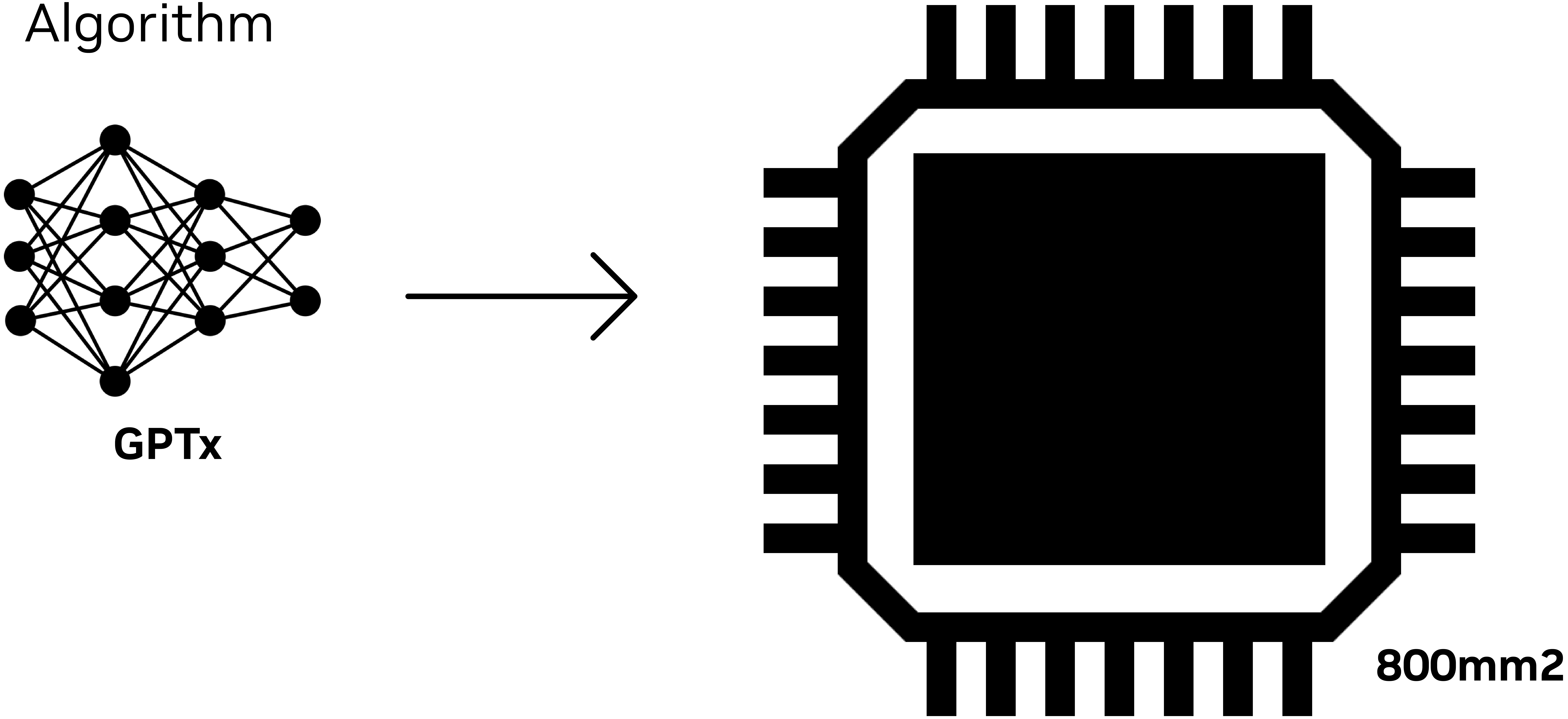
Mind the Gap: Attainable Data Movement and Operational Intensity Bounds for Tensor Algorithms

Qijing Huang, Po-An Tsai, Joel Emer, Angshuman Parashar

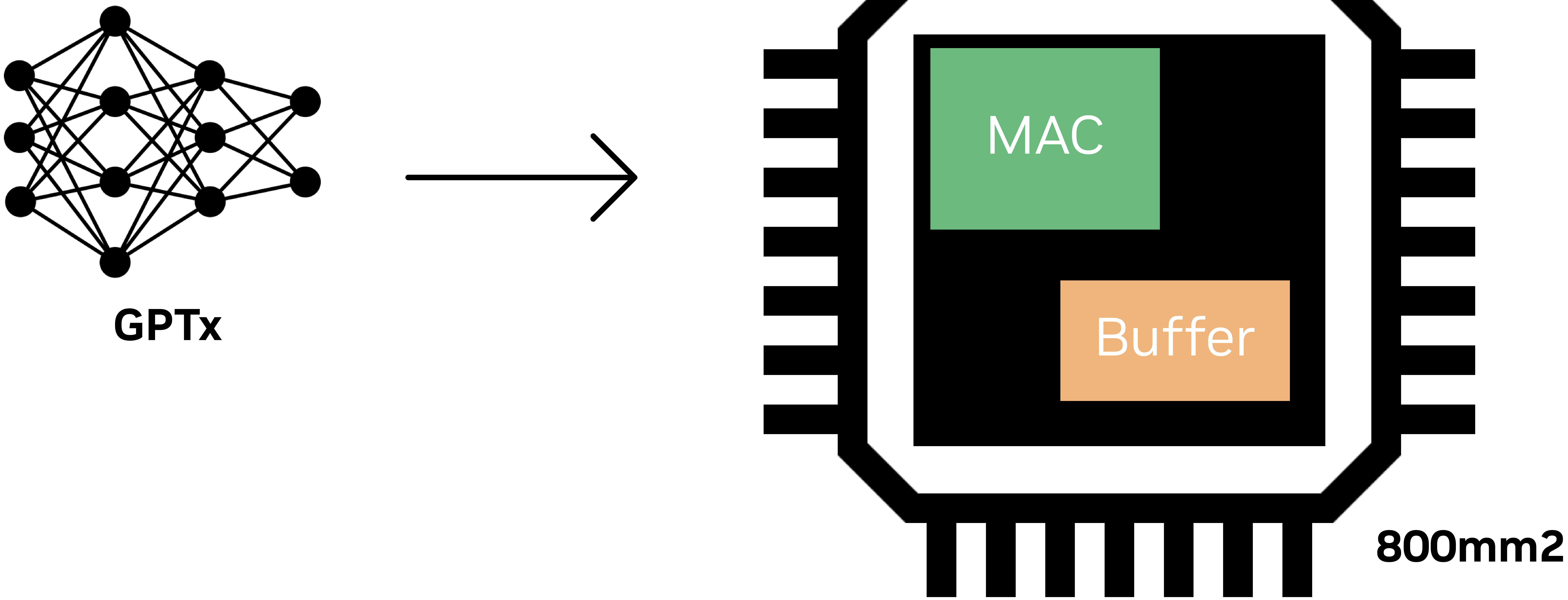
NVIDIA, MIT CSAIL

Motivation: A design challenge

Algorithm



Motivation: A design challenge



Algorithm

GPTx

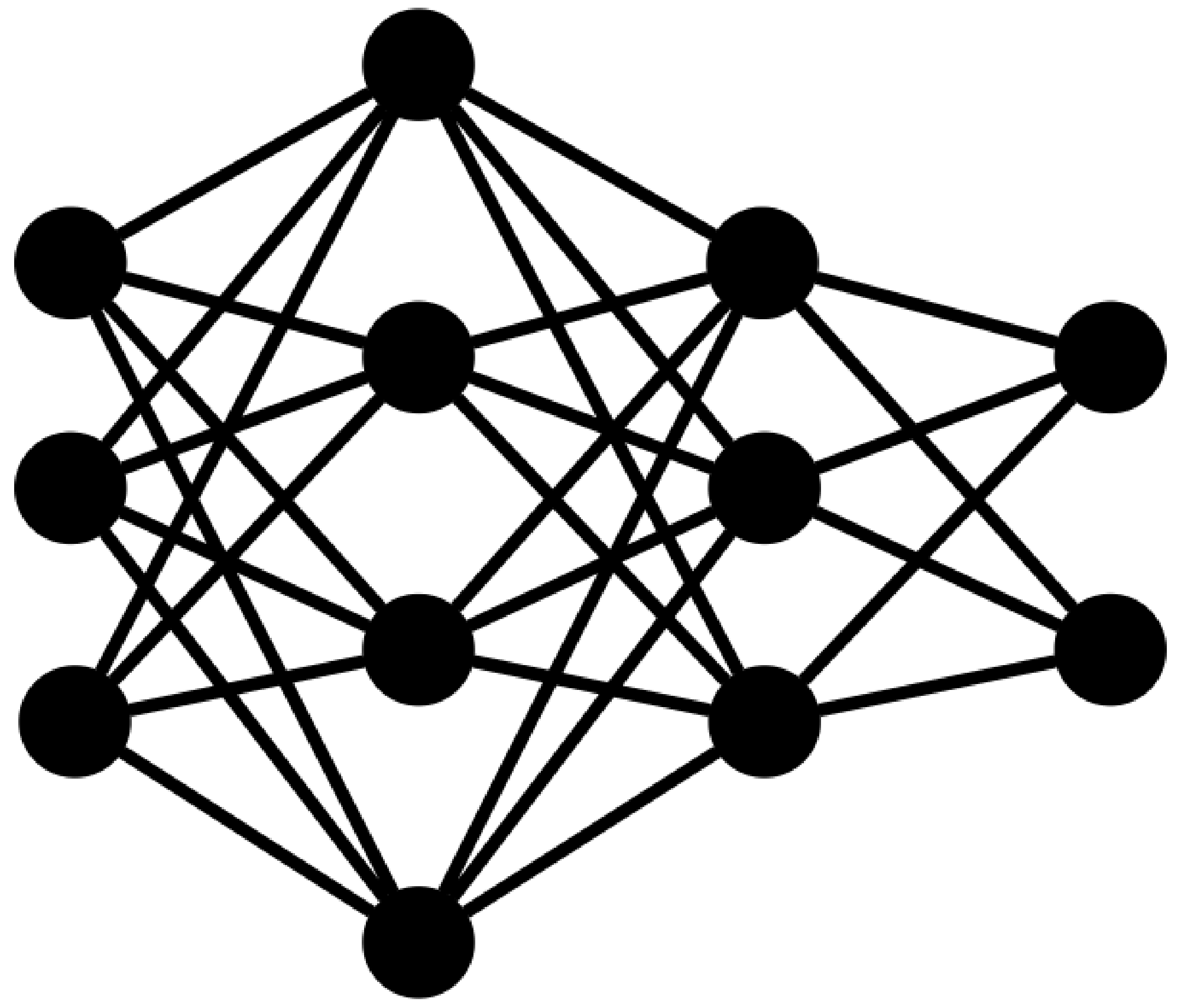
MAC

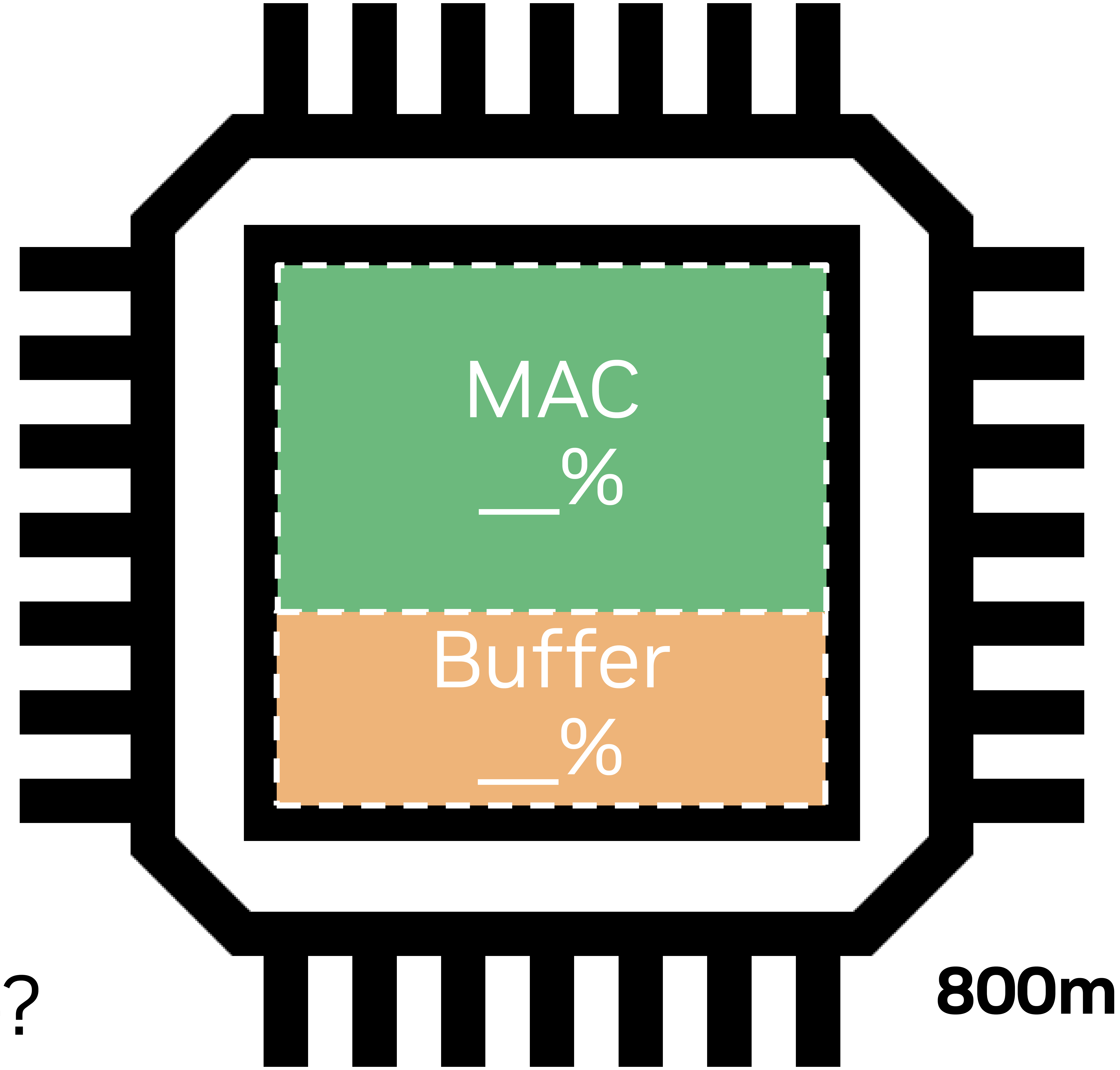
Buffer

800mm2

Motivation: A design challenge

Algorithm

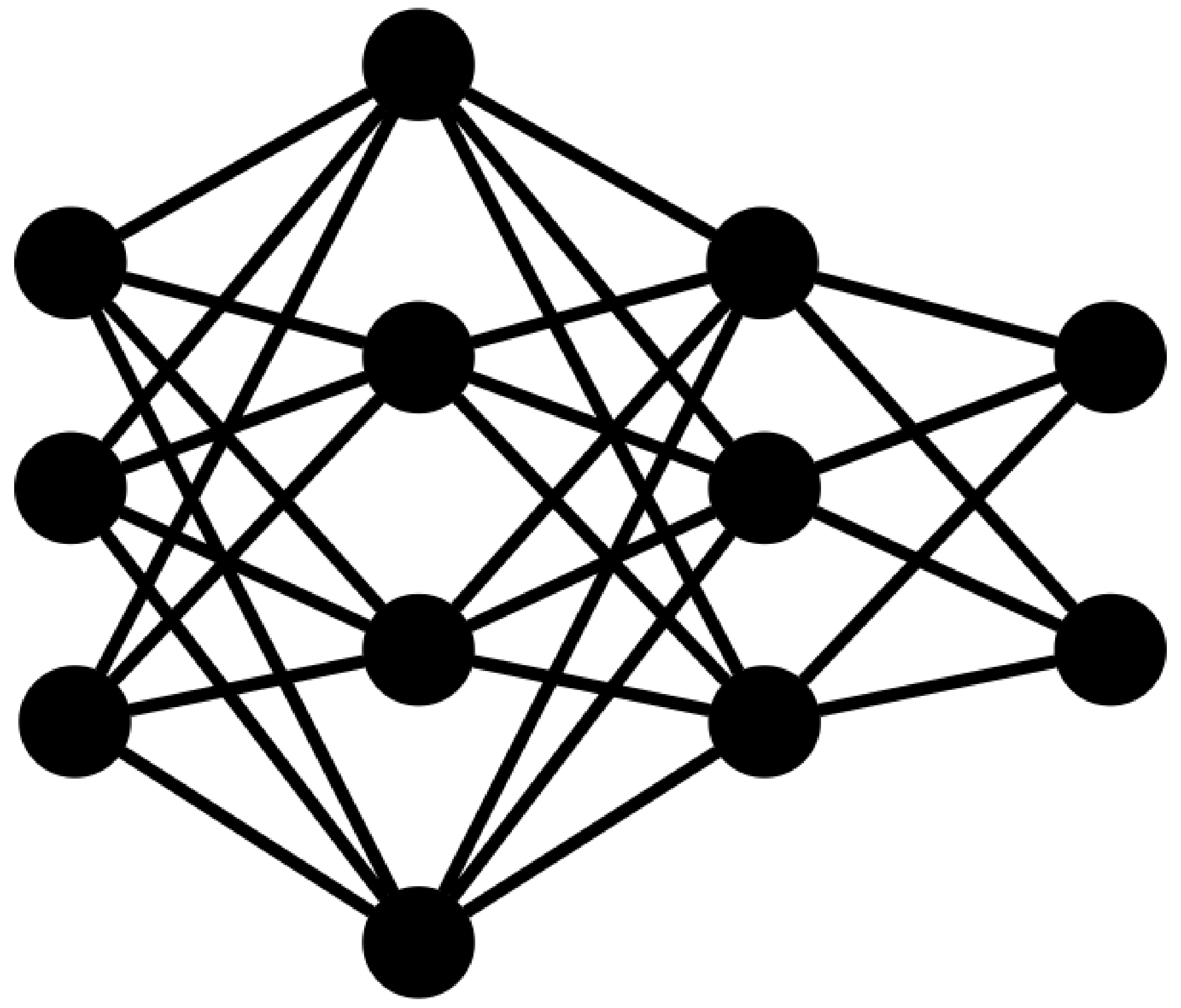




How to provision chip area between storage and compute?

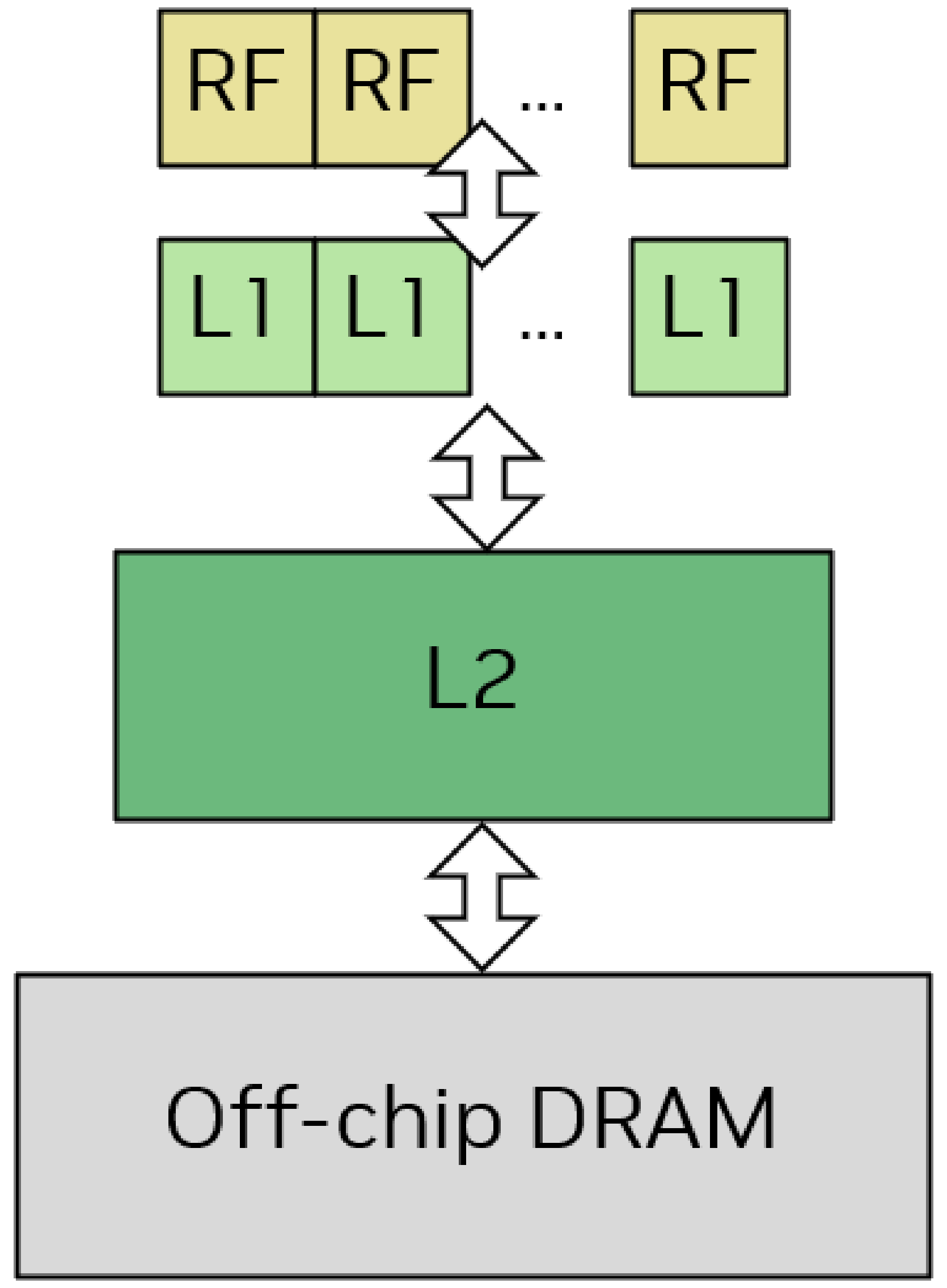
Approach 1: Design space exploration

Algorithm

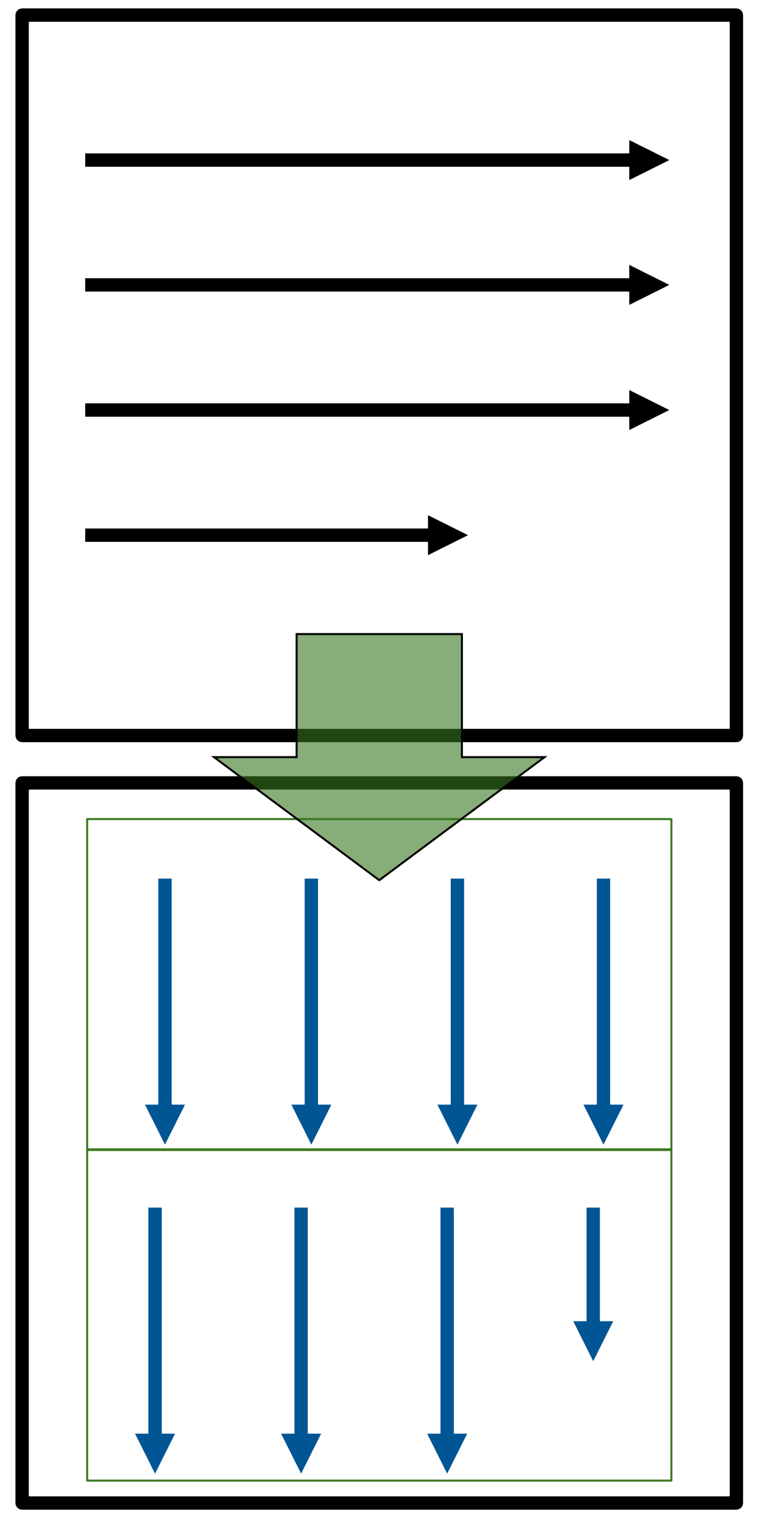


GPTx

Architecture Design Space



Mapping Space



Evaluation Time

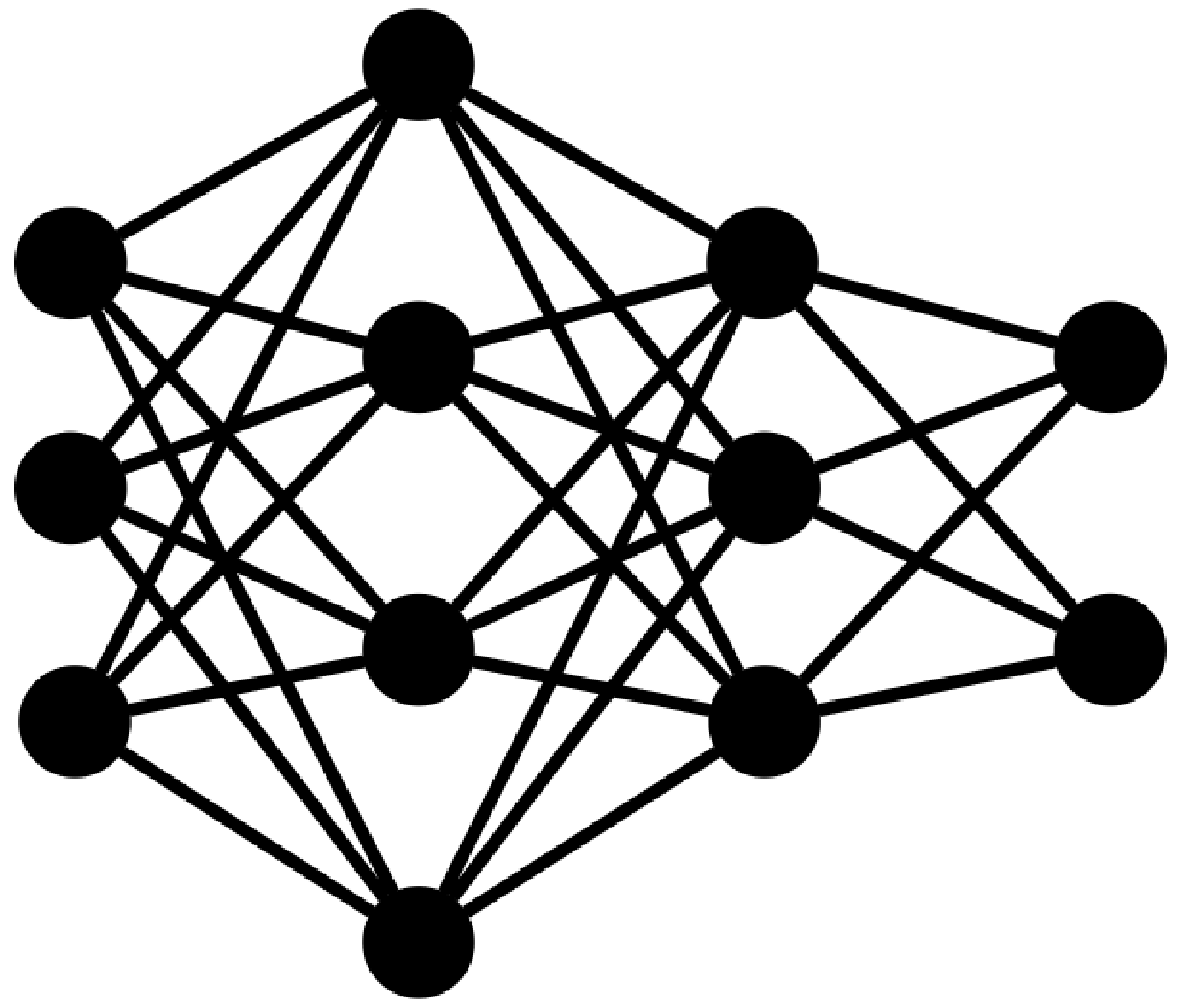
Tool	Eval Time
Timeloop	0.01s
FPGA	2 mins
VCS	10 mins
Power	6 hrs

