Profiling Triton Kernels for PyTorch 2

Shunting Zhang, Horace He, Yang Chen
PyTorch

Inductor Triton Backend

- Inductor generates Triton kernels for GPUs
  - Triton kernel templates: mm, bmm, conv, etc
  - codegen-ed Triton kernels: fused pointwise, reduction
- Auto-tuning
  - select the best kernel implementation from multiple backends
  - determine the best kernel configs

Triton Kernel Performance

- Improving the performance of Triton kernels is important
- Two common sources of performance issues
  - Inductor may generate less-than-ideal Triton kernels
  - The Triton compiler may miss optimization opportunities

Profiling Triton Kernels

- Inductor profiling utilities
  - standalone benchmarks for individual Triton kernels
  - Inductor’s built-in bandwidth profiler
  - custom ncu-based profiler
- Low device memory bandwidth may indicate optimization opportunities

Inductor Built-in Bandwidth Profiler

- Estimate memory traffic by counting the number of input and output bytes
  - overestimate memory traffic when only part of inputs/outputs are accesses
  - underestimate memory traffic when inputs/outputs need to be accessed multiple times, e.g. softmax with large reduction numel
- No extra dependency
  - Profiling data generated along the auto-tuning process

Custom ncu-based Profiler

- More accurate memory bandwidth data
  - reads profiling metrics from ncu’s output
- More dependencies
  - ncu
  - extra setup such as custom profiling scripts