G10: Enabling An Efficient Unified GPU Memory and Storage Architecture with Smart Tensor Migrations

Haoyang Zhang*, Yirui Eric Zhou*, Yuqi Xue, Yiqi Liu, Jian Huang

Systems Platform Research Group

*Co-primary authors.

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Artifacts Available V1.1 · Artifacts Evaluated Functional V1.1 · Results Reproduced V1.1 · Artifacts Evaluated Reusable V1.1

Large DNN Workloads Are Hungry for Memory

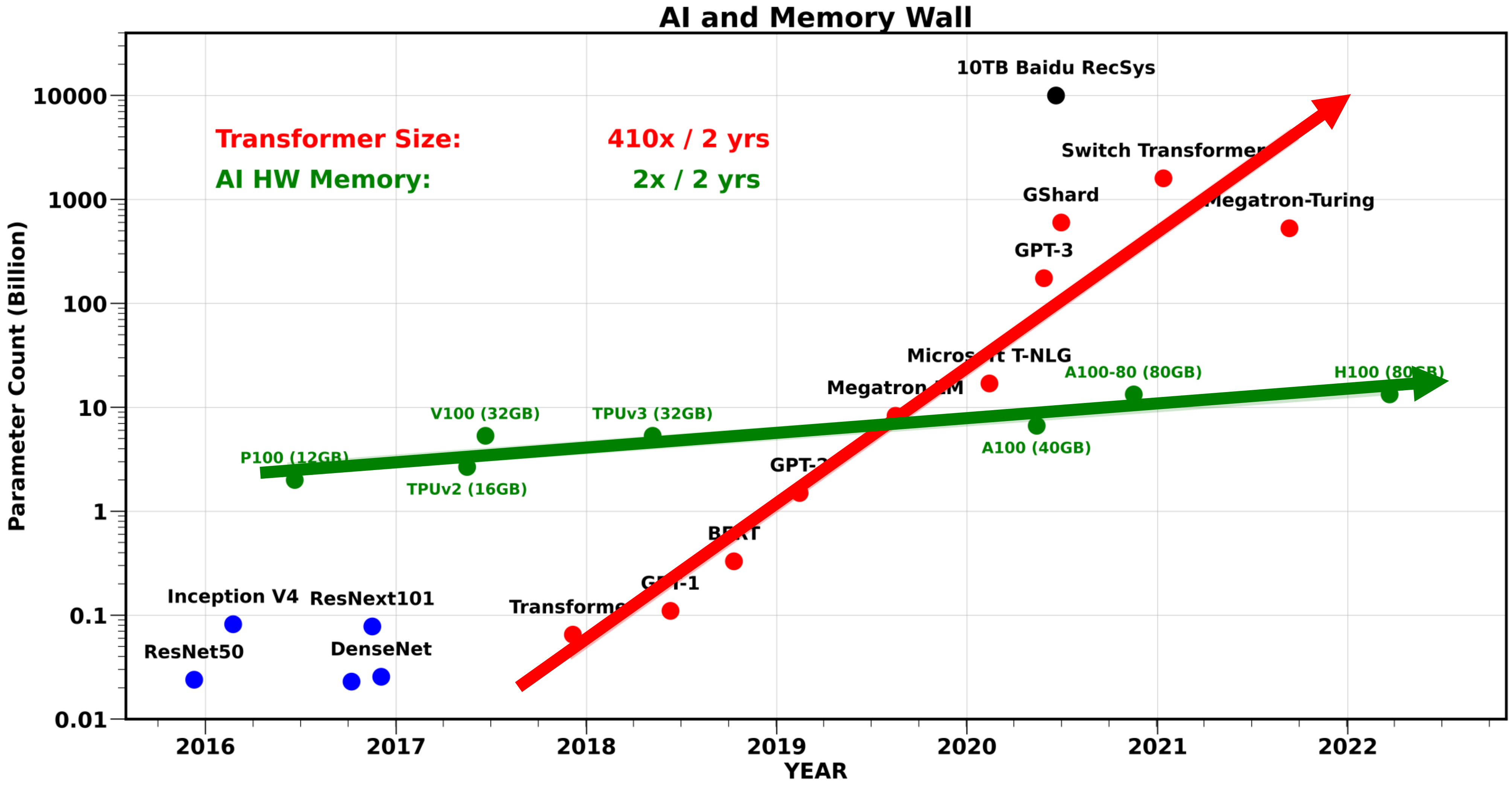
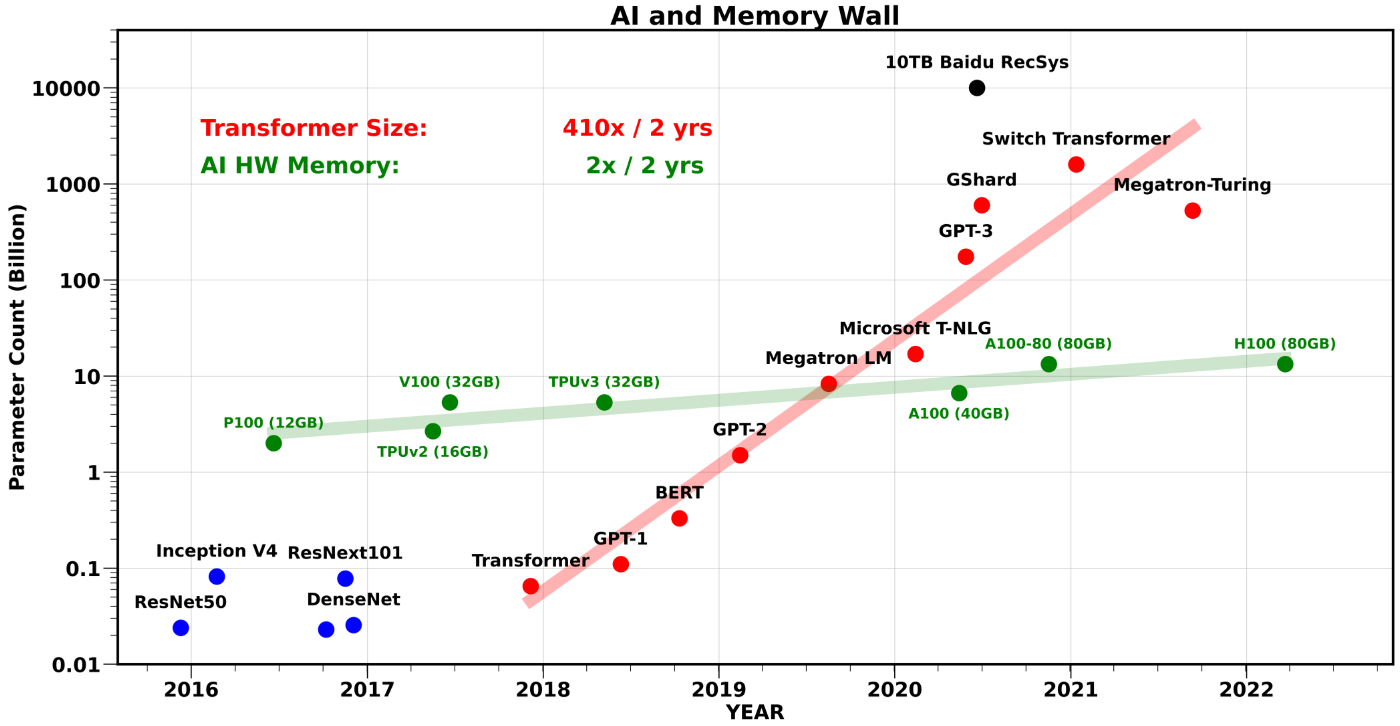
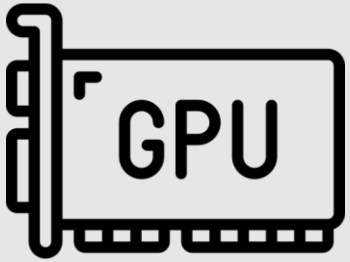


Photo credit: https://github.com/amirgholami/ai_and_memory_wall

Large DNN Workloads Are Hungry for Memory

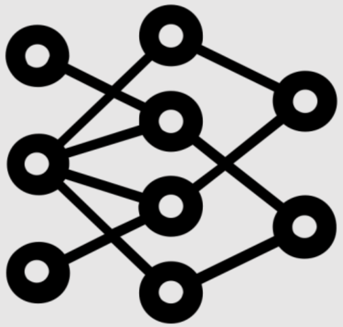




Tens of GBs

On-board Memory

vs.

