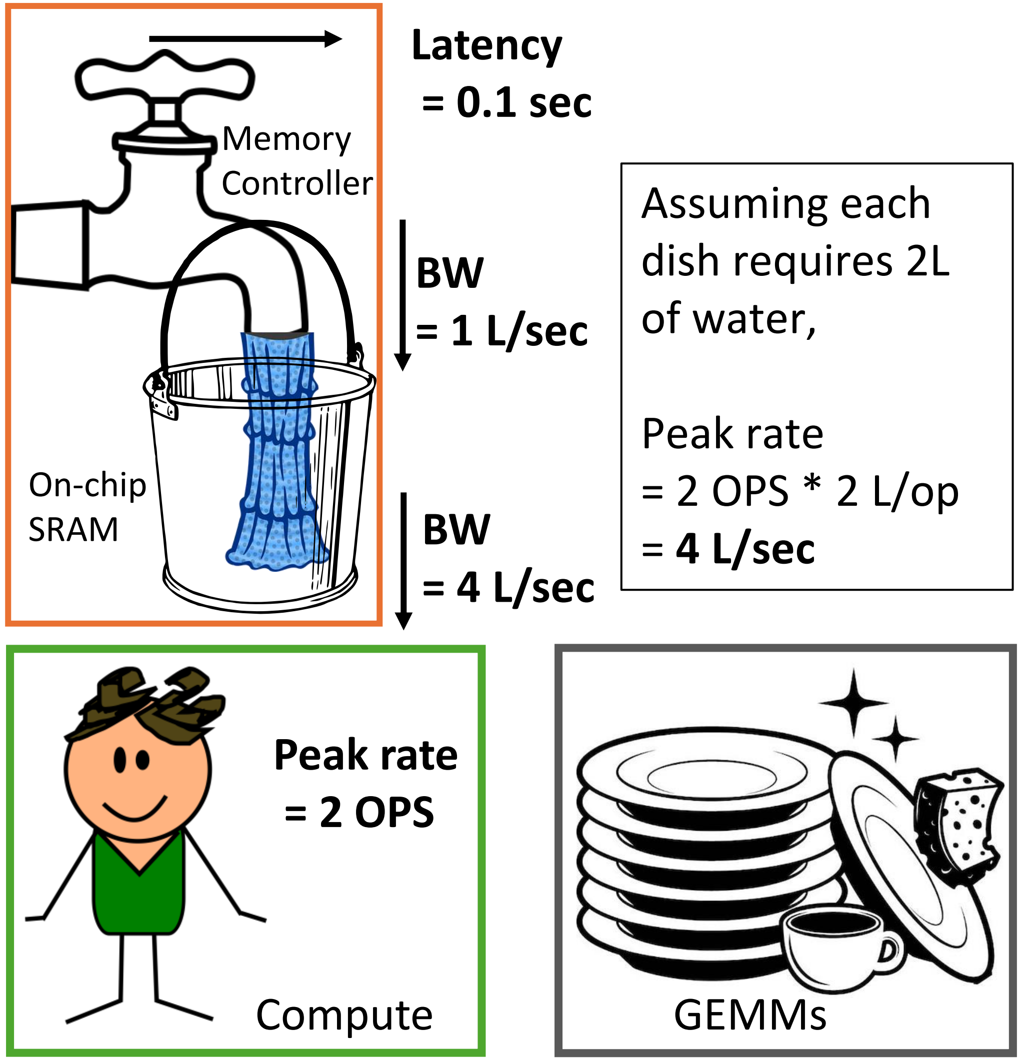
Considering some dishes are not heavily soiled – can share same water for cleaning.

2L effectively acts as 4L for you.



Memory bound

Impact of Data Reuse on Performance (2)



Assuming each dish requires 2L of water,

Peak rate
= 2 OPS * 2 L/op
= **4 L/sec**

(a) Water Analogy with caches/bucket

Case 3: Reuse factor (F_{reuse}) = 4

OPs = 100

Data (effective) = 200L

Data (required)

= 2 *100 / F_{reuse}

= **50 L**

T_{data} = max(T_{tap} , T_{bucket})

= max(50.1, 200/4)

= 50.1 sec

T_{comp} = Data/R_{peak}

= 200/4

= 50 sec

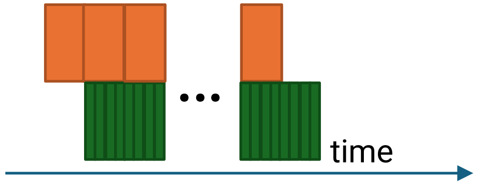
(T_{tap} ~ T_{comp})

R_{obs} = 100/50.1 OPS

~ 2 OPS (R_{peak})

