

Kevin-32B V.S. CudaForge(ours)

Initialization

- Input: {Model Arch in Python}
- Output: {ModelNew}
- {CUDA Strategies}
- {Requirements}



CUDA_KERNEL matmul_atb_kernel
 INPUT: A ($K \times M$), B ($K \times N$)
 OUTPUT: C ($M \times N$) = $A^T \times B$

GRID: (ceil_div(N,16), ceil_div(M,16)) blocks
 BLOCK: 16×16 threads
 FOR each thread (global row i = blockIdx.y*16 + ty, global col j = blockIdx.x*16 + tx):

Kevin-32B

Round 1

Compile and evaluate this kernel ✓

The performance for this kernel is:
{Speedup}.
 Please analyze the provided kernel and try to produce an improved CUDA kernel.

Compile and evaluate this kernel ✓

Let's use NCU to see what happened while the kernel was running.



GPU PROFILING RESULTS:
 - SM Utilization: 91.94% warp occupancy
 - Compute Load: 2.06M active cycles, 444.7M instructions
 - Occupancy Bottleneck: Register-limited (4 vs 16 blocks)

CudaForge(ours)

Round 2

CUDA_KERNEL matmul_atb_optimized:
 INPUT: A ($K \times M$), B ($K \times N$)
 OUTPUT: C ($M \times N$) = $A^T \times B$

GRID: (M/32, N/32) blocks
 BLOCK: 32×32 threads
 SHARED_MEM: tile_A[32×33], tile_B[32×33] // +1 padding

FOR each thread(i,j):
 $C[i,j] = \sum_{k=0 \text{ to } K-1} LDG(A[k,i]) \times LDG(B[k,j])$

FIX MEMORY BOTTLENECK (24% stalls):
 1. Add shared memory tiling: ($K^3 \times C_i \times \text{float}$) \leq 64KB
 2. Cooperative load + sync + unroll loops
 3. Reduce registers to ≤ 32 for 8×128 blocks/SM



Generating the improved kernel following the instruction from Judge...

Human

Coder



NCU



Testing



Judge