FASTEN: Fast GPU-accelerated Segmented Matrix Multiplication for Heterogeneous Graph Neural Networks

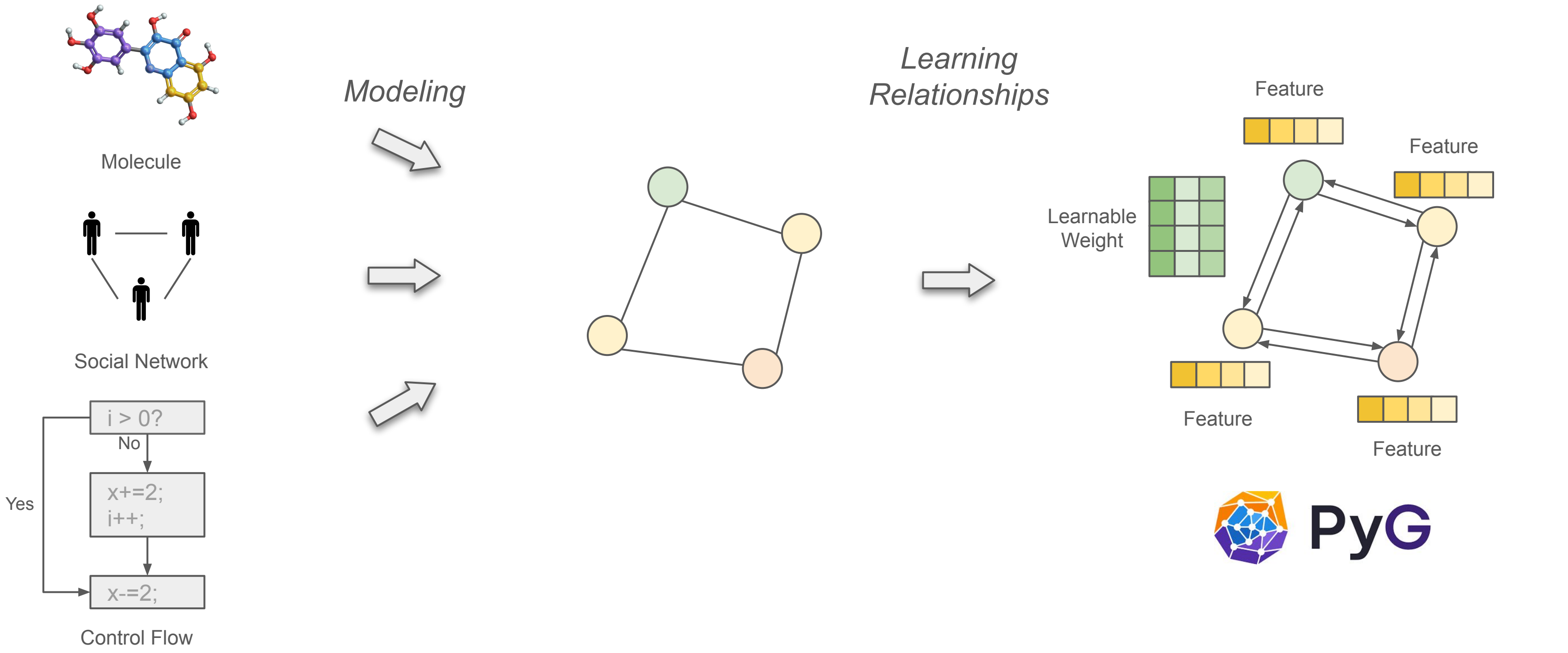
Keren Zhou*, Karthik Ganapathi Subramanian, Po-Hsun Lin, Matthias Fey, Binqian Yin, and Jiajia Li
kzhou6@gmu.edu