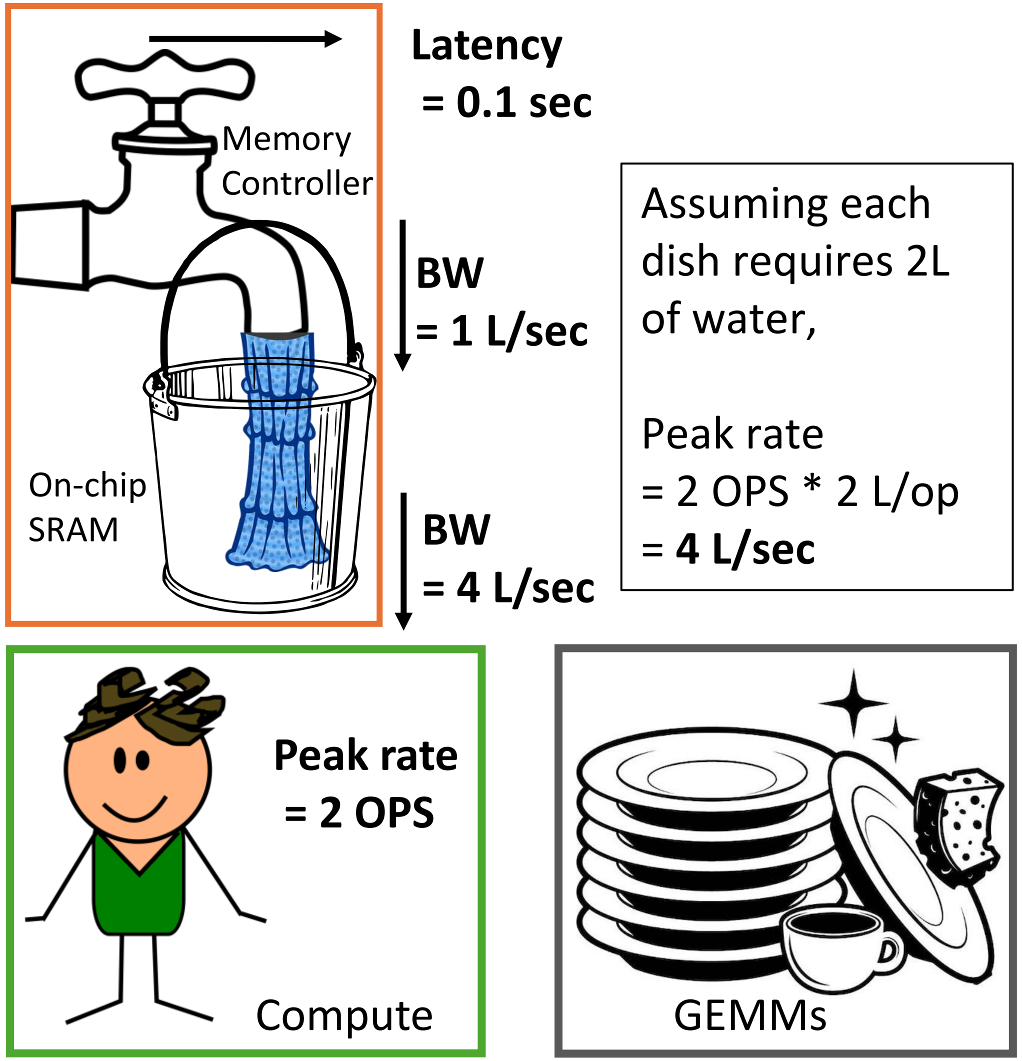
Considering dishes are soiled only slightly – more dishes can share same water for cleaning.

2L effectively acts as 8L for you.



Balanced

Impact of Data Reuse on Performance (3)



(a) Water Analogy with caches/bucket

Case 3: Reuse factor (F_{reuse}) = 8

OPs = 100

Data (effective) = 200L

Data (required)

= 2 *100 / F_{reuse}

= **25 L**

T_{data} = max(T_{tap} , T_{bucket})

= max(25.1, 200/4)

= 50 sec

T_{comp} = Data/R_{peak}

= 200/4

= 50 sec

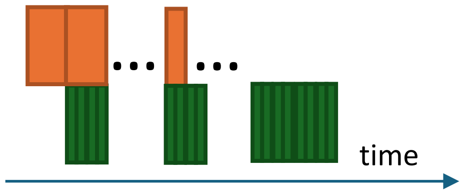
(T_{tap} < T_{comp})

R_{obs} = 100/50 OPS

= 2 OPS (R_{peak})

Say some notorious kid have put already clean dishes in the sink – more water reuse.

2L effectively acts as 16L for you.