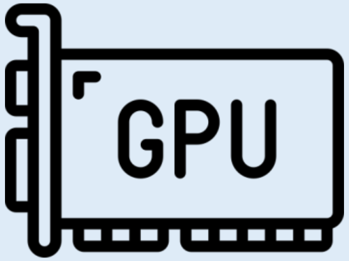


Tens of TBs

DNN Model Size

Photo credit: https://github.com/amirgholami/ai_and_memory_wall

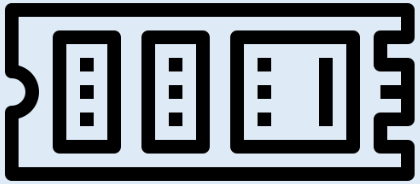
Expanding GPU Memory with Flash Memory



Tens of GBs

On-board Memory

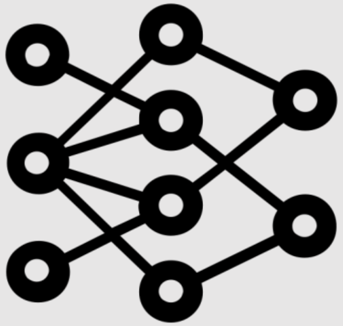
+



Tens of TBs

Flash Memory

---------------VS.-----------------



Tens of TBs

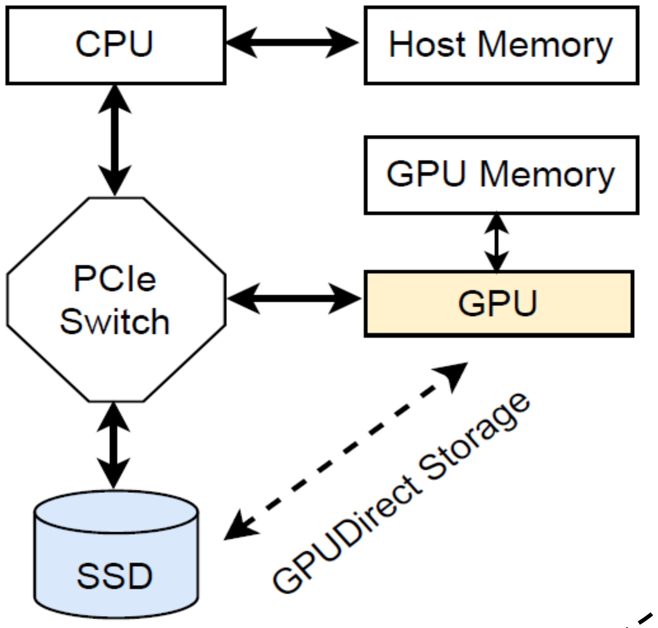
DNN Model Size

Large Capacity

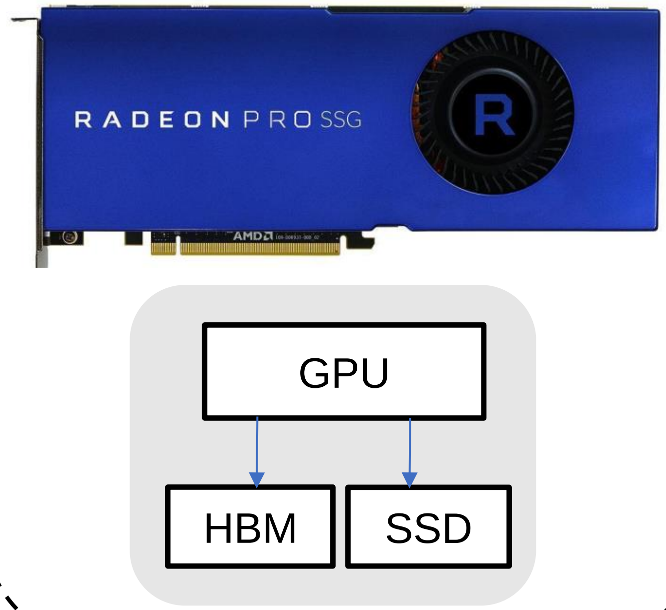
Low Cost

State-of-the-Art Solutions Are Not Efficient Enough

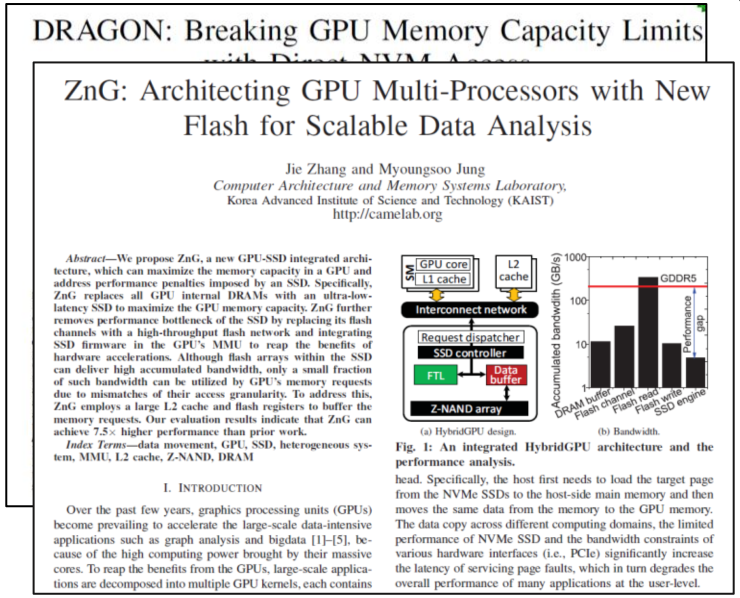
GPUDirect Storage



AMD SSG



Academic Research



DRAGON: Breaking GPU Memory Capacity Limits

ZnG: Architecting GPU Multi-Processors with New Flash for Scalable Data Analysis

Jie Zhang and Myoungsoo Jung
Computer Architecture and Memory Systems Laboratory,
Korea Advanced Institute of Science and Technology (KAIST)
http://camelab.org

Abstract—We propose ZnG, a new GPU-SSD integrated architecture, which can maximize the memory capacity in a GPU and address performance penalties imposed by an SSD. Specifically, ZnG replaces all GPU internal DRAMs with an ultra-low-latency SSD to maximize the GPU memory capacity. ZnG further removes performance bottleneck of the SSD by replacing its flash channels with a high-throughput flash network and integrating SSD firmware in the GPU's MMU to reap the benefits of hardware accelerations. Although flash arrays within the SSD can deliver high accumulated bandwidth, only a small fraction of such bandwidth can be utilized by GPU's memory requests due to mismatches of their access granularity. To address this, ZnG employs a large L2 cache and flash registers to buffer the memory requests. Our evaluation results indicate that ZnG can achieve 7.5× higher performance than prior work.

Index Terms—data movement, GPU, SSD, heterogeneous system, MMU, L2 cache, Z-NAND, DRAM

(a) HybridGPU design. (b) Bandwidth.

Fig. 1: An integrated HybridGPU architecture and the performance analysis.

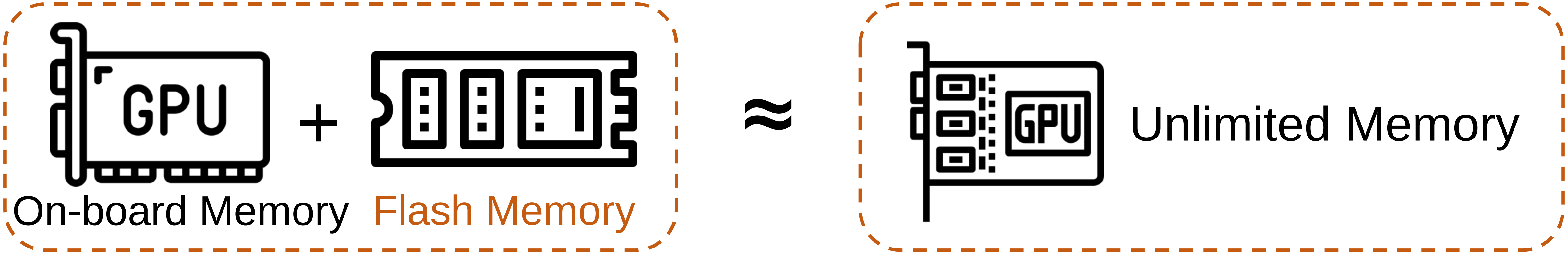
I. INTRODUCTION

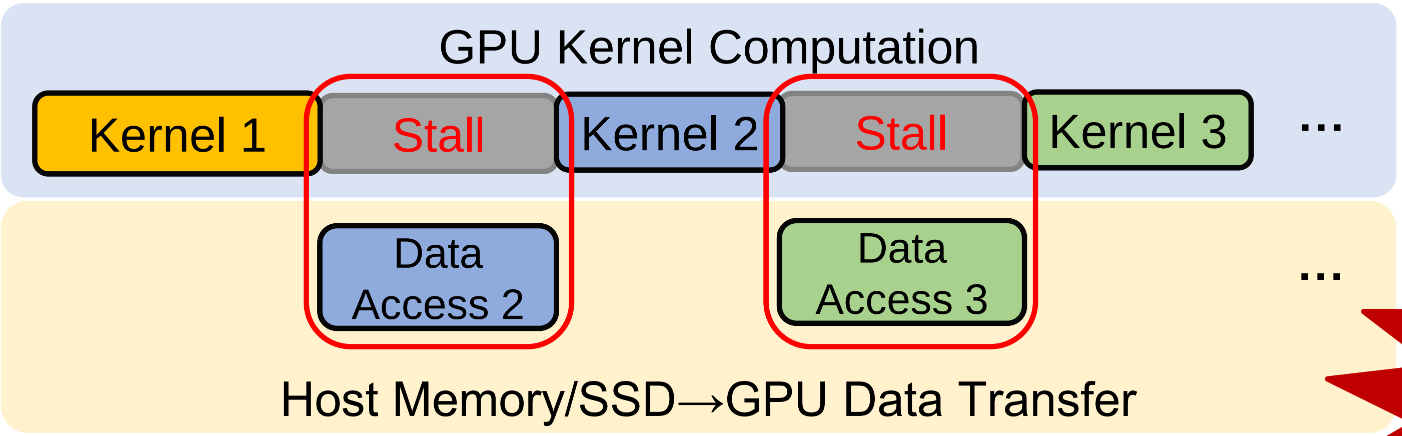
Over the past few years, graphics processing units (GPUs) become prevailing to accelerate the large-scale data-intensive applications such as graph analysis and bigdata [1]–[5], because of the high computing power brought by their massive cores. To reap the benefits from the GPUs, large-scale applications are decomposed into multiple GPU kernels, each contains

head. Specifically, the host first needs to load the target page from the NVMe SSDs to the host-side main memory and then moves the same data from the memory to the GPU memory. The data copy across different computing domains, the limited performance of NVMe SSD and the bandwidth constraints of various hardware interfaces (i.e., PCIe) significantly increase the latency of servicing page faults, which in turn degrades the overall performance of many applications at the user-level.

