ECLIP: Energy-efficient and Practical Co-Location of ML Inference on Spatially Partitioned GPUs

Ryan Quach*, Yidi Wang†, Ali Jahanshahi*, Daniel Wong*, Hyoseung Kim*

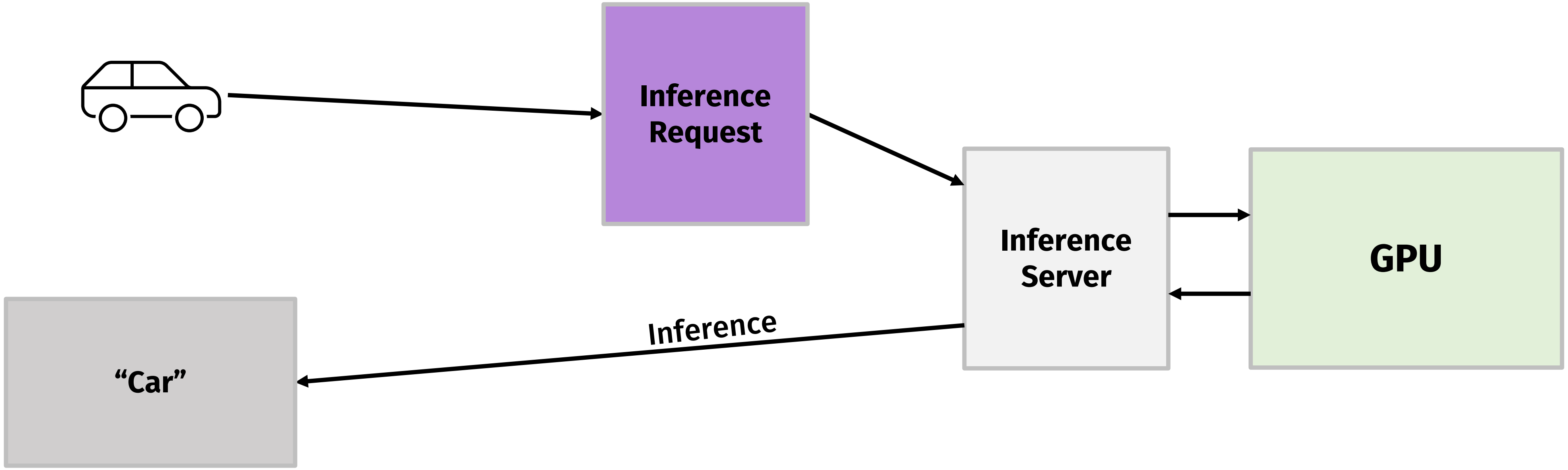
*University of California, Riverside
†Santa Clara University

ML Inference in Everyday Life

ML inference is becoming mainstream

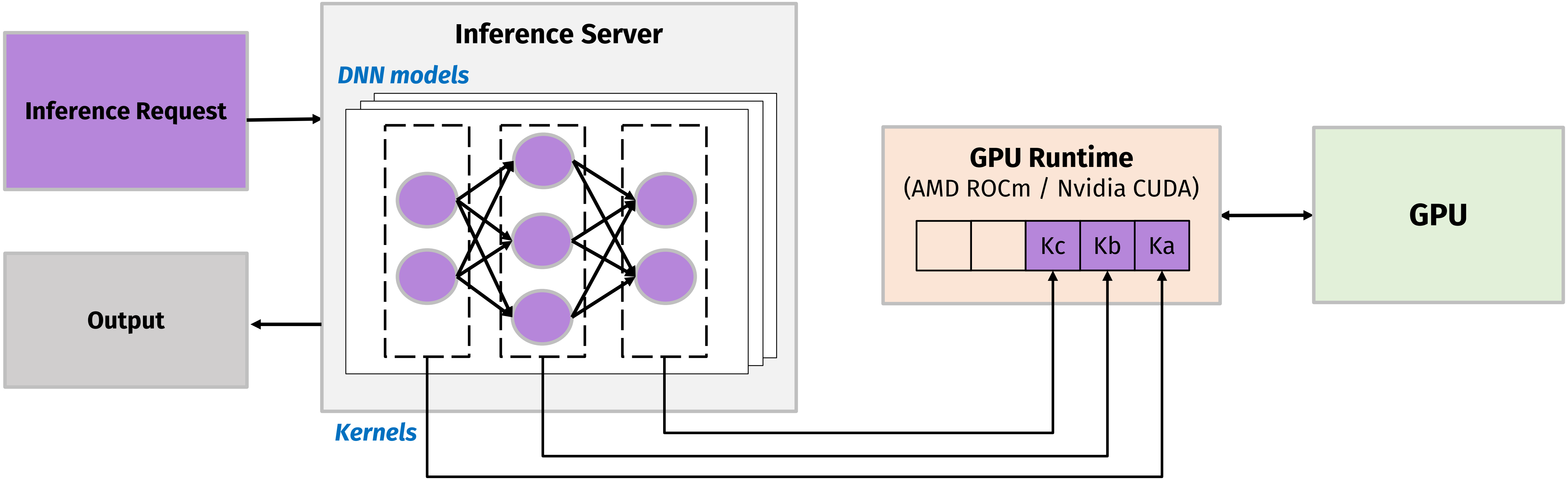
Self Driving Cars
Inference for car sensors – extremely deadline sensitive

Unlocking Smartphones
Inference on Face ID – inconvenient if slow



How do Inference Servers Work?

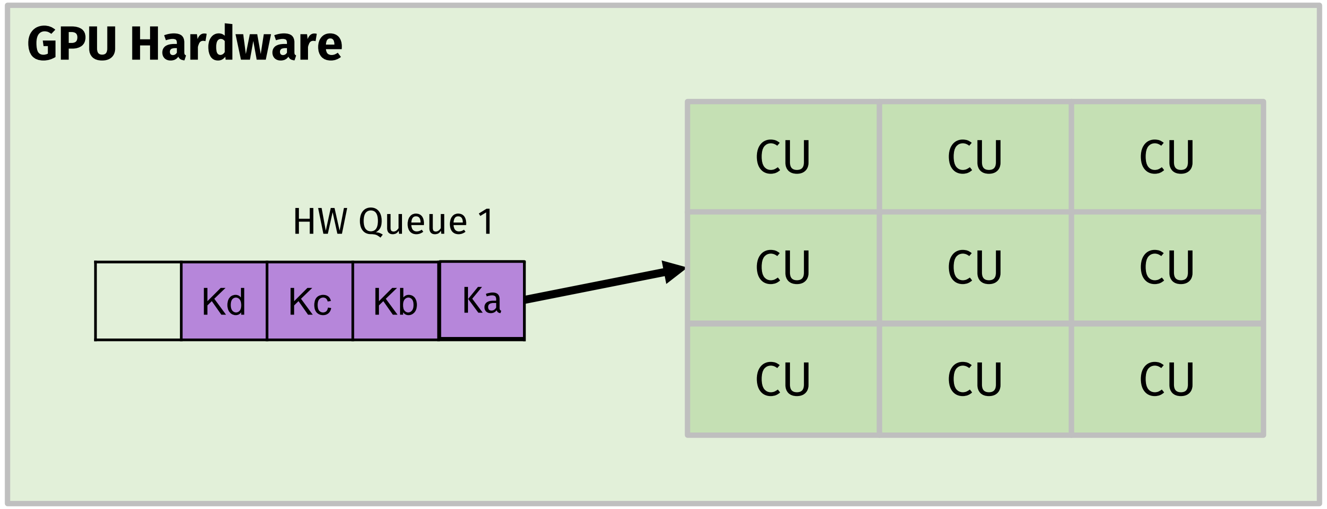
Each inference request involves launching multiple kernels, upwards of several hundred kernels



GPU Compute Resources

Within GPUs, kernels are dispatched to Compute Units (CUs)

- CU = Streaming Multiprocessors in NVIDIA terminology

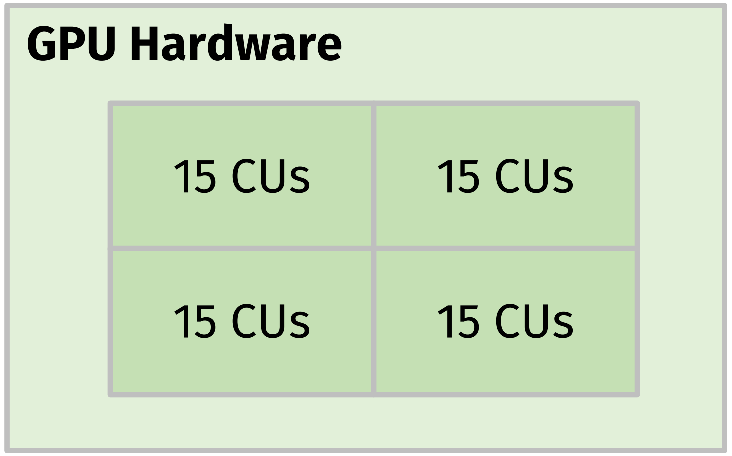


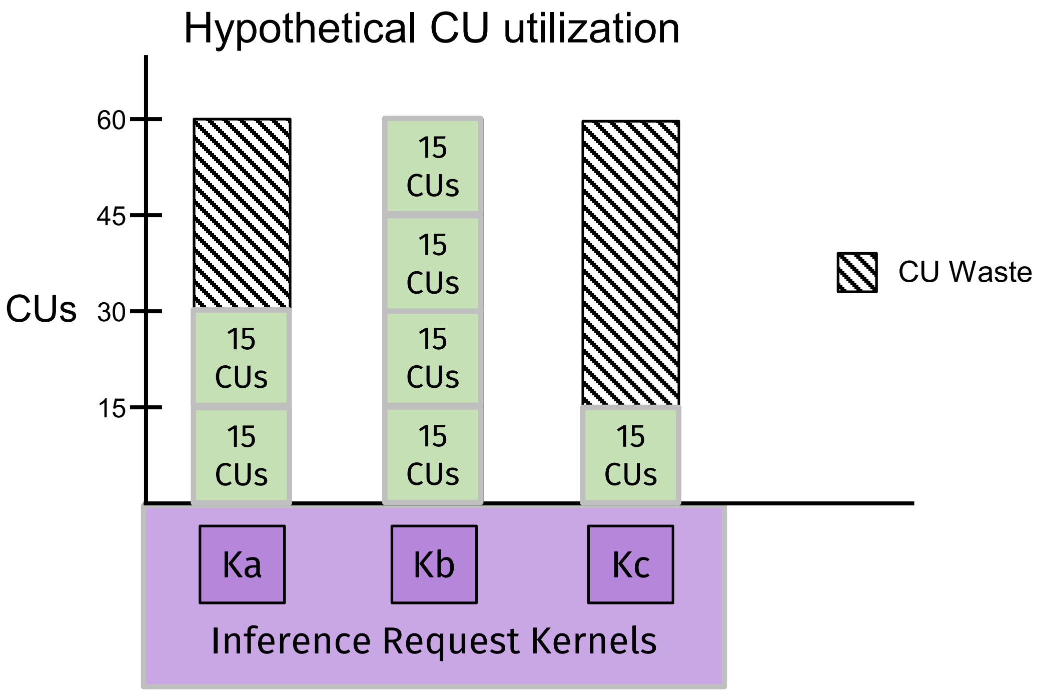
Issue: Inference Kernels underutilize the GPU's compute resources

Problem: Underutilized GPUs Waste Power

Inference kernels frequently underutilize the GPU

- Inference kernels do not need all CUs [1]
- Idling CUs cannot be power gated [2]





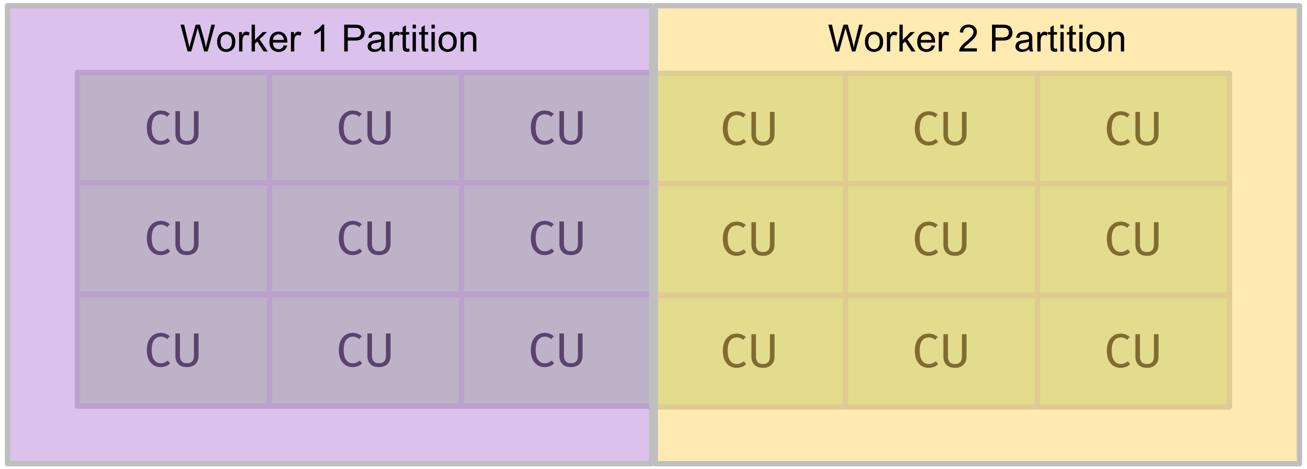
Opportunity to share the GPU among workloads

[1] M. Chow, A. Jahanshahi, and D. Wong, “Krisp: Enabling kernel-wise right-sizing for spatial partitioned gpu inference servers,” in 2023 IEEE International Symposium on High-Performance Computer Architecture
[2] Y. Wang, M. Karimi, Y. Xiang, and H. Kim, “Balancing energy efficiency and real-time performance in gpu scheduling,” in 2021 IEEE Real-Time Systems Symposium

Sharing a GPU Among Workloads

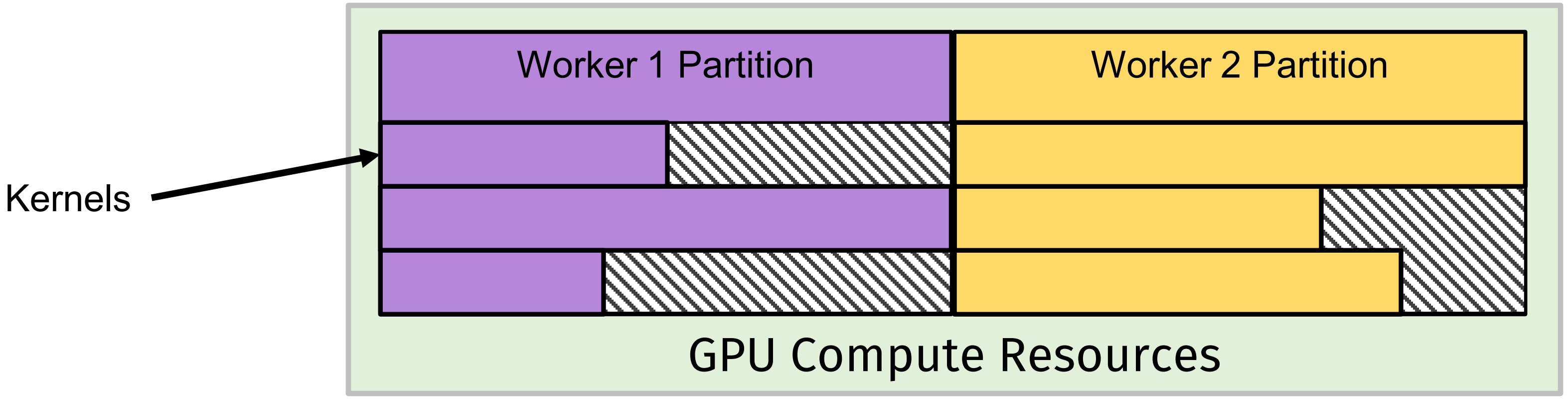
Increase GPU utilization by co-locating inference models

- Achieved through Spatial Partitioning
- Potentially improves throughput
- Can increase energy efficiency

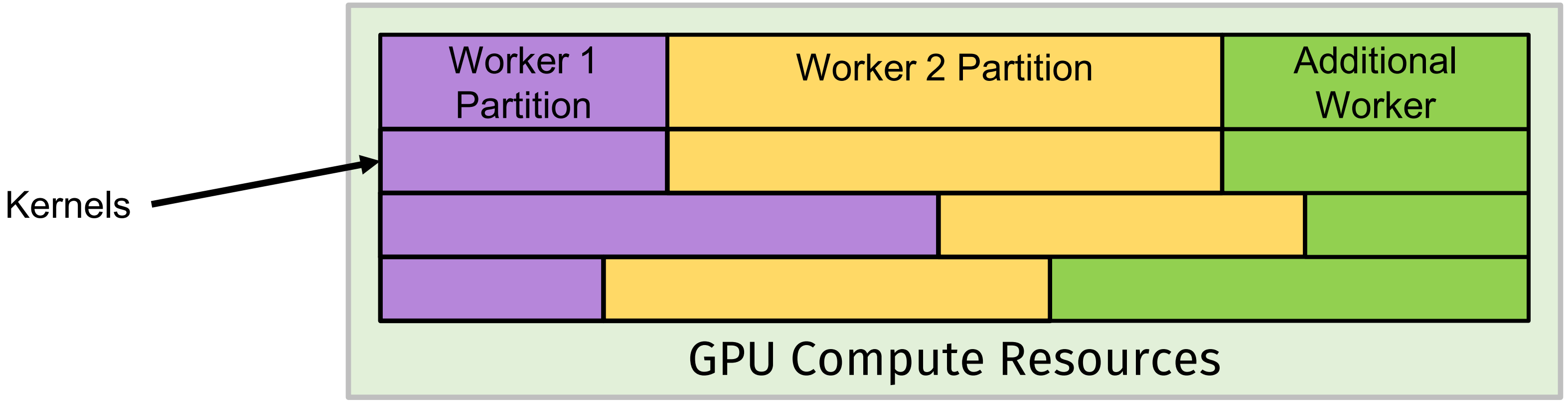


Limitations of GPU Spatial Partitioning

Model-grain Right-Sizing:
- leaves GPU underutilized



Kernel-grain Right-Sizing:
- better utilization but requires custom hardware modifications to extend AMD CU Masking

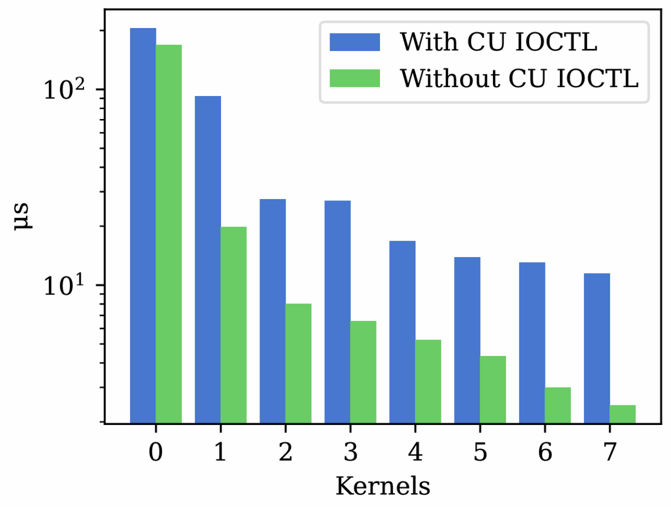


Goal: How can we achieve kernel grain benefits, without the custom hardware?

CU Masking IOCTL calls are Expensive

Challenge 1

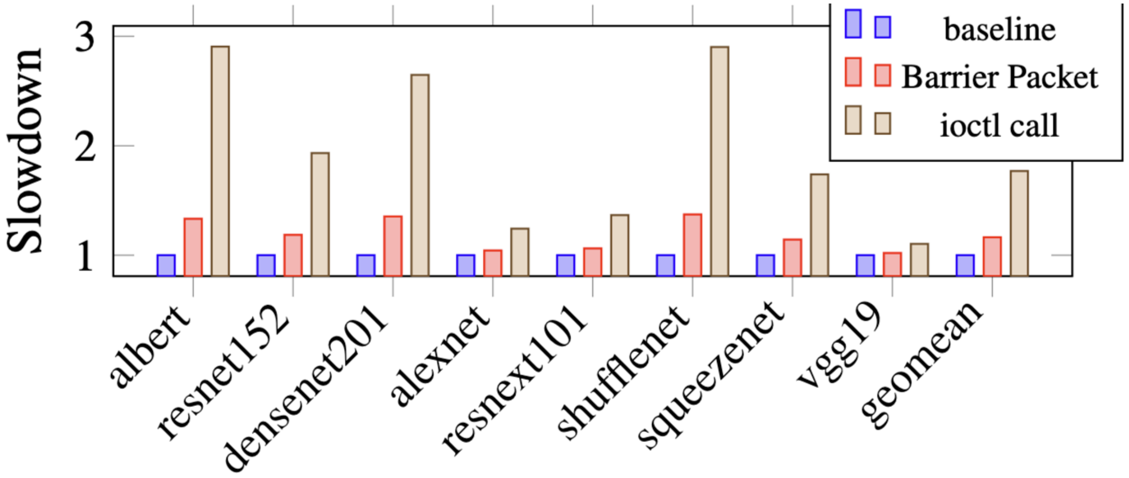
- CU Mask IOCTL cost is unpredictable and expensive
- Not viable to use for every kernel launch



Lower is better

Our Solution

[1]



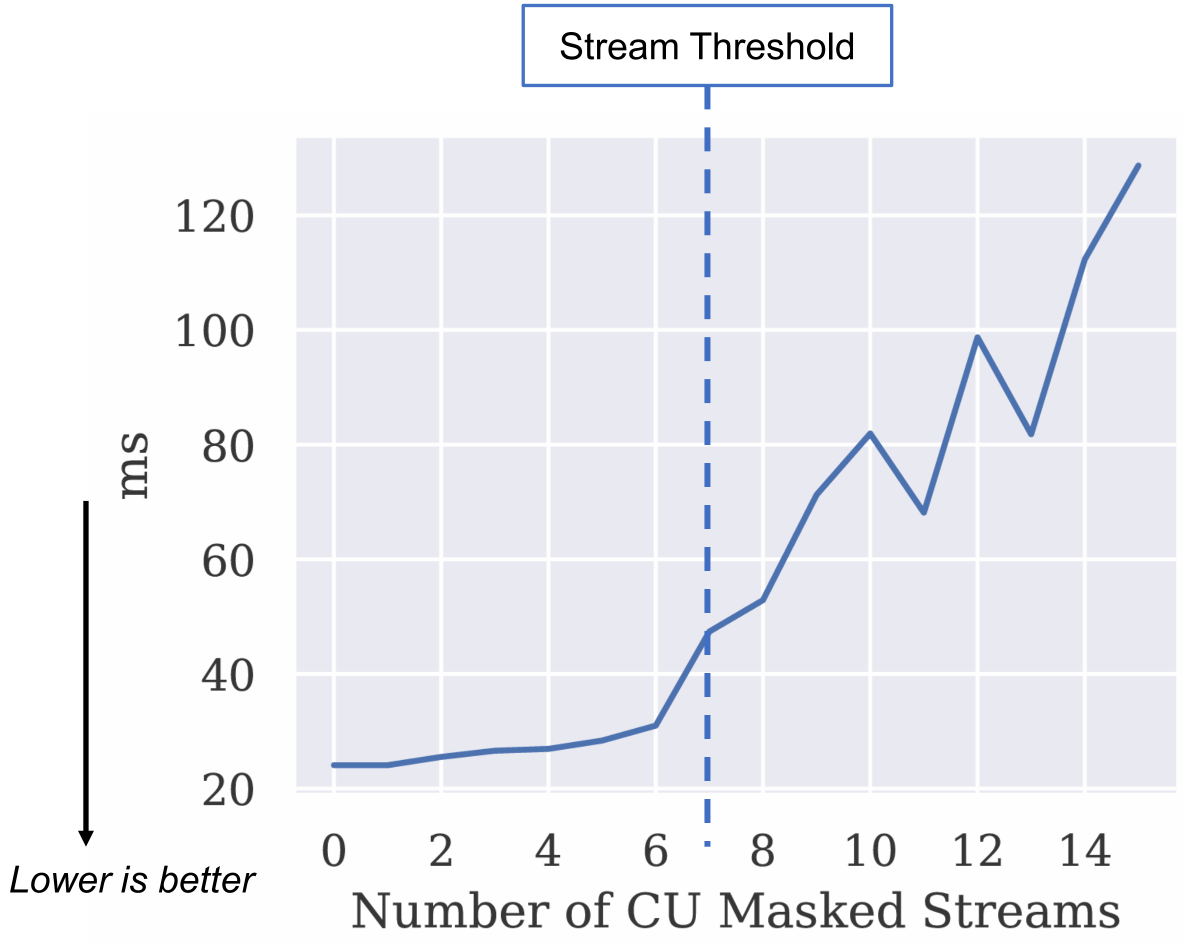
Create pool of CU-Masked stream

Instead of directly CU-masking every kernel:

- Pre-allocate CU-Masked streams
- Dispatch kernels to CU-Masked streams

[1] M. Chow, A. Jahanshahi, and D. Wong, “Krisp: Enabling kernel-wise right-sizing for spatial partitioned gpu inference servers,” in 2023 IEEE International Symposium on High-Performance Computer Architecture

Limited Number of CU-Masked Streams



Challenge 2

- Additional streams causes slowdown
- Exceeding 7 streams leads to significant and unpredictable slowdown

Our Solution

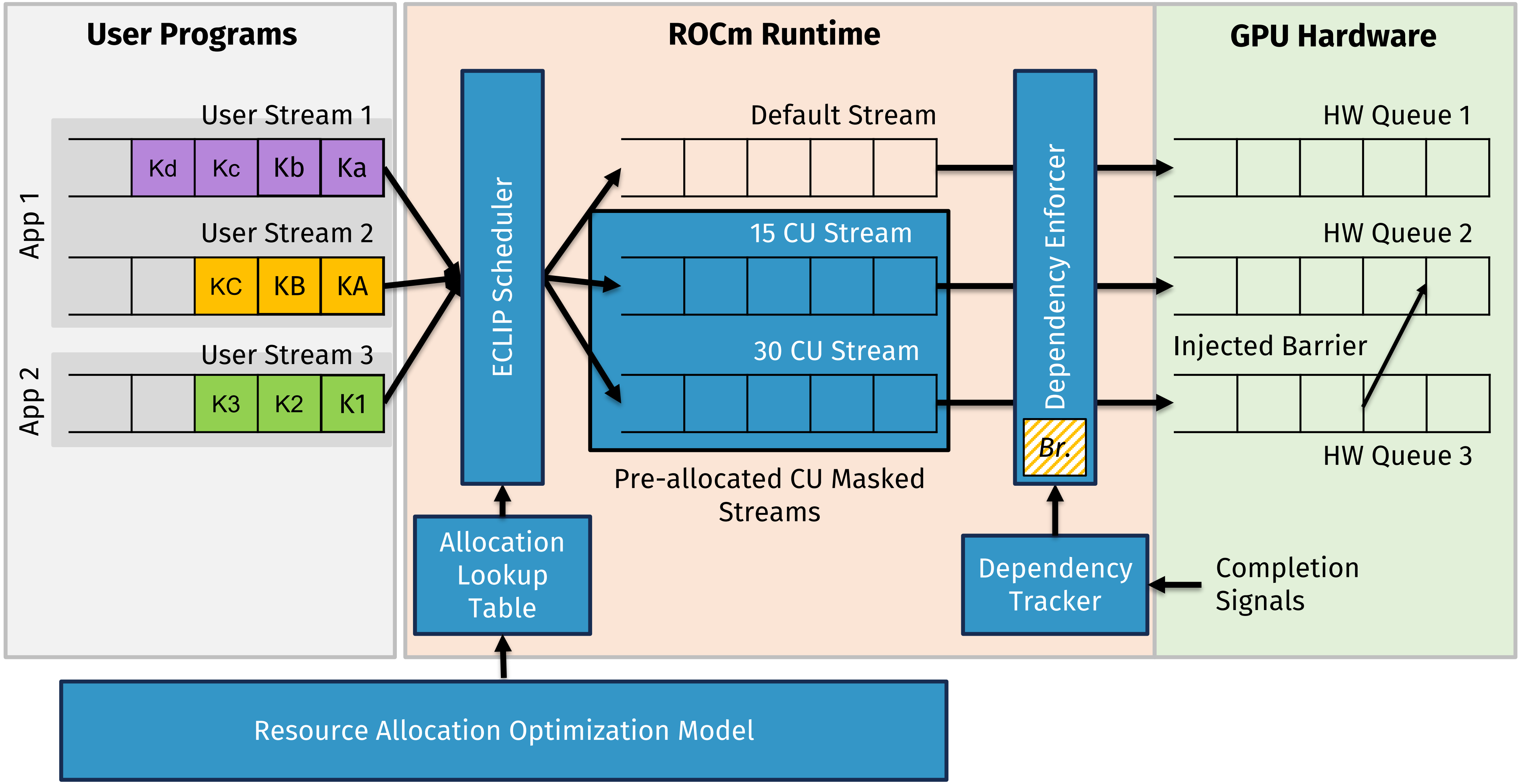
- Limit number of streams and reuse them
  - Carefully share across multi-worker scenario

Introducing ECLIP

Energy-efficient Kernel-wise Spatial Partitioning with Minimal Spatial Partitioning Overheads on Real-World GPUs

1. **Pre-allocated CU masked streams:**
   - Avoids costly CU masking IOCTL calls (challenge 1)
   - Strict budget on number of streams (challenge 2)
2. **Runtime Scheduler**:
   - Redirects kernels to pre-allocated streams
   - Enforces data dependencies
3. **Optimization Model:**
   - Achieves energy efficiency through minimal execution time & fairness
   - Assigns kernels to pre-allocated CU masked streams for all workers

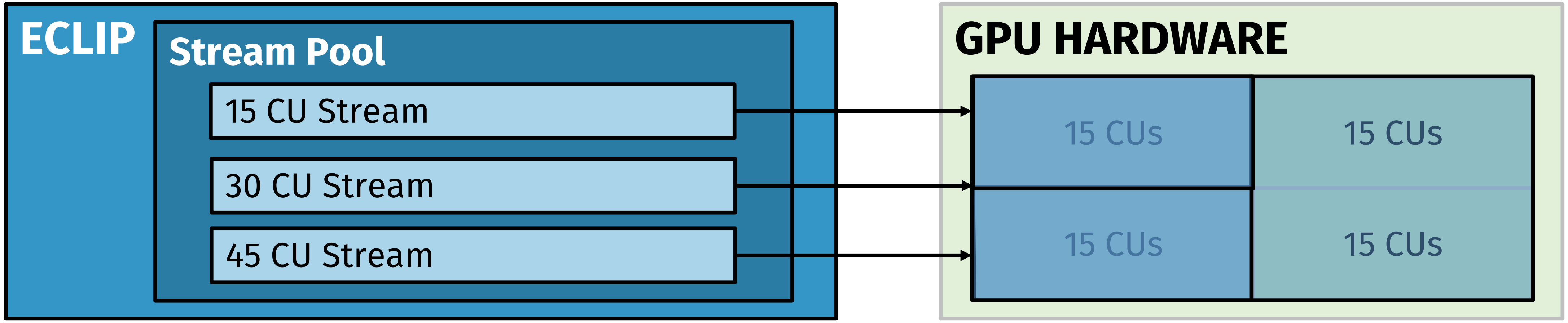
How does ECLIP work?



CU Masked Stream Pool

CU masked streams are created only once.

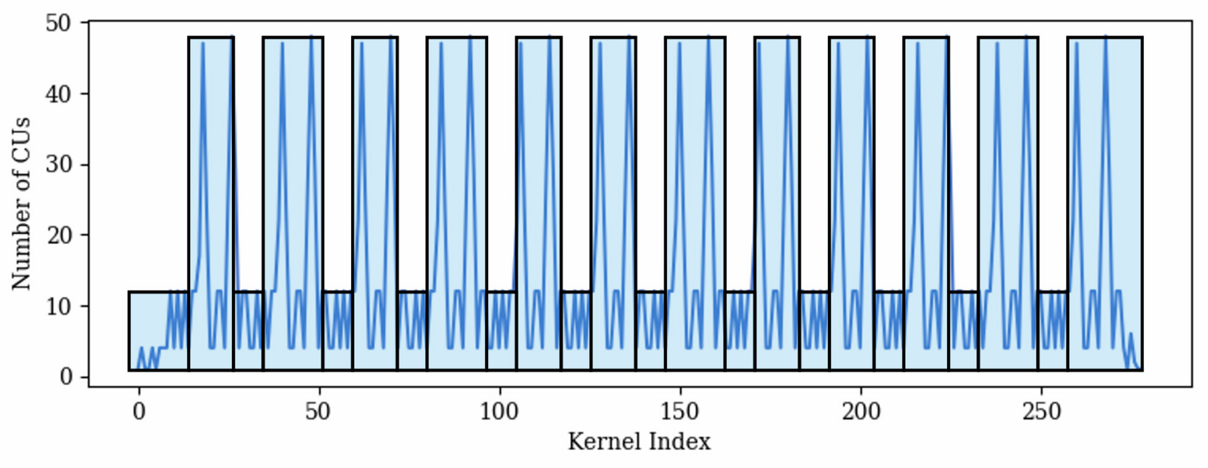
1. For each worker, create streams with different # of CUs
   - Considering H/W characteristics: e.g., shader engine size (15 CUs/SE)
2. Limit the total number of streams (due to Challenge 2)
   - CU overlap between workers is inevitable; our optimization minimizes this slowdown

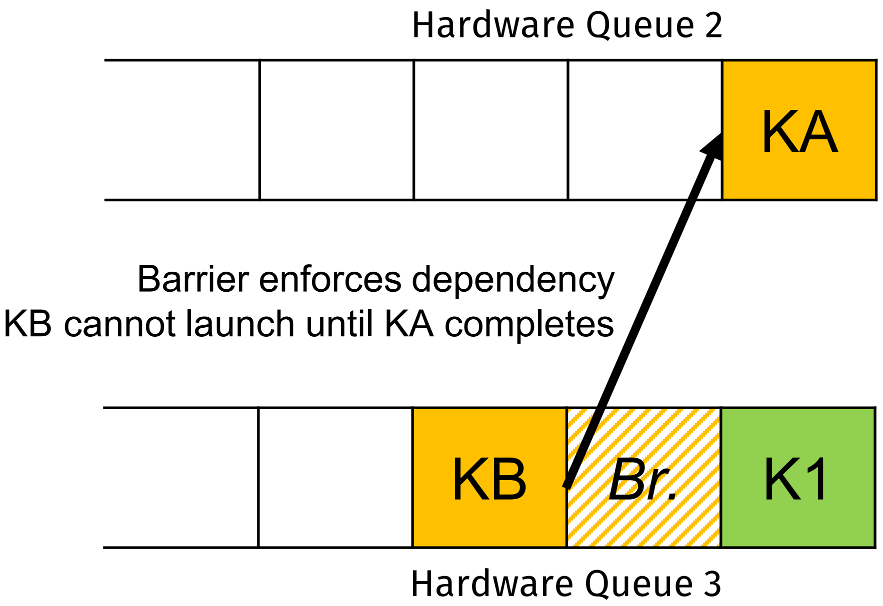


Enforcing Data Dependency

Switching streams can cause dependency issues

- Barrier packets are a ROCm feature
- Used to enforce dependencies across hardware queues
- Barrier packet latency is significant when frequently used
  - Our optimization determines when to switch





Mapping Kernels to CU Masked Streams

Questions to answer:

1. How many CUs kernels will be allocated
2. When to switch CU streams

To minimize energy consumption & overhead

Formulate optimization model:

Slowdown factor from other workers

- Objective Function: $\forall w, \text{minimize} \sum_{k \in w} e_k$ $e_k = \beta_k * (1 + \alpha_k)$

Minimize the **execution time** of all kernels in each worker w, with equal weights for all workers for **fairness**

- Why Fairness important?
  - Disparate completion times among workers lead to idling CUs, wasting energy

Evaluation

Evaluation

Workload Scenarios

1. **Baseline:** Unmodified runtime
2. **Model-Wise:** Model-wise right-sizing
3. **Kernel-Wise^{CU_Mask} (KW^{CU_Mask}):** Uses CU Mask IOCTL to switch CU allocations *every* kernel, like KRISP [1]
4. **Kernel-Wise^{Stream} (KW^{Stream}):** Uses ECLIP to switch between CU streams for *nearly every* kernel
5. **ECLIP:** Our full ECLIP implementation

Experiment Setup

CPU: 2x AMD EPYC 7302 16 core

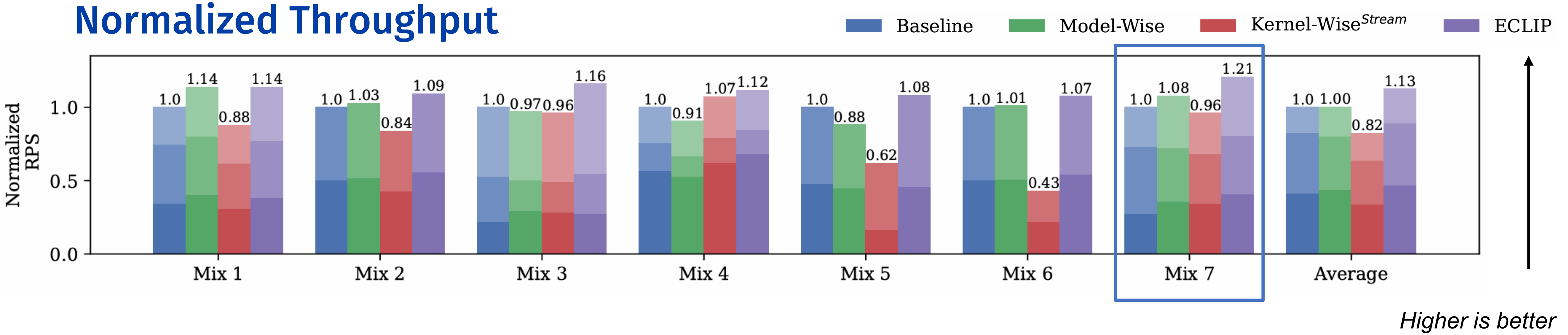
GPU: AMD Radeon Instinct MI50 (60 CUs)

Models Evaluated: ALBERT, DenseNet201, ResNet152, ResNeXt101, ShuffleNet, AlexNet, and vgg19

[1] M. Chow, A. Jahanshahi, and D. Wong, “Krisp: Enabling kernel-wise right-sizing for spatial partitioned gpu inference servers,” in 2023 IEEE International Symposium on High-Performance Computer Architecture

ECLIP Improves Throughput

Normalized Throughput

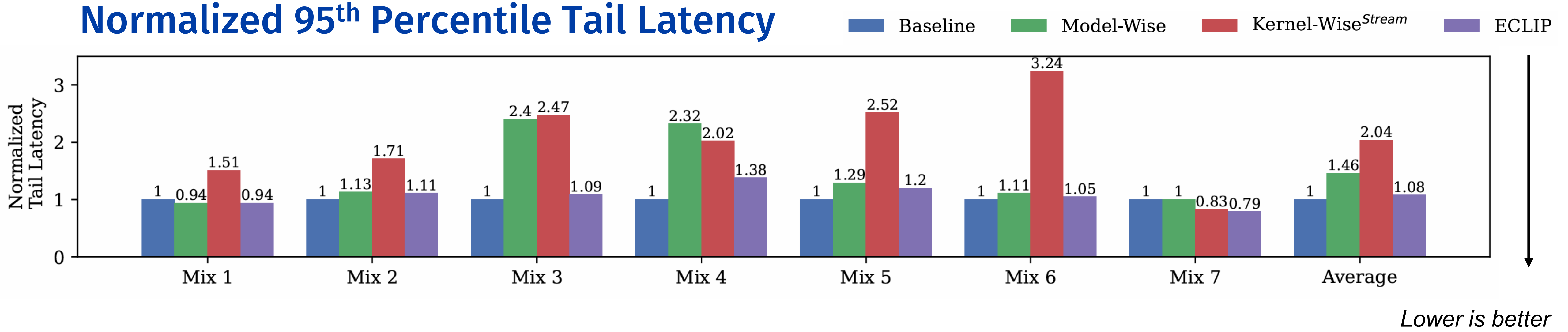


Best shown by Mix 7 (highlighted) – baseline workload allows kernels from one worker to cut ahead, despite having 3 identical models

On average, 13% higher throughput

ECLIP doesn't sacrifice Tail Latency

Normalized 95th Percentile Tail Latency

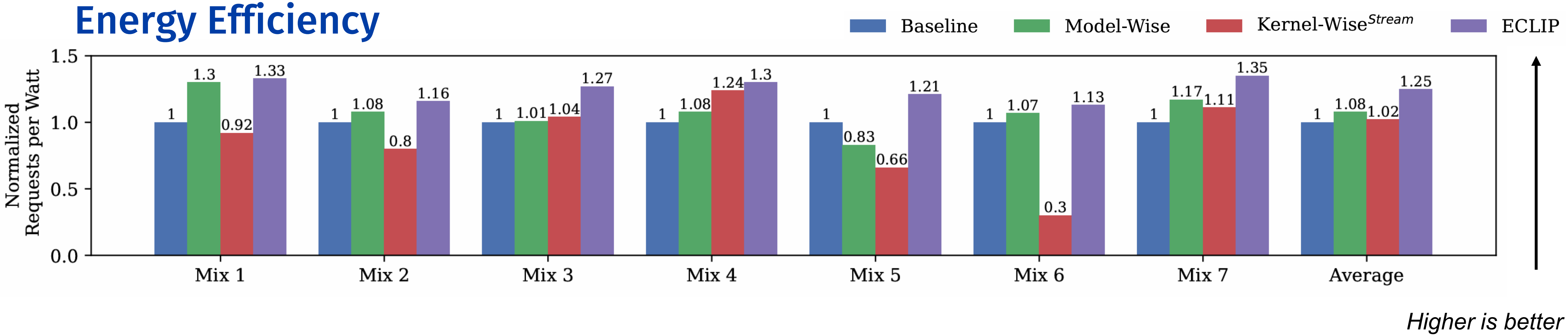


Excessive Barrier Packets, and uninformed CU sharing all lead to tail latency spikes
Model-wise granularity too coarse, can be slowed from resource contention
ECLIP circumvents these issues and ensures tail latency does not spike

Tail latency only rises 8%

ECLIP Improves Energy Efficiency

Energy Efficiency



The difference between occasionally idling CUs and better utilized CUs is not high
With faster throughput, energy is significantly conserved.

On average, 25% more requests per watt

Conclusion

- ECLIP enables energy-efficient kernel-wise ML inference on real GPUs
  - **Pre-allocated CU-masked Streams:** addresses overhead issues
  - **Runtime Scheduler:** coordinates stream switching while preserving data dependencies
  - **Optimization Model:** selects streams to minimize energy and overhead

- Key Results
  - ML inference throughput (+13%) and energy efficiency (+25%)
  - ECLIP does not compromise fairness and does not exhibit tail latency increases
    - Can occur in other partitioning approaches

Thank You!

Questions?