Compute bound

Roofline: Memory vs Compute intensive

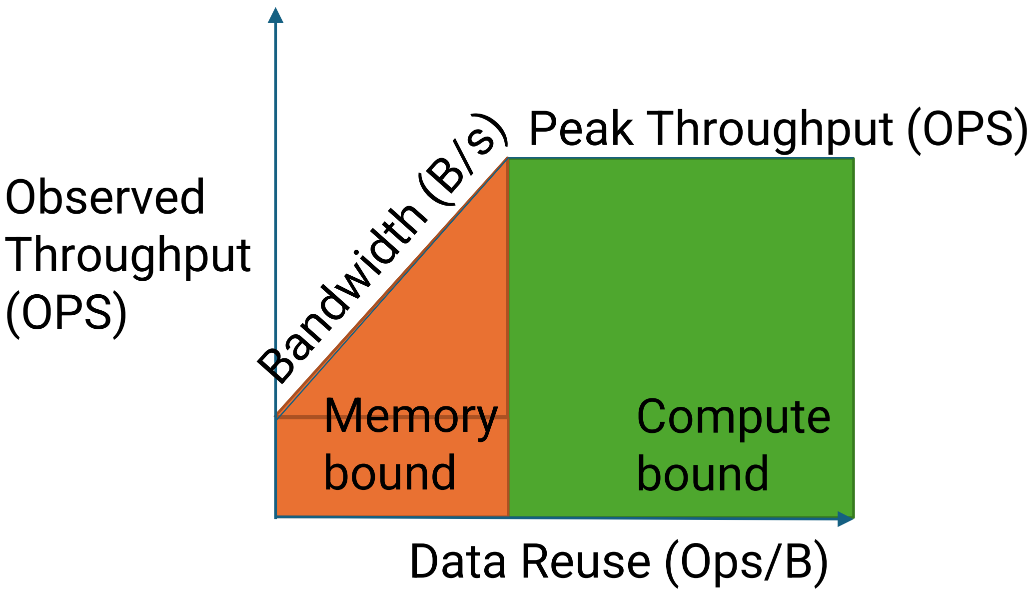
A workload is considered memory intensive if

- Time to fetch data > Time to perform computations
- $\frac{Data\ Accesses\ (B)}{Bandwidth\ (\frac{B}{sec})} > \frac{Total\ number\ of\ computations\ (ops)}{Peak\ Throughput\ (\frac{ops}{sec})}$

Data Reuse affects the nature of workload

- Operations/Byte (Arithmetic Intensity)
- Low reuse -> memory bound region
  - Performance limited by memory bandwidth
- High reuse -> compute bound region
  - Performance limited by peak throughput

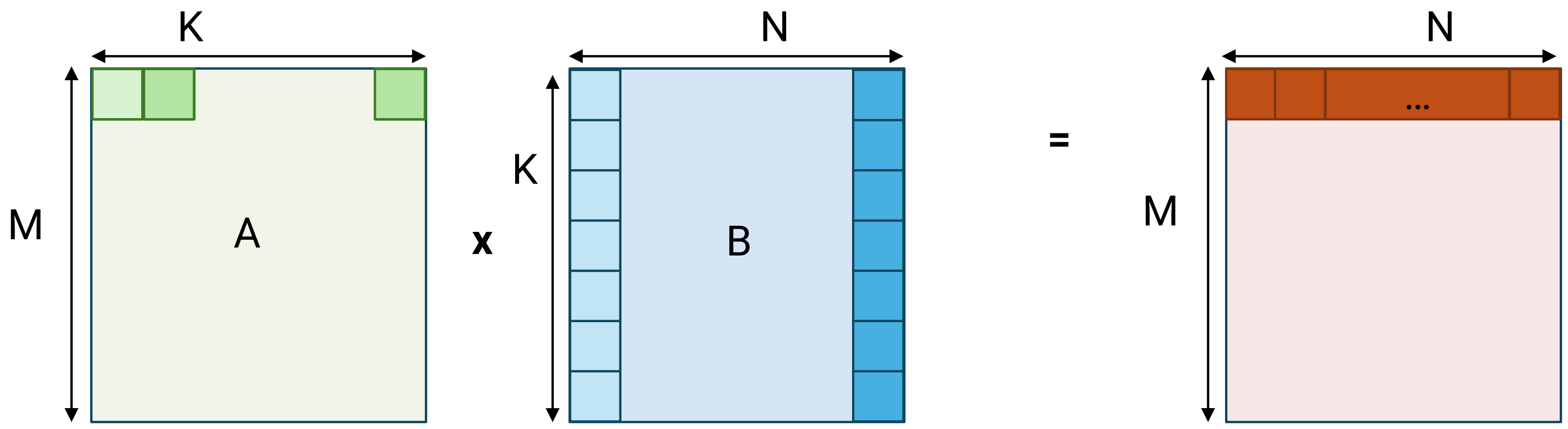
Roofline Performance Model



Arithmetic Intensity of GEMMs (1)

- Theoretical (or algorithmic) AI for GEMMs = $\frac{2MNK}{(MK+KN+MN)}$
- Will the following GEMMs be in compute bound or memory bound region?
  - For a GPU with 130 TFLOPS peak throughput and 1000 GB/s DRAM bandwidth
  - GEMM (1024, 1024, 1024)?