Flash Bandwidth is the Bottleneck!

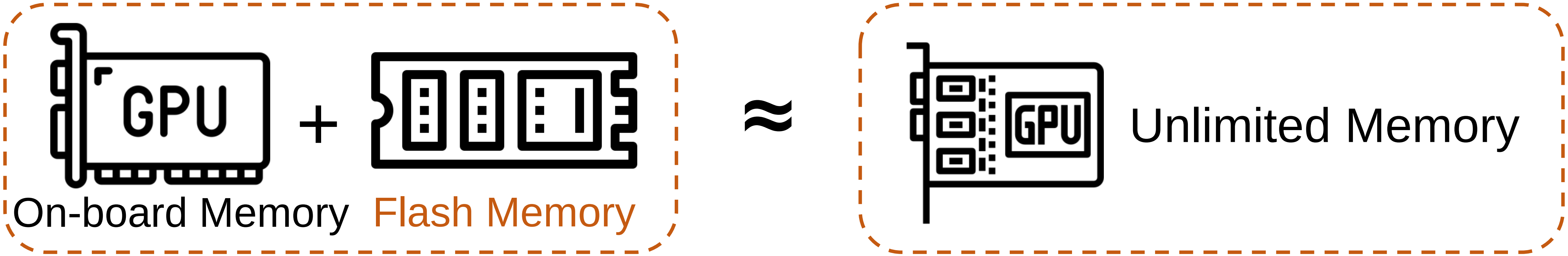
Goal: Expand GPU Memory While Achieving Near-Ideal Performance

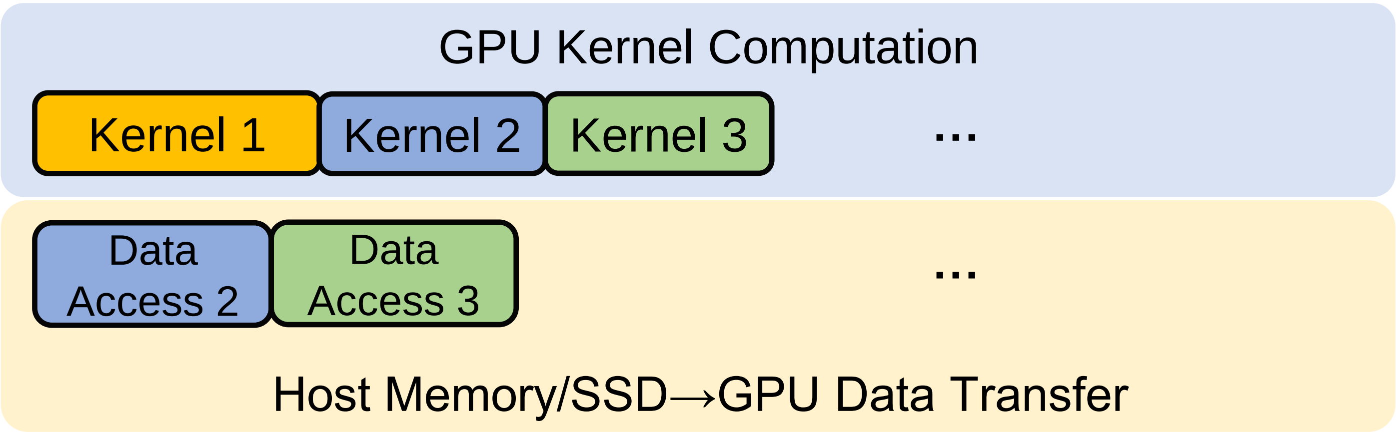




Overlap Data I/O and Computation

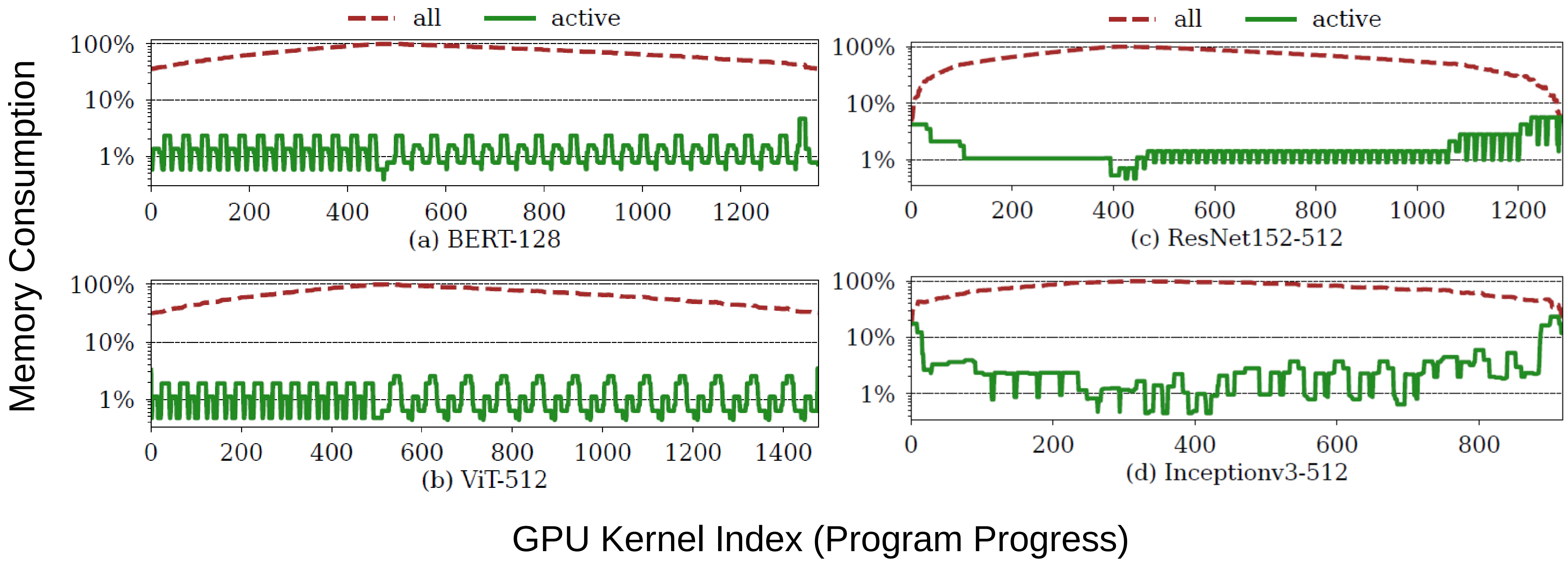
Goal: Expand GPU Memory While Achieving Near-Ideal Performance