$\sim 10^{17}$ × **$\sim 10^{5}$** × **0.01s**

= 31T years

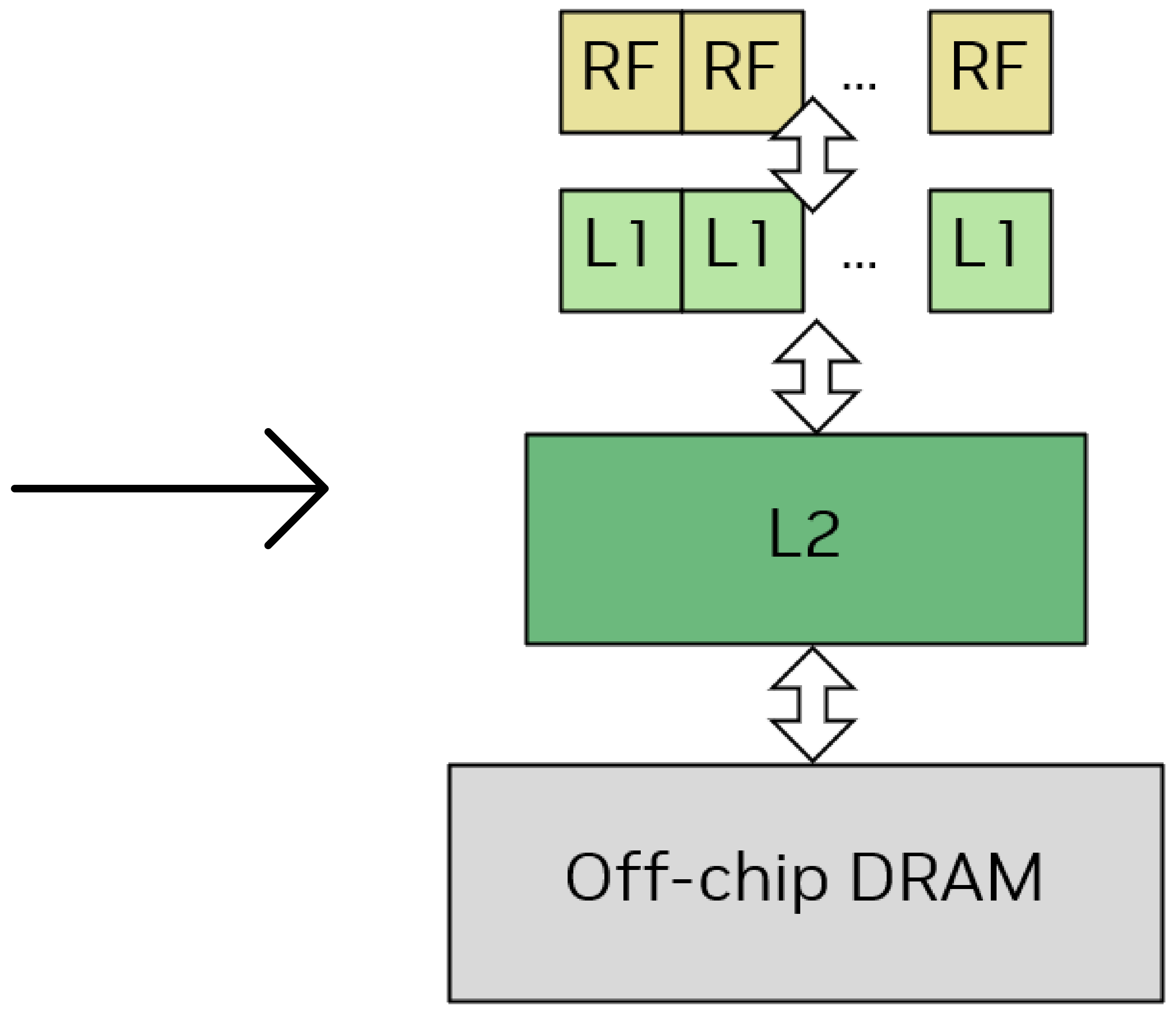
Approach 1: Design space exploration

Algorithm

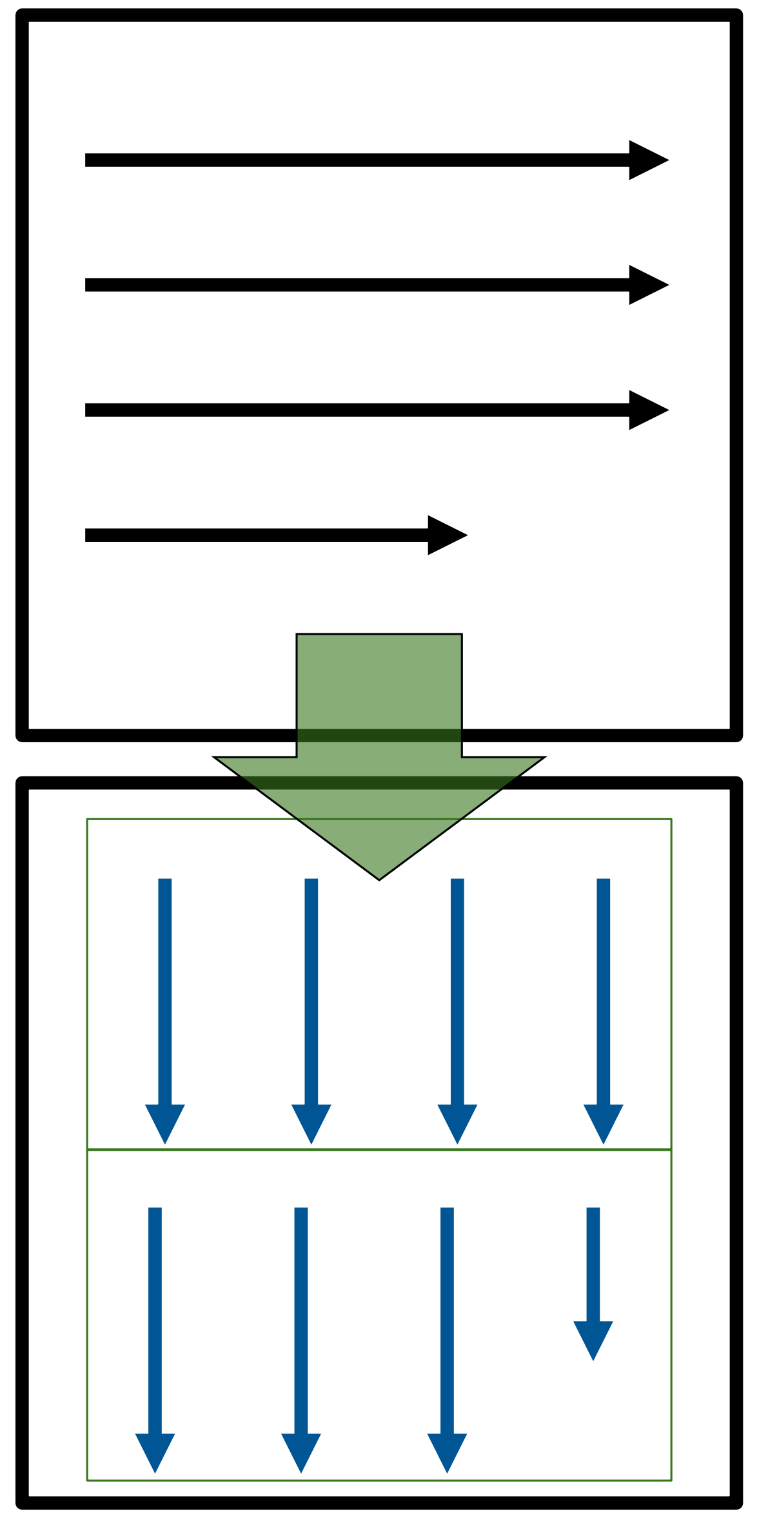


GPTx

Architecture Design Space



Mapping Space



Evaluation Time

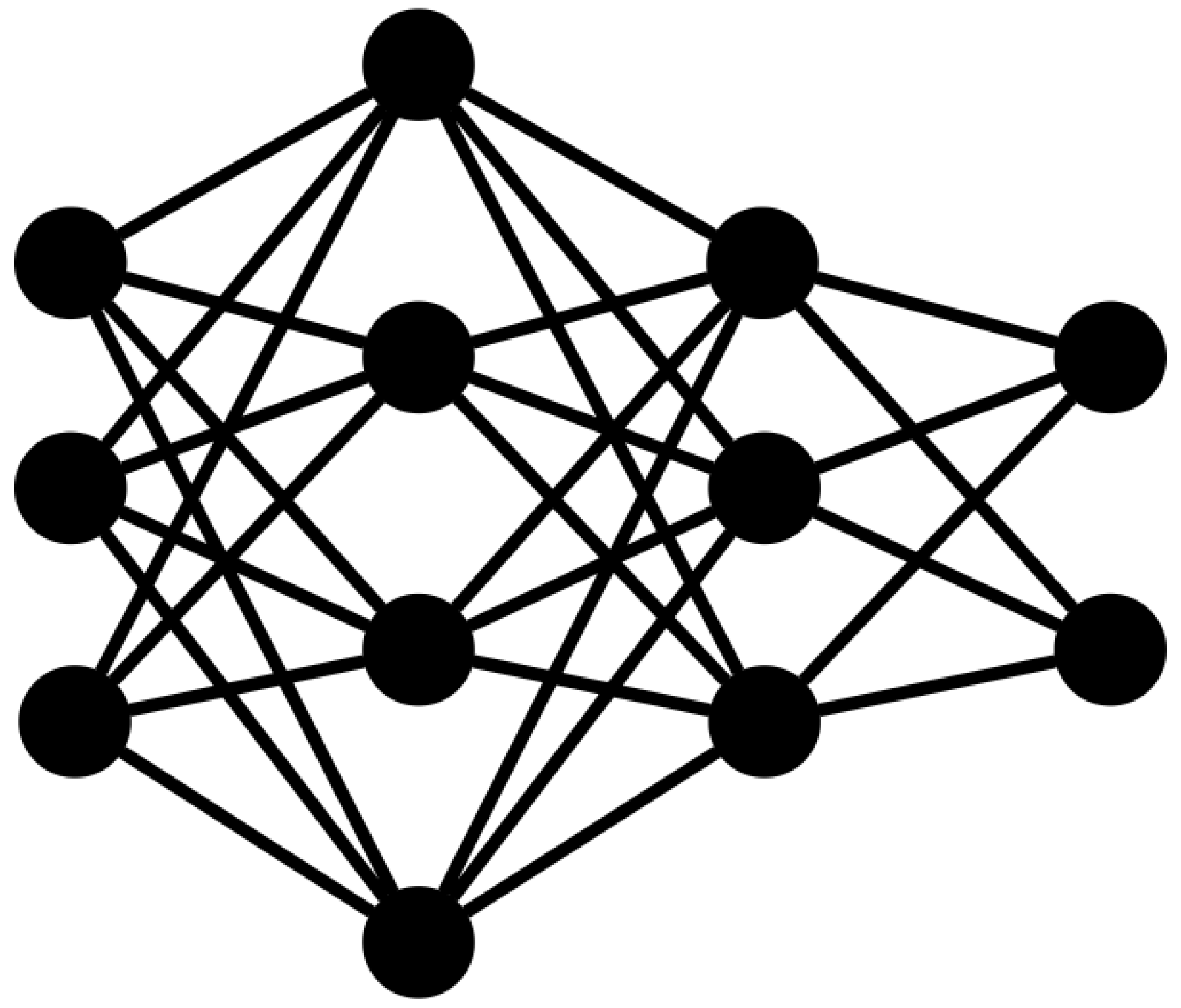
Tool	Eval Time
Timeloop	0.01s
FPGA	2 mins
VCS	10 mins
Power	6 hrs

- Time-consuming and costly
- No optimality guarantee
- Lack of design insight

Approach 2: Roofline model analysis

“Speeds and feeds”

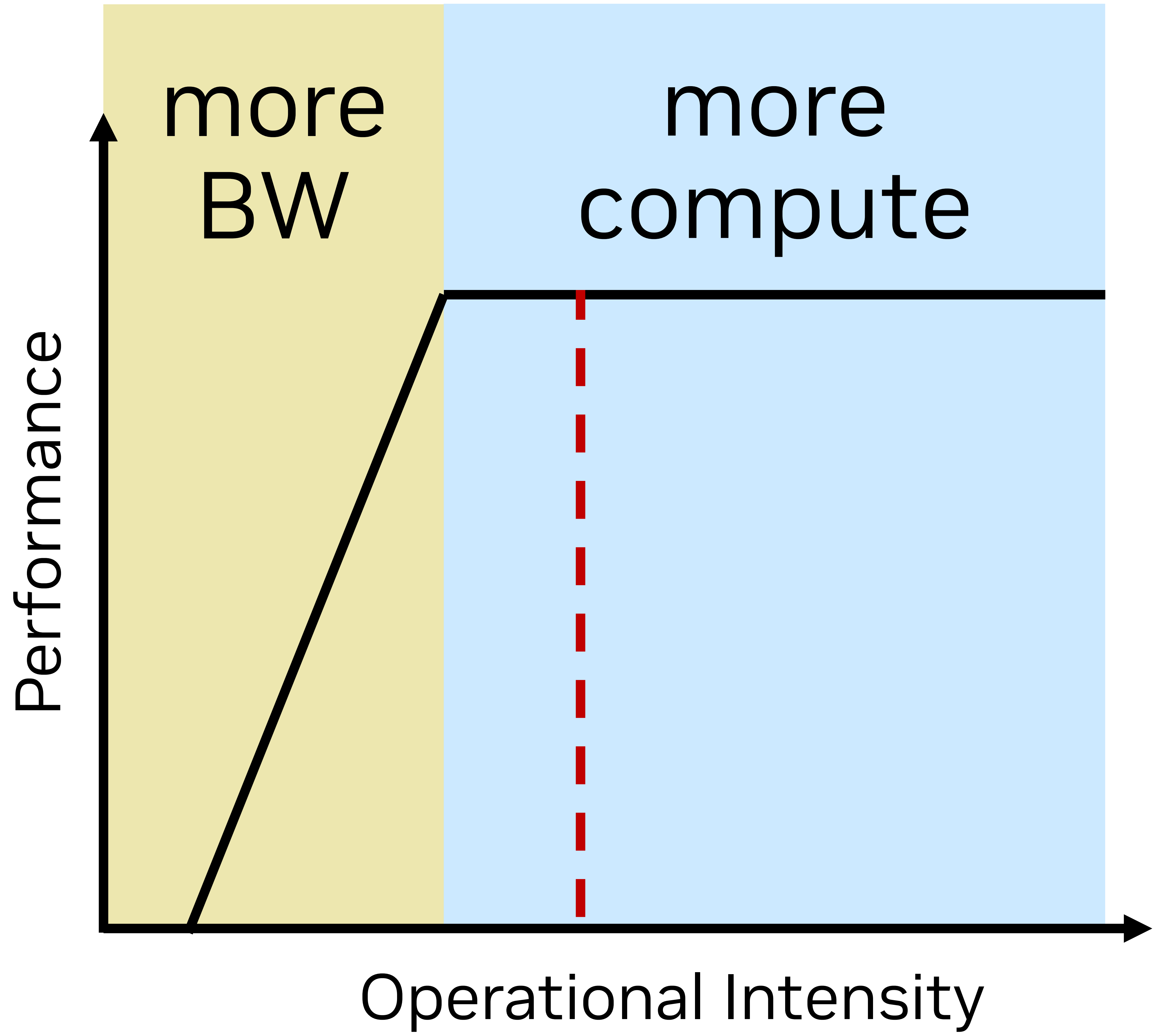
Algorithm



GPTx

Algo max operational intensity (OI)

$$= \frac{\text{total compute}}{\text{algo min accesses}}$$

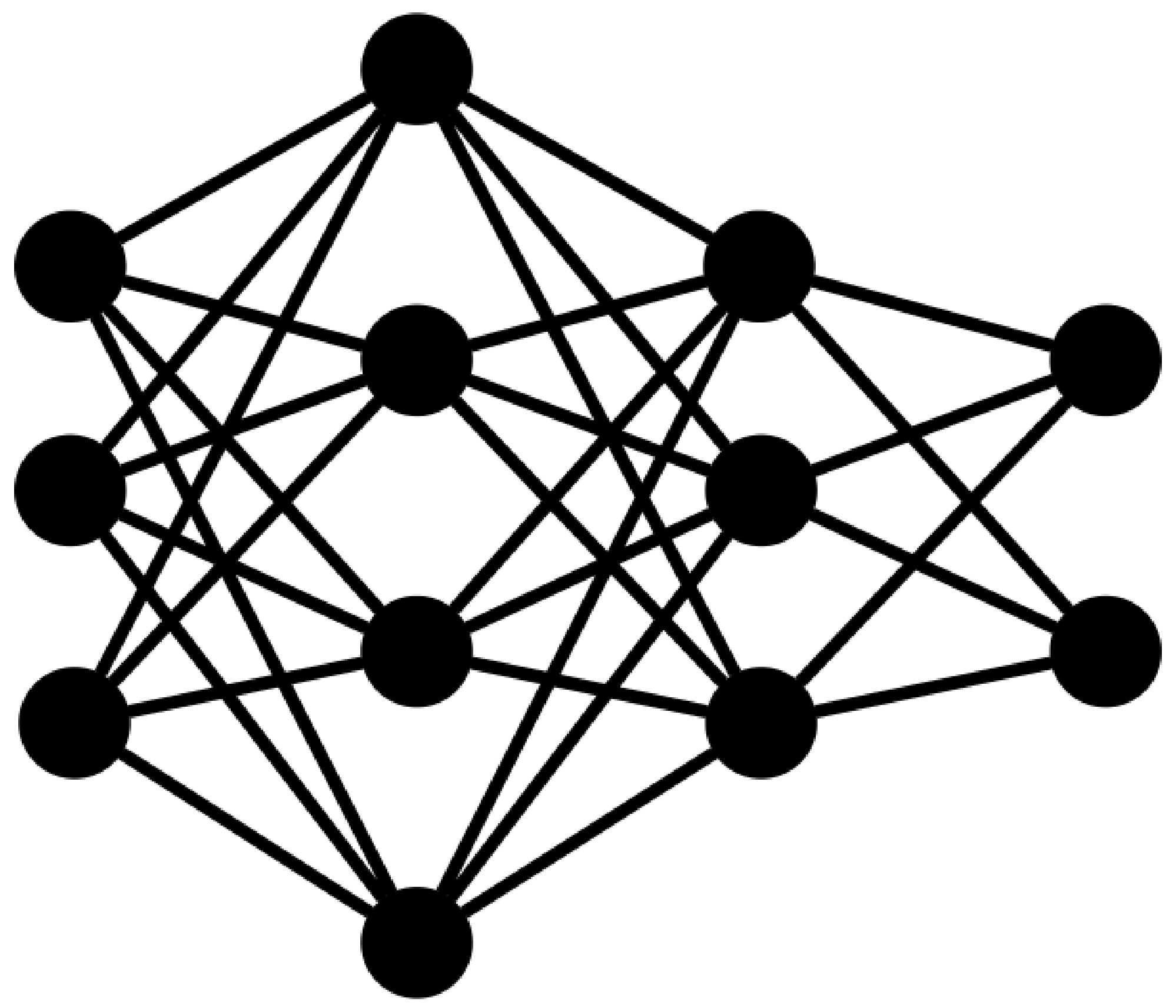


Roofline Model

Approach 2: Roofline model analysis

"Speeds and feeds"

Algorithm



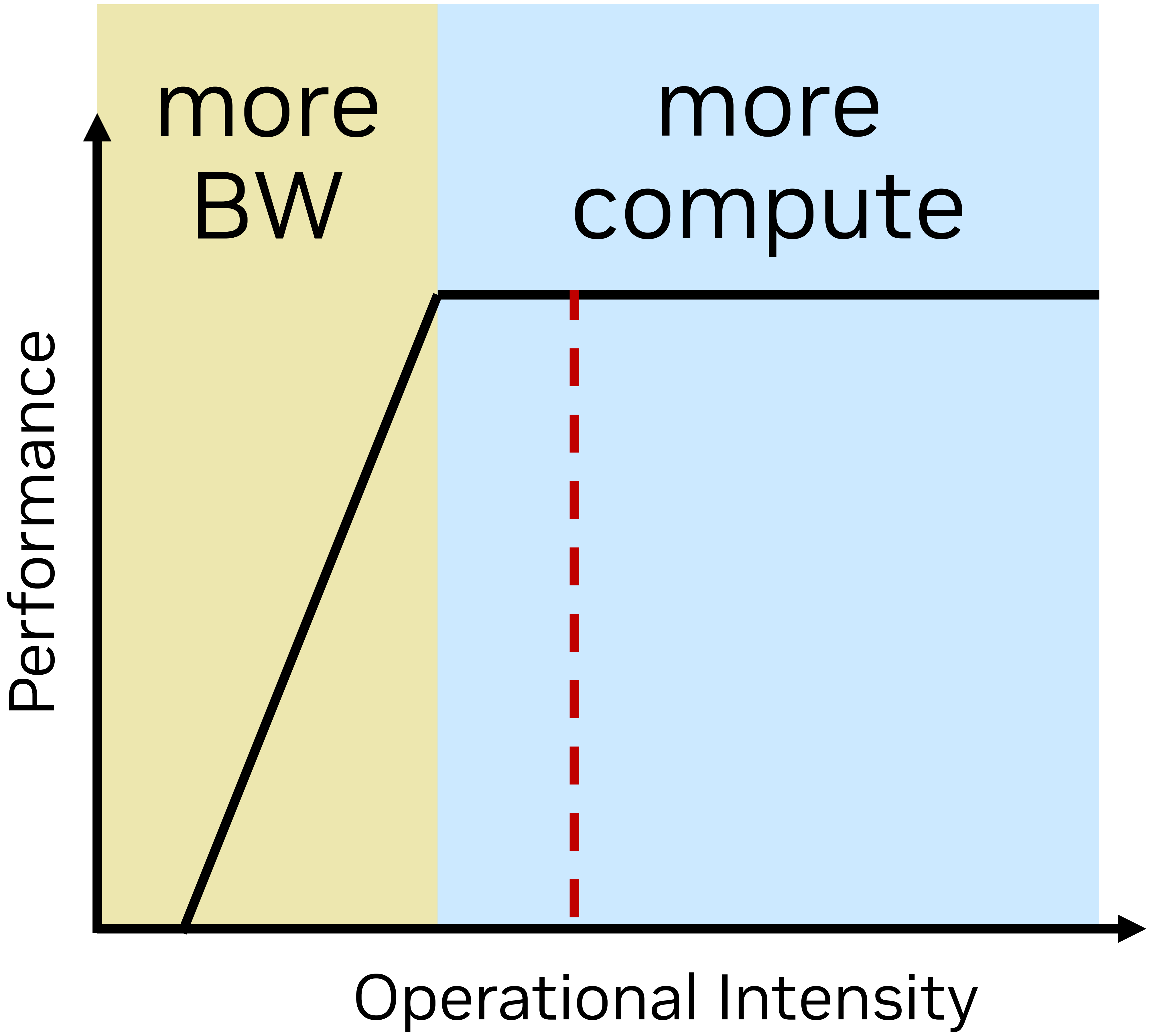
GPTx

→

Algo max operational intensity (OI)