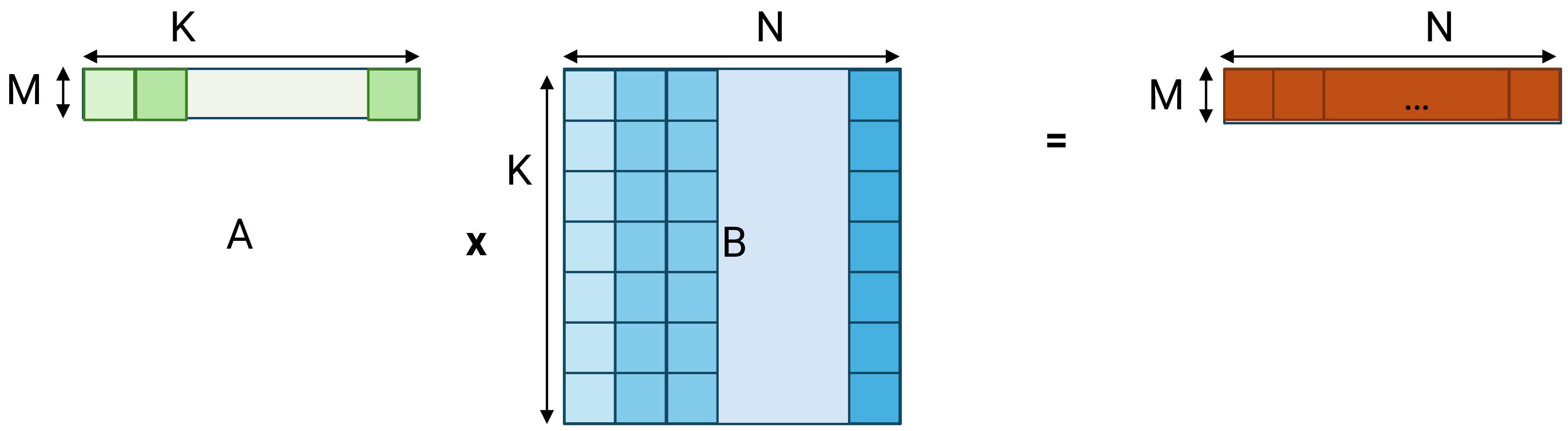
    = 2M/3



Arithmetic Intensity of GEMMs (2)

- Theoretical (or algorithmic) AI for GEMMs = $\frac{2MNK}{(MK+KN+MN)}$
- Will the following GEMMs be in compute bound or memory bound region?
  - For a GPU with 130 TFLOPS peak throughput and 1000 GB/s DRAM bandwidth
  - GEMM (1024, 1024, 1024)
  - GEMM (4, 1024, 1024)?

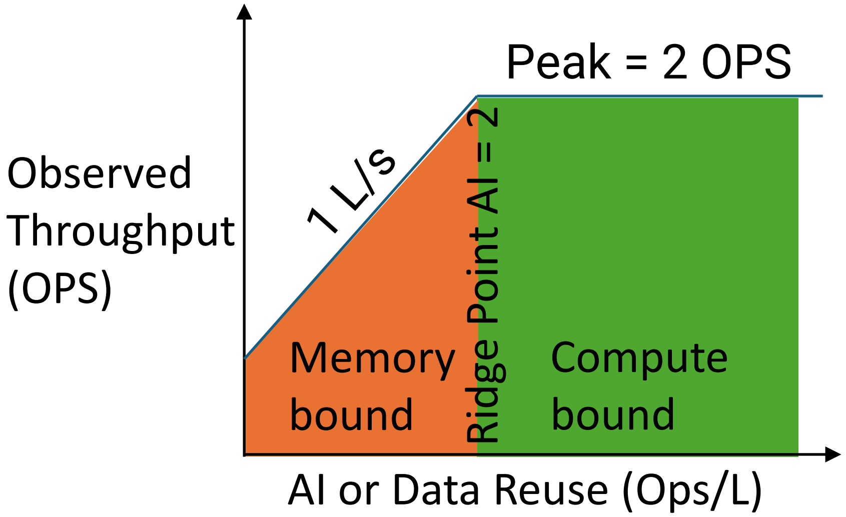
    =~ 2M



Roofline: Memory vs Compute intensive

A workload is considered memory intensive if

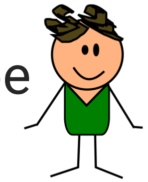
- Time to fetch data > Time to perform computations
- $\frac{Data\ Accesses\ (B)}{Bandwidth\ (\frac{B}{sec})} > \frac{Total\ number\ of\ computations\ (ops)}{Peak\ Throughput\ (\frac{ops}{sec})}$



AI or Data reuse = (1 Op/2 L)*F_{reuse}
Ridge Point AI = 2 OPS/1 L/s = 2
Ridge Point F_{reuse} = 2/(1/2) = 4

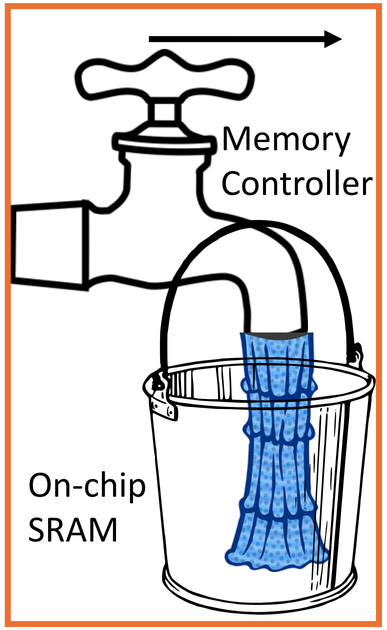
Factors affecting workload nature:

- Data Reuse or Arithmetic Intensity (AI)
  - Water shared by multiple dishes/how much they are soiled (GEMM shape)
- Hardware Specifications
  - physical limitations on how many dishes can be washed per second (peak throughput)
  - Limited tap water bandwidth (main memory bandwidth)



Is it always necessary for a workload to be either in the memory bandwidth- or compute- bound region?