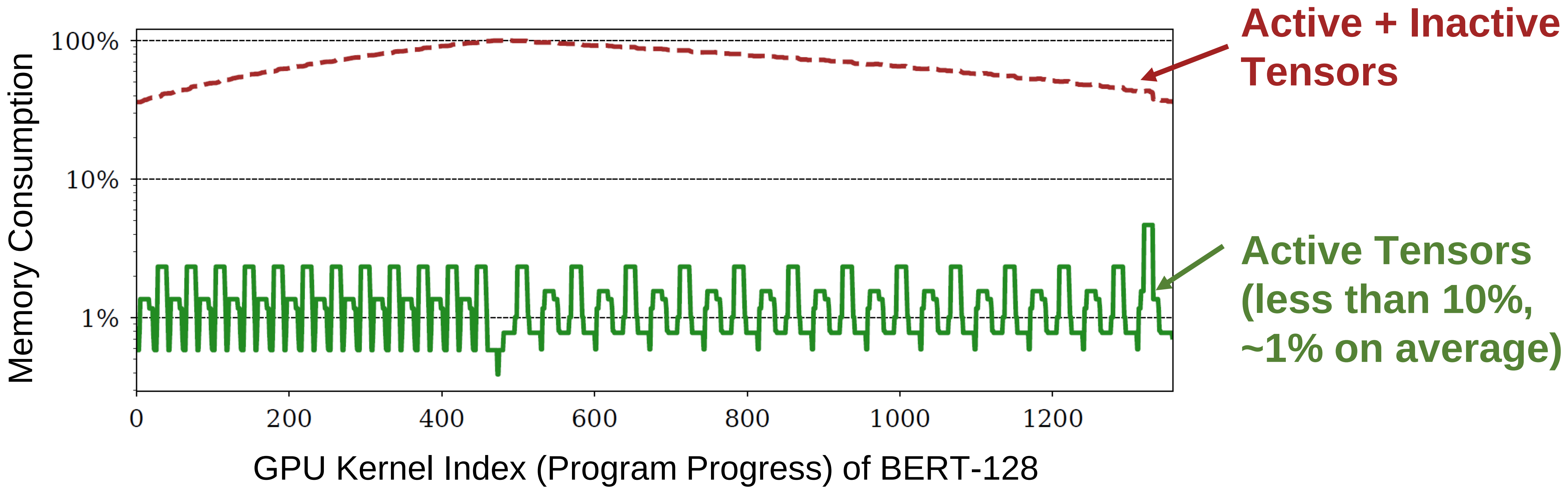


Overlap Data I/O and Computation → Achieve Near-Ideal Performance with Slow Memory

Observation 1: Active Tensors Require Only A Small Portion of GPU Memory

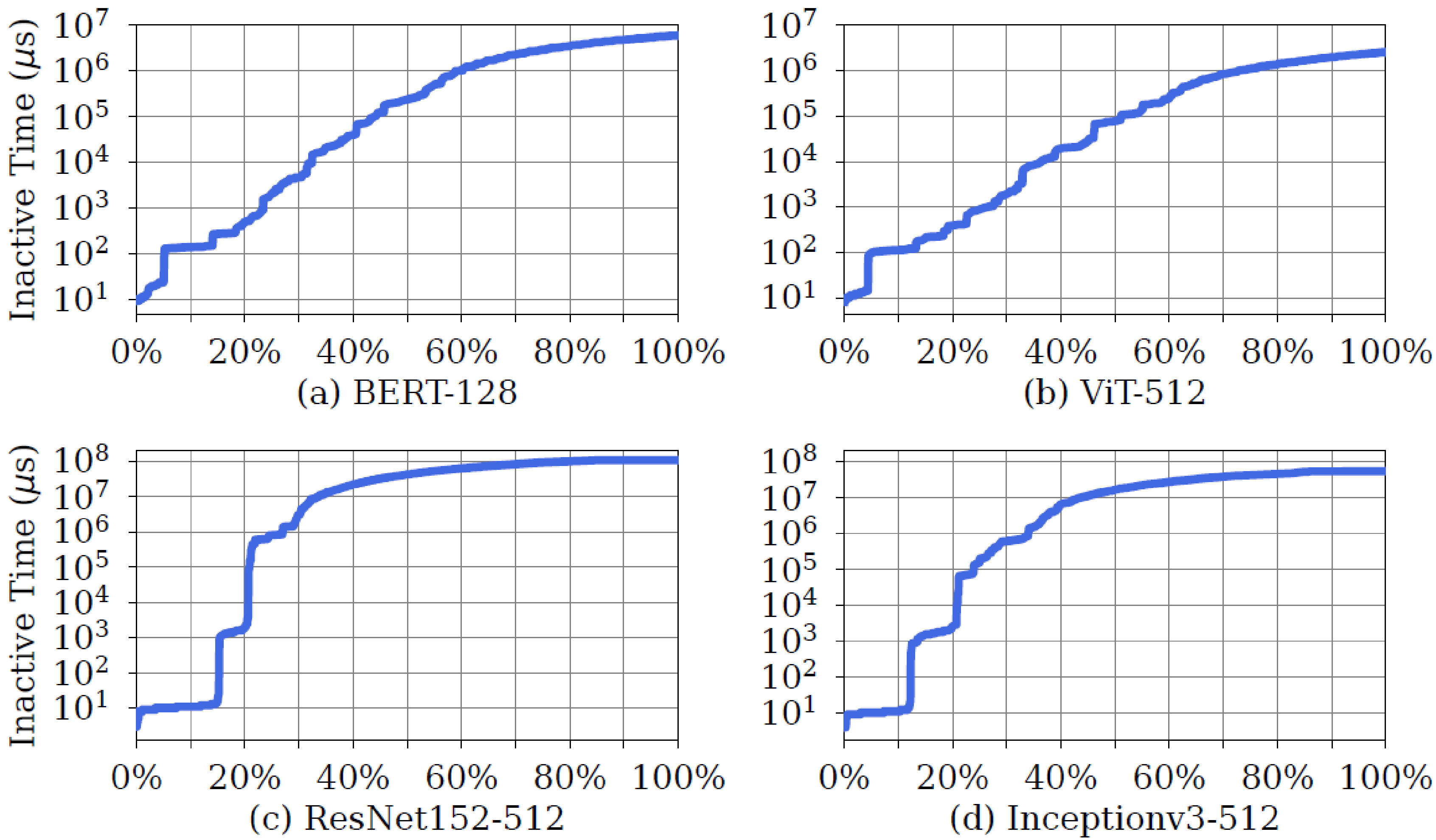


Observation 1: Active Tensors Require Only A Small Portion of GPU Memory



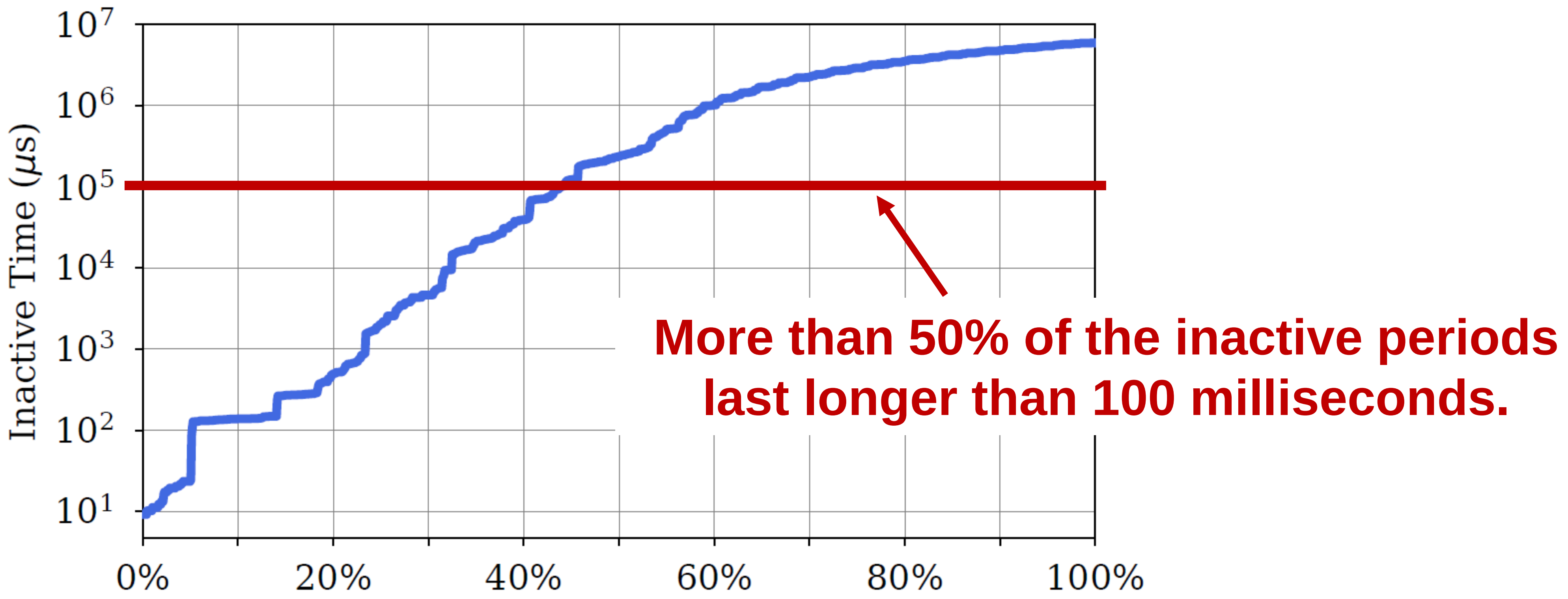
Most Tensors Are Inactive and Can Be Swapped Out During Training

