Lab 11: GPU Programming - Mixture of Experts

Moreh Vietnam

Mixture of Experts

Mixture of Experts (MoE) is a neural network architecture designed to increase model capacity without proportionally increasing computation.

It works by activating only a subset of "expert" sub-networks for each input, making it highly efficient for large-scale models like LLMs (i.e. Mistral, DeepSeek).

Mixture of Experts

Structure: Input → Router → Top-k Experts Selected → Expert Processing → Weighted Output

Input Tokens: [Token1, Token2, Token3]

↓ ↓ ↓

Router: [Expert1: 0.7, Expert3: 0.3] (Token1)

[Expert2: 0.9, Expert1: 0.1] (Token2)

[Expert3: 0.6, Expert2: 0.4] (Token3)

↓ ↓ ↓

Experts: Expert1 processes Token1 (weight=0.7) + Token2 (weight=0.1)

Expert2 processes Token2 (weight=0.9) + Token3 (weight=0.4)

Expert3 processes Token1 (weight=0.3) + Token3 (weight=0.6)

↓ ↓ ↓

Output: Weighted sum of expert outputs per token.

Problem

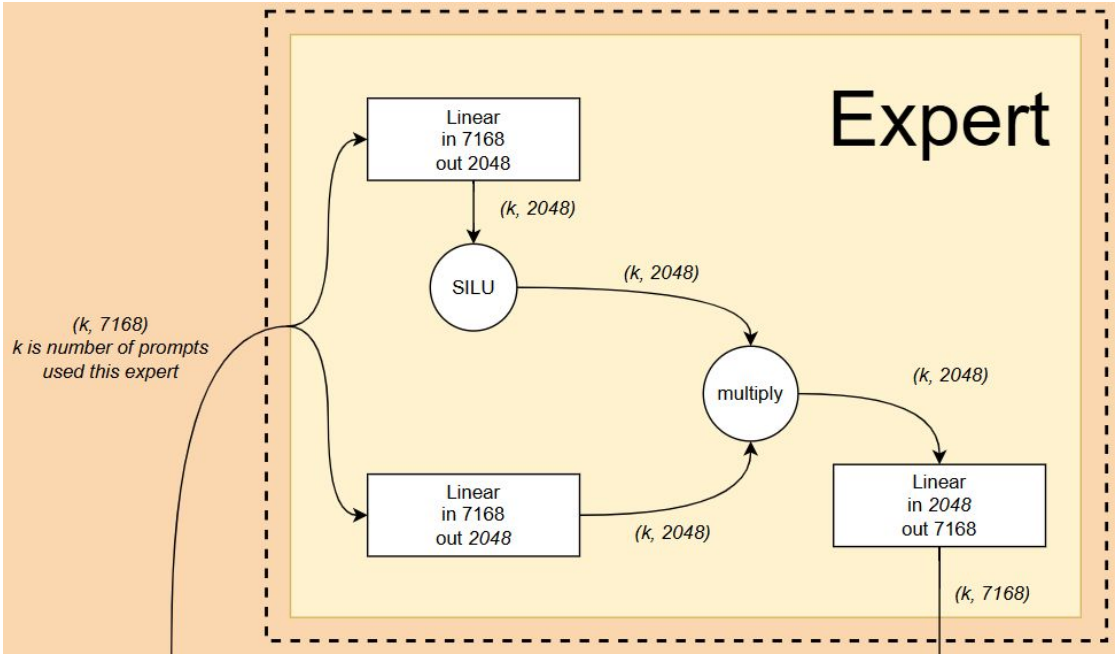
The pain point of MoE is expert processing (via a Feed Forward layer) and summing for output.

Your task is to implement this part in C/C++ and optimize it.

The baseline is an implementation using Pytorch.

Feed Forward Layer

Here is an example using DeepSeek V3 configuration



Practice

You are given a zip file “lab11_moe.zip” containing the following files:

- “README.md”: A note for installation, testing and benchmarking
- “setup.py”: A script used for compiling and binding your code into Python package
- “test_moe.py”: Unit testing and performance benchmarking script
- “csrc/moe.cpp”: A sample script. You should modify this file.

Your task is to **implement and optimize** Mixture of Experts module on **one MI250 GPU** using this template. You must also submit **a report** describing your algorithm and technique used.

Your work must be done inside the “csrc” folder, you can modify “moe.cpp”, or adding more files for better coding experience. On the other hand, any modifications outside may causes errors during evaluation, which can lead your final score to 0.

There are almost no limitations of technique used. However, it’s illegal for using 3rd party library in your code (you can use **torch::empty()**).

Practice

For compilation, binding, and installation:

- `srun --time=05:00 --pty --gres=gpu:0 python3 setup.py develop --install-dir ./install`

For testing and benchmarking:

- `srun --time=02:00 --pty --gres=gpu:1 python3 test_moe.py # For testing and benchmarking`
- `TEST_ONLY=1 srun --time=02:00 --pty --gres=gpu:1 python3 test_moe.py # For testing`
- `BENCHMARK_ONLY=1 srun --time=02:00 --pty --gres=gpu:1 python3 test_moe.py # For benchmarking`

For development:

- **Using HIP instead of CUDA**
- **torch::empty() for memory allocation on GPU**

Practice Server

- Time limit for srun has been extended to 5 minutes.
- If there are any complain about shortage of computation resources, the time limit will be reduced.
- You can upload or download your script to practice server with `scp`

Evaluation

This is a performance competition. The evaluation will be processed on a **MI250** machine. Therefore, I suggest working on practice server.

2 points for your report.

There are 10 test cases in “test_moe.py”, at most 1 point for each case, **8 points** in total.

For each test case with correct result (diff less than 5%):

- Faster than naive implementation by 10%: **0.4 point**
- Best implementation in class: **0.4 point**
  - Other students get points adjusted for their implementation performance relative to the best and the naive implementation.
- Slower than naive implementation: **Nothing**

Submission

A zip file with similar structure with “lab11_moe.zip”, named “<your_student_id>_moe.zip”.

Your report should be included in the zip file too.

Due date: 6AM 28/04/2025