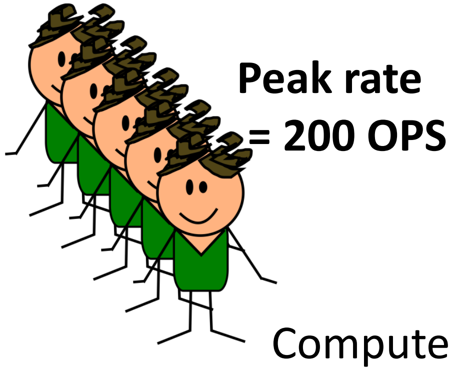
Examples of Memory-bound Scenarios



Latency
= 1 sec

BW
= 10 L/sec

BW
= 400 L/sec



Assuming each dish requires 2L of water,

Peak rate
= 200 OPS * 2 L/op
= **400 L/sec**

Case 6: Reuse factor (F_{reuse}) = 1
OPs = 1000
Data (effective) = 2000 L
Data (required)
= 2 *1000 / F_{reuse}
= 2000 L
T_{data} = max(T_{tap} , T_{bucket})
= max(1+2000/10, 2000/400)
= 201 sec
T_{comp} = Data/R_{peak}
= 2000/400
= 5 sec
(T_{data} >> T_{comp})
R_{obs} = 1000/201 OPS
= 4.98 OPS << 200 OPS

(a) Memory-**bandwidth** bound

Case 7: Reuse factor (F_{reuse}) = 4
OPs = 4
Data (effective) = 8 L
Data (required)
= 2 *4 / F_{reuse}
= 2 L
T_{data} = max(T_{tap} , T_{bucket})
= max(1+2/10, 8/400)
= 1.2 sec
T_{comp} = Data/R_{peak}
= 8/400
= 0.02 sec
(T_{data} >> T_{comp})
R_{obs} = 4/1.2 OPS
= 3.33 OPS << 200 OPS

(b) Memory-**Latency** bound

Conclusion

- Proposed a simple waterflow-analogy for showcasing the impact of data reuse on AI hardware performance.
- Connected the concepts to the roofline performance model with worked-out examples.
- Used in the first offering of AI Hardware course at Purdue – where students were from different backgrounds (from pure algorithm to circuit designers).
- The given slides can be used prior to introducing dataflow or scheduling strategies, lowering the barrier to AI hardware concepts.

Q&A

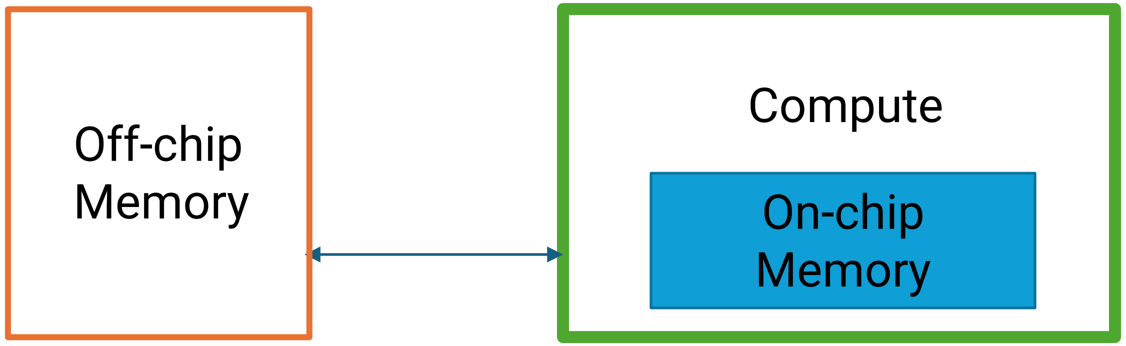
Thank you. Any feedback?

PURDUE UNIVERSITY | Elmore Family School of Electrical and Computer Engineering

APPENDIX

Recap: Evaluating DNNs

- Simple memory-compute model
- Arithmetic Intensity (data reuse)
- Roofline performance model
  - Memory bound region
  - Compute bound region