Observation 2: Many Tensors Are Unused for A Long Time



Distribution (CDF) of tensor inactive period lengths

Observation 2: Many Tensors Are Unused for A Long Time



Distribution (CDF) of tensor inactive period lengths for BERT-128

Many Tensors Can Be Safely Swapped Out to Slow Memory

G10: Break the GPU Memory Wall with Smart Tensor Migrations

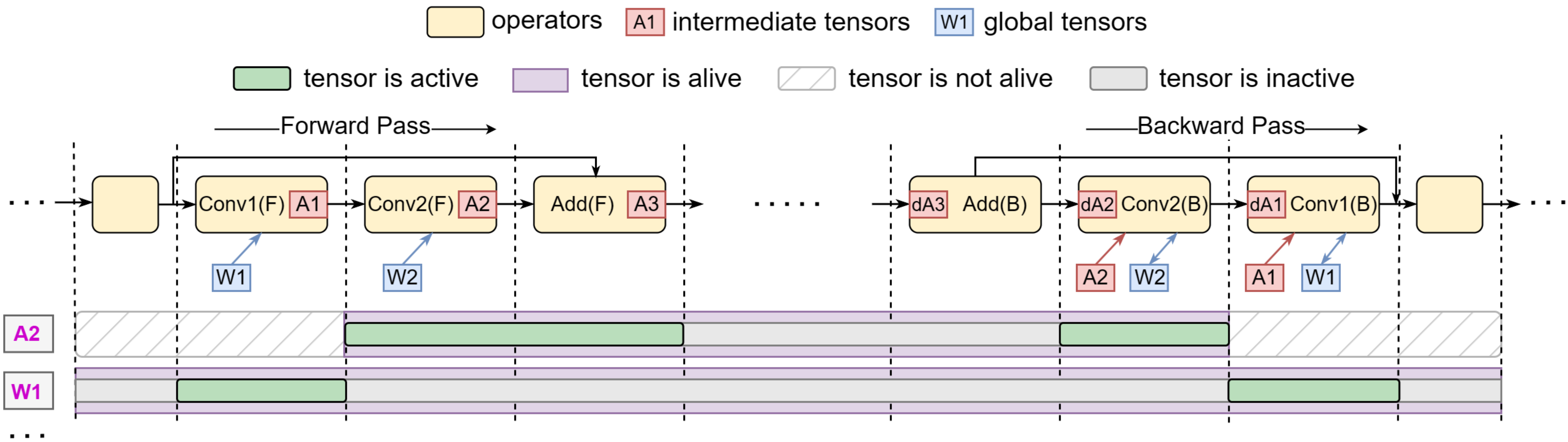
Extract the Semantic Knowledge of Tensors in DNN Models

Schedule Tensor Migrations using Semantic Knowledge

Simplify the Heterogeneous Memory Management for GPU

G10

Tensor Liveness Analysis with ML Compiler Support



With offline compile-time profiling, we can:

1. Estimate the **active time** of each tensor
2. Estimate the **lifetime** of each tensor
3. Estimate the **inactive periods** of each tensor

Tensor Liveness Analysis with ML Compiler Support

operators A1 intermediate tensors W1 global tensors

tensor is active tensor is alive tensor is not alive tensor is inactive

Semantic Knowledge of A DNN Model

Inactive Tensor Table

Index ID	Tensor ID	Size	Start Kernel ID	End Kernel ID	Inactive Time
1	35	500MB	5	920	151 millisecs
2	36	2GB	2	924	153 millisecs
3	37	1GB	6	26	7 millisecs

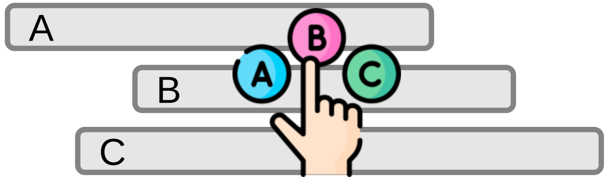
⋮

Kernel Time Table

Kernel ID	Estimated Exe. Time
1	1.0 millisecs
2	2.6 millisecs
3	0.01 millisecs

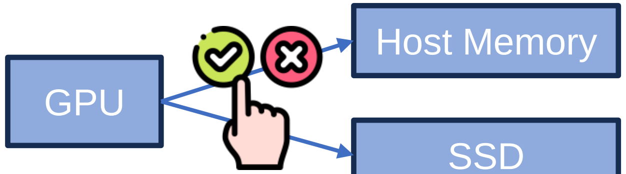
⋮

Enabling Smart Tensor Migrations with Rich Semantic Knowledge



What

Identify the Most Beneficial Inactive Tensor

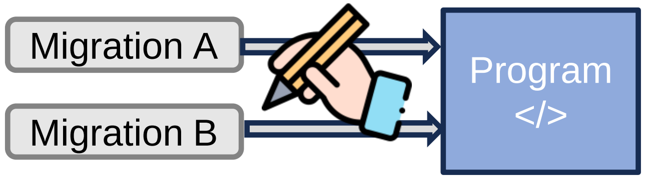


Where

Decide Eviction Destination

When

Decide the Best Time for Migrations

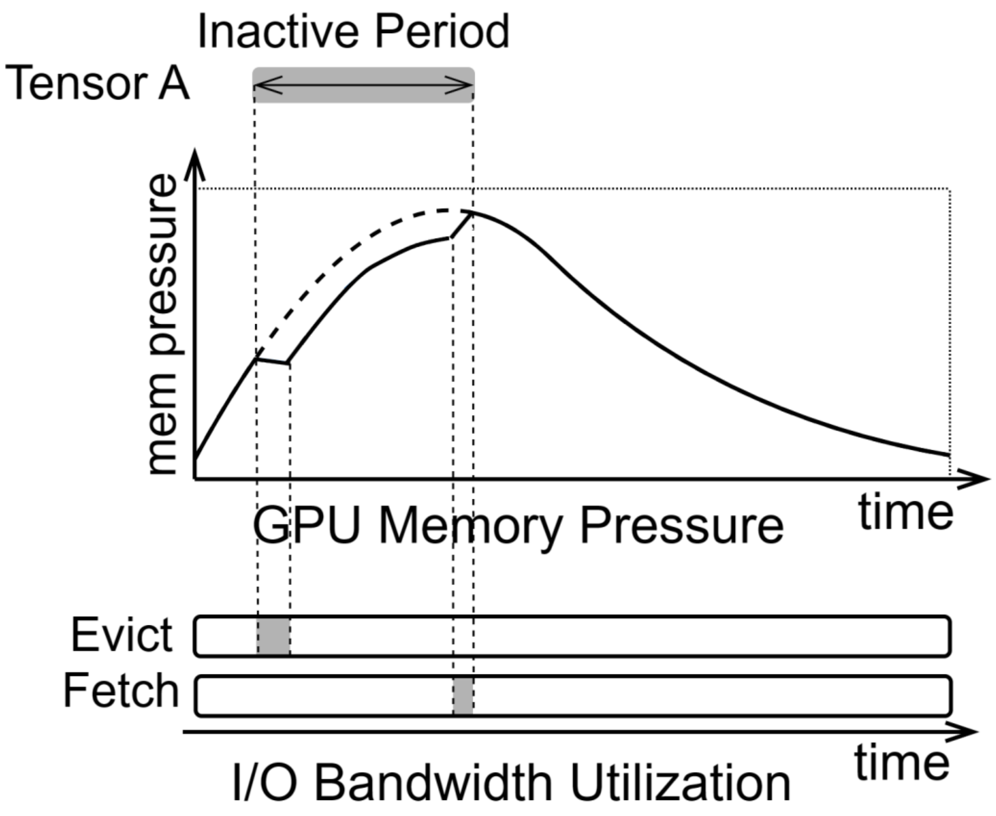


How

Instrument GPU Program with Migration Instructions

Deciding the Most Beneficial Tensor is a Dynamic Optimization Problem

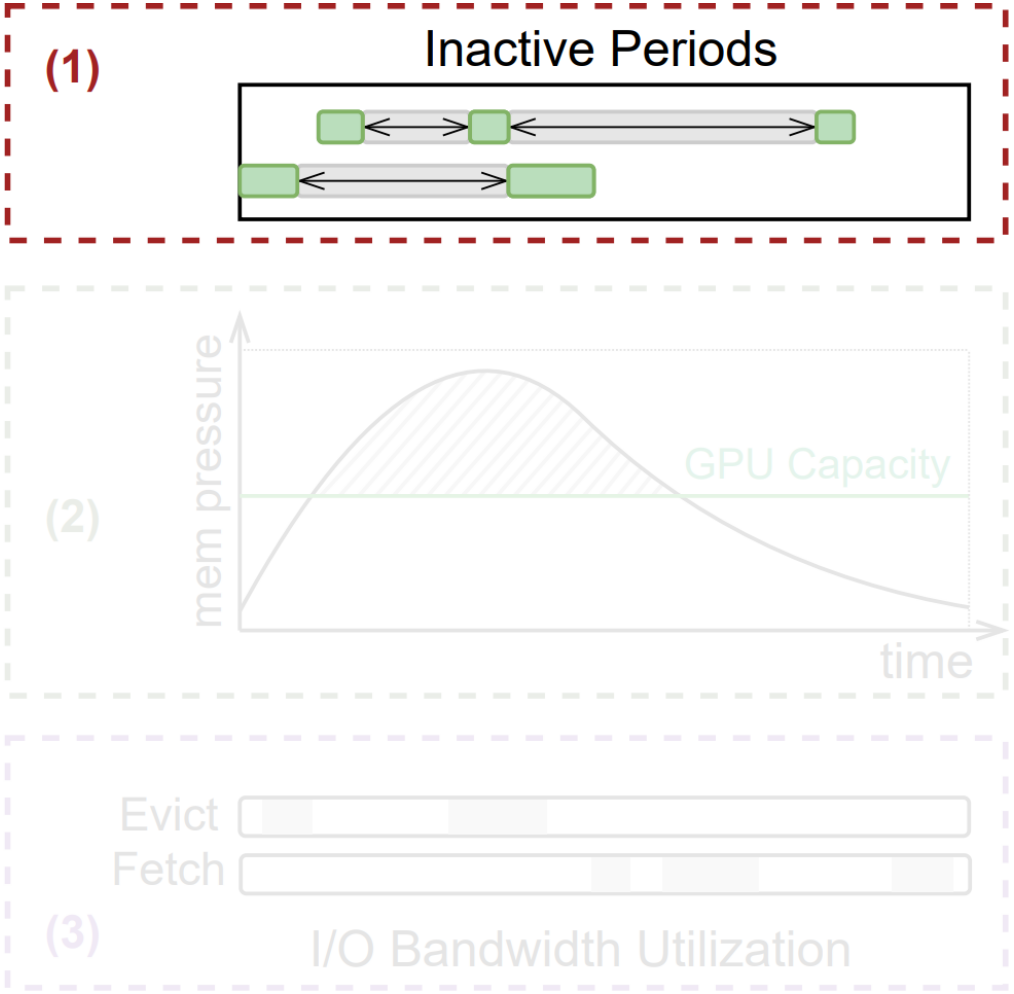
Decision Strategy: Choose tensors with the largest size and longest inactive time.



