15745 Milestone: Hardware-aware Compilation of Tensor Programs for Low-Precision Computing

Jinqi Chen, Zhibo Chen
jinqic@andrew.cmu.edu, zhiboc@andrew.cmu.edu

https://www.andrew.cmu.edu/user/zhiboc/15745-fa24/

1 Major Changes

We have made no major changes to our project so far. We are focusing on constructing the scheduling policy for low-precision matrix multiplication, specifically INT4 x FP16 case.

2 What we have accomplished so far.

We successfully downloaded and ran the BitBLAS benchmark on NVIDIA GPUs. This includes diving into the tiny little details of the benchmarks. This leaves us having a much clearer picture of the BitBLAS framework, and we propose to just improve two aspects of a single kind of benchmark: speed and accuracy.

We identified some key insights and limitations in current scheduling framework. Considering the complexity of low-precision GEMM kernel optimization, we extensively studied tutorials such as CUTLASS and investigated CUDA's instructions in depth. After we get a feel of the key performance aspects of INT4 x FP16 kernel, we are in the process of redesigning and rewriting the scheduling policy with good insights.

3 Meeting Your Milestone.

We met our 75% goal in "understanding how current scheduler works by changing memory layouts and access patterns, and knowing clearly what optimization contributes to the performance gain on hardware accelerators." Our progress in implementing a new scheduler has been slightly delayed, as devising research-level improvements over the current framework presents significant challenges, but we managed to gain fruitful insights in the process.

Towards the 100% goal, we are now implementing better scheduling policy and profiling on GEMM.

4 Surprises.

Initially, the framework contains many bugs for us to fix. We then discovered that it does not automatically use MMA on H100 GPUs, resulting a running time 4x slower than torch.matmul. Regarding dequantized MMA policies, we discovered an interesting insight. One policy, which fetches quantized weight to shared memory and dequantize in registers, then stores the dequantized result to shared memory for future compute, takes a longer time but has a higher accuracy. Another policy, which prefetch quantized weight to shared memory using async copy and load to warp memory for dequantize, takes 35% shorter time, but has varying results in every run. We suspect that this policy has some synchronization issues (does not insert barrier properly, for example).

5 Revised Schedule

We plan to concentrate our following work fully on writing the scheduling policy and squeezing the last bit of performance out of the compiled kernel.

6 Resources Needed.

None. We have stable access to H100 GPUs which guarantees our work.