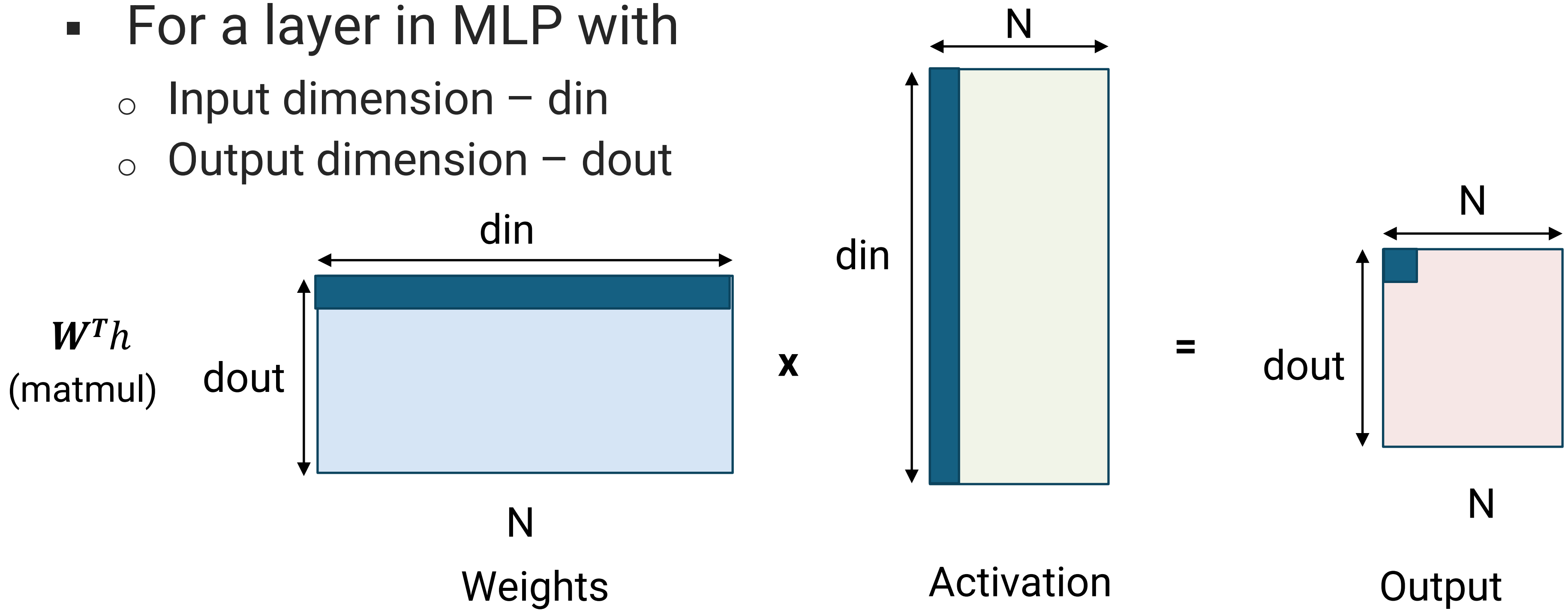


Question:

For a GPU with 130 TFLOPS peak throughput and 1000 GB/s DRAM bandwidth, what will be the minimum value of arithmetic intensity for it to be compute bound? (transition AI)

Recap: Matrix Multiplications in DNN Models (1)

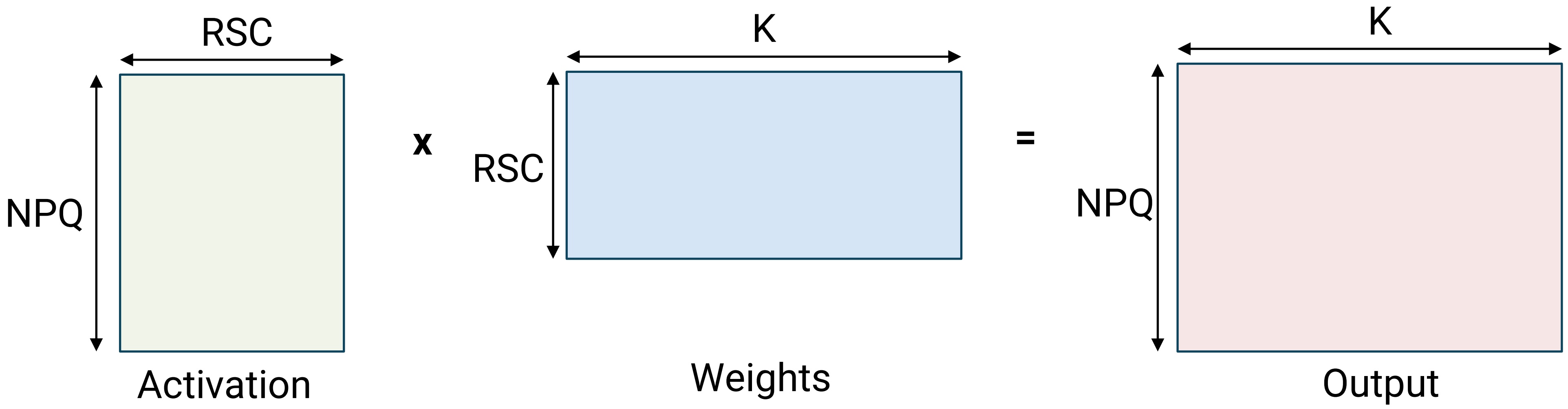
- For a layer in MLP with
  - Input dimension – din
  - Output dimension – dout



  - Pytorch: nn.Linear

Recap: Matrix Multiplications in DNN Models (2)