Evicting a tensor will affect GPU memory pressure and I/O bandwidth utilization.

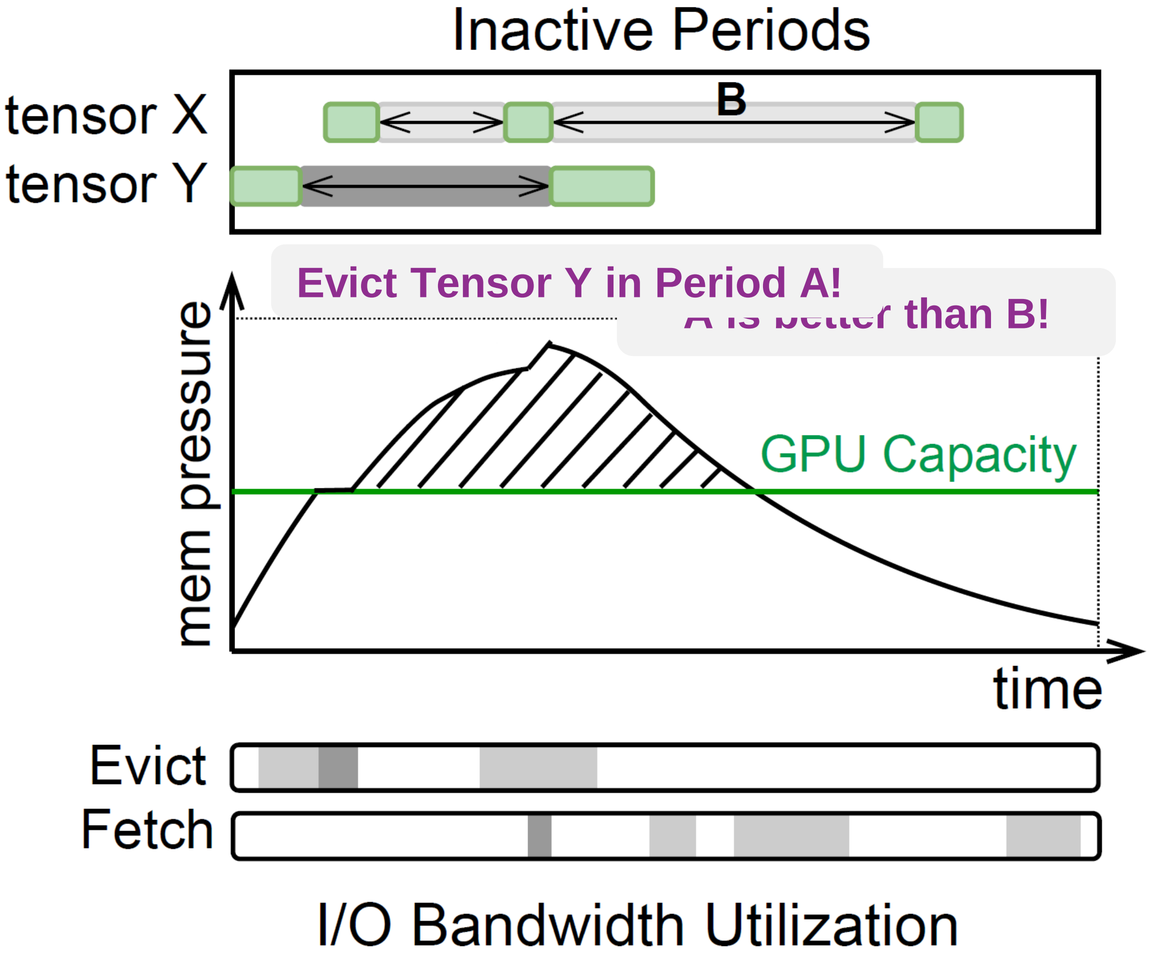
Each tensor migration will affect the subsequent decisions

Utilizing Dynamic Algorithm to Decide Which Tensor to Evict



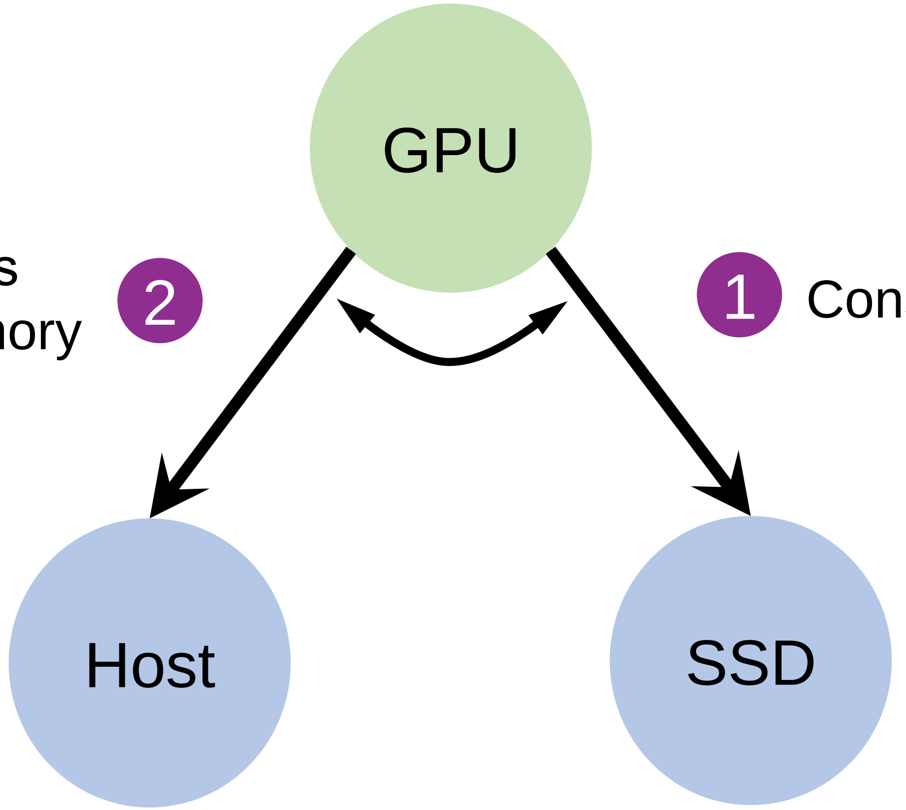
- **Dynamic Algorithm**: Keep track of
  (1) inactive tensor periods,
  (2) GPU memory pressure, and
  (3) estimated I/O bandwidth utilization.