$$= \frac{\text{total compute}}{\text{algo min accesses}}$$

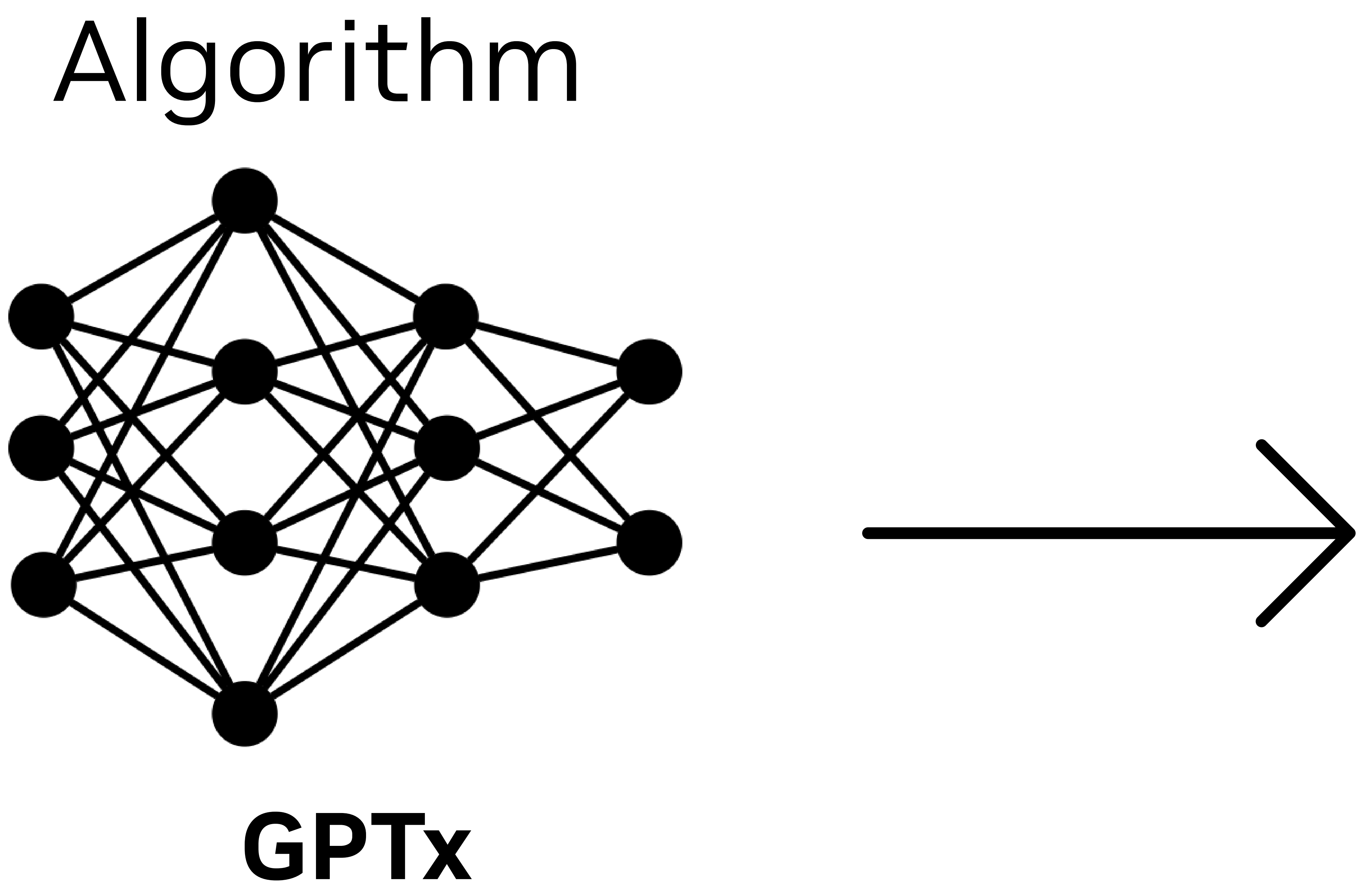
→

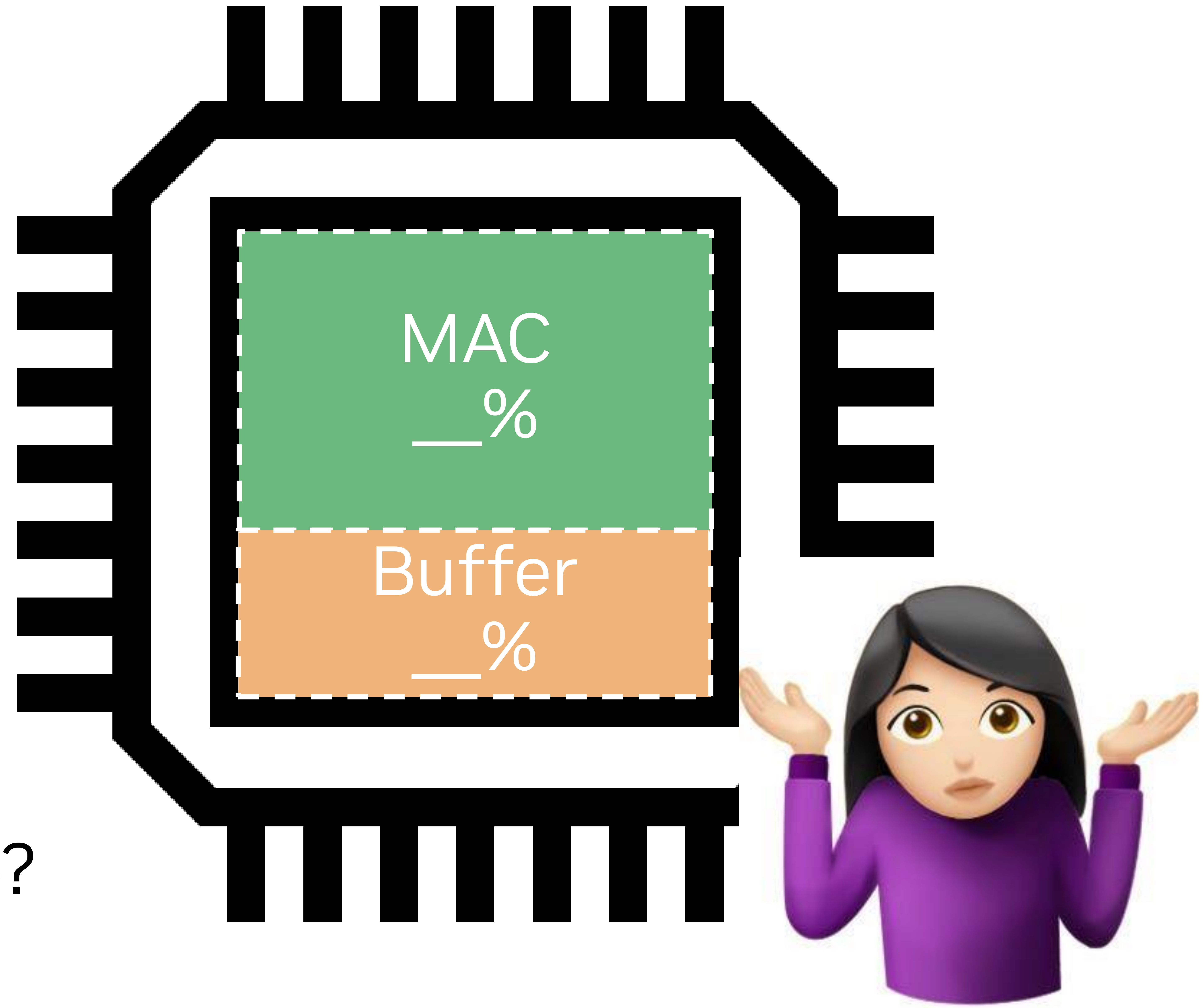


Roofline Model

- No buffer storage tradeoffs are present in the analysis

Motivation: Lack of design tools

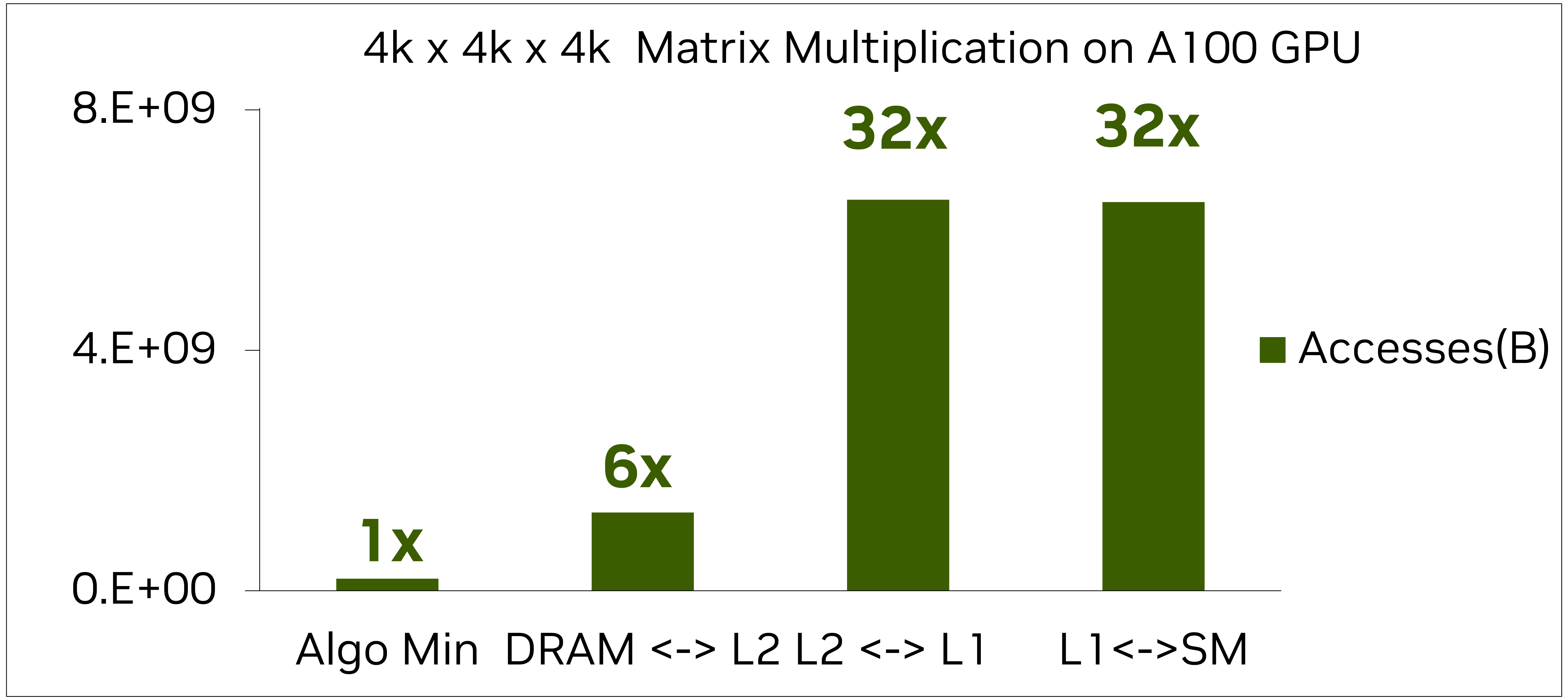




How to provision chip area between storage and compute?

What is missing?

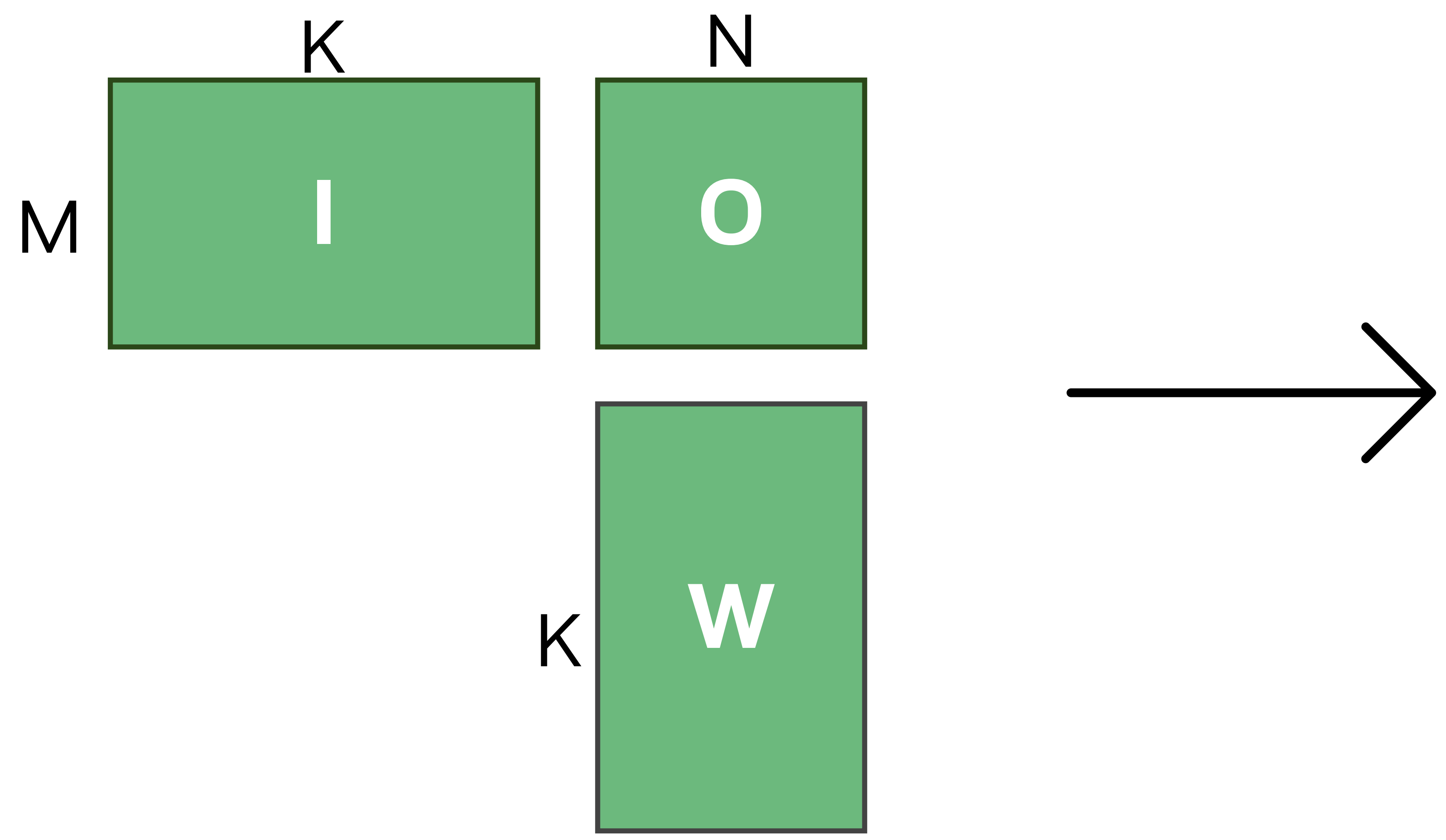
- The workload **does not** always operate with *algorithmic minimal accesses*, or equivalently, *algorithmic maximal OI*.



- Actual *backing-store accesses* and *OI* depend on the **mapping** and **buffer sizes**.

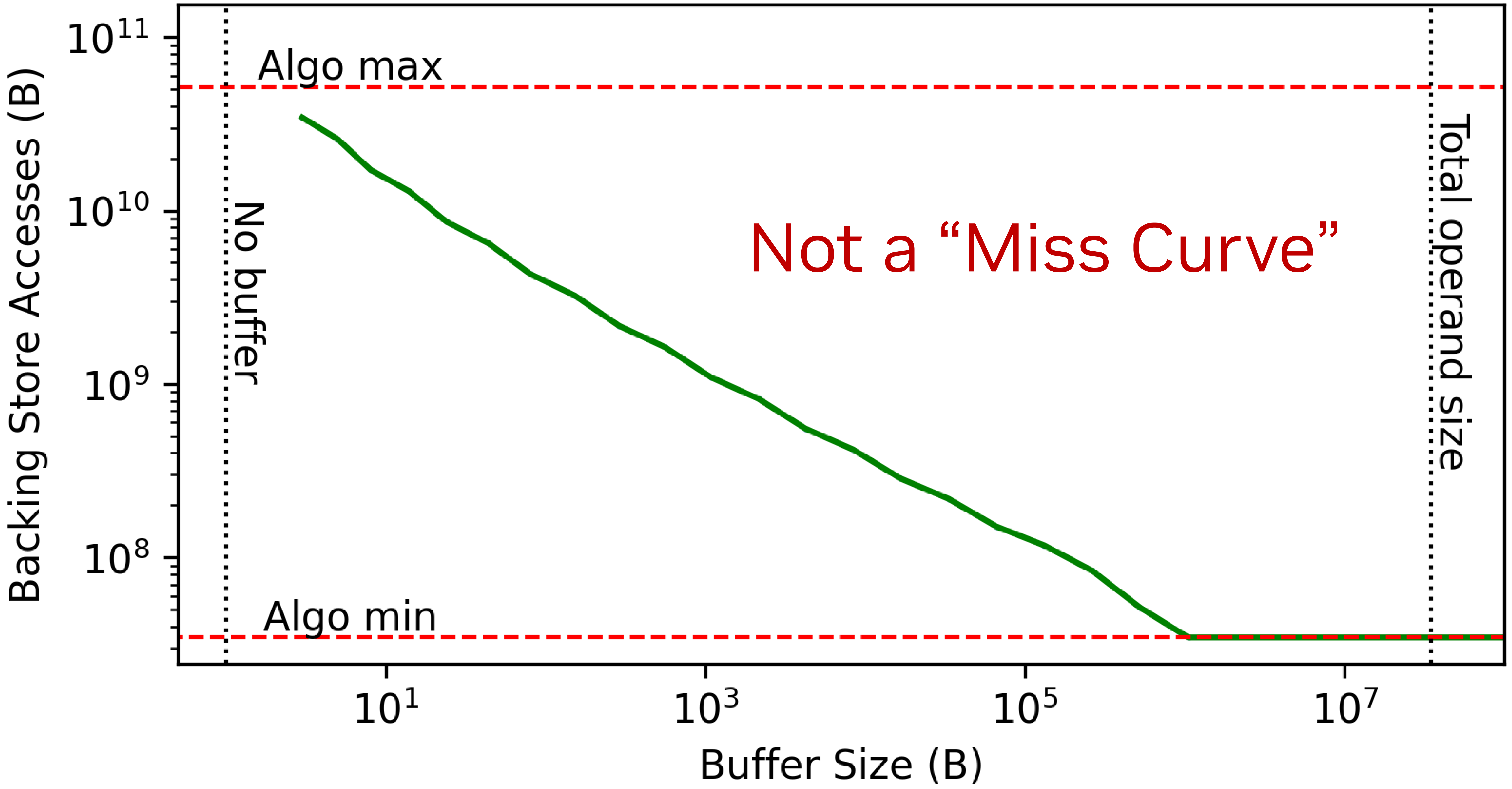
A desirable data movement bound

Matrix Multiplication (GEMM) Einsum:



$$O_{m,n} = I_{m,k} W_{k,n}$$

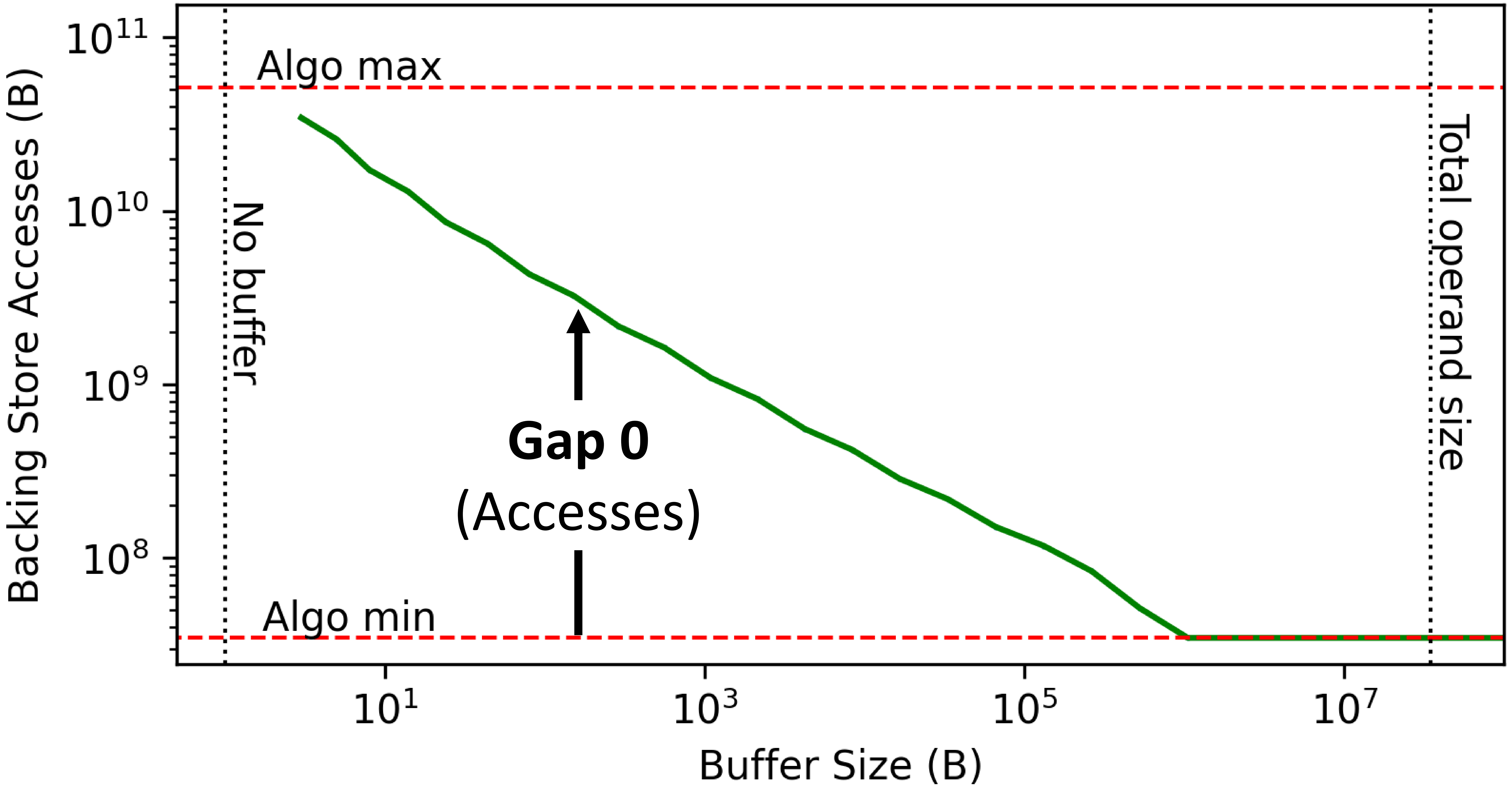
M – output row dim
K - reduction dim
N – output column dim



Not a "Miss Curve"

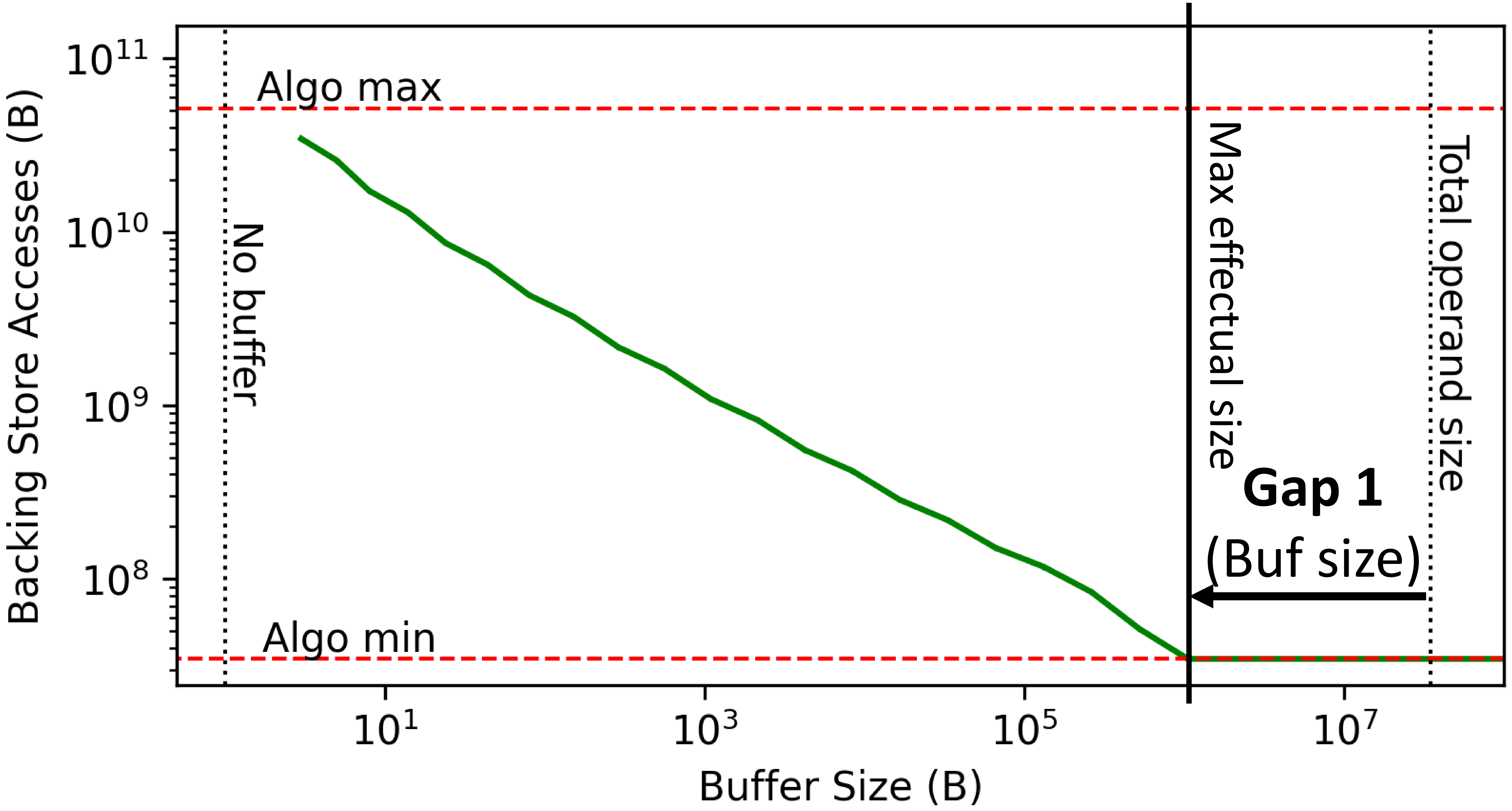
"Ski-slope Diagram"

Mind the gap: key design questions



[Gap 0] Given a buffer capacity, what is the minimal attainable *data access count*?

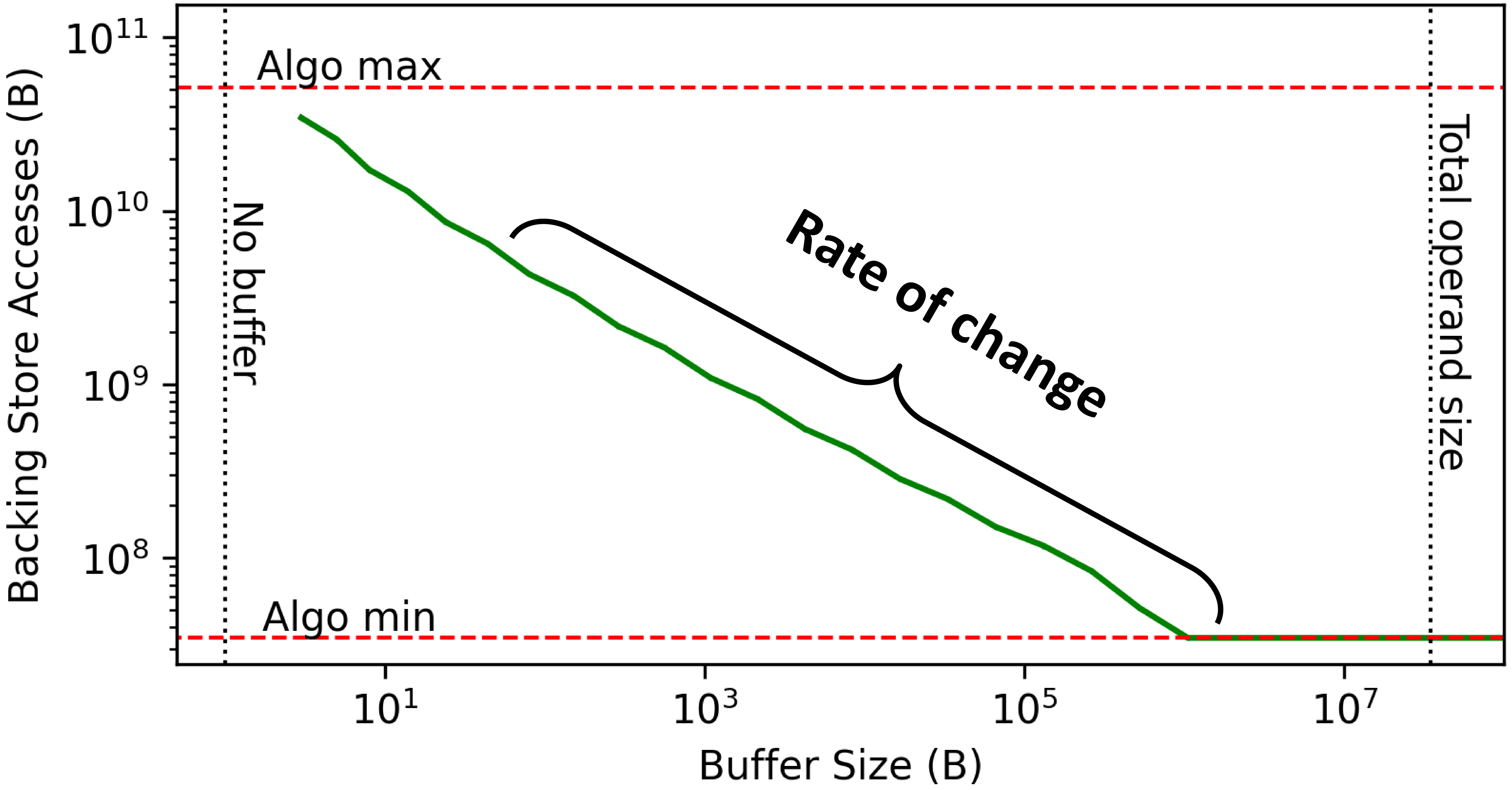
Mind the gap: key design questions



[Gap 0] Given a buffer capacity, what is the minimal attainable *data access count*?