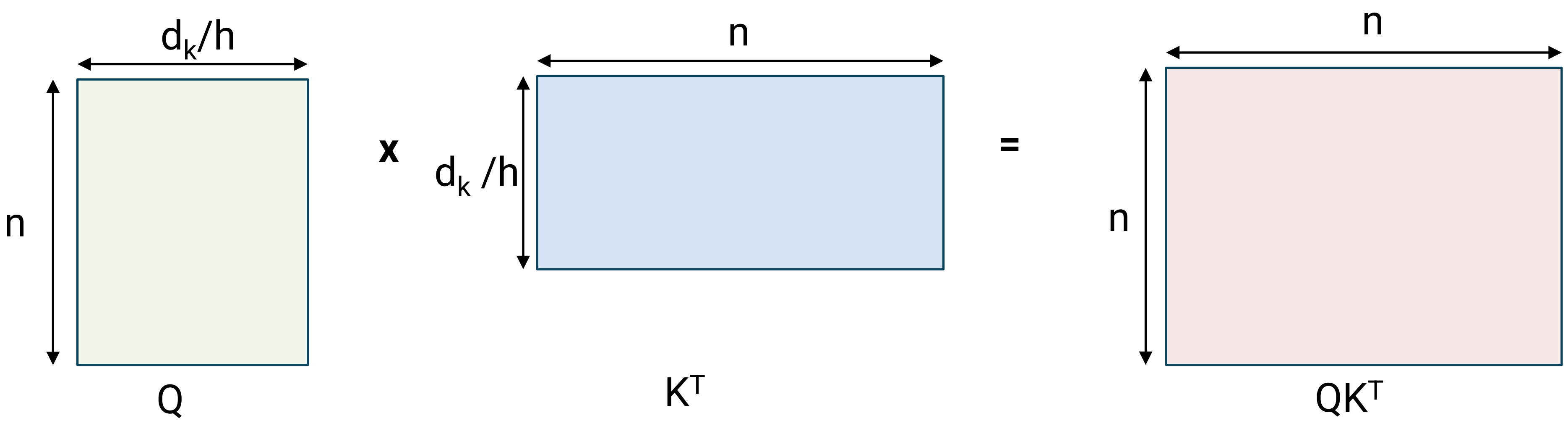
- For a convolution layer with
  - Filter size – R x S x C
  - Input fmap size – H x W x C
  - Output fmap size – P x Q x K
  - Batch size – N



  - Implicit GEMM algorithm using im2col transformation
    - (GEMM – general matrix multiplication)

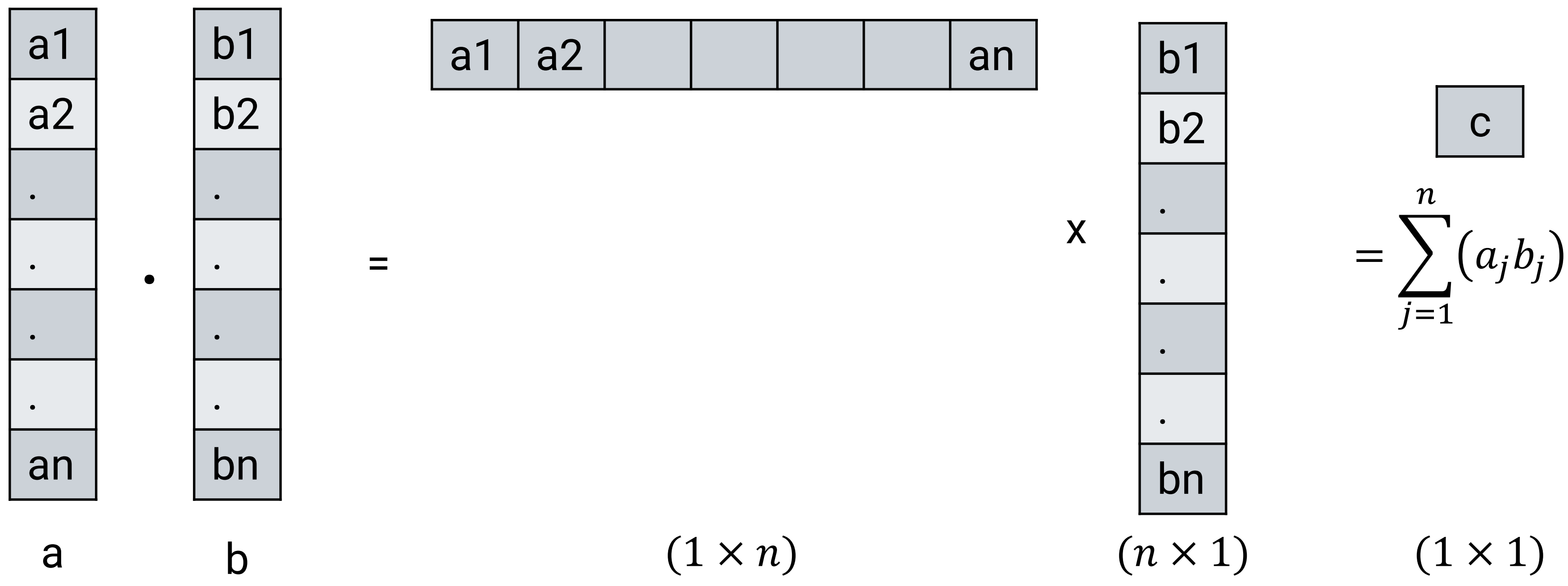
Recap: Matrix Multiplications in DNN Models (3)

- For an attention layer with
  - sequence length – n
  - embedding dimension – $d_{k\ (=v=q)}$
  - heads = h
  - Static matrix multiplications – Q, K, V, Output
  - Dynamic matrix multiplications – QK^T, QK^TV
  - e.g.