Modeling Graph Structure

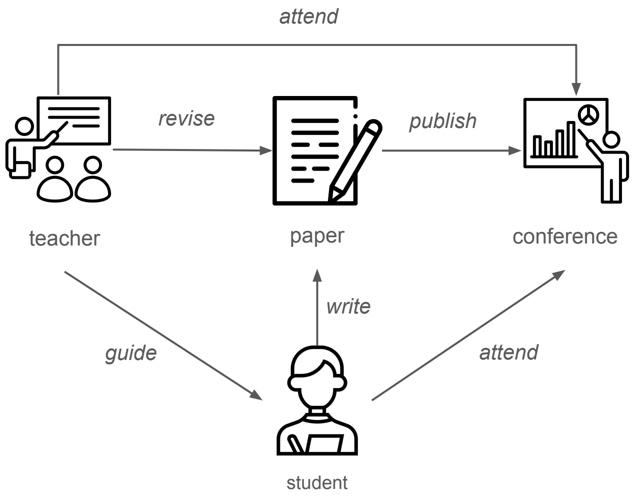
Applications

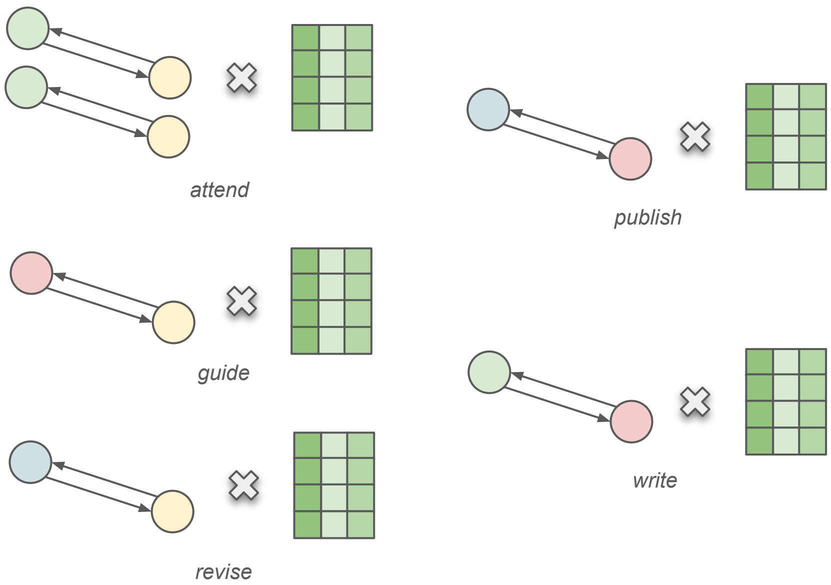
Graphs

Graph Neural Networks



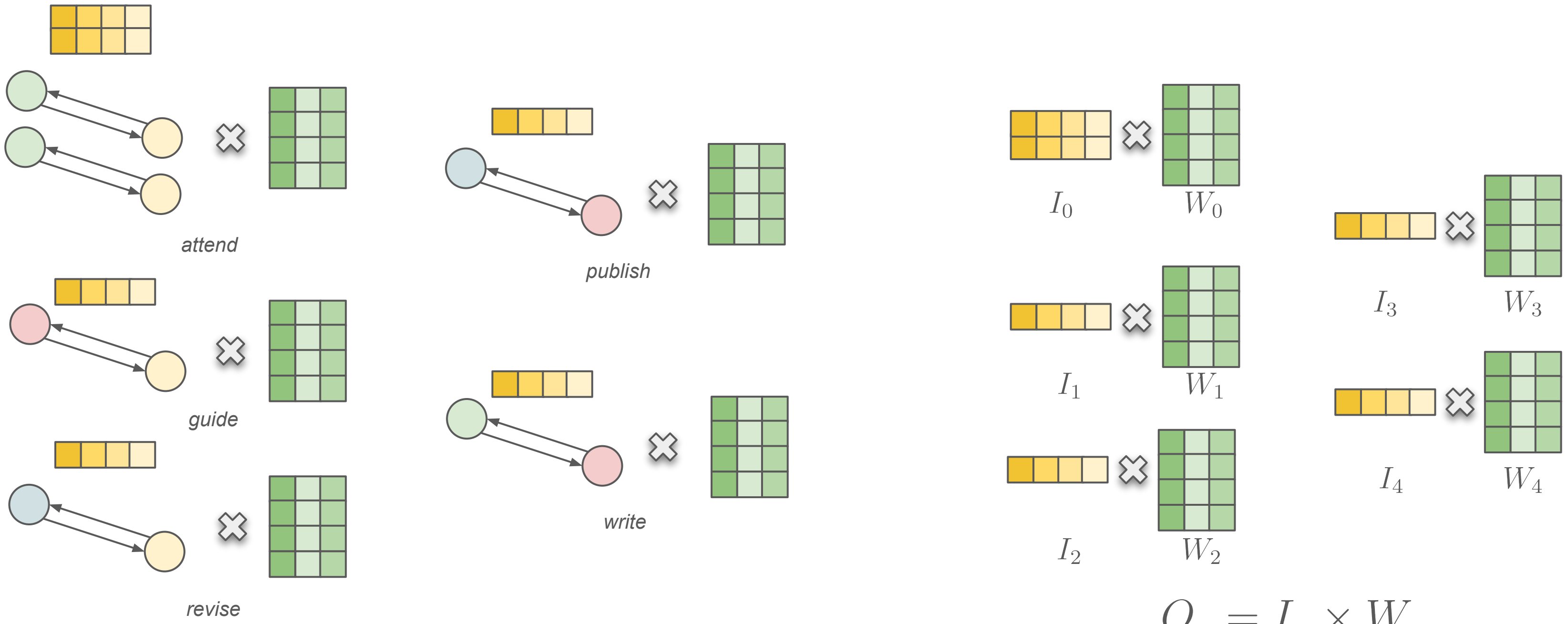
Heterogeneous Graphs





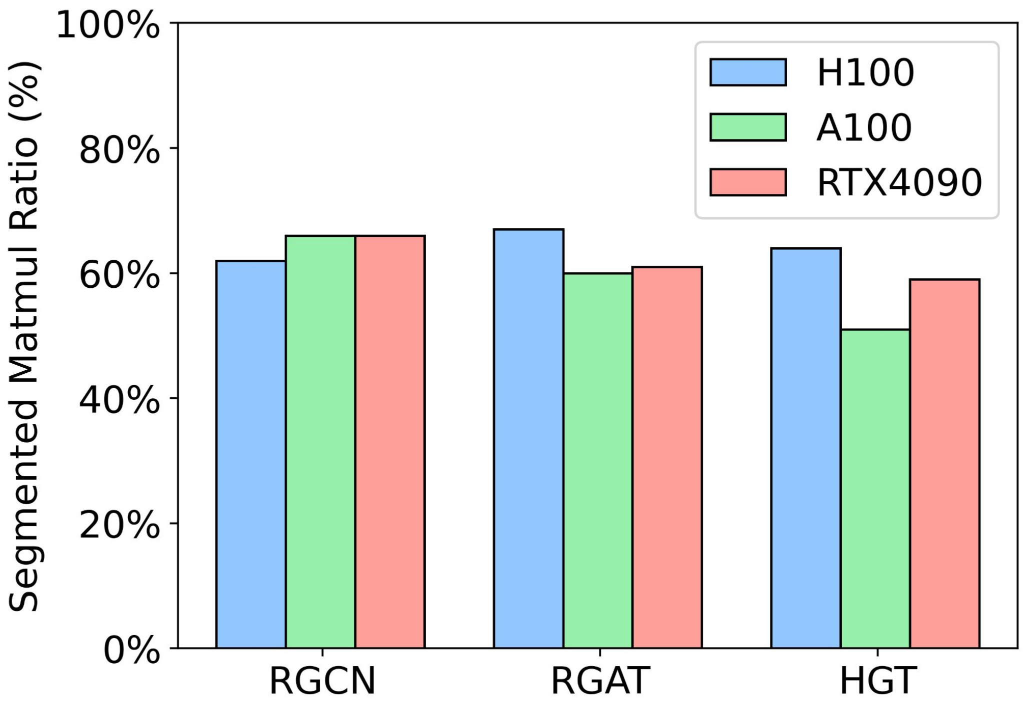
Research Network

Segmented Matmul



$$O_\tau = I_\tau \times W_\tau$$

Segmented Matmul Takes Significant Time



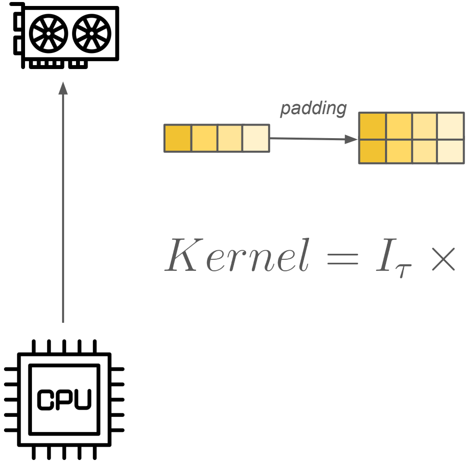
Existing Implementations

- Launch kernels in sequence
- High launch overhead
- Low GPU utilization

- Pad inputs to the same shape
- Extra compute
- Extra memory

$$Kernel_0 = I_0 \times W_0$$
$$Kernel_1 = I_1 \times W_1$$
$$Kernel_2 = I_2 \times W_2$$

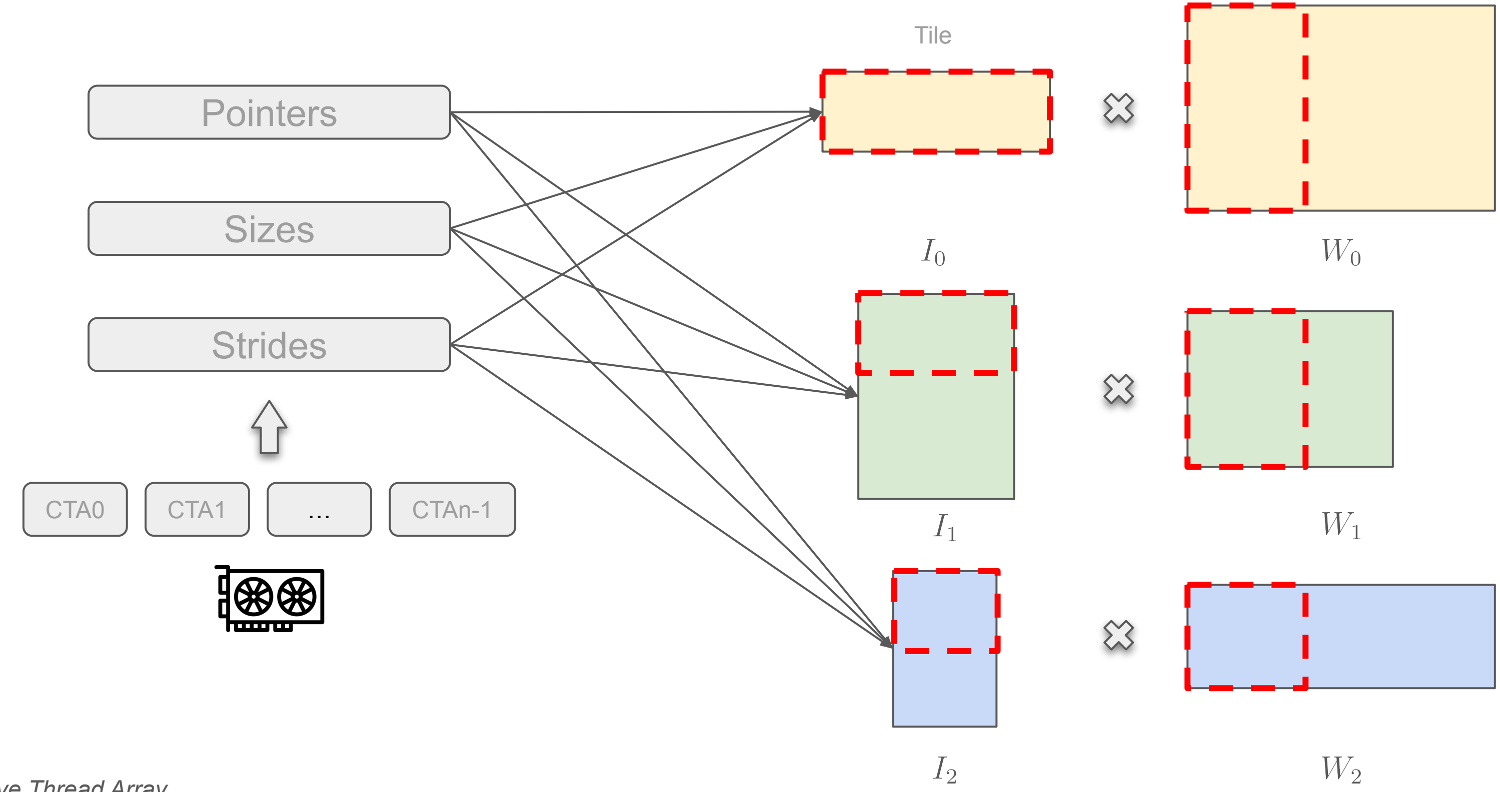
Loop over Matmul



$$Kernel = I_\tau \times W_\tau$$

Batched Matmul

Grouped Matmul

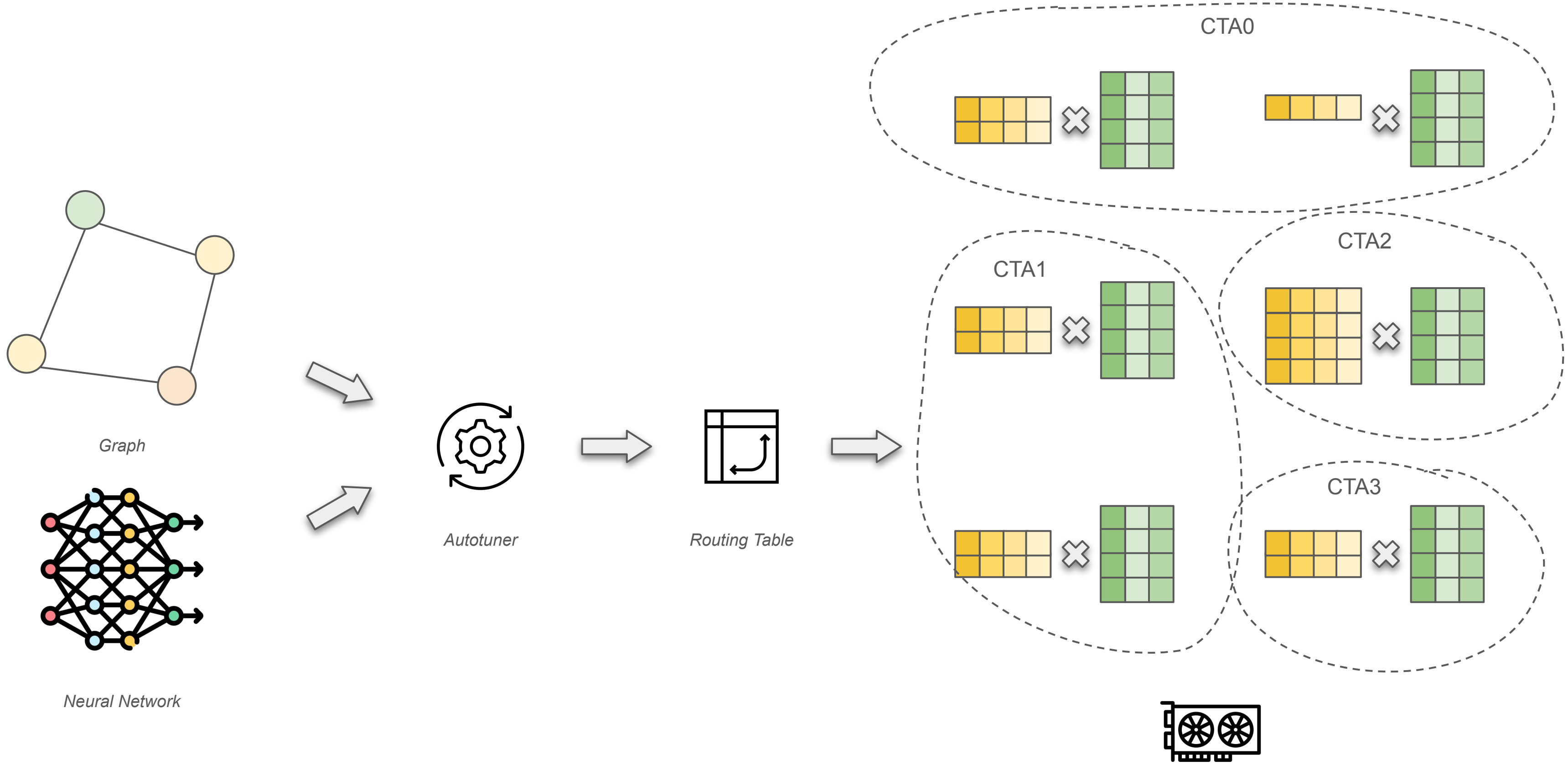


CTA: Cooperative Thread Array

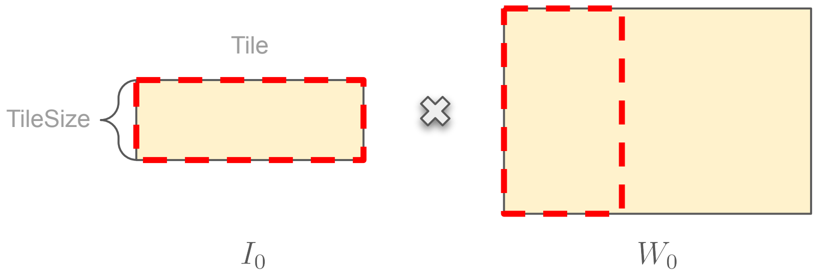
Inefficiencies of Grouped Matmul

- Grouped matmul uses many ***indirect memory access*** to auxiliary data structures
- The round-robin scheduling mechanism does not take ***data locality*** into account (e.g., W_0=W_1)
- Grouped matmul overlooks the tile sizes of input and weight matrices assigned to each CTA, causing ***workload imbalance***

Design of FASTEN



Routing Table



TypeIdx	Start	End
t0	0	10
t1	10	70
t2	70	100
…		

TileSize=32

TypeIdx	Start	End	Next
t0	0	10	NULL
t1	10	42	
t1	42	70	NULL
t2	70	100	NULL
…			

Routing Table Optimization

- Divide tiles into ***large*** and ***small*** tiles
- A large tile (i.e., a block) consists of *B* small tiles with size divisible by TileSize
- Small tiles can have indivisible tile sizes

TypeIdx	Start	End	Next	Large?
t0	0	10	NULL	N
t1	10	42	NULL	Y
t1	42	74	NULL	N/A
t1	74	106	NULL	N/A
t1	106	138	NULL	N/A

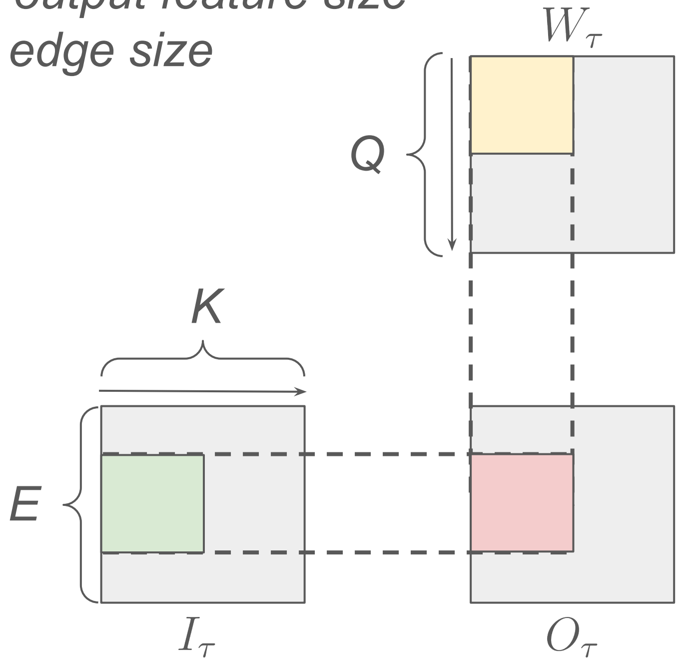
Merge B=4

Read one row to get B tiles

Basic Algorithm

TypeIdx	Start	End	Next
t0	0	10	NULL
t1	10	42	
t1	42	70	NULL
t2	70	100	NULL
…			

K: input feature size
Q: output feature size
E: edge size

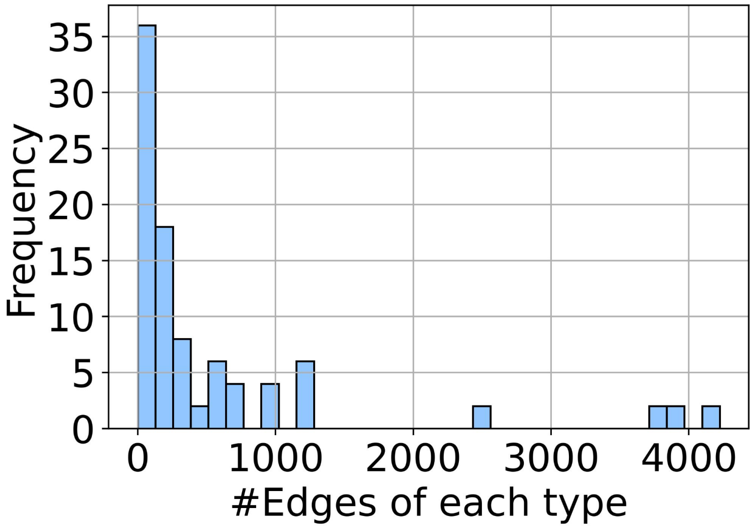


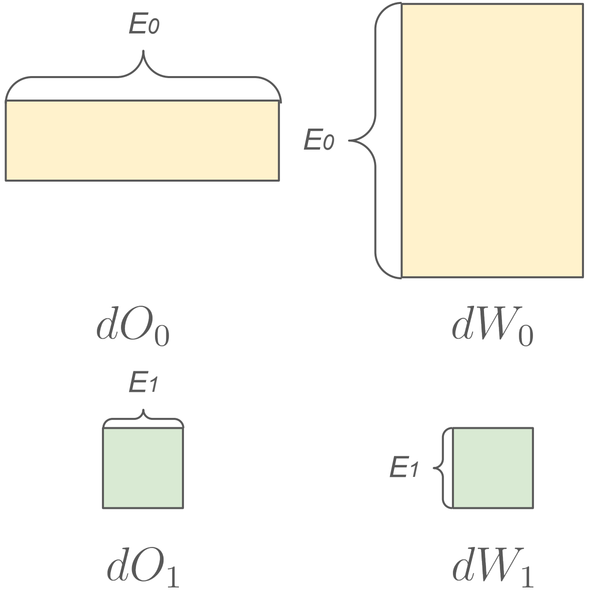
Optimization

- Algorithm
  - Dynamic tiling
  - Tile reordering
  - Persistent processing
- Implementation
  - Pipeline asynchronous load and (asynchronous) compute
  - Prefetch shared memory data
  - Tensor core TF32
  - Register blocking

Backward Observation

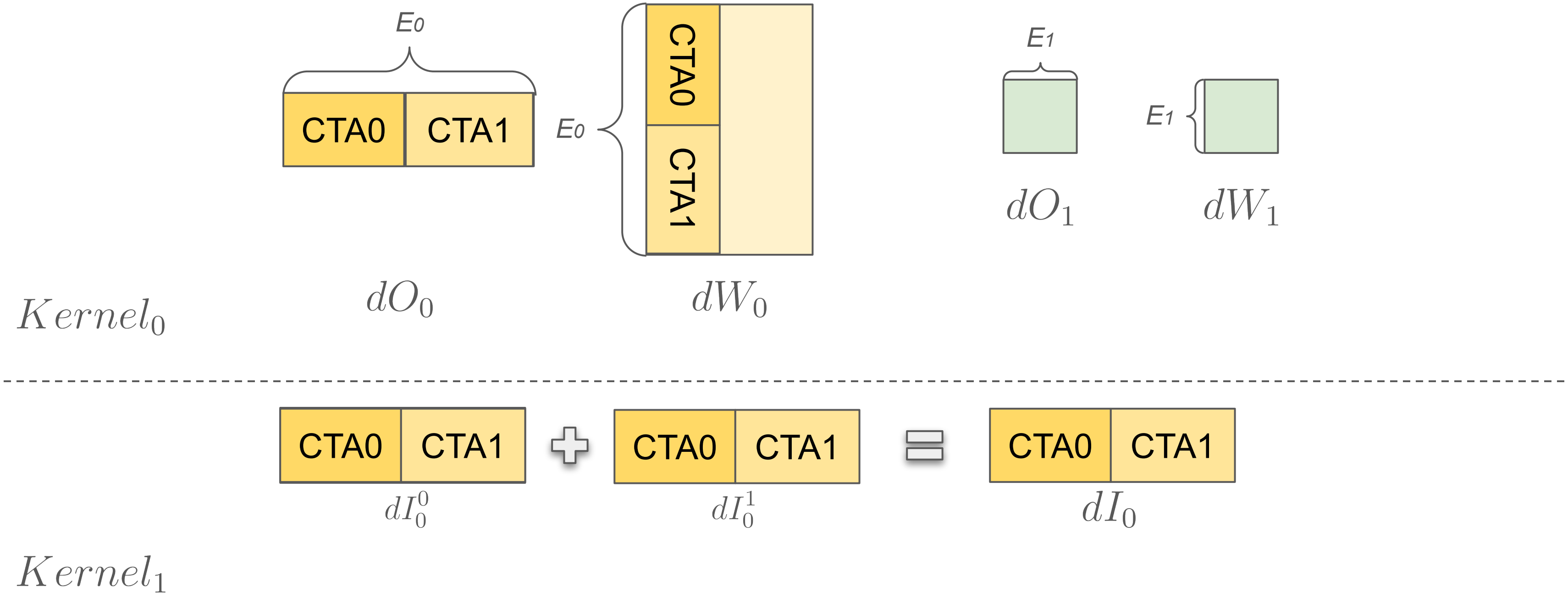
- Significant imbalance of edge types (e.g., E_0 *vs* E_1)





Backward Optimization - 3D Parallelization

- Split the K dimension across multiple CTAs and accumulate



Performance Modeling

$Occupancy$

$Utilization$

$$Time_{total} = N_{waves} \times Time_{wave}$$

$Time_{indexing}$

$Time_{load}$

$Time_{compute}$

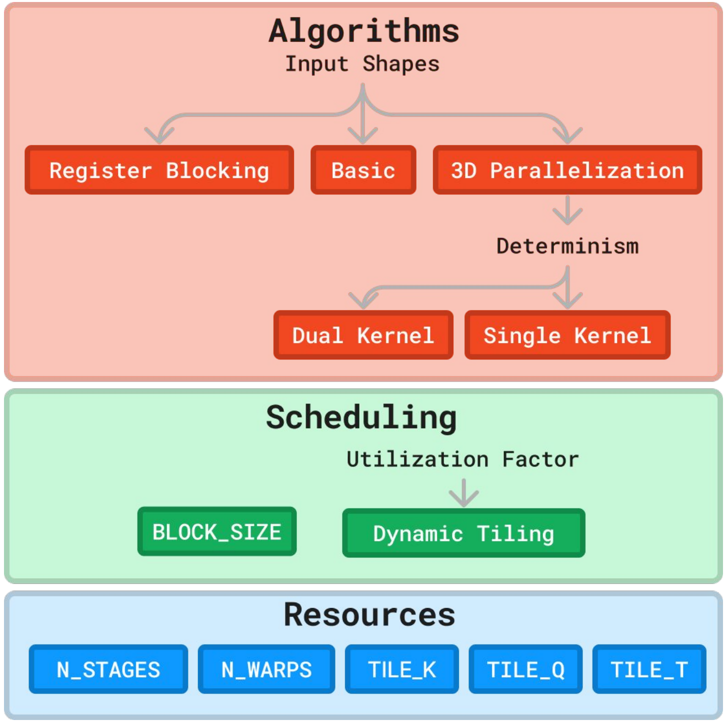
$Time_{store}$

Parallel Efficiency:
The ratio of ideal number of waves to N_{waves}

Compute Efficiency:
The ratio of $Time_{compute}$ to $Time_{wave}$

Multi-level Autotuning

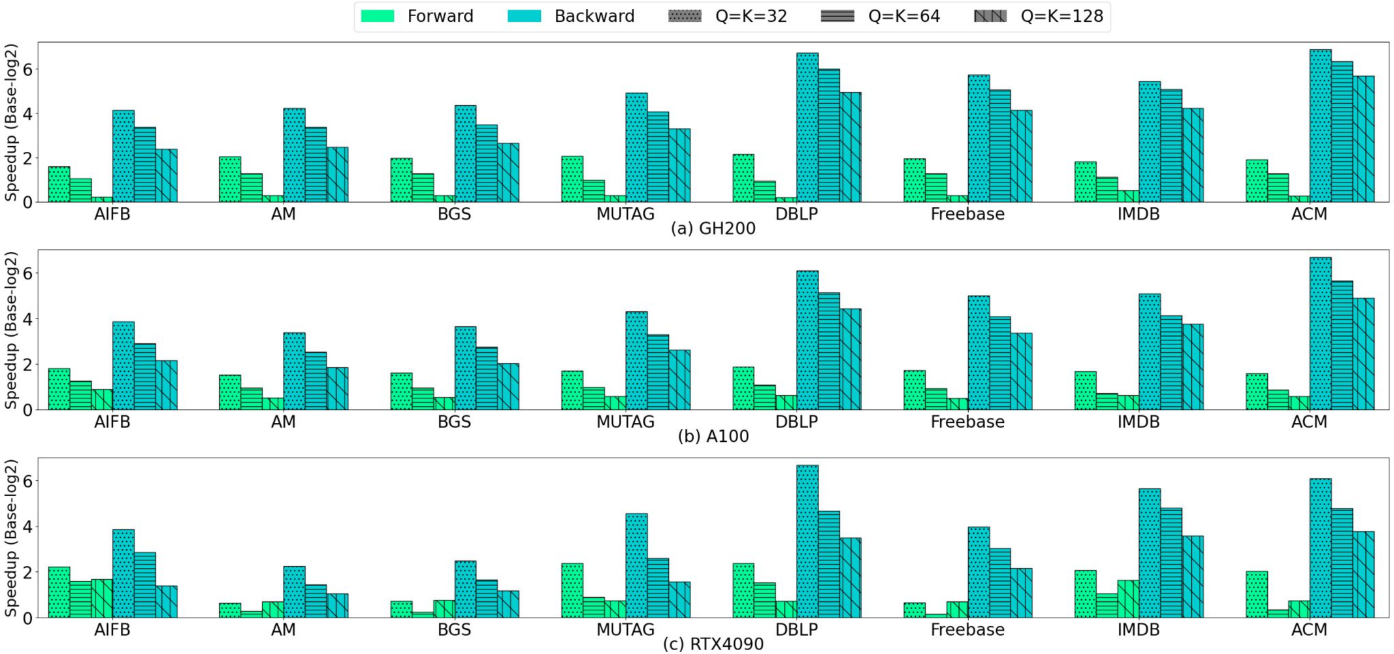
- Configuration keys
  - Average number of edges
  - Standard deviation of edges
  - Feature size
- Configuration pruning
  - Resource-based
  - Heuristic-based
  - Shape-based
  - Efficiency-based
  - Algorithm-based



Experiments - Setup

- GH200
  - 96GB GPU Memory, 132 SMs, 989 TF32 TFLOP/s, 4TB/s Bandwidth
- A100 SXM
  - 80GB GPU Memory, 108 SMs, 156 TF32 TFLOP/s, 2TB/s Bandwidth
- RTX4090
  - 24GB GPU Memory, 128 SMs, 82.6 TF32 TFLOPS, 1TB/s Bandwidth
- Eight heterogeneous graphs
  - AIFB, AM, BGS, MUTAG, DBLP, Freebase, IMDB, ACM

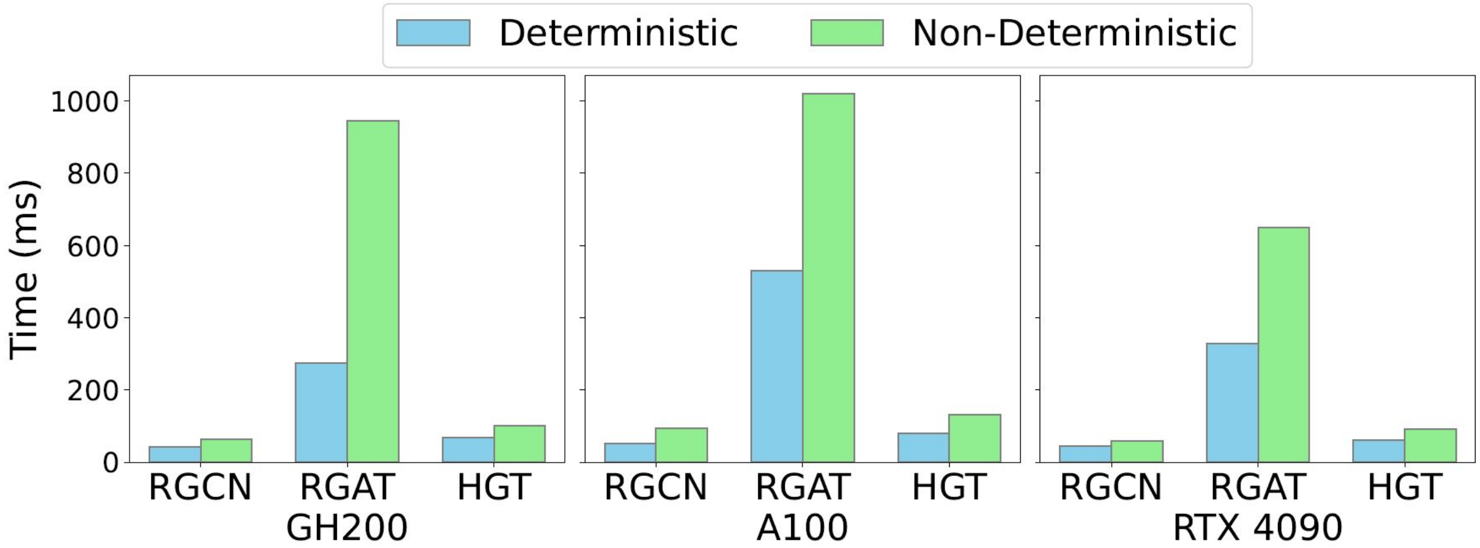
Experiments - Operators



FASTEN *vs* CUTLASS

Forward: 1.11x-5.21x speedup; Backward: 2.07x-117.54x speedup

Experiments - End-to-end



FASTEN *vs* PyG

Up to 3.53x speedup

Takeaway

- Operator optimization is critical for heterogeneous graphs
- Key innovations
  - Tile-based routing table
  - 3D Parallelization
  - Multi-level autotuning
- Deep-Learning-Profiling-Tools/fasten (github.com)