[Gap 1] How much buffer capacity is required to achieve full data reuse?

Mind the gap: key design questions



[Gap 0] Given a buffer capacity, what is the minimal attainable *data access count*?

[Gap 1] How much buffer capacity is required to achieve full data reuse?

[rate of change of Gap 0] How does a workload benefit from incremental increase in buffer capacity?

Orojenesis → ***Oro - Genesis***

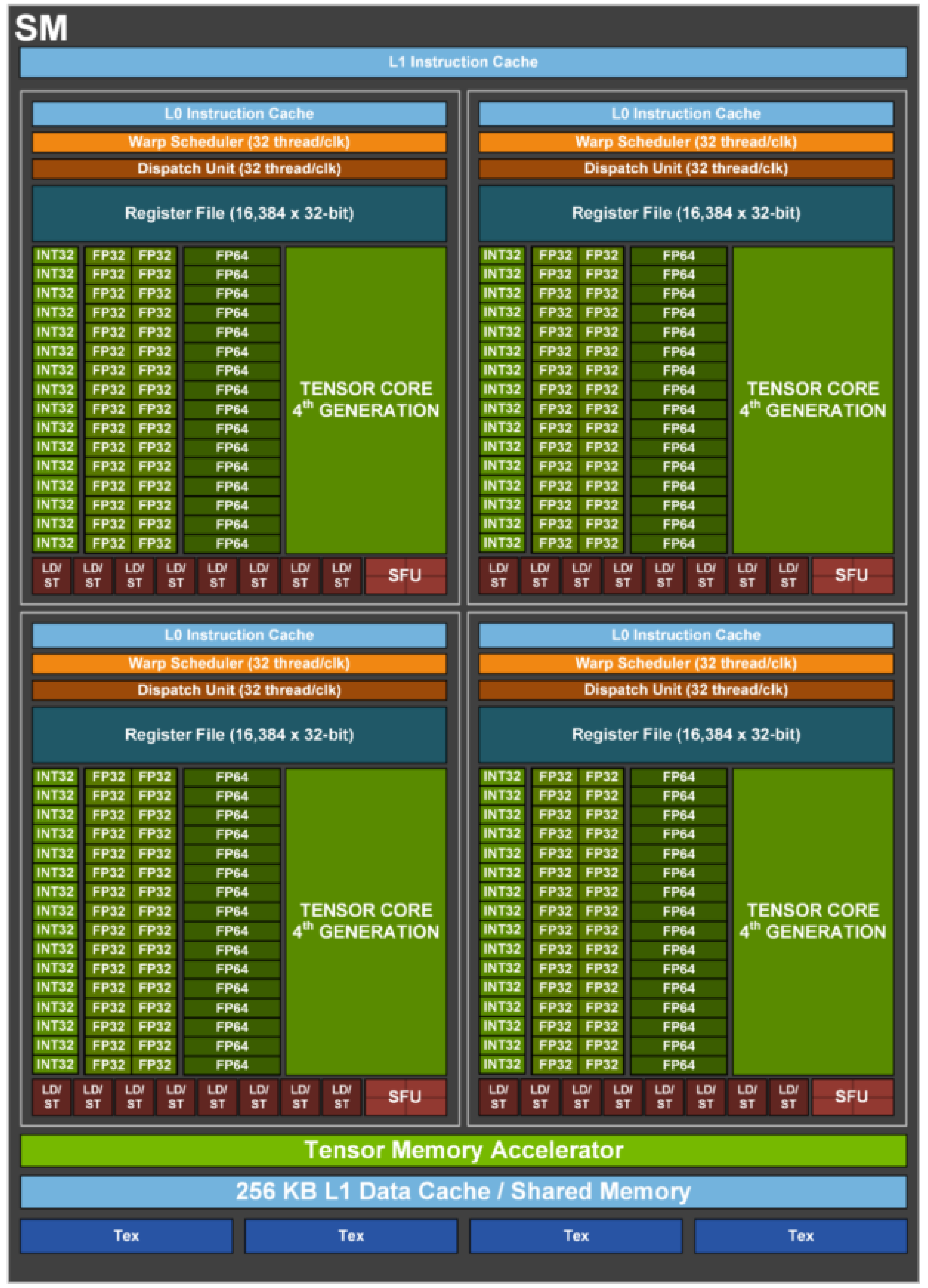
↓ Mountain ↓ Creation

Bound derivation via exhaustive search enabled by:

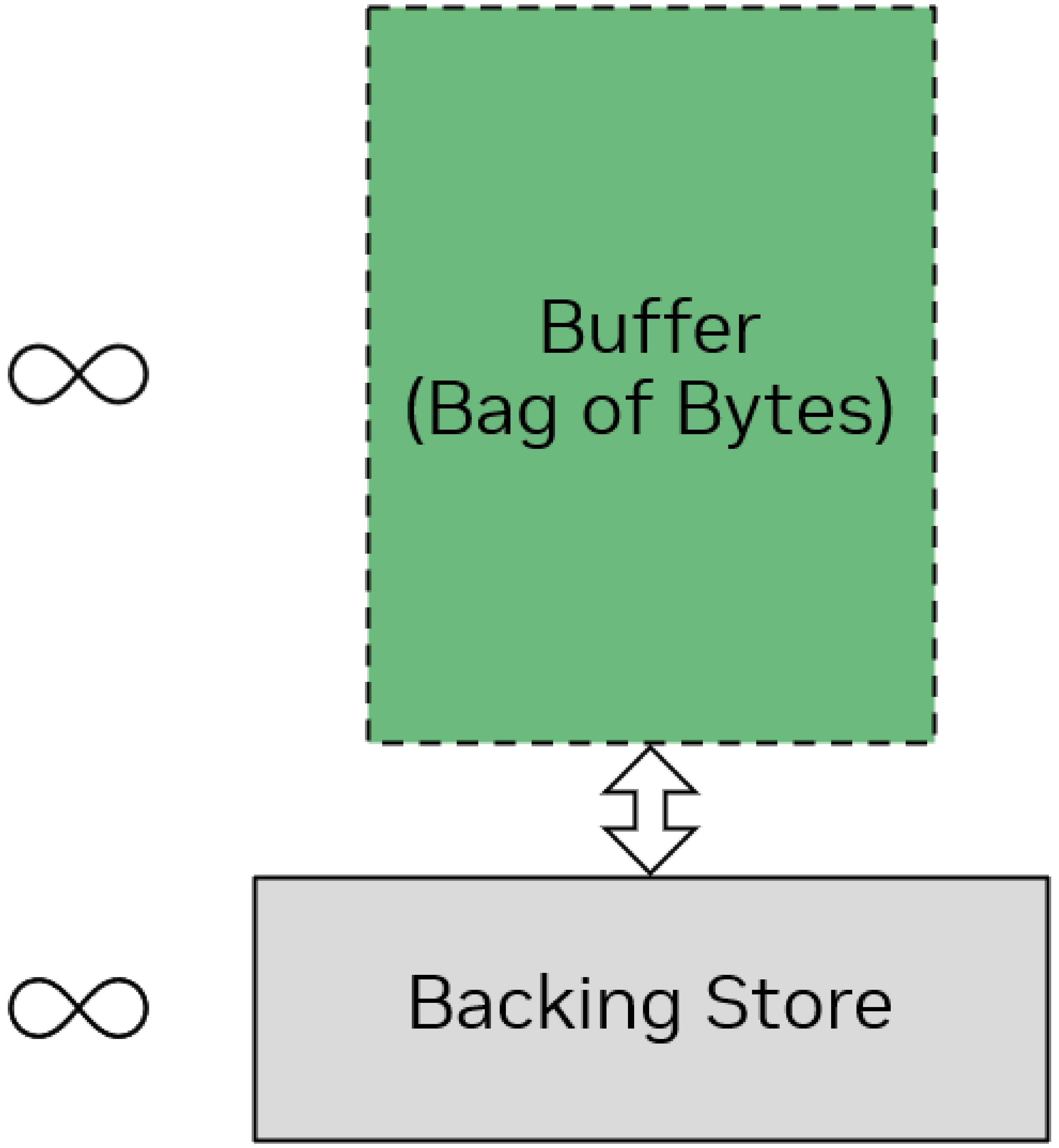
1. Simplified architecture
2. Mapping-derived buffer requirements

The *Snowcat* Architecture

Enables exhaustive search


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE 4th GENERATION
LD/ST
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

Real Design



∞

Buffer
(Bag of Bytes)

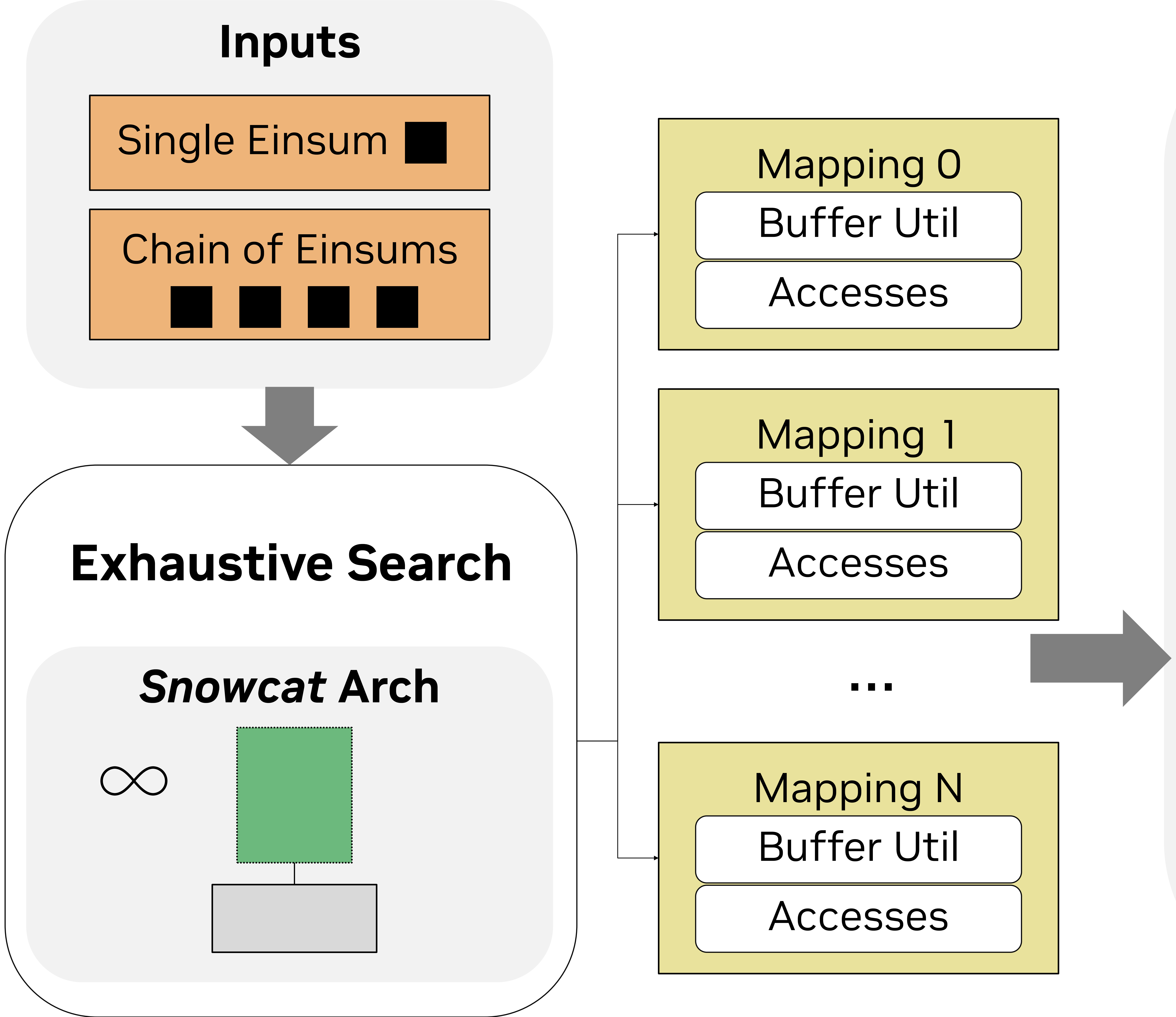
∞

Backing Store

Snowcat
Architecture

The *Orojenesis* Methodology

A single exhaustive mapping search per workload



Inputs

Single Einsum ■

Chain of Einsums ■ ■ ■ ■

Exhaustive Search

Snowcat Arch

∞

Mapping 0

Buffer Util

Accesses

Mapping 1

Buffer Util

Accesses

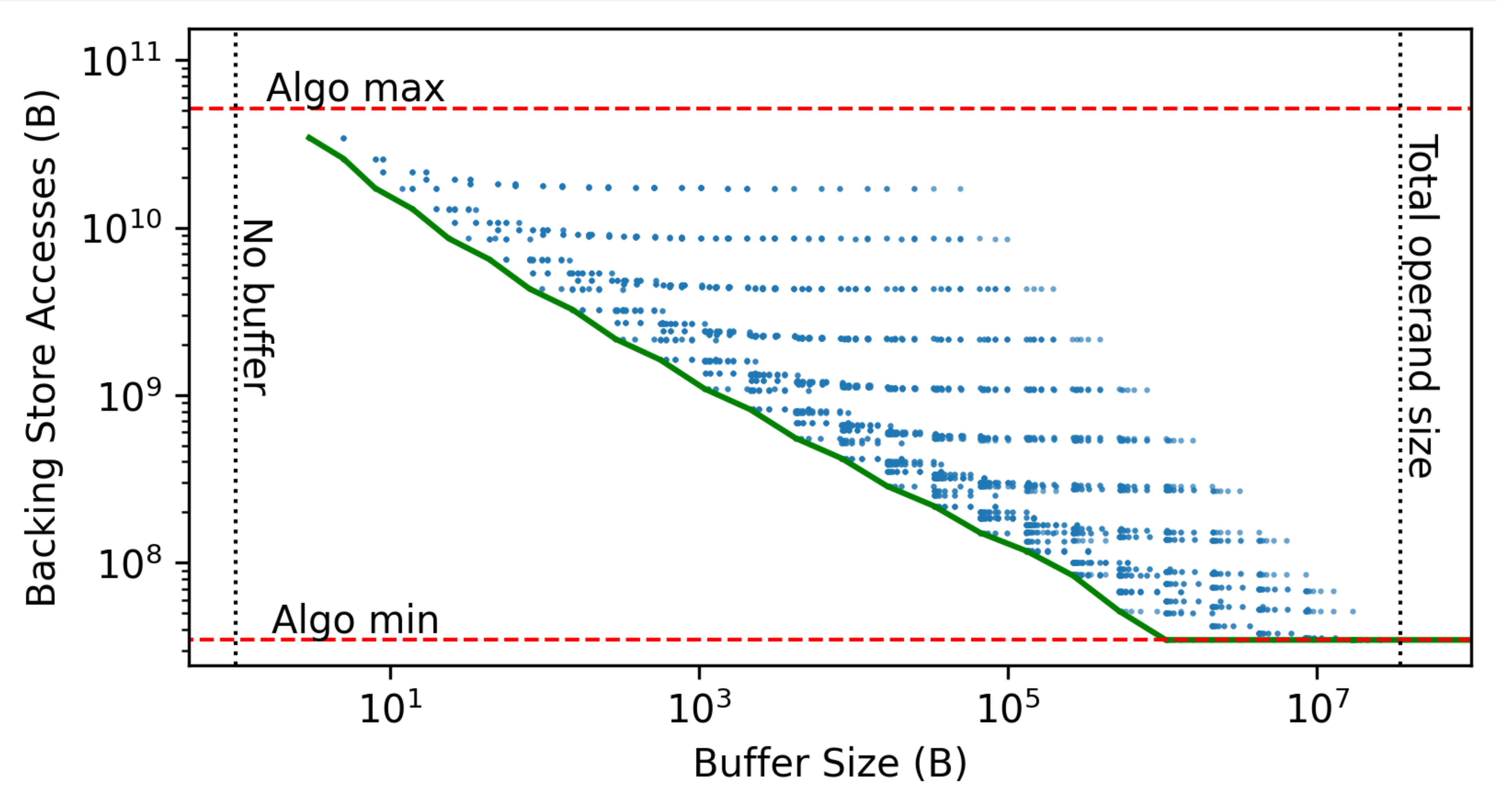
...

Mapping N

Buffer Util

Accesses

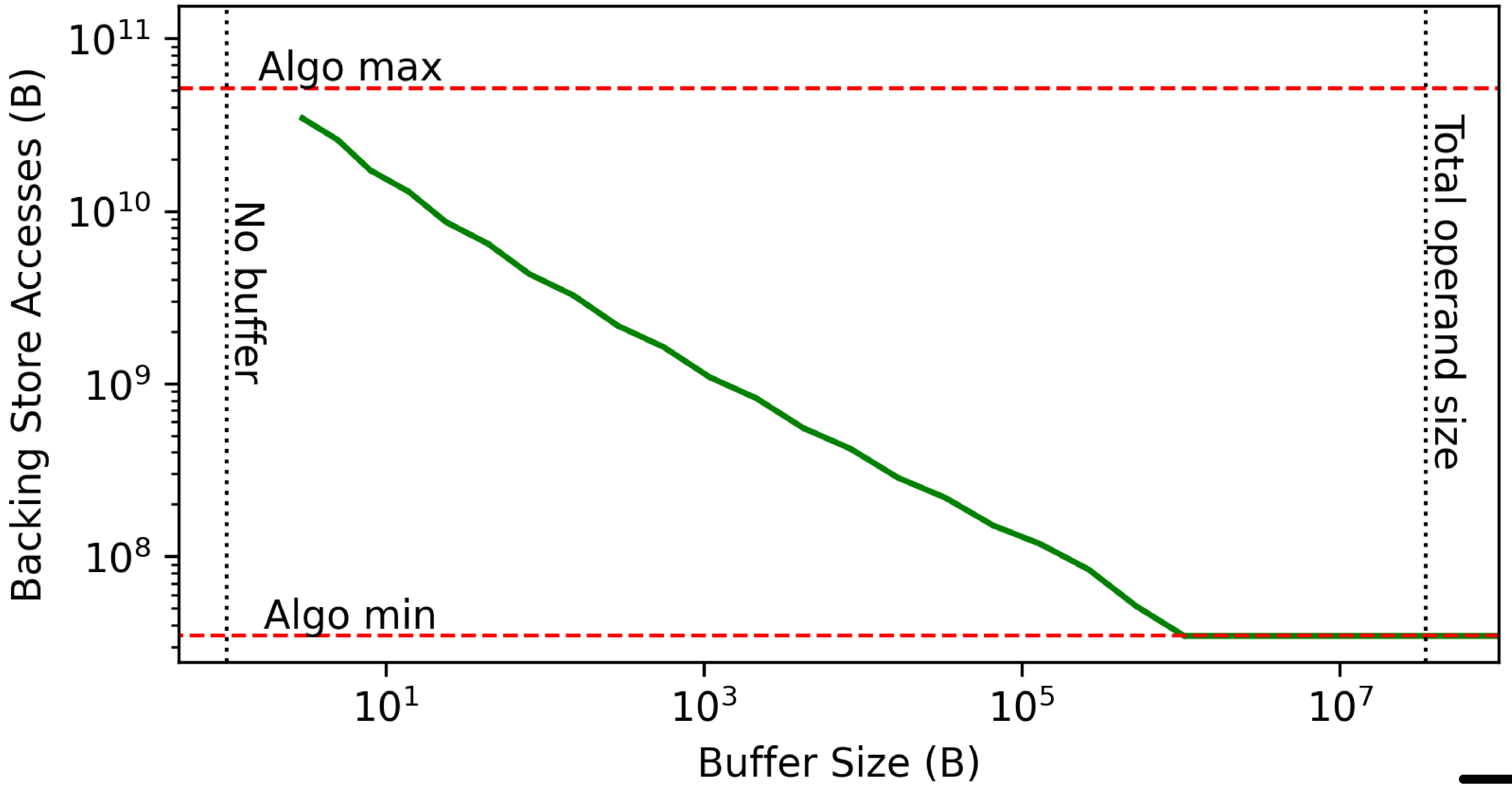
Ski-slope Diagram



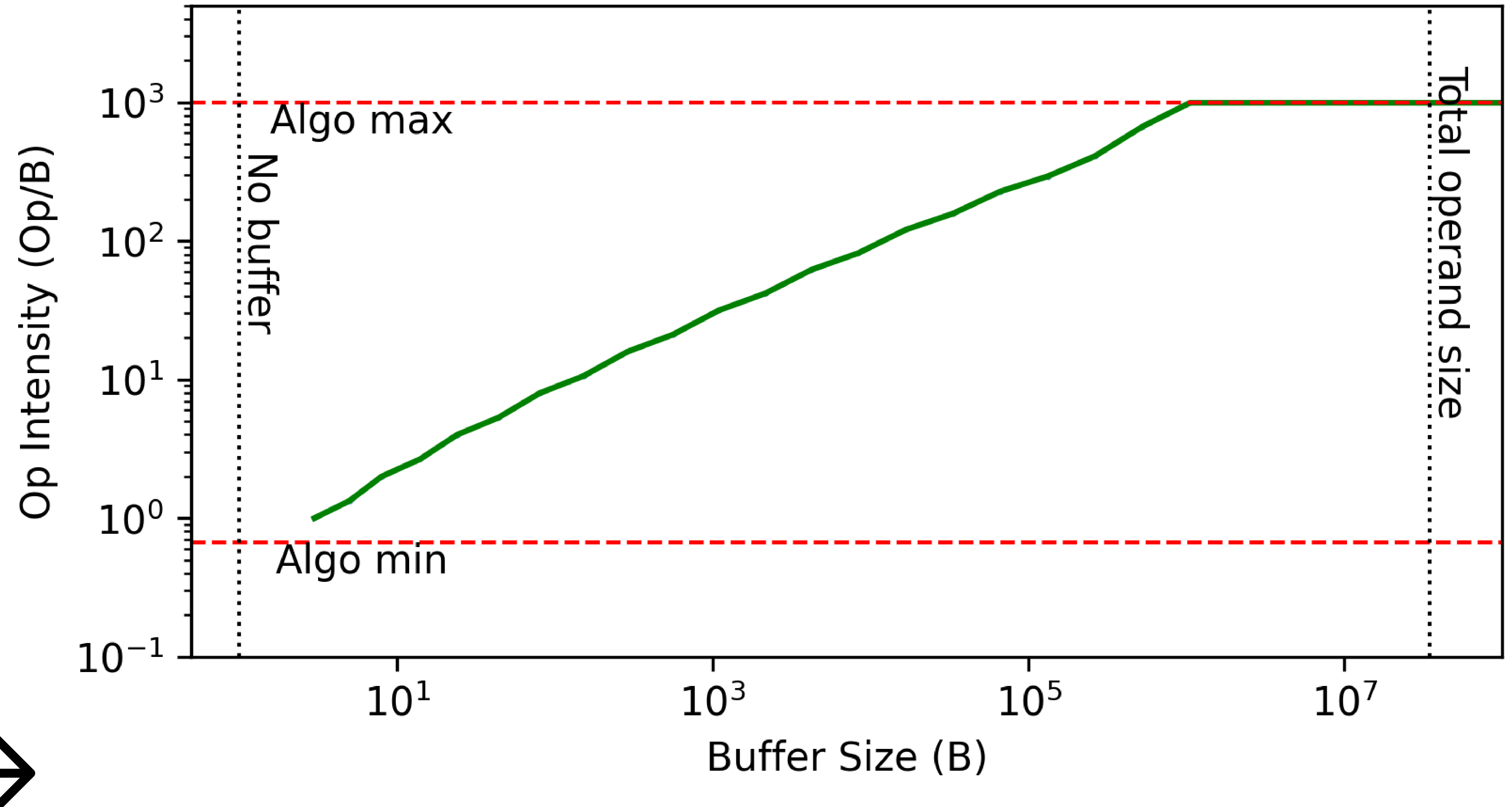
min(buffer util, accesses)

OI Bound Derivation

Data Movement Bound



OI Bound



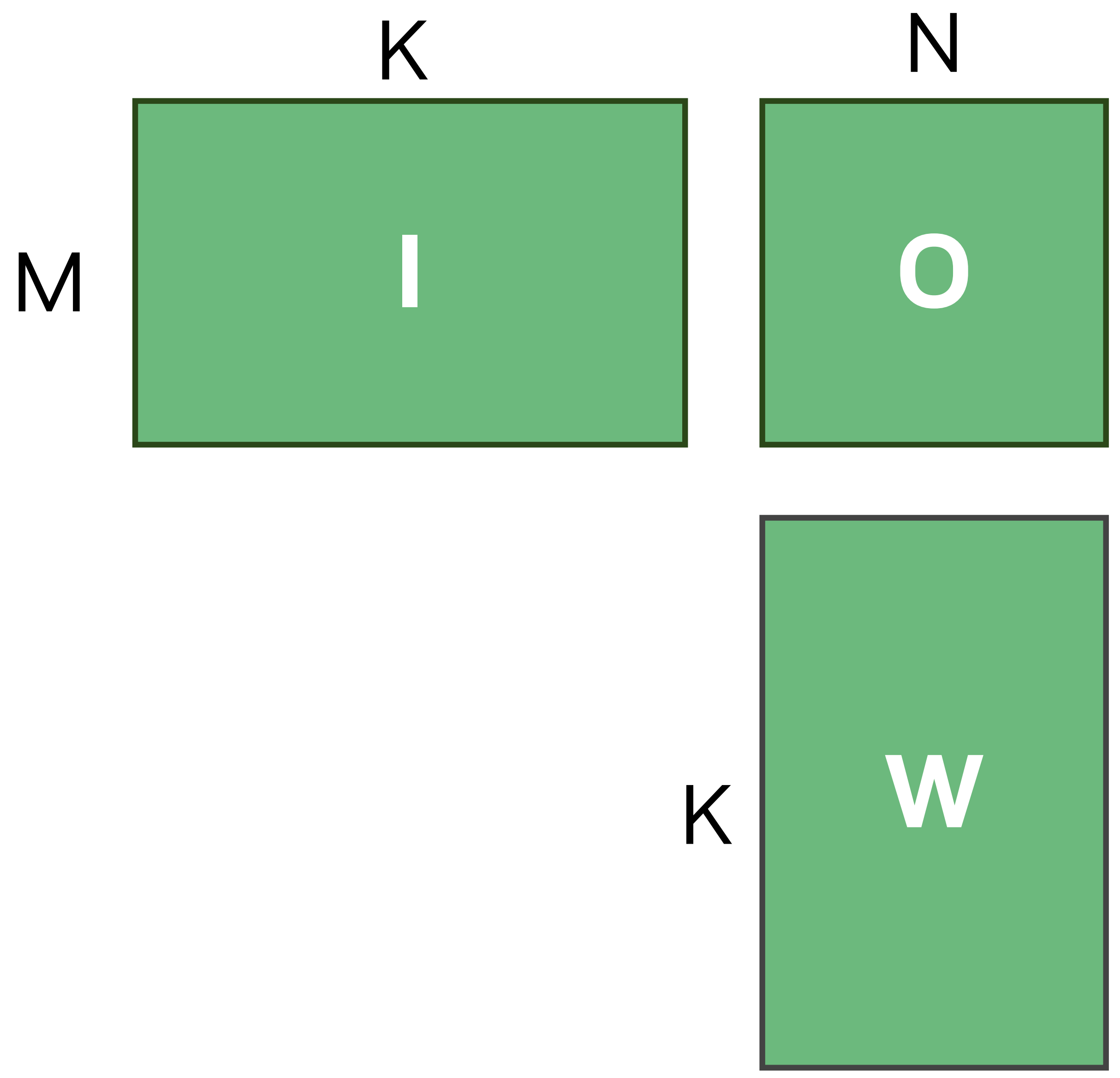
Inverse

&

Multiply by
total operations

Example *Orojenesis* Analysis

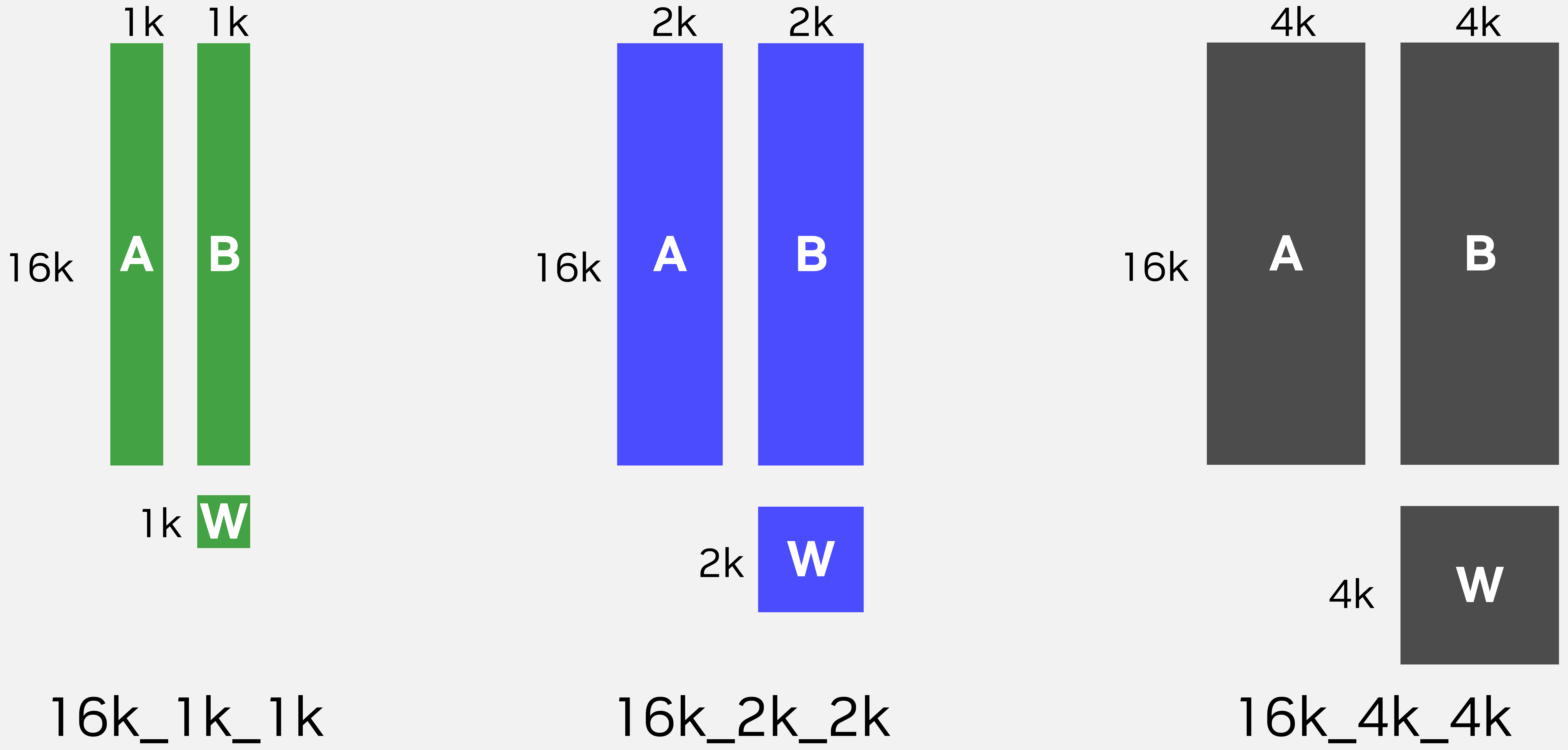
GEMM Einsum:



$$O_{m,n} = I_{m,k} W_{k,n}$$

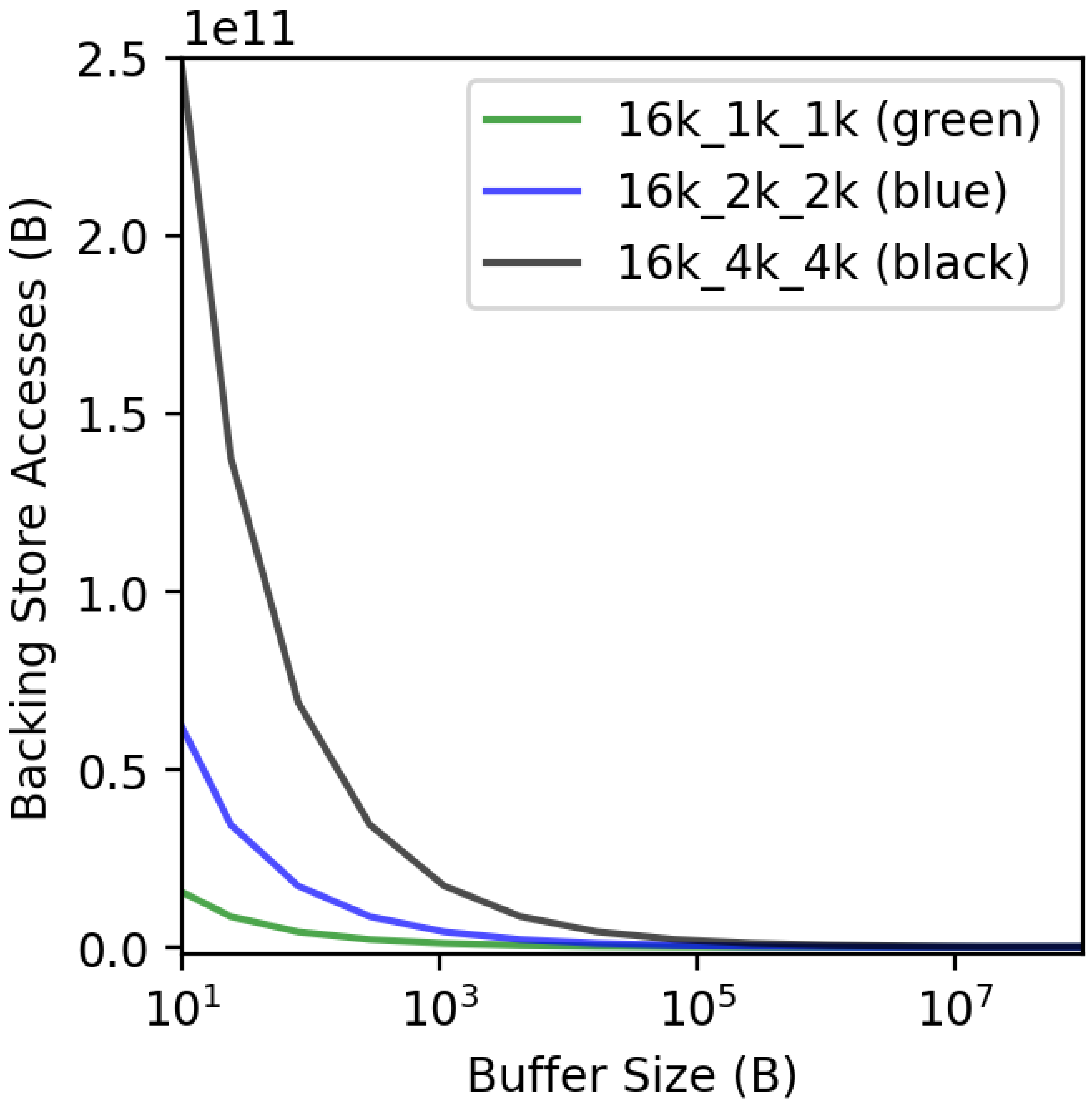
M – output row dim
K - reduction dim
N – output column dim

Input Examples

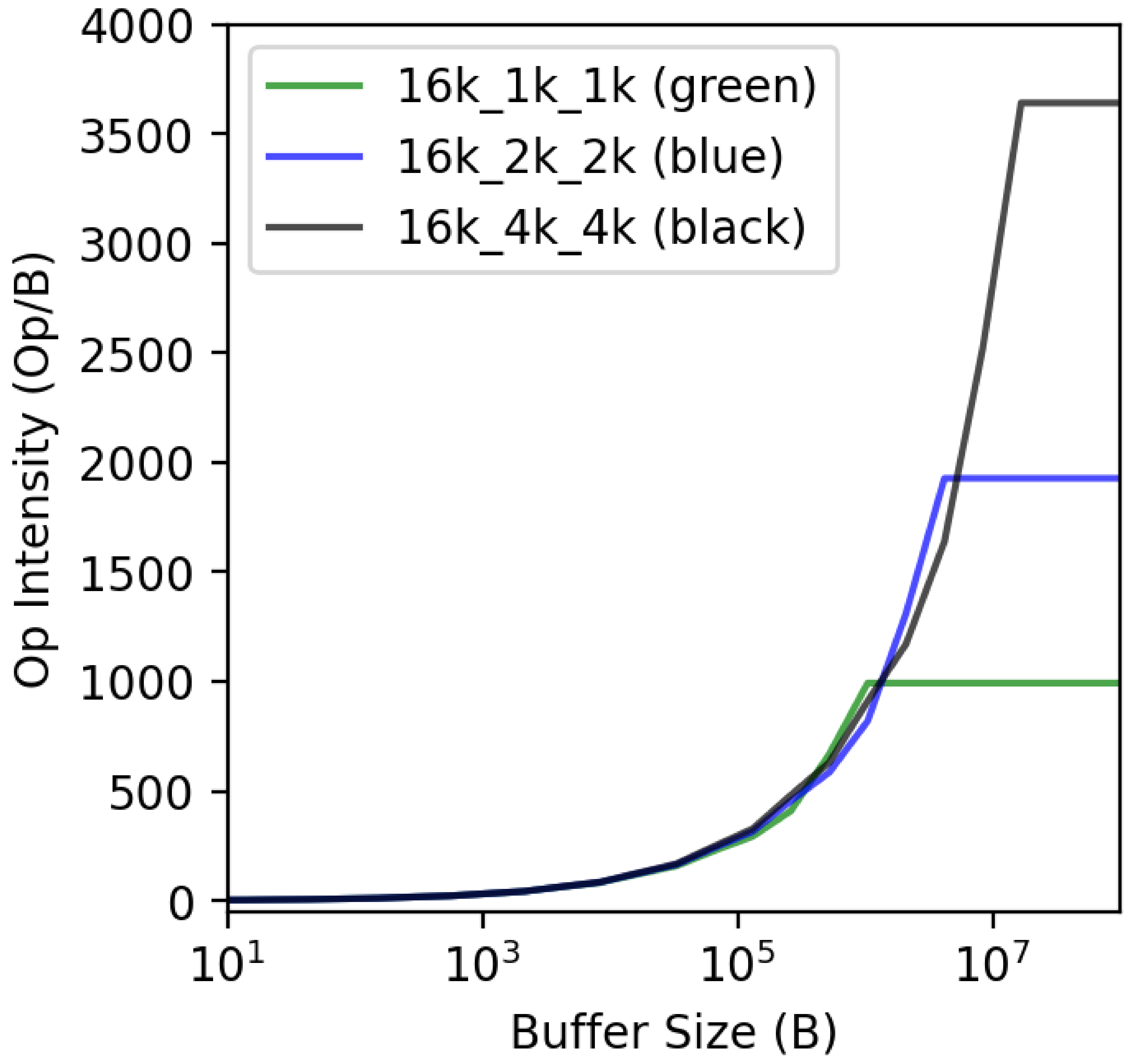


Example *Orojenesis* Analysis

Data Movement Bound

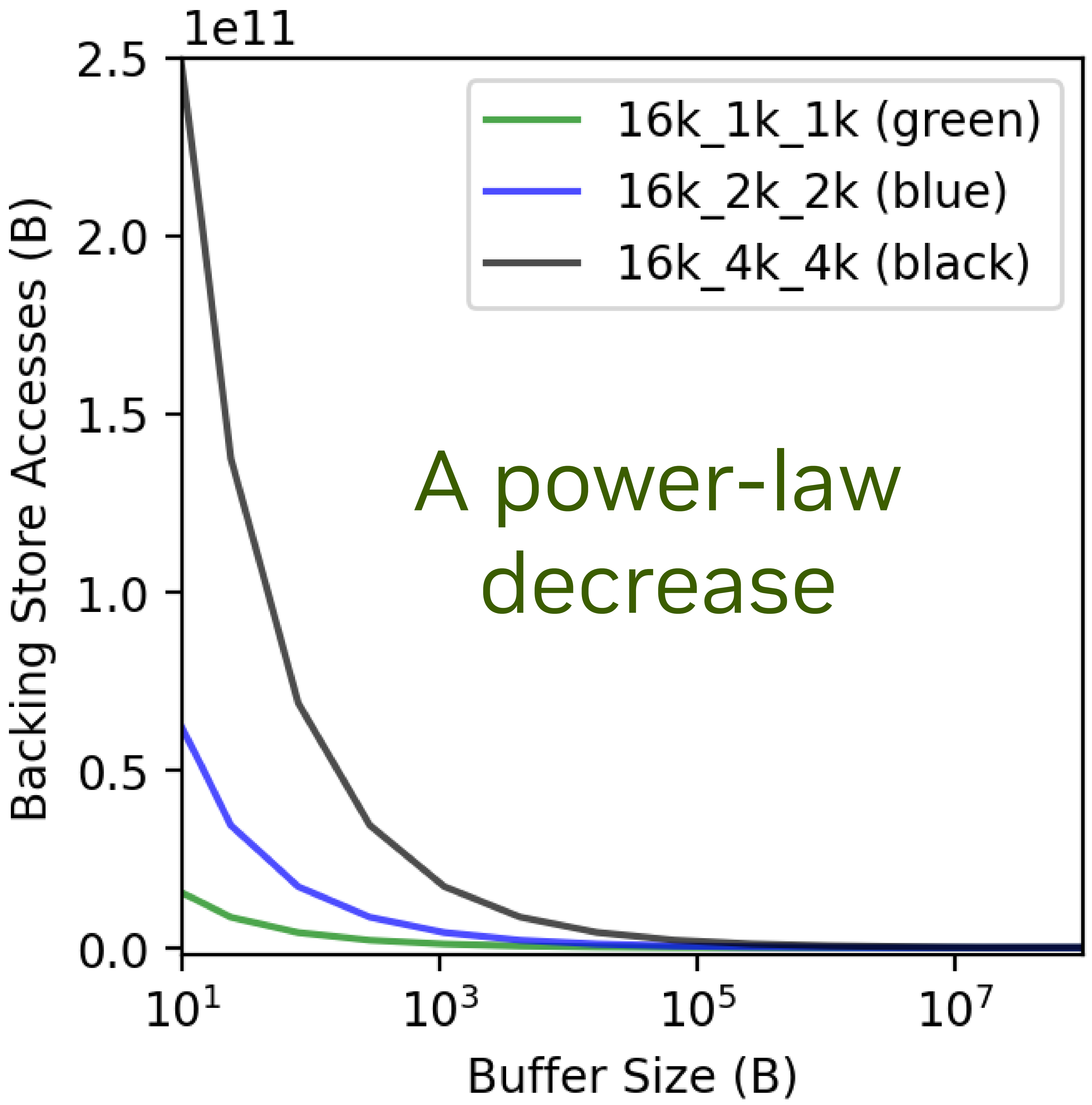


OI Bound



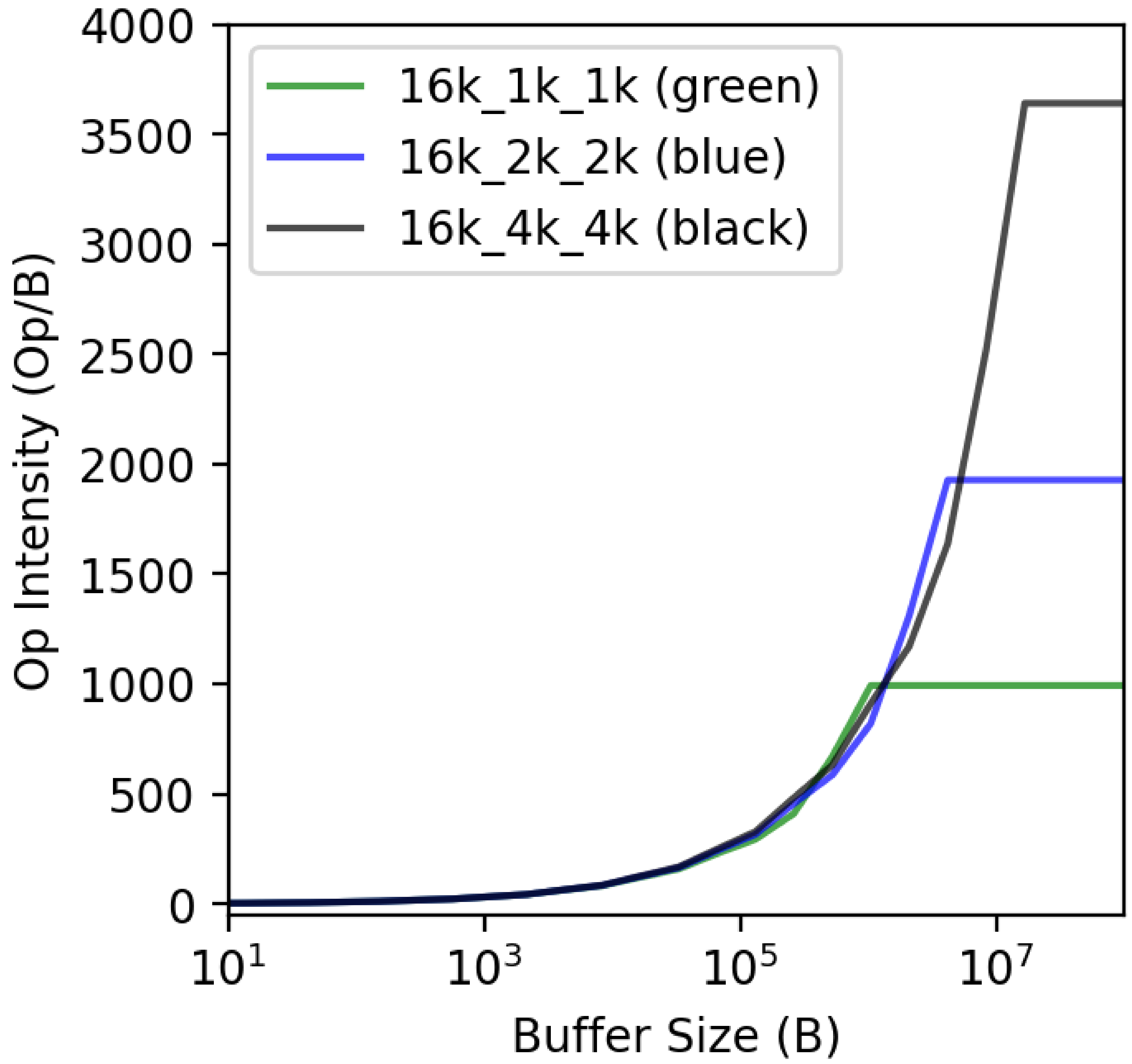
Example *Orojenesis* Analysis

Data Movement Bound

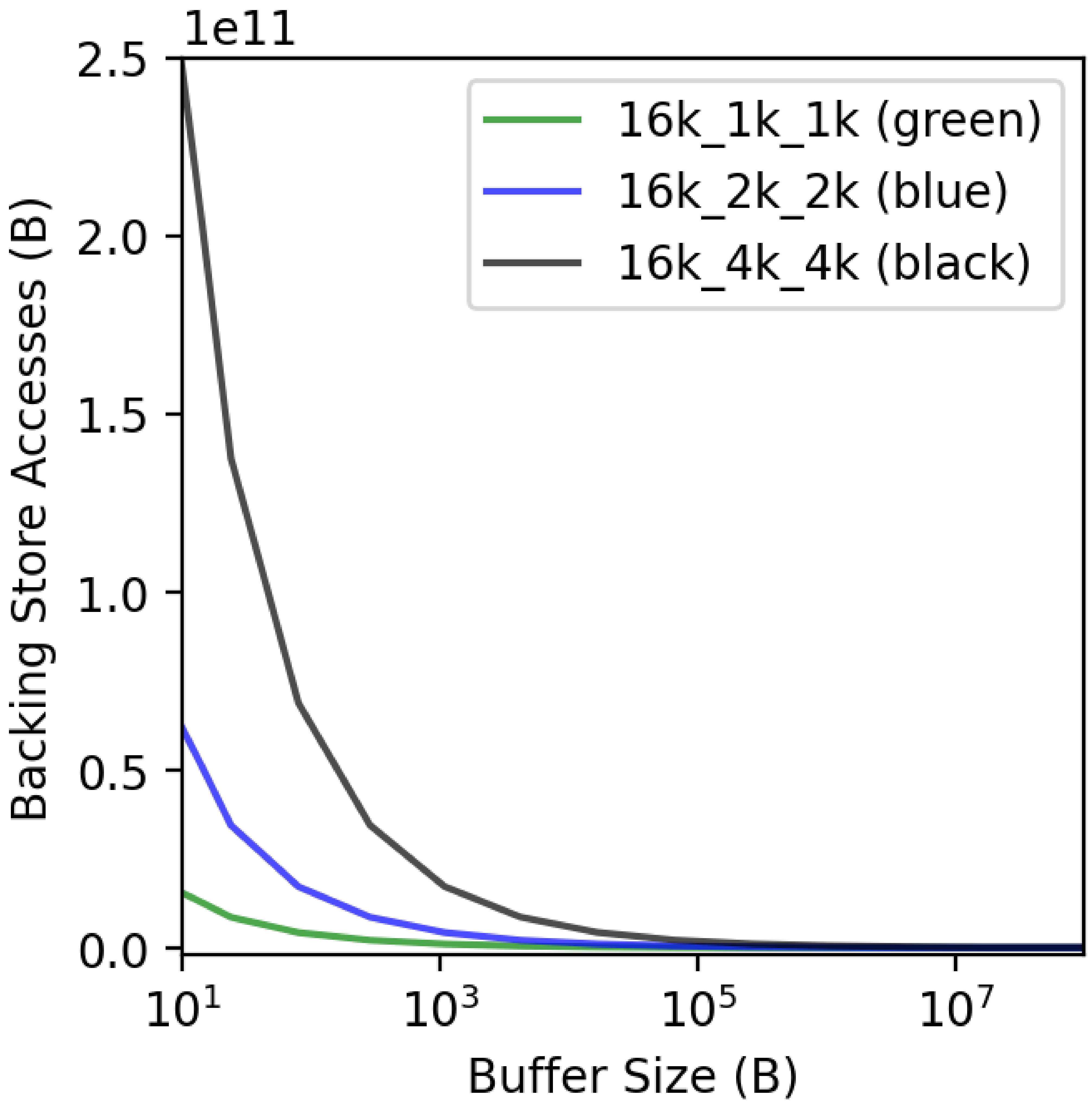


A power-law decrease

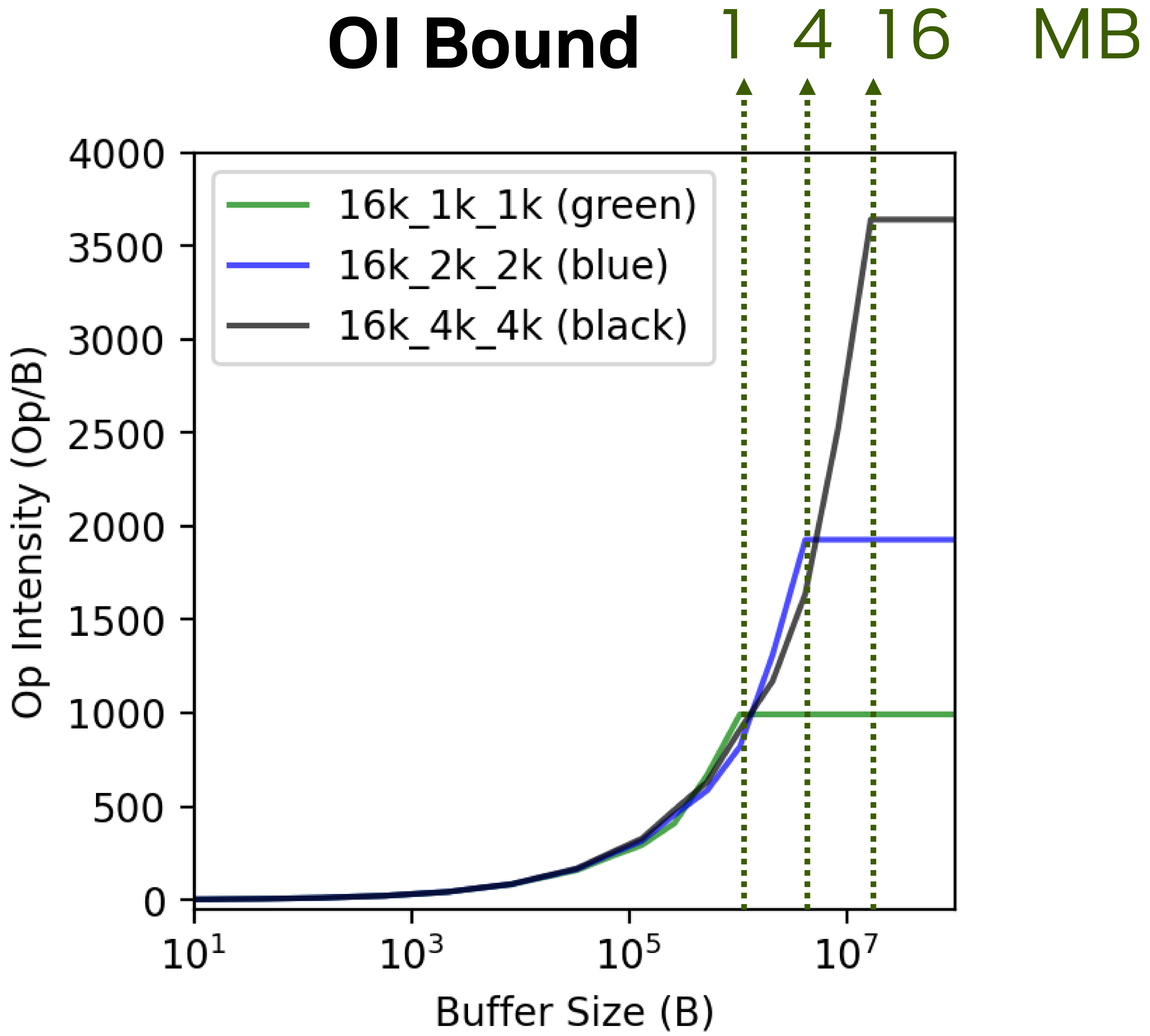
OI Bound



Example *Orojenesis* Analysis



Data Movement Bound



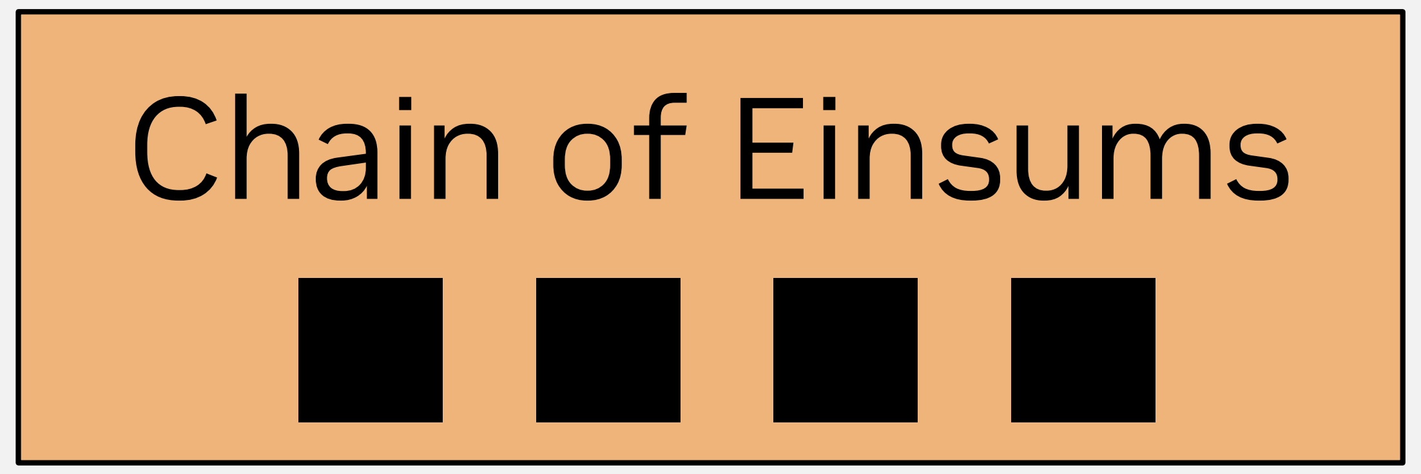
OI Bound

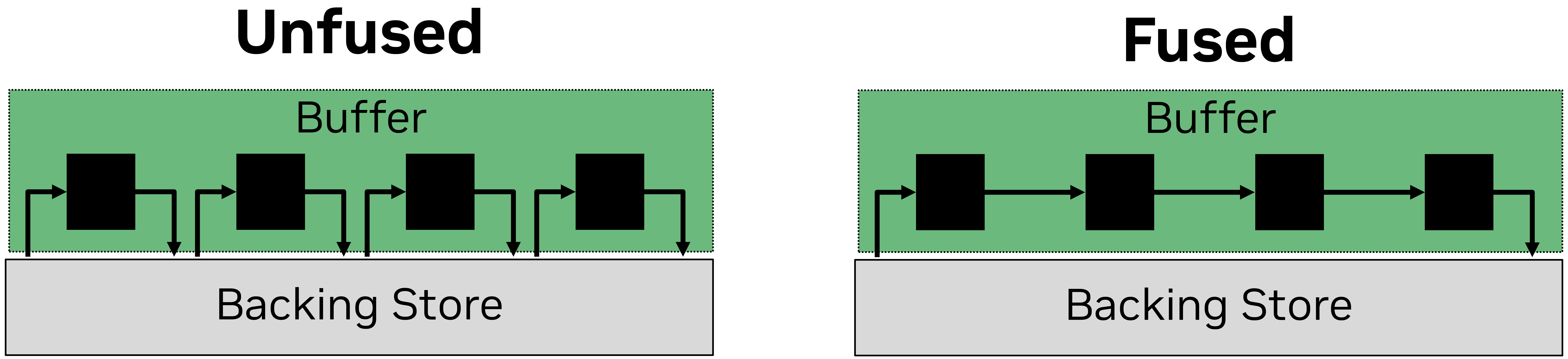
The maximal effectual buffer size of a GEMM is approximately its **smallest operand size**

#1: Orojenesis produces bounds that reveal powerful design insights

The *Orojenesis* Fusion Flow

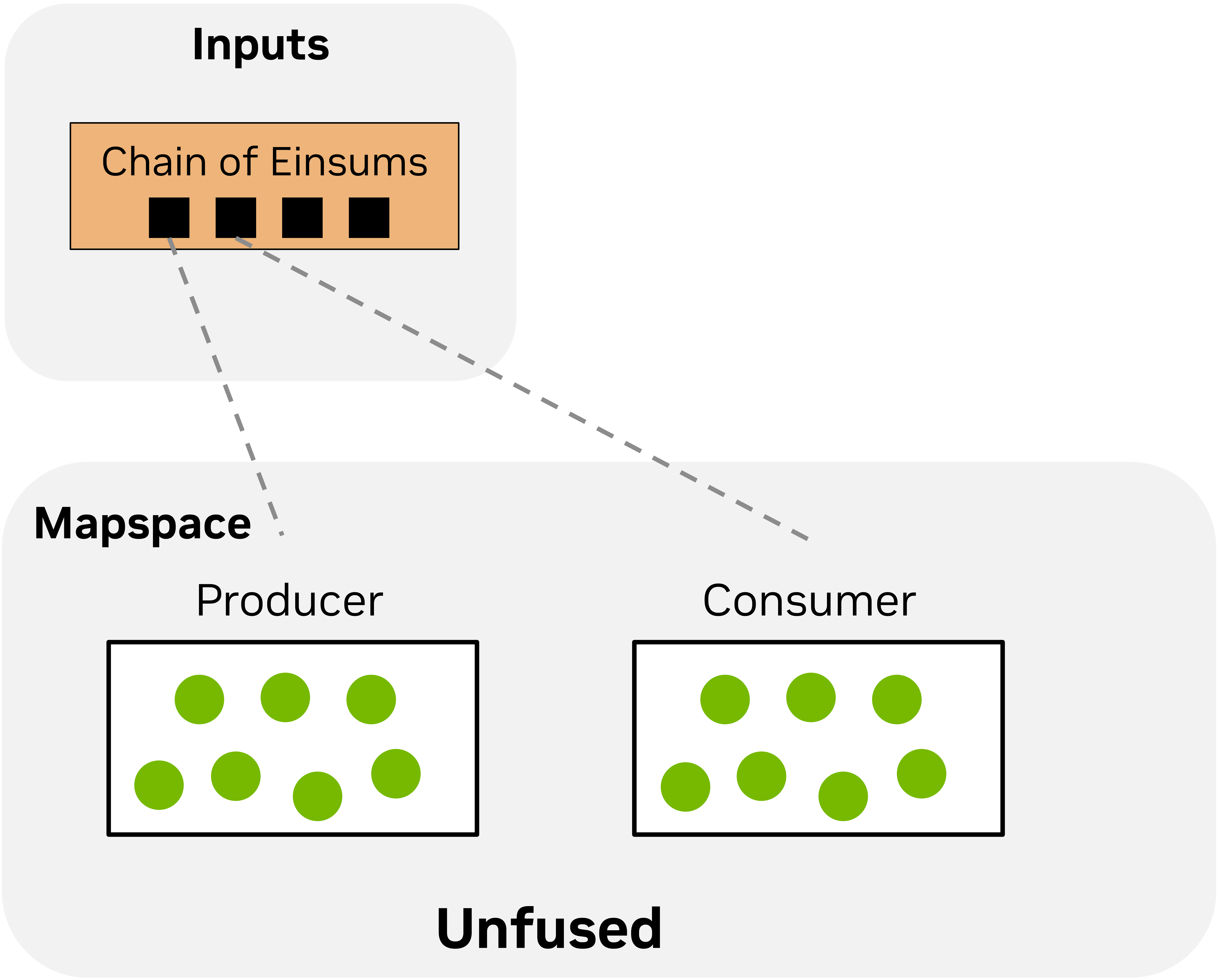
Inputs





Fusion is an effective technique to minimize data movement for a chain of operations