Basic Computations in AI Workloads

- Dot Product (**a** . **b** = **a**^{T}**b** = c)
  - Product is a scalar value
  - Vectors typically treated as column vectors
  - Used to measure similarity, also referred as inner product



$$= \sum_{j=1}^{n} (a_j b_j)$$

Number of operations?

MAC (multiply and accumulate)

Basic Computations in AI Workloads

- Matrix Vector Multiplication (**A**^{T}**b** = **c**)
  - Product is a vector
  - Each element in output vector is dot product of row of A^T and the vector b

a_{11}	a_{12}	.	a_{1m}
a_{21}	a_{22}	.	.
.	.	.	.
.		.	.
.			
.			
a_{n1}	a_{n2}	.	a_{nm}

$A\ (n \times m)$

b1
b2
.
.
.
.
bn

$b\ (n \times 1)$

Basic Computations in AI Workloads

- Matrix Vector Multiplication (**A^Tb** = **c**)
  - Product is a vector
  - Each element in output vector is dot product of row of A and the vector b

a_{11}	a_{21}	.	.	.	.	a_{n1}
a_{12}	a_{22}	.	.	.	.	a_{n2}
.	.	.	.	.	.	.
a_{1m}	a_{2m}	.	.	.	.	a_{nm}

$A^T(m \times n)$

x

b1
b2
.
.
.
.
bn

$(n \times 1)$

=

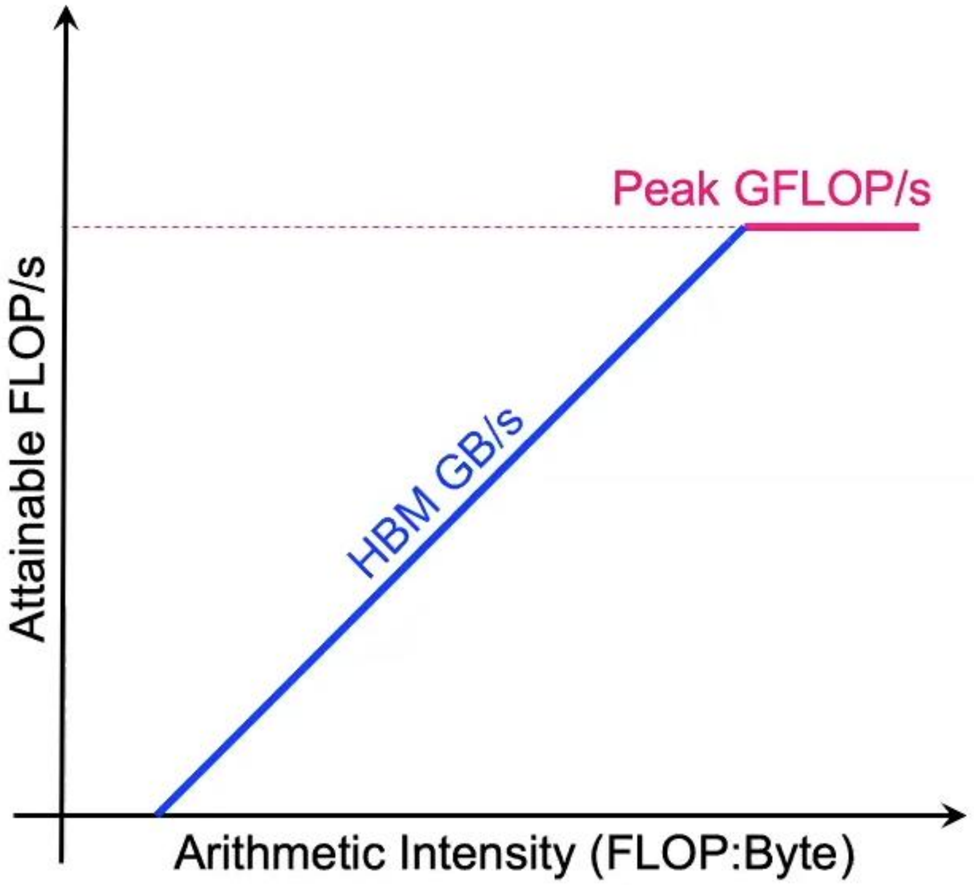
c1
c2
.
.
cm

$(m \times 1)$

$$c_i = \sum_{j=1}^{n} (a_{ij} b_j)$$

Recap: Roofline Performance Model

- GFLOPS = GFLOP/s
  - Giga Flop Operations per second
  - Throughput
- Arithmetic Intensity (AI) = Reuse
  - $\frac{Total\ number\ of\ operations\ (ops)}{Data\ Accesses\ (B)}$
  - $\frac{Total\ FLOPs}{Data\ data\ movement\ (B)}$
  - FLOPs/Byte
- Transition @ AI
  - Time to fetch data = Time to compute



Transition @ AI = Peak GFLOPs/Peak GB/s