Utilizing Dynamic Algorithm to Decide Which Tensor to Evict



1. Estimate the impact for evicting each inactive tensor
2. Choose the one which will reduce the highest memory pressure (a) , and cause the least I/O bandwidth pressure (b + c)
3. Update the impact of this decision

Deciding the Eviction Destination Based on the Available SSD Bandwidth



1 Consider SSD first

2 If SSD's current traffic is saturated, try host memory

3 If both SSD and Host are busy, go for SSD.

When Should We Migrate Tensors to Eliminate Data Access Stalls

1. Start eviction once the tensor becomes inactive

2. Calculate the time needed to fetch the tensor

$$t = \frac{\text{Tensor Size}}{\text{Available I/O Bandwidth}}$$

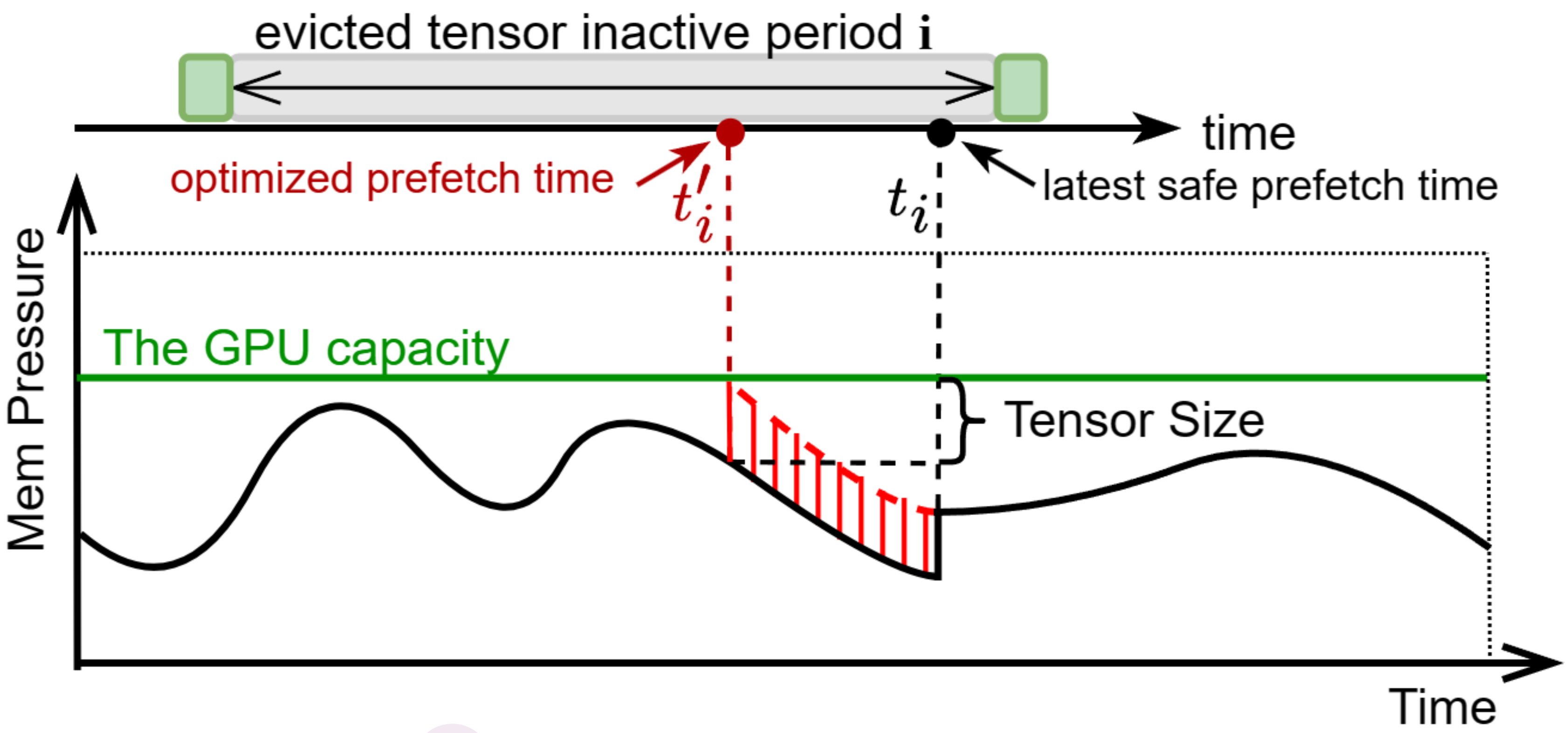
Stalls May Happen Due to Performance Variations at Runtime

Prefetching Too Early will Lose Eviction Benefits

Eliminate Potential Stalls without Losing Eviction Benefits!

Eliminate Potential Stalls with Smart Prefetching Algorithm

2 Re-schedule prefetch time t_i -> t_i'



1 Try to find an earlier time

How to Implement Tensor Migrations

New Instructions

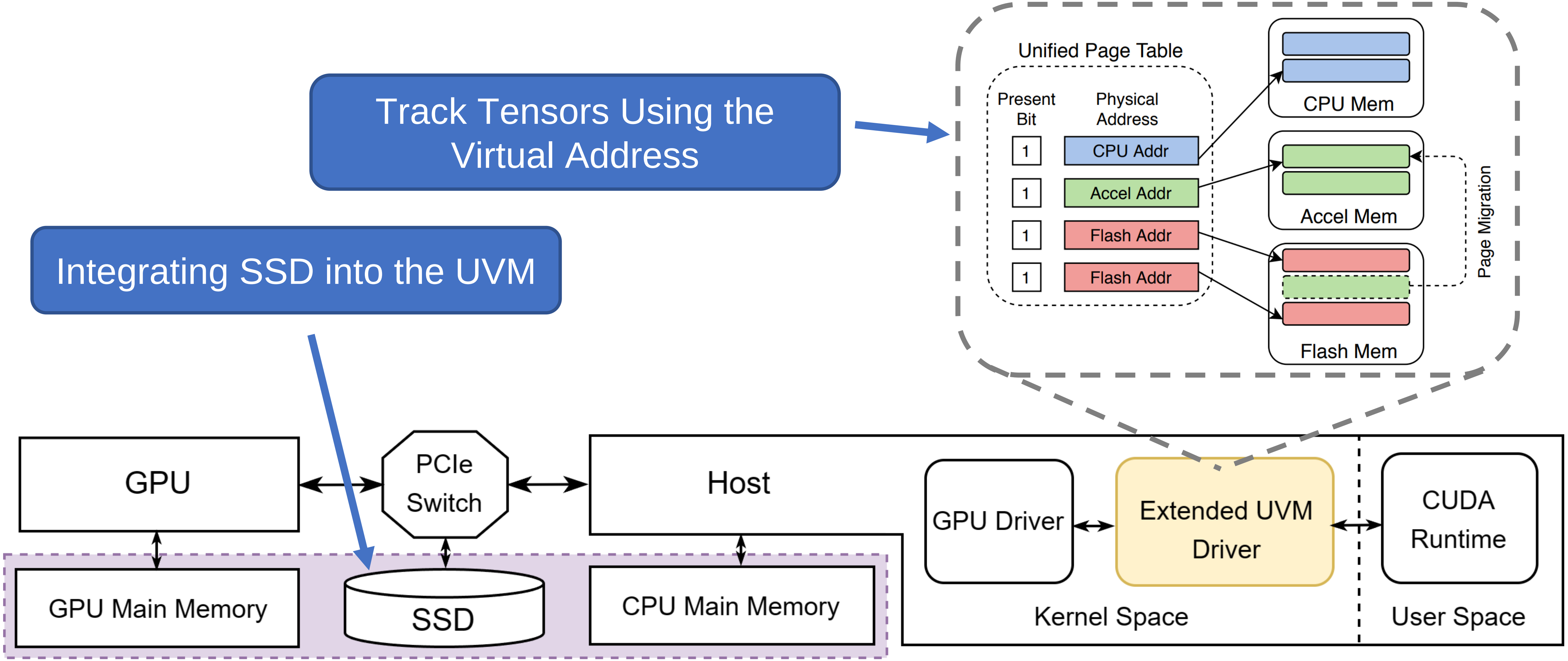
Instruction	Objective
g10_pre_evict()	Evict Inactive Tensor
g10_prefetch()	Prefetch Active Tensor
g10_alloc()	Allocate Alive Tensor
g10_free()	Discard Dead Tensor