The *Orojenesis* Fusion Flow



Inputs

Chain of Einsums

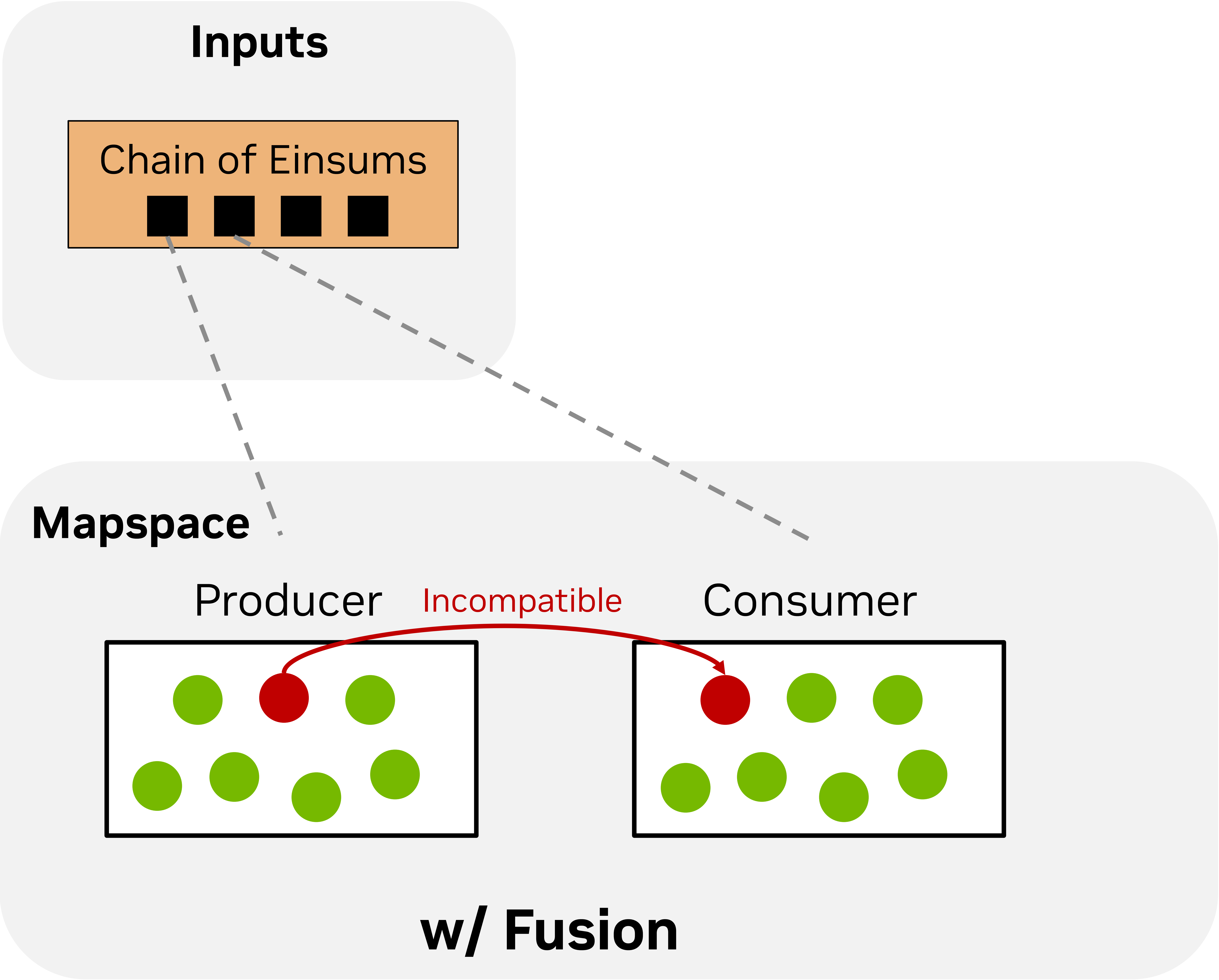
Mapspace

Producer

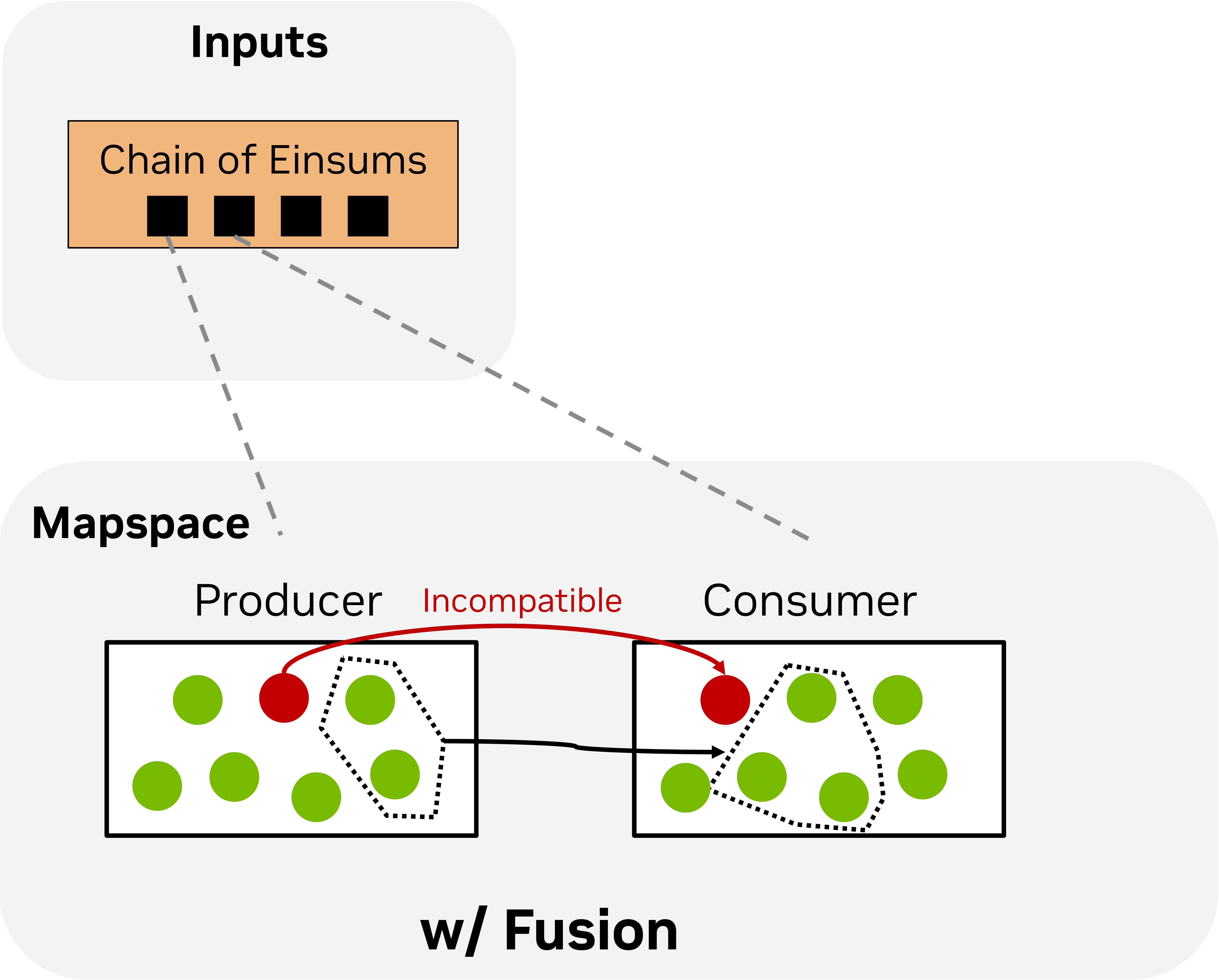
Consumer

Unfused

The *Orojenesis* Fusion Flow



The *Orojenesis* Fusion Flow



Inputs

Chain of Einsums

Mapspace

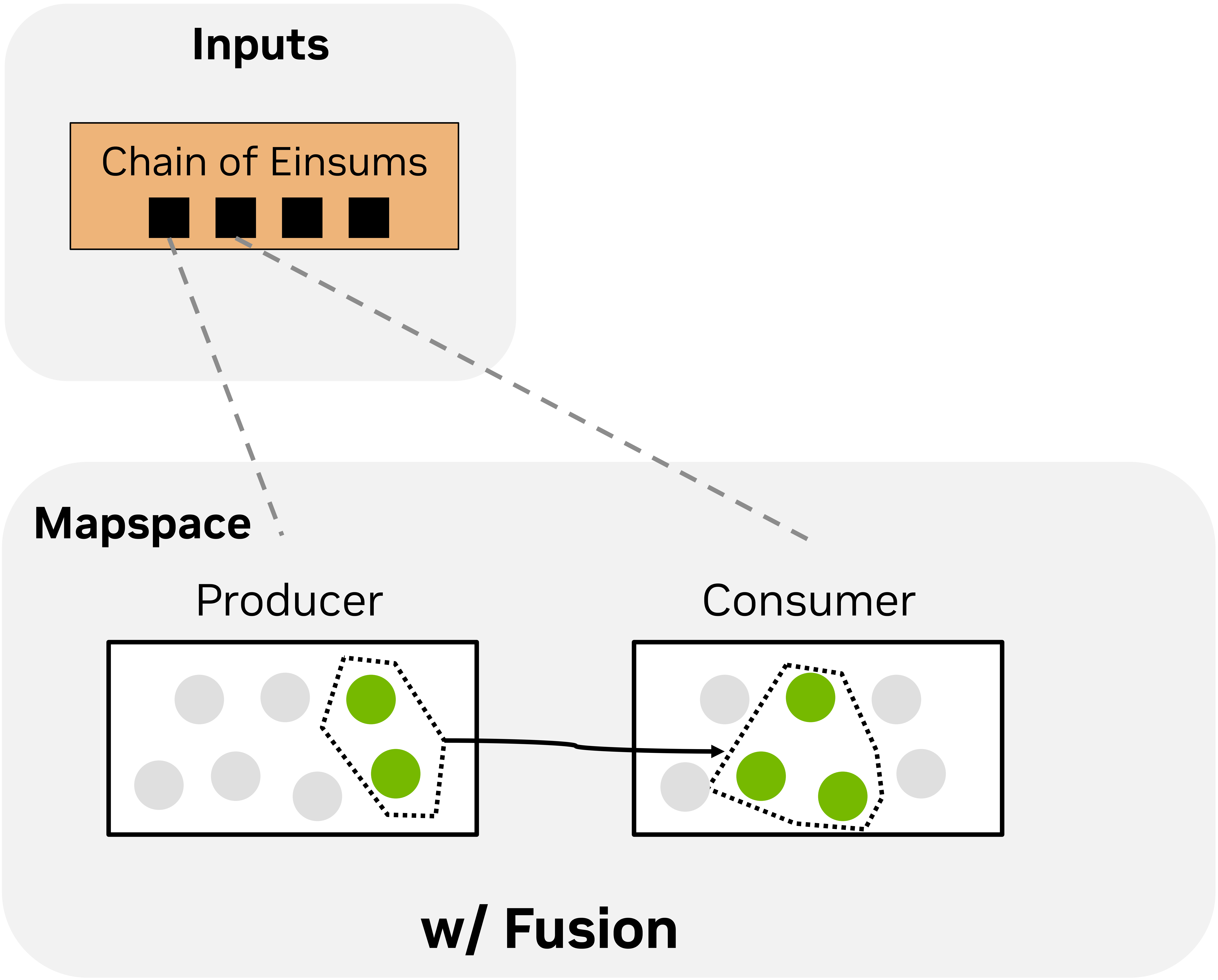
Producer

Incompatible

Consumer

w/ Fusion

The *Orojenesis* Fusion Flow



Inputs

Chain of Einsums

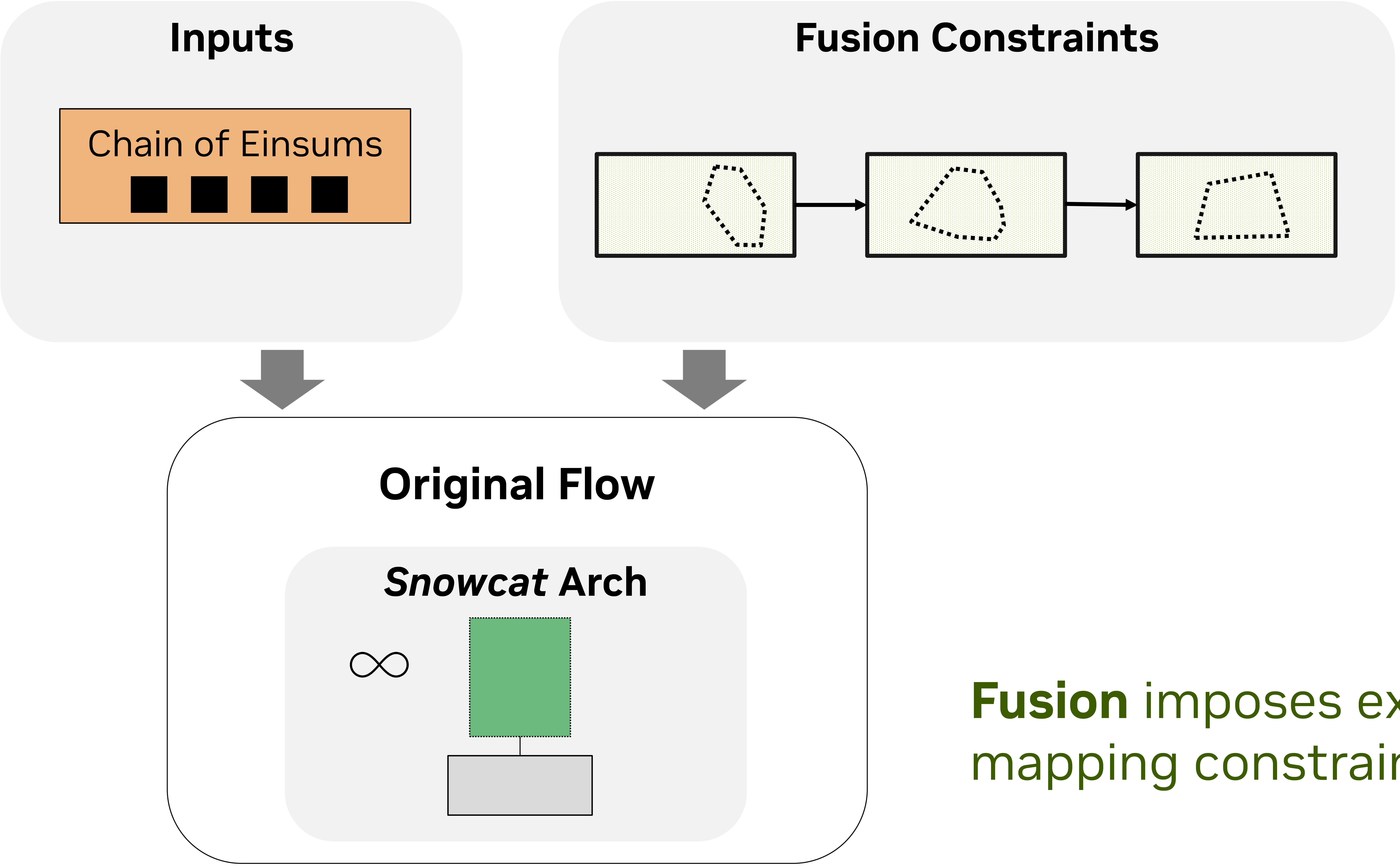
Mapspace

Producer

Consumer

w/ Fusion

The *Orojenesis* Fusion Flow



Inputs

Chain of Einsums

Fusion Constraints

Original Flow

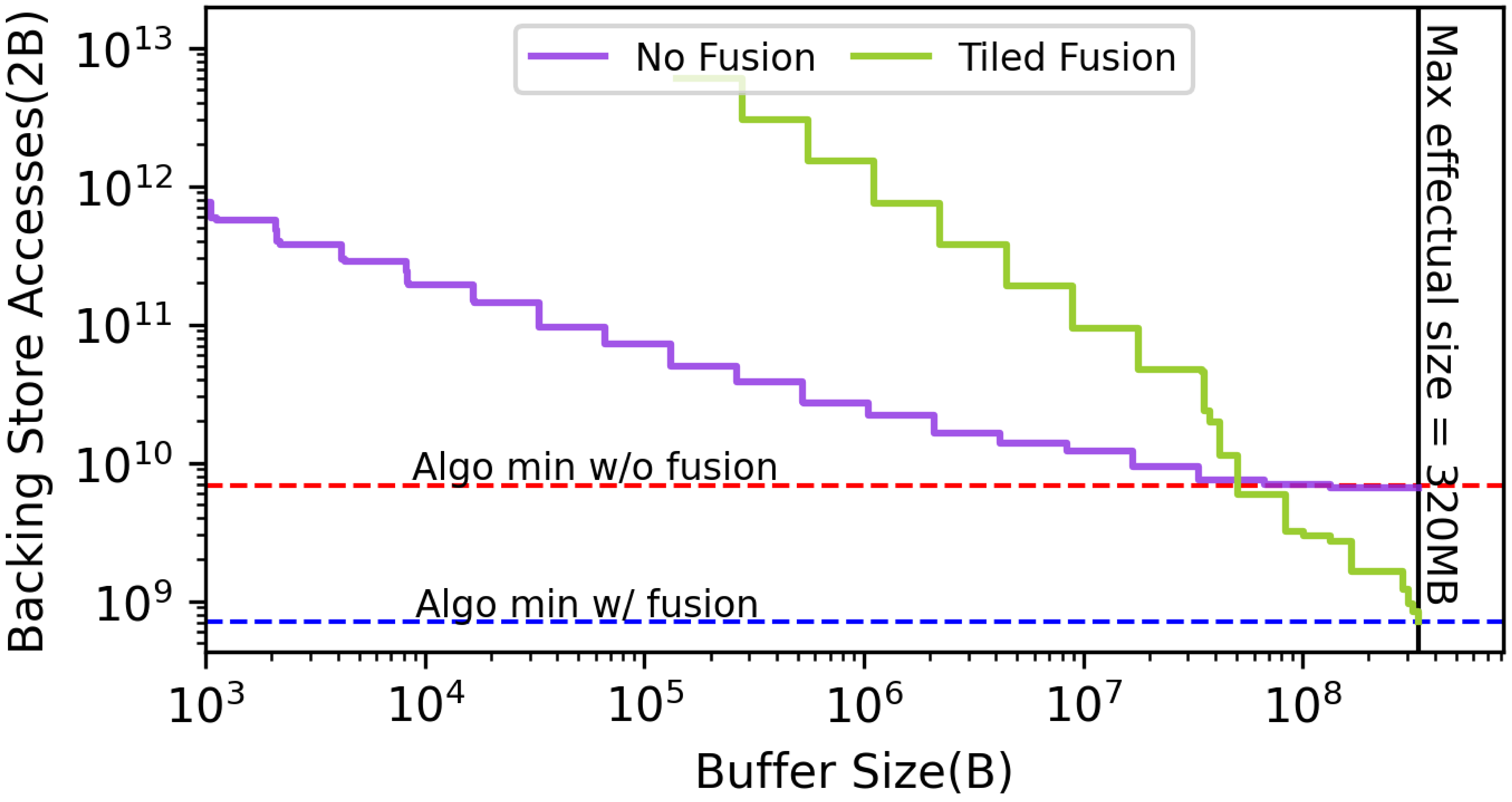
Snowcat Arch

∞

Fusion imposes extra intra-layer mapping constraints

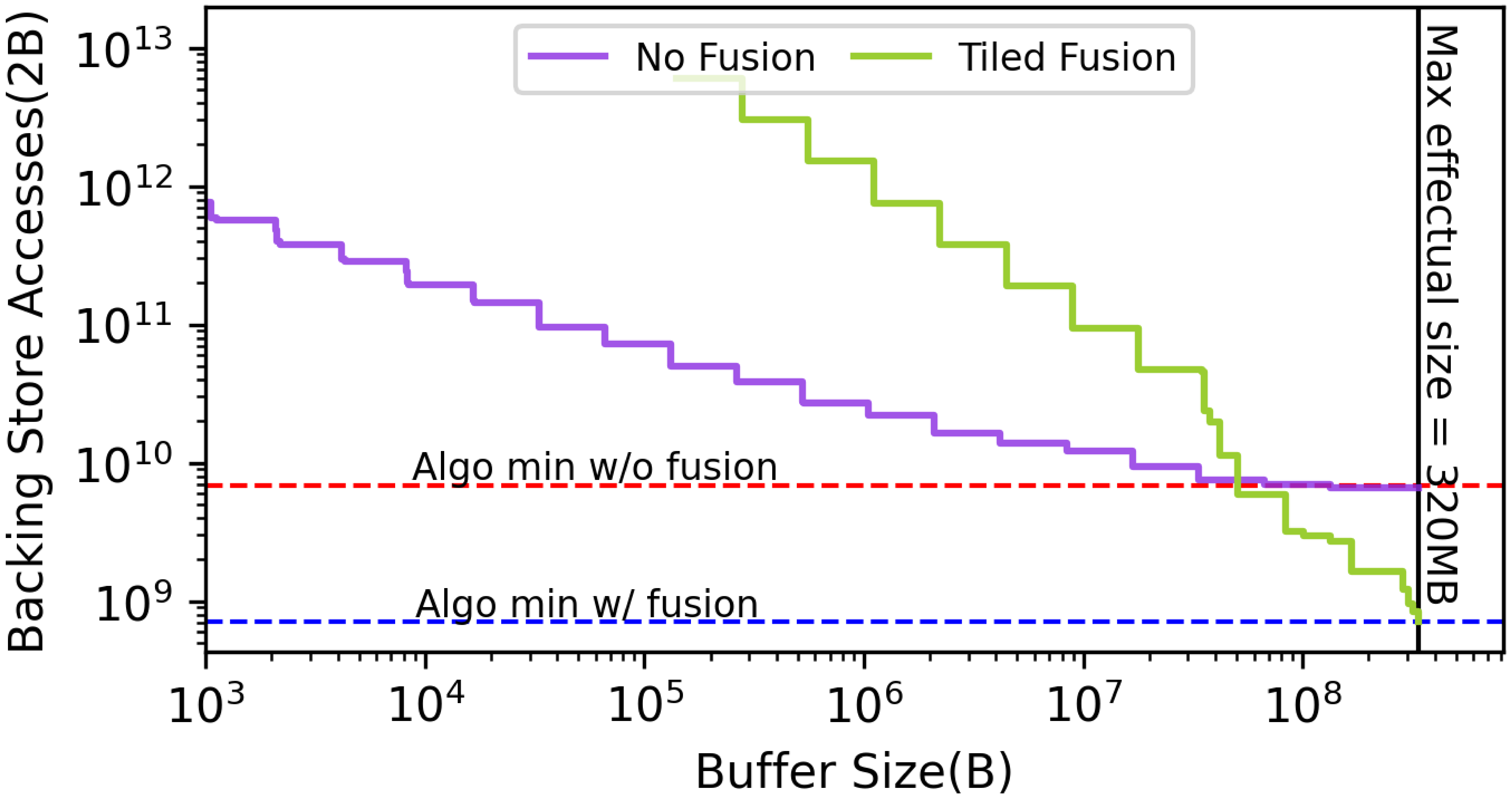
Fusion Analysis

A chain of 6 operations in GPT-6.7b block



Fusion Analysis

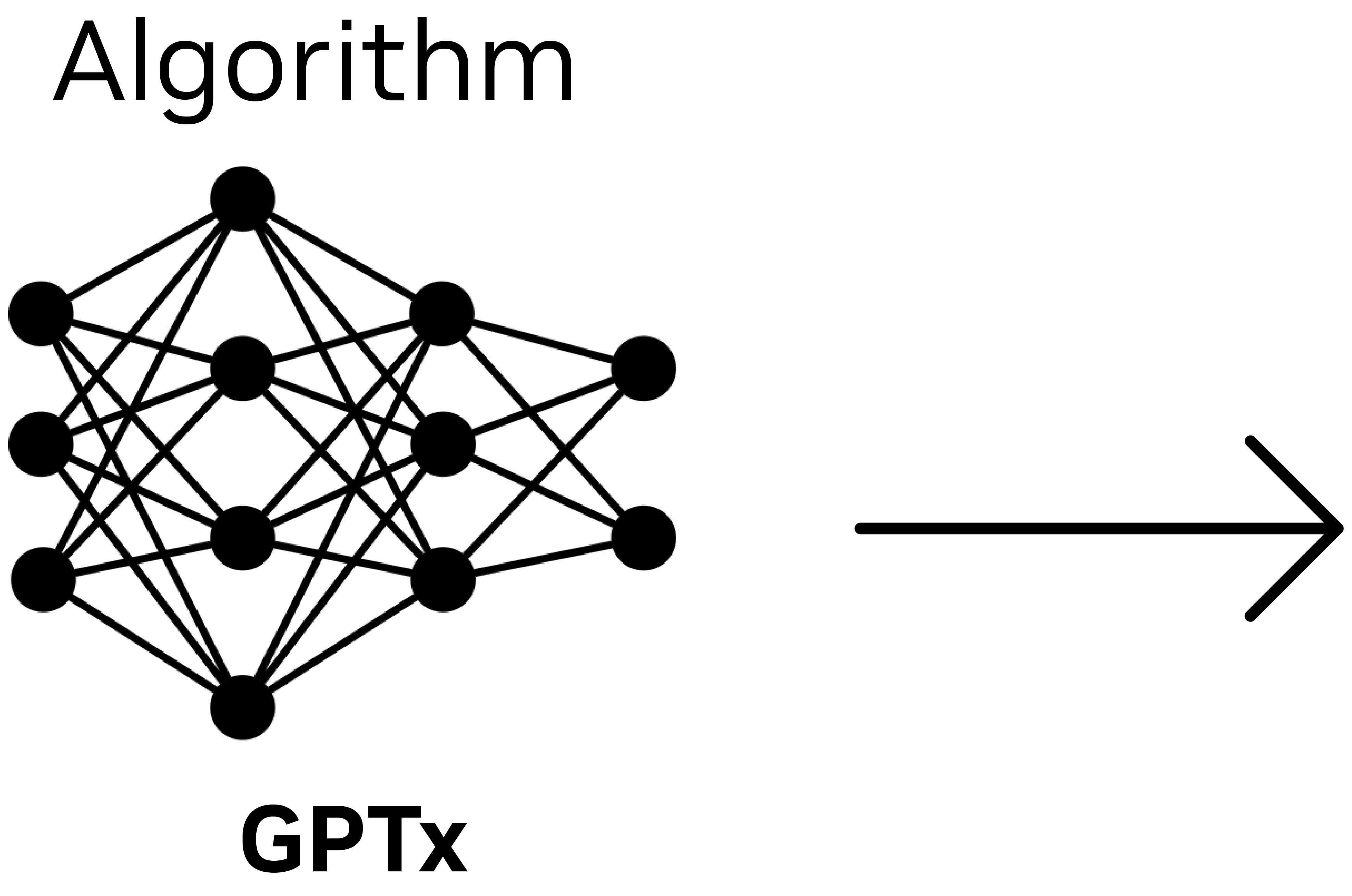
A chain of 6 operations in GPT-6.7b block

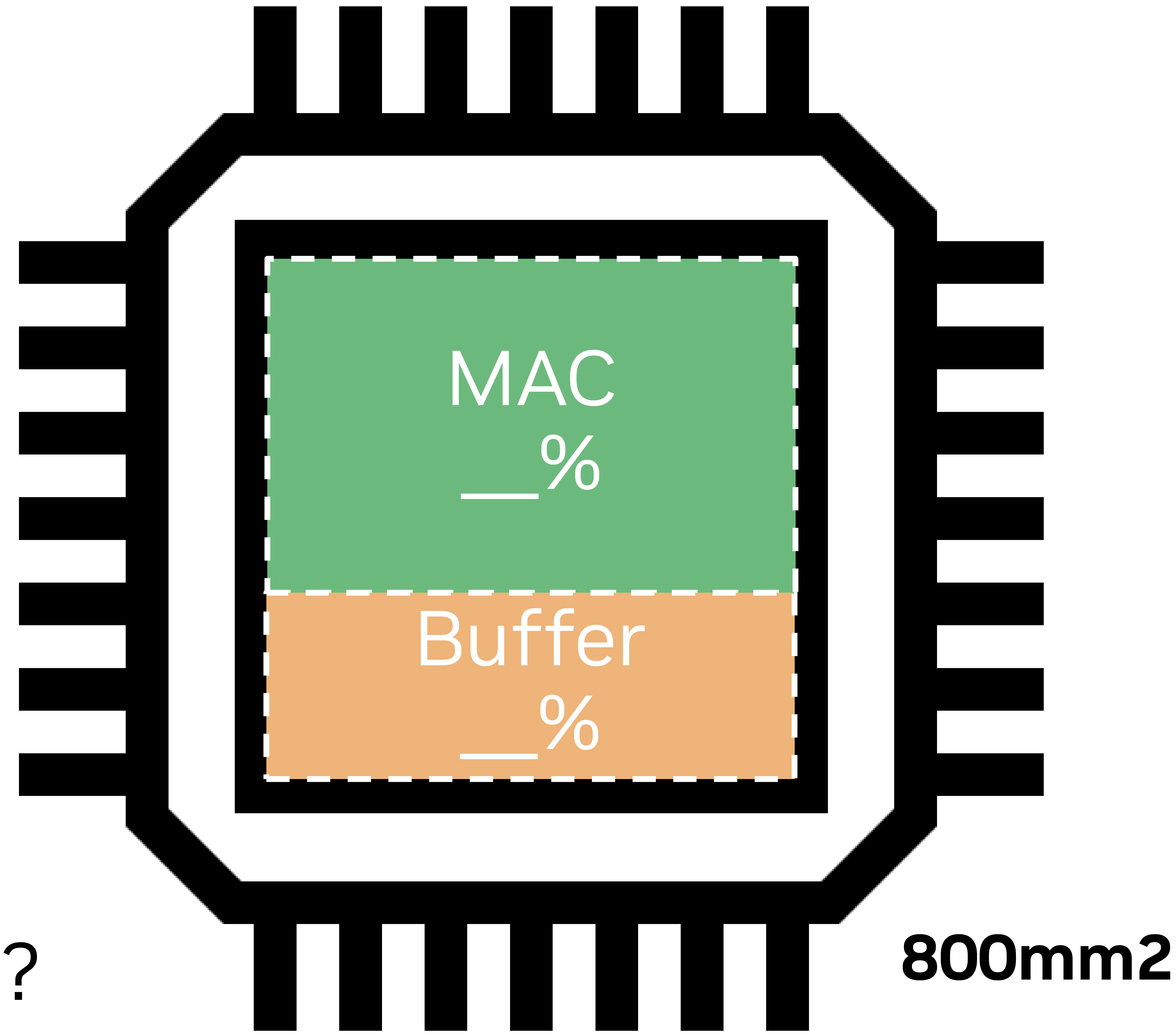


Fusion is effective when buffer size is large

#2: Orojenesis comprehends complex workload optimizations (e.g. fusion)

Motivation: A design challenge



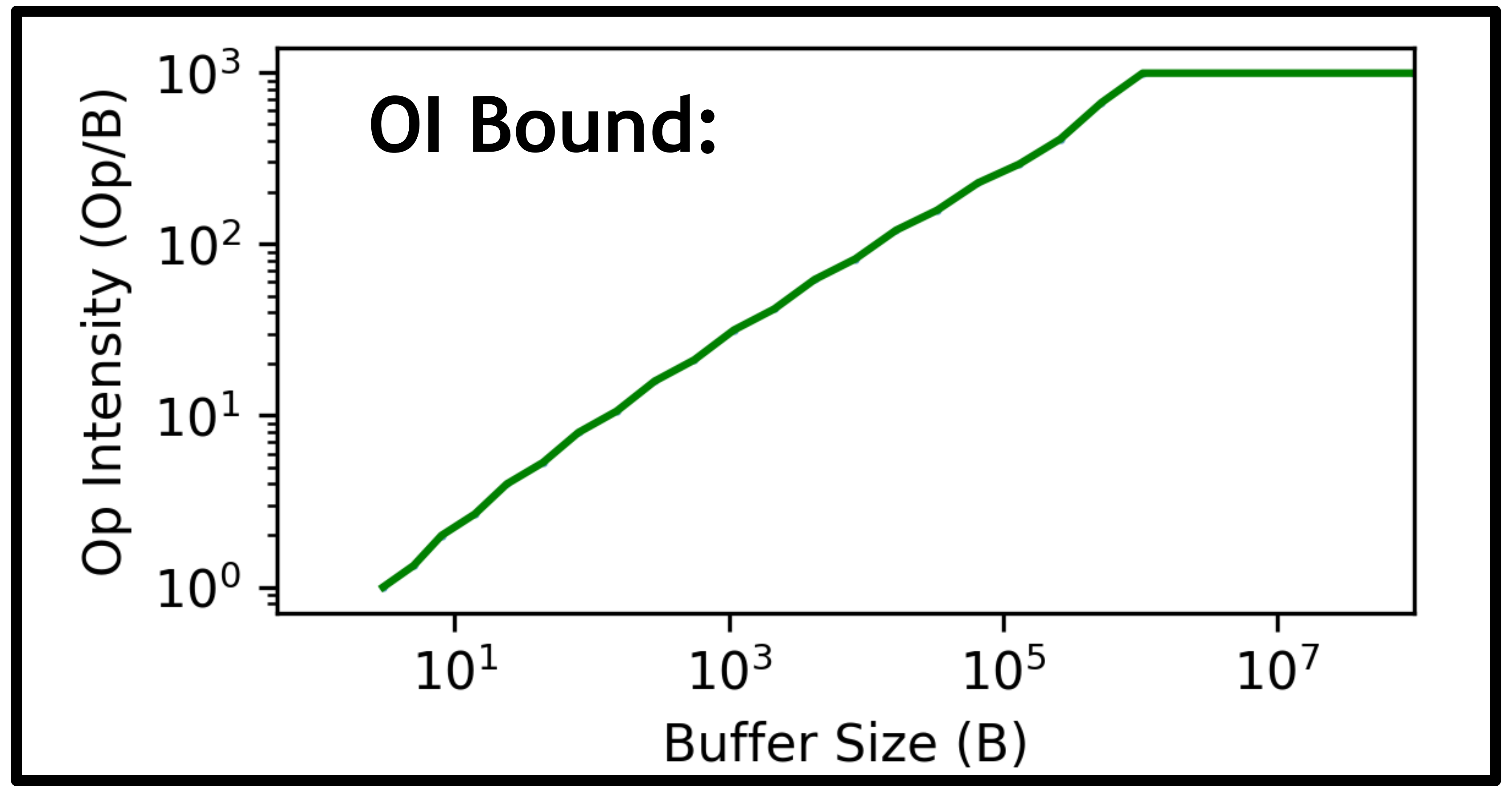


How to provision chip area between storage and compute?

Orojenesis Performance Model

Algo

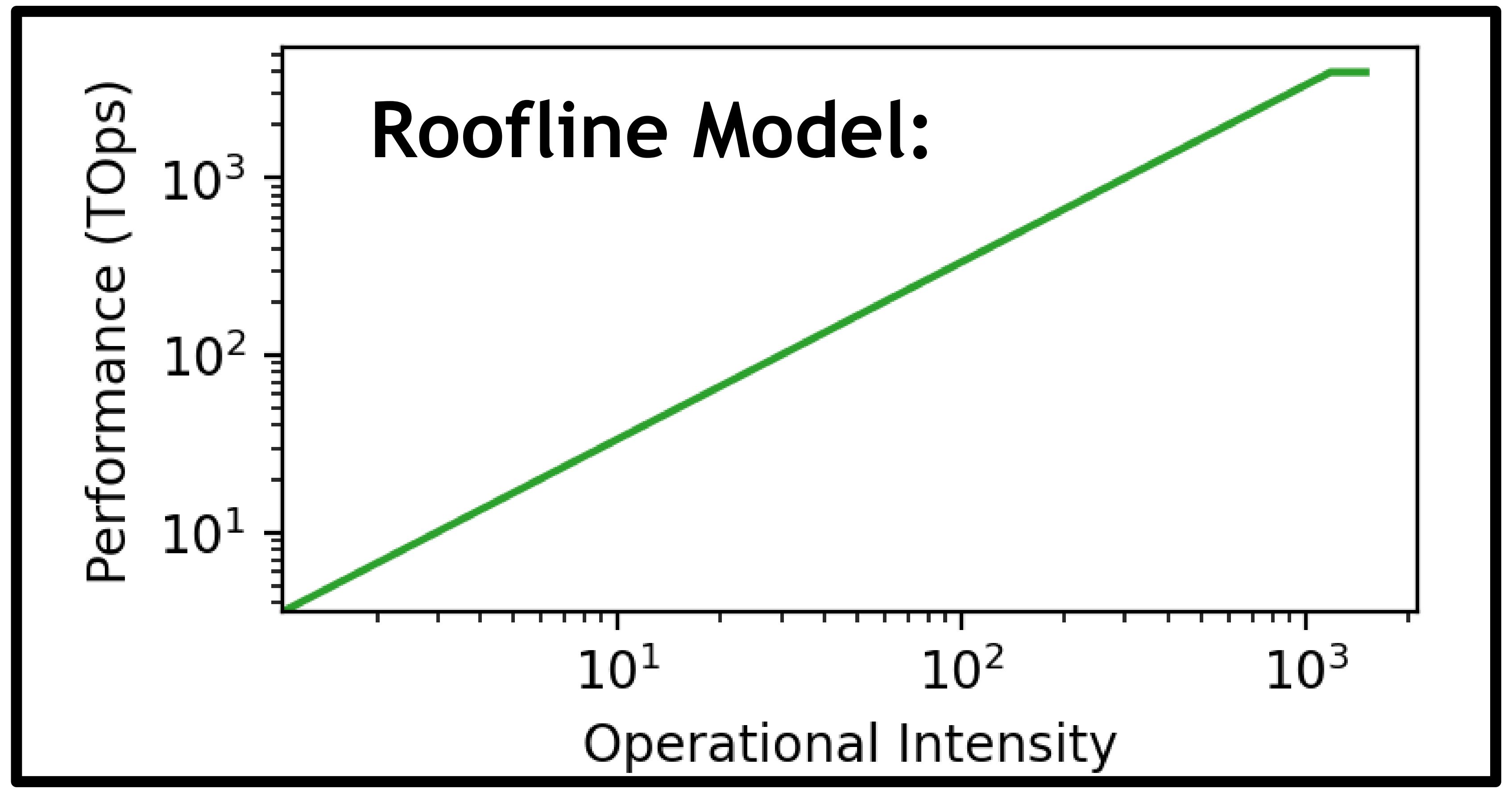
Input:
Buffer Size



Output:
OI

Hardware TOPs and BW

Input:
OI



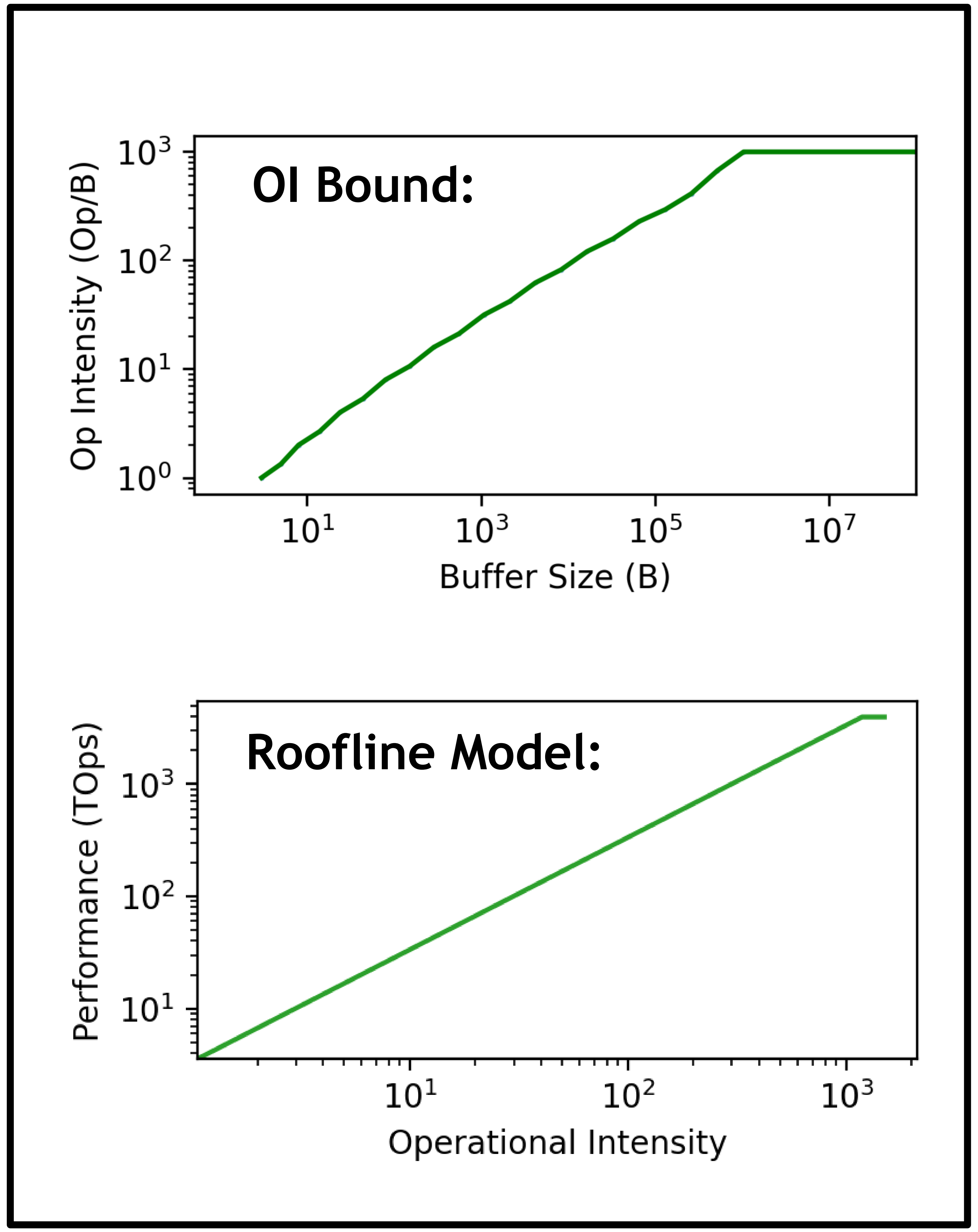
Output:
Performance

Orojenesis Performance Model

Algo

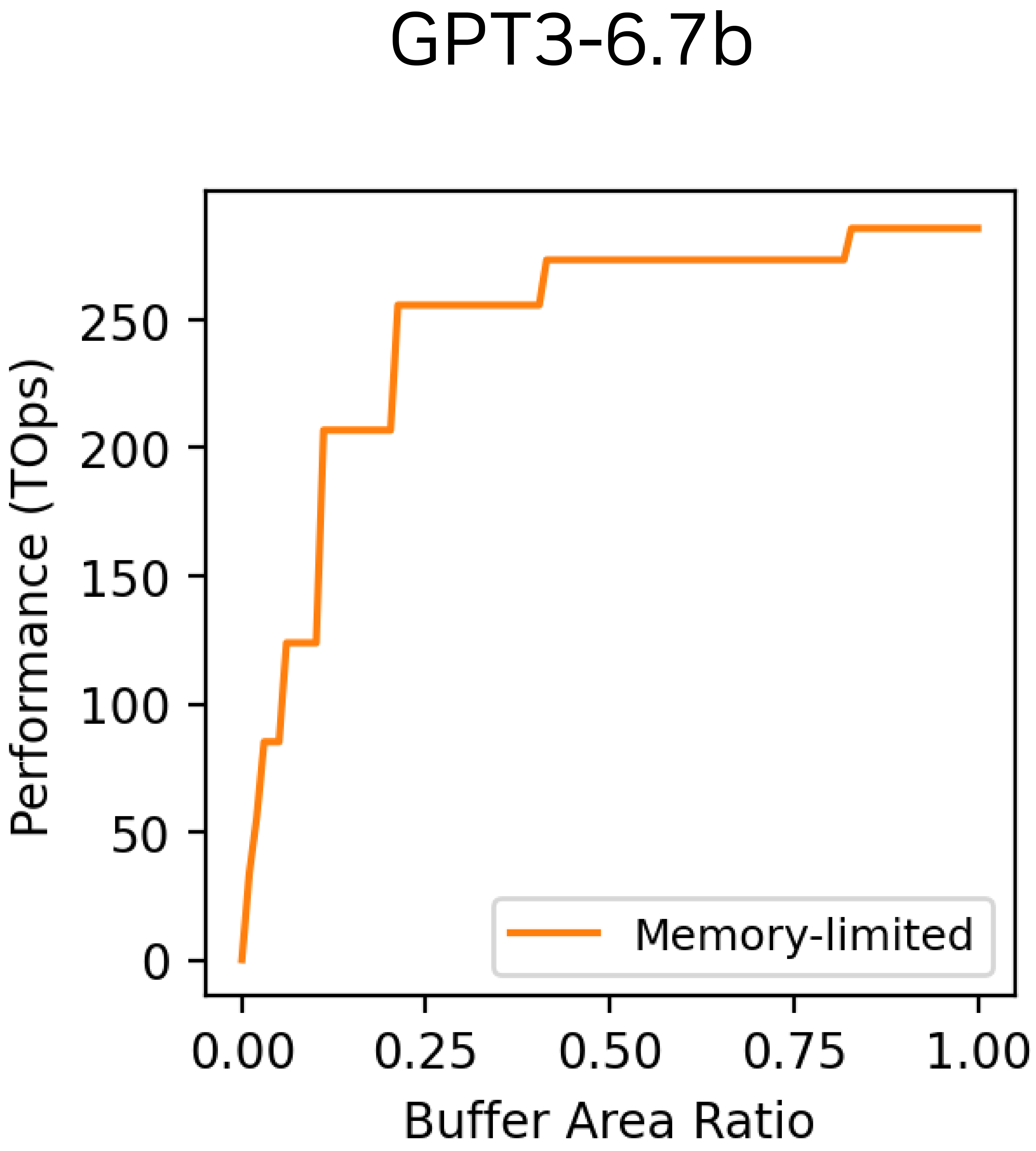
Hardware TOPs and BW

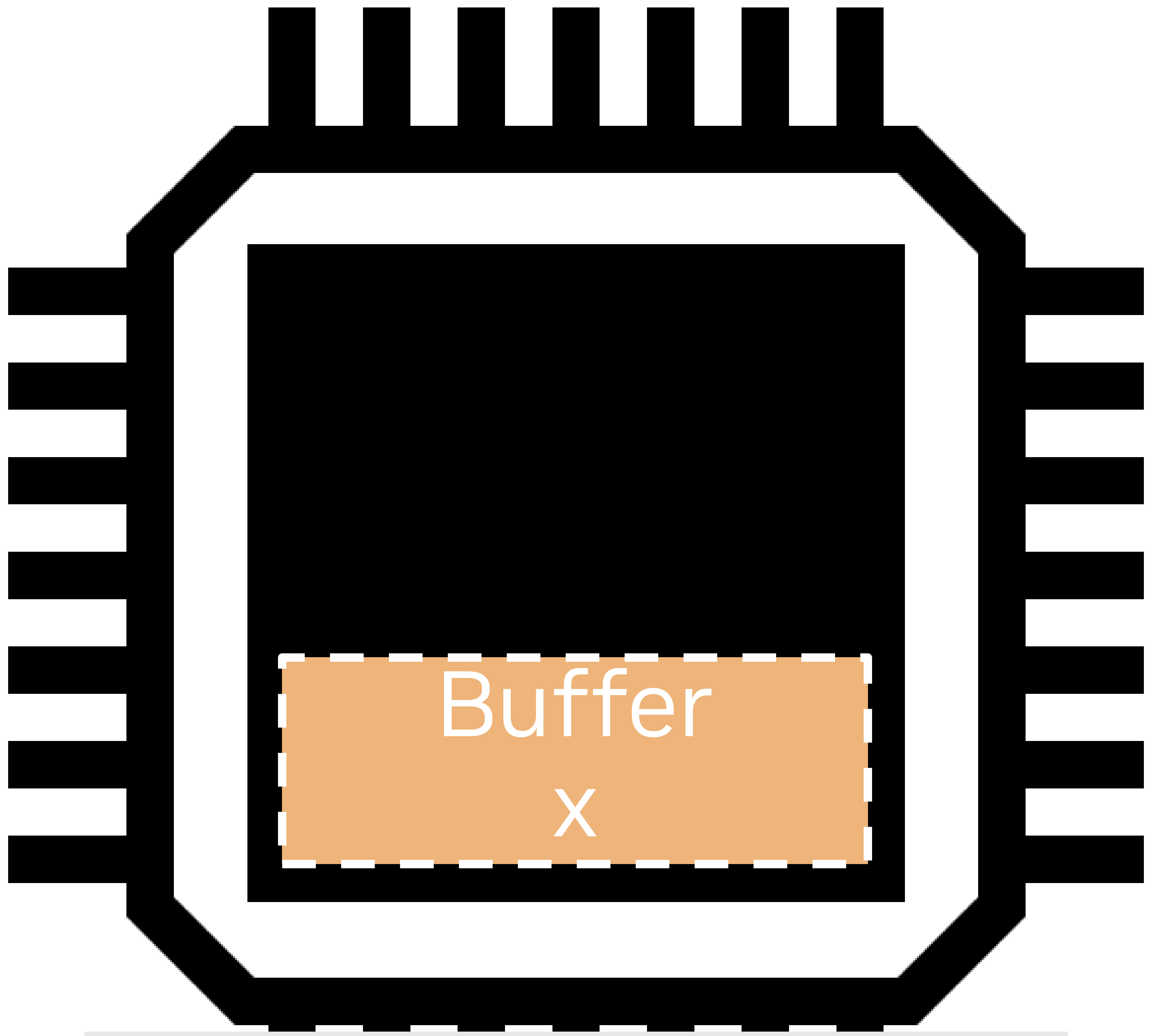
Input:
Buffer Size



Output:
Performance

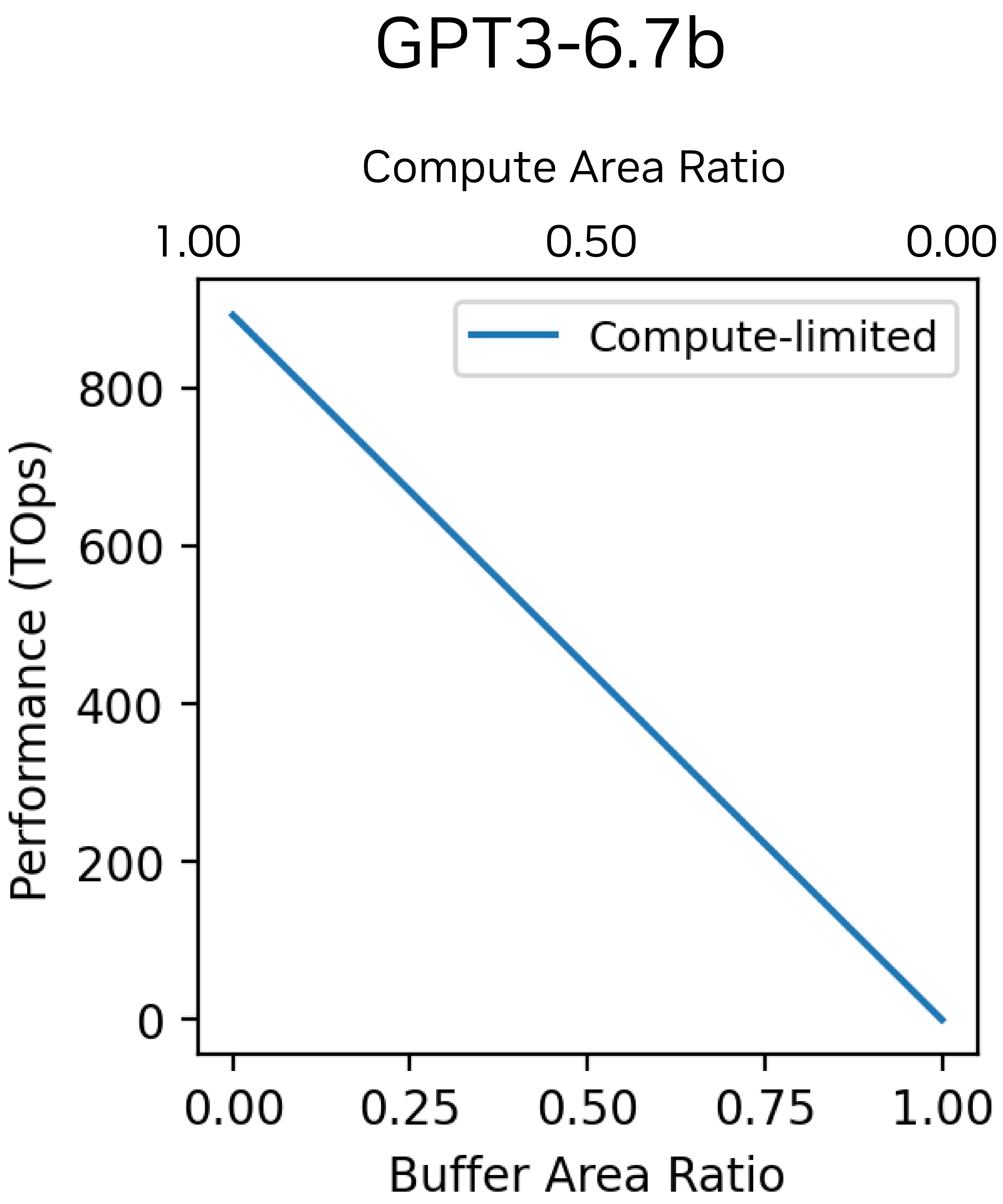
Orojenesis for DSE

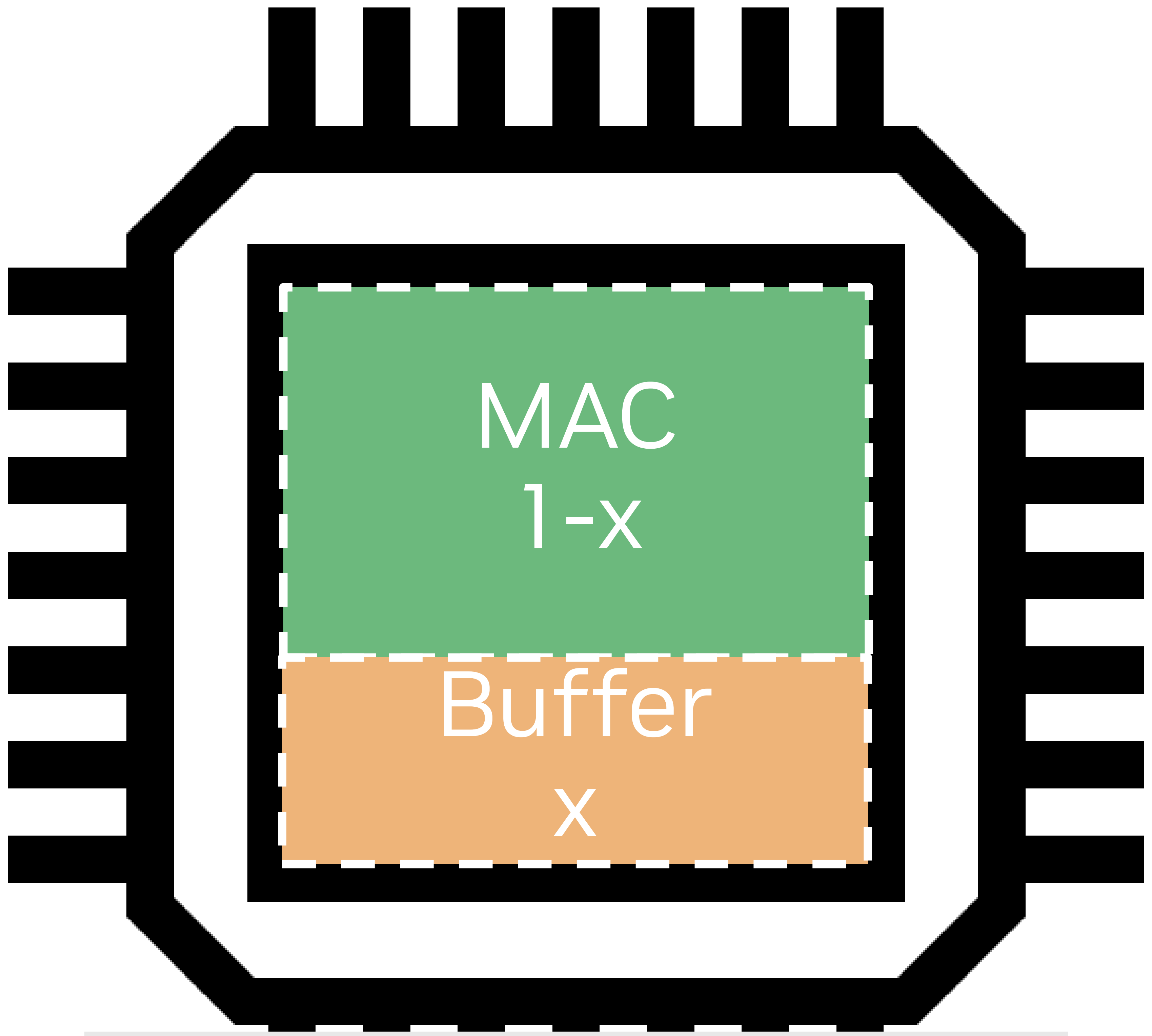




HW Specs:
Total chip area
Area per Byte
Area per MAC
Backing-store BW
Frequency

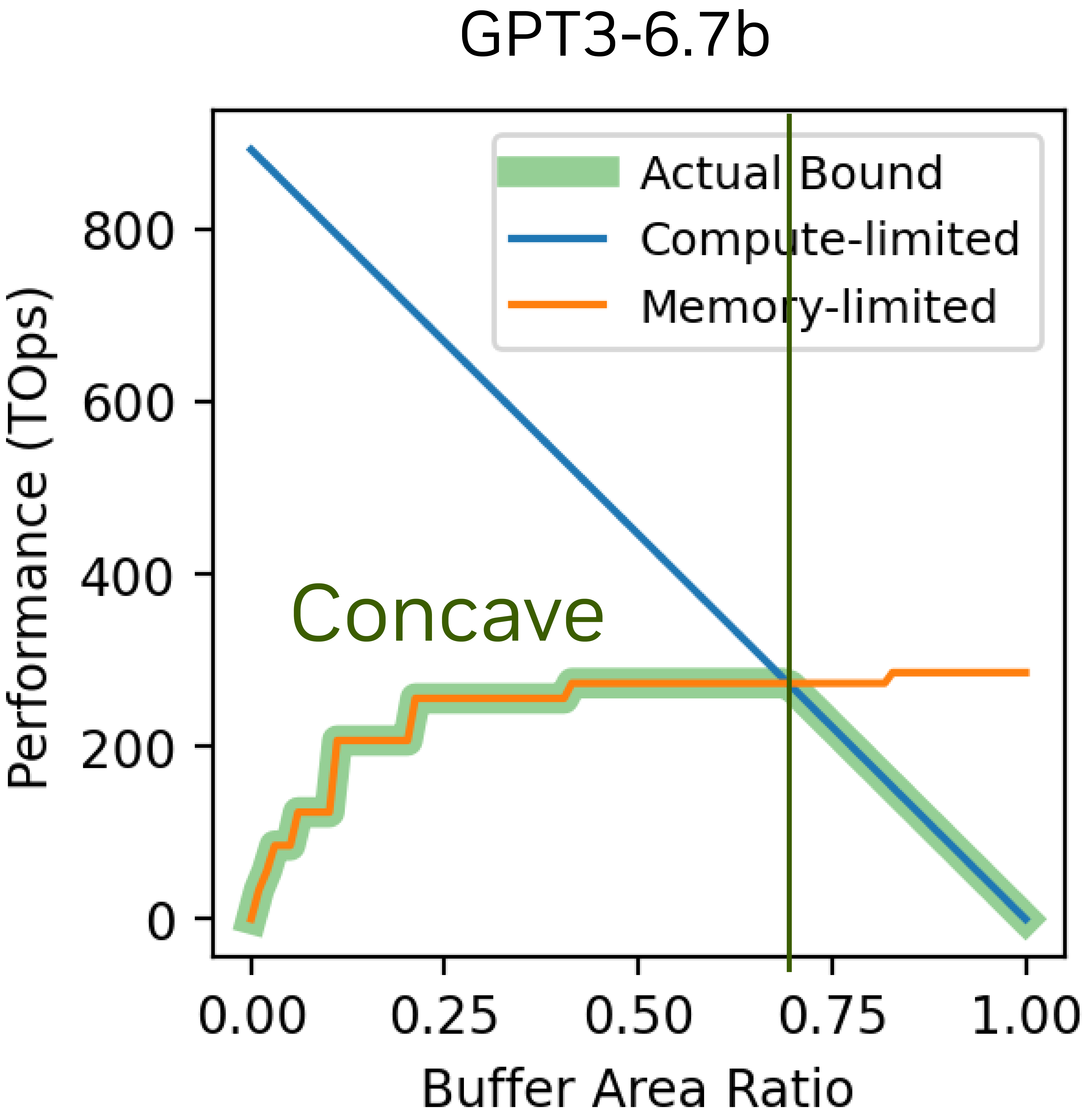
Orojenesis for DSE

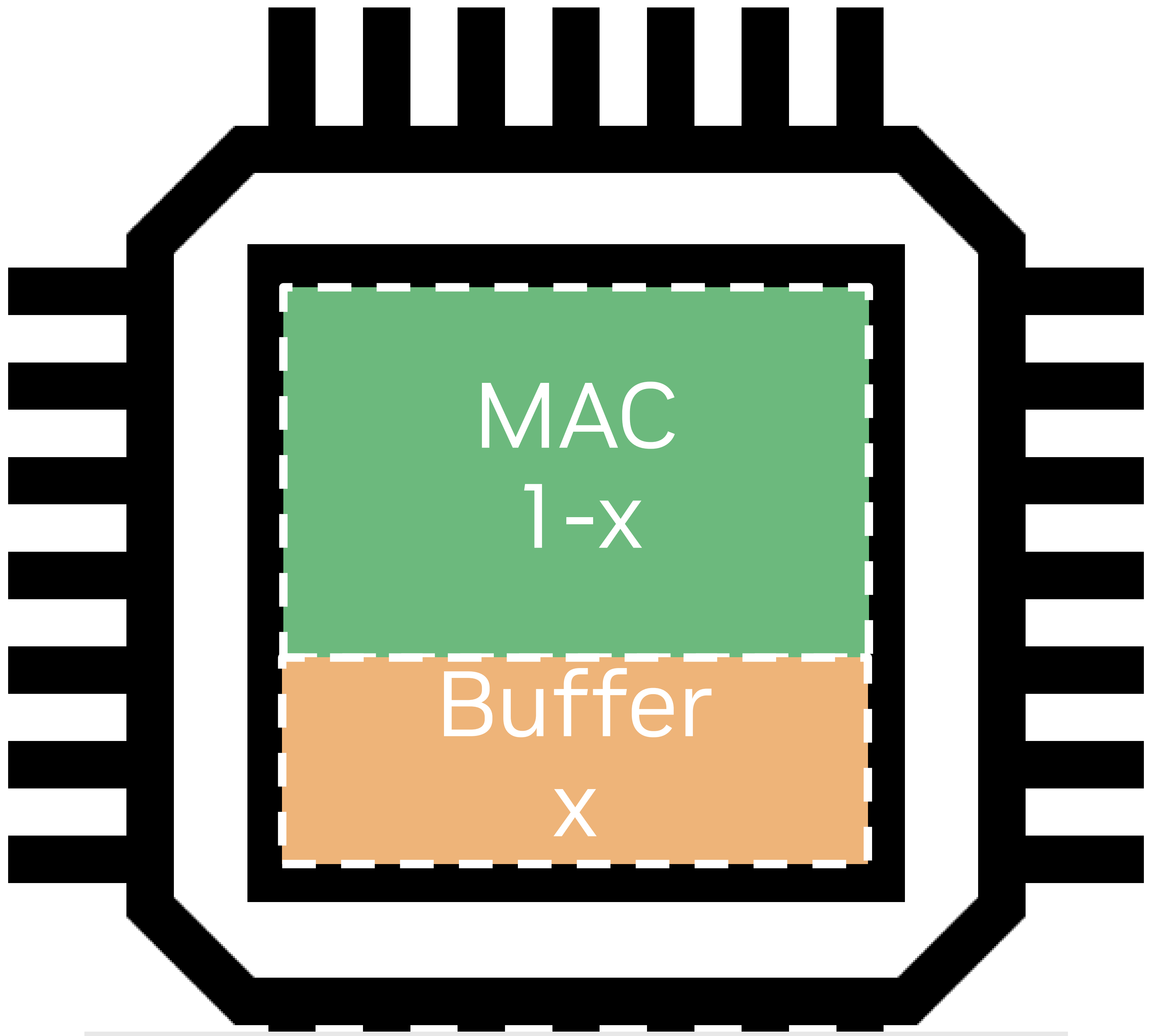




HW Specs:
Total chip area
Area per Byte
Area per MAC
Backing-store BW
Frequency

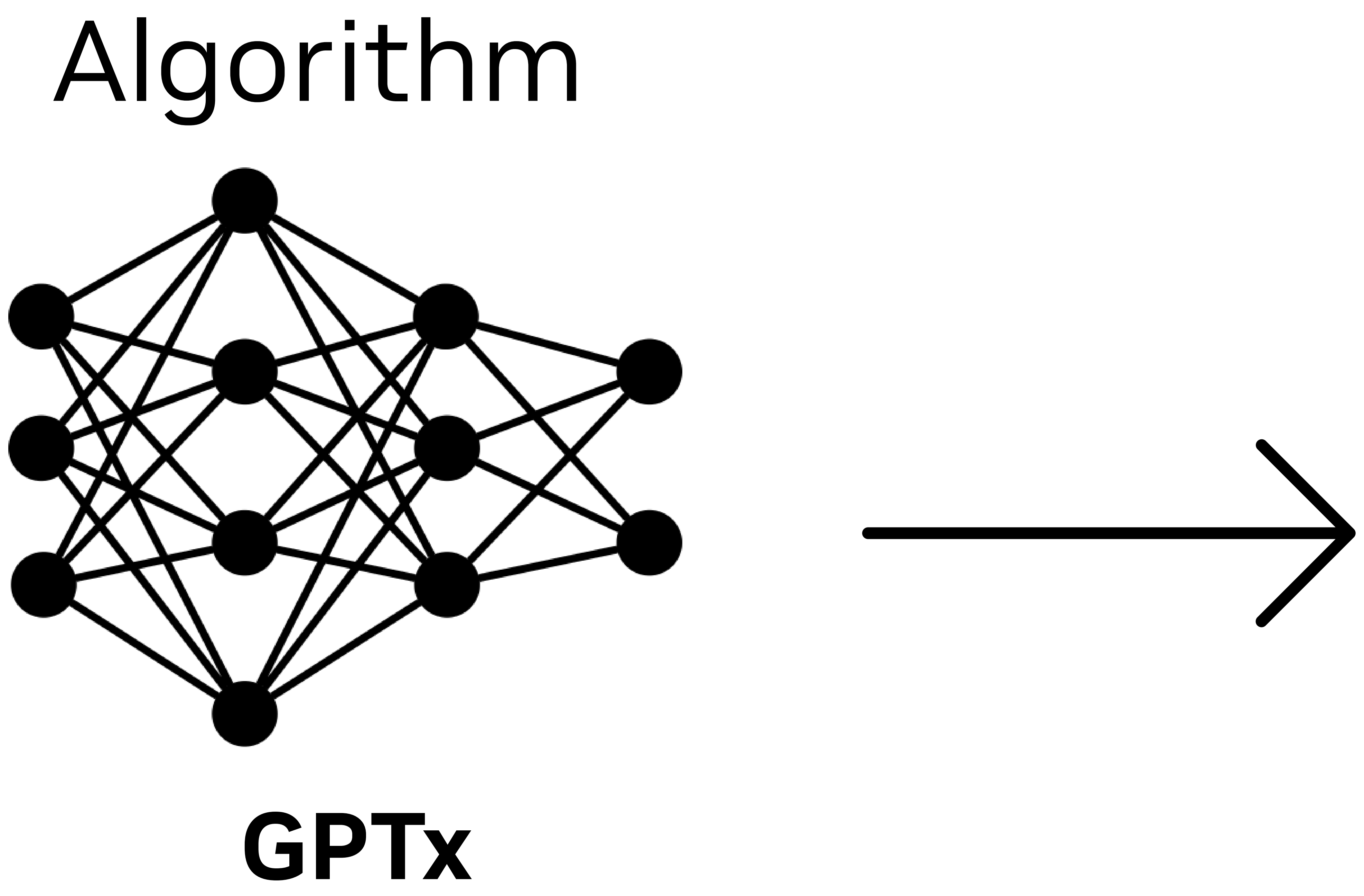
Orojenesis for DSE

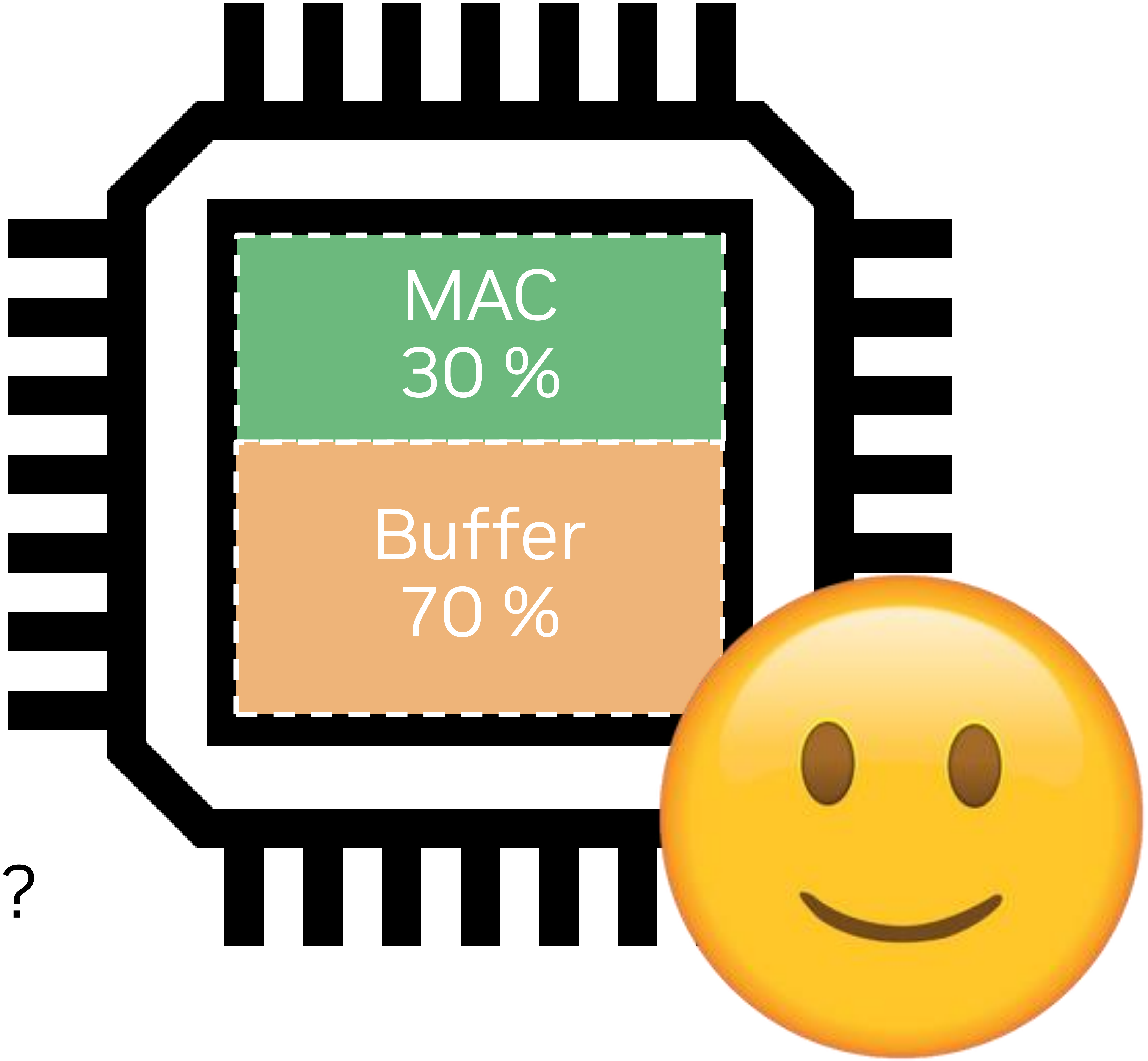




HW Specs:
Total chip area
Area per Byte
Area per MAC
Backing-store BW
Frequency

Motivation: A design challenge





How to provision chip area between storage and compute?

#3: Orojenesis complements the roofline model to provide buffer size suggestions

Orojenesis

- A radically new approach for early-stage architectural DSE
- Offers **visualization** and **insights** for design tradeoffs
- Can be **a powerful addon** to the roofline performance model

Webpage: https://timeloop.csail.mit.edu/orojenesis