G10 Instrumented Program

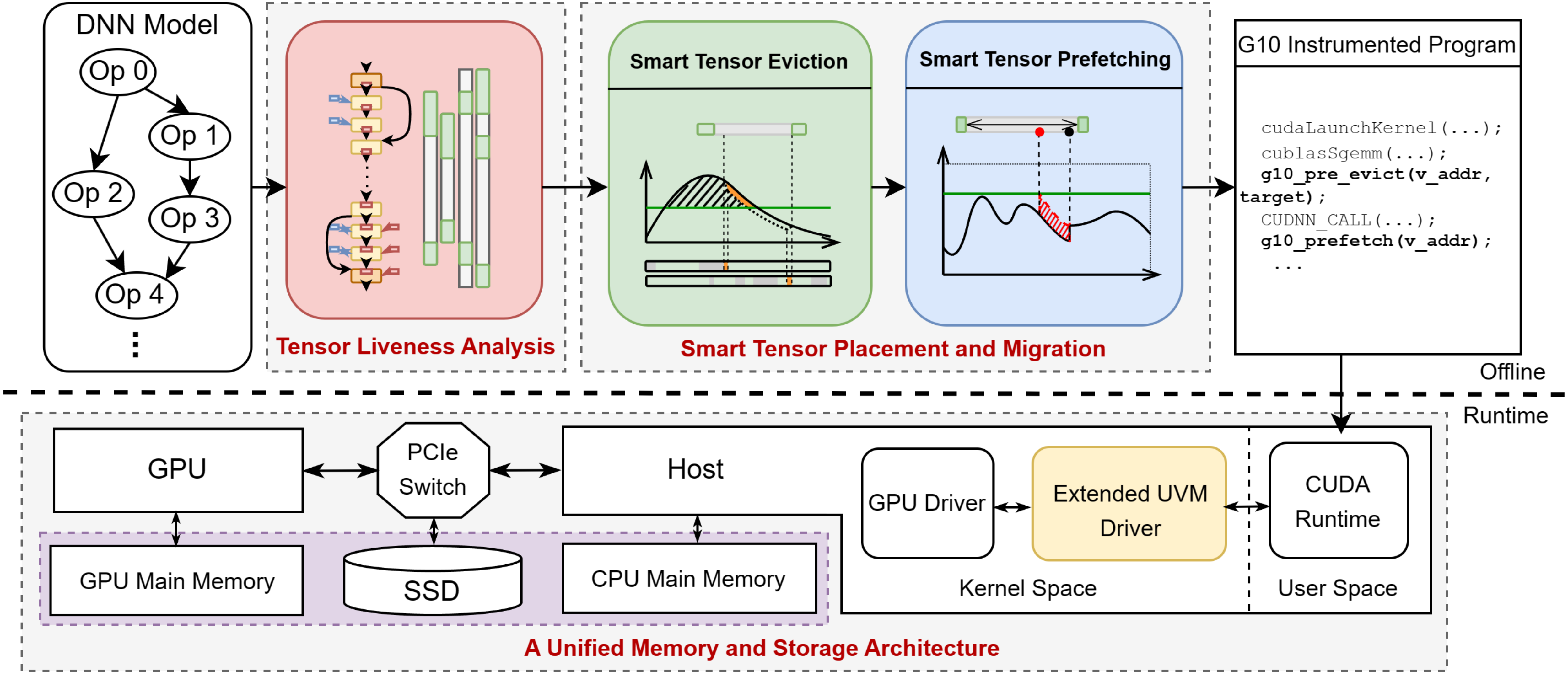
```
...
g10_alloc(tensor20, 40960);
g10_prefetch(tensor23, 40960);
// Kernel 2 ReLU(input, output)
forward_ReLU_l2(tensor5, tensor5);
...
g10_alloc(&tensor22, 77073360);
g10_alloc(&tensor2914, 4110417920);
// Kernel 3 MaxPool2d(input, output)
forward_MaxPool2d_l3(tensor5, tensor20);
...
// Kernel 4 Conv2d(input, output, filter, workspace)
forward_conv2d_l4(tensor20, tensor22, tensor23, tensor2914);
g10_free(tensor2914);
g10_pre_evict(tensor23, 40960, SSD);
...
```

G10 will migrate tensors across GPU memory, host memory and Flash memory transparently

A Unified GPU Memory and Storage Architecture



Put It All Together



G10 Evaluation

Implementation

Trace-driven simulator based on UVMSmart and GPGPU-Sim

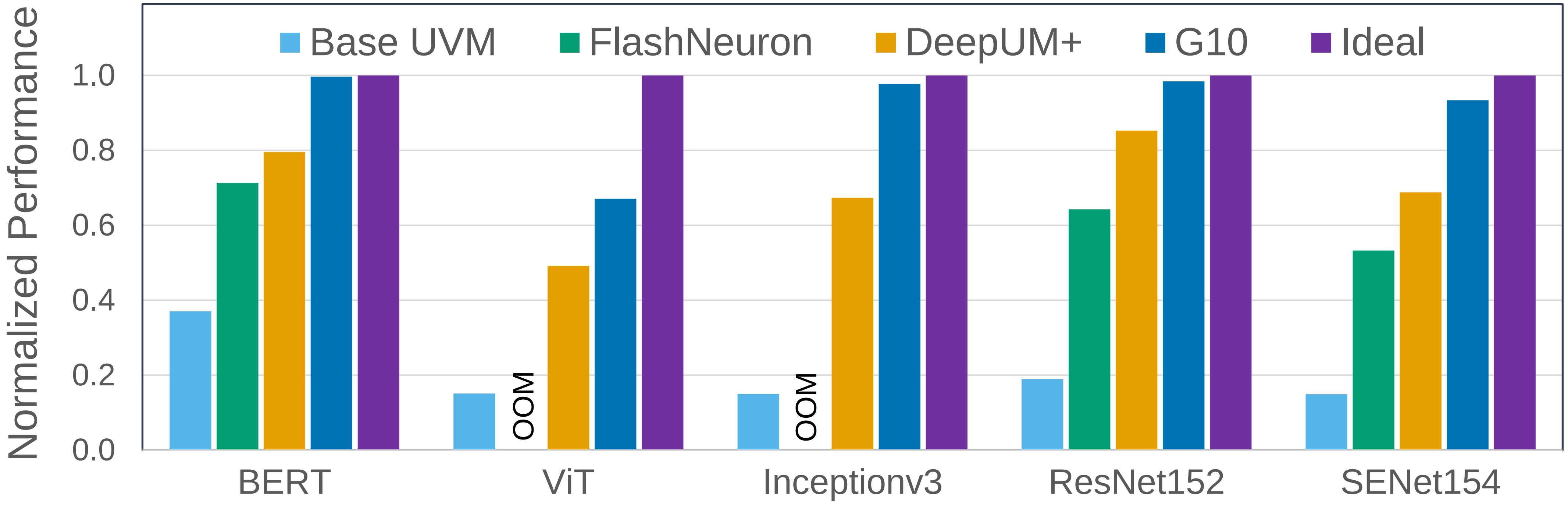
Benchmarks

BERT, ViT, ResNet, Inceptionv3, SENet

Baselines

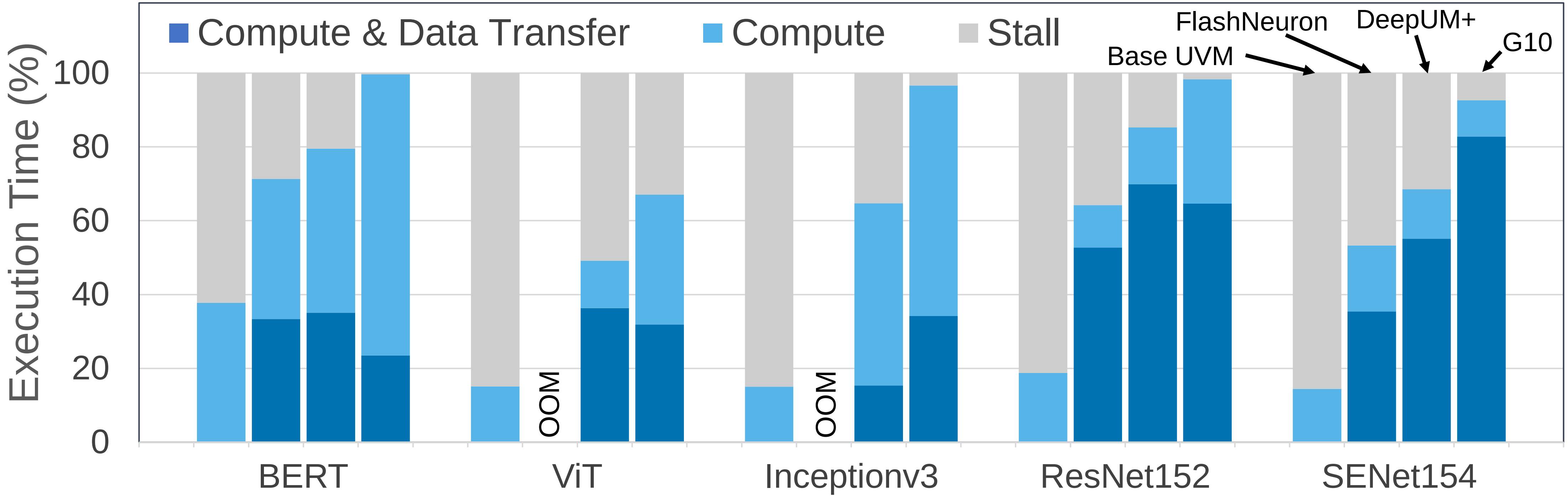
- **Ideal:** GPU with infinite memory
- **Base UVM:** Basic GPU-CPU-SSD UVM system w/o smart migrations
- **DeepUM+:** UVM system with correlation-based prefetch
- **FlashNeuron:** Direct GPU-SSD communication w/ selective tensor offloading

Performance Benefit of G10 for Training Large Models



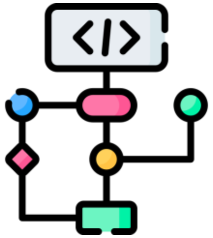
With limited GPU memory, G10 achieves 90.3% of the ideal performance

Performance Breakdown of G10

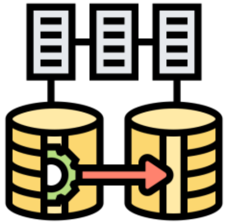


G10 incurs the least stall time, as it achieves better I/O and computation overlapping

G10 Summary



Smart Tensor Migration Mechanism for DNN Workloads



A Unified Memory and Storage Architecture for Simplifying Memory Management

Achieves 90.3% of the Ideal Performance

Thank You!

Haoyang Zhang

Yirui Eric Zhou, Yuqi Xue, Yiqi Liu, Jian Huang

Systems Platform Research Group

UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN