AMD Developer Cloud Resources:

1. Create an account & login to https://amd.digitalocean.com/
2. Create team and add collaborators
3. Request additional credits to try out AMD GPUs

A Simple Exercise:

export HIP_VISIBLE_DEVICES="0/1/2/3/4/5/6/7"

python -m sglang.bench_offline_throughput --model-path /data/mixtral / --dataset-name random --num-prompts 256 --random-input-len 256 --random-output-len 256 --random-range-ratio 1 --tp-size 1 --trust-remote-code --disable-cuda-graph --disable-radix-cache

Docker Environment [SGLang]:

docker pull lmsysorg/sglang:v0.5.5-rocm700-mi30x

docker run -it --rm --ipc=host --cap-add=SYS_PTRACE --network=host --device=/dev/kfd --device=/dev/dri -v /data:/data/ --security-opt seccomp=unconfined --group-add video --privileged lmsysorg/sglang:v0.5.5-rocm700-mi30x

Deepseek Offline Scenario:

python -m sglang.bench_offline_throughput --model-path /data/deepseek-v3/ --dataset-name random --num-prompts 256 --random-input-len 256 --random-output-len 256 --random-range-ratio 1 --tp-size 8 --trust-remote-code --disable-cuda-graph --disable-radix-cache

Deepseek Server Scenario:

TP attention & TP MoE:

nohup python3 -m sglang.launch_server --model /data/deepseek-v3/ --port 3000 --trust-remote-code --disable-cuda-graph --disable-radix-cache --tp-size 8 > deepseek_server.log 2>&1 &

python3 -m sglang.bench_one_batch_server --model /data/deepseek-v3/ --base-url http://localhost:3000 --batch-size 256 --input-len 256 --output-len 256

DP attention & TP MoE:

nohup python3 -m sglang.launch_server --model /data/deepseek-v3/ --port 3000 --trust-remote-code --disable-cuda-graph --disable-radix-cache --tp-size 8 --enable-dp-attention --dp-size 8 > deepseek_server.log 2>&1 &

python3 -m sglang.bench_one_batch_server --model /data/deepseek-v3/ --base-url http://localhost:3000 --batch-size 256 --input-len 256 --output-len 256

SGLang AITER Flags:

sudo sh -c \\'echo 0 > /proc/sys/kernel/numa_balancing

export SGLANG_USE_AITER=1

Docker environment [vLLM]:

docker pull rocm/vllm:latest

docker run -it --rm --ipc=host --cap-add=SYS_PTRACE --network=host --device=/dev/kfd --device=/dev/dri -v /data:/data/ --security-opt seccomp=unconfined --group-add video –privileged rocm/vllm:latest

Mixtral 8x7B Offline Scenario:

vllm bench throughput --model /data/mixtral/ --dataset-name random --num_prompts 256 --input-len 256 --output-len 256 --enforce-eager --tensor-parallel-size 8

Mixtral 8x7B Server Scenario:

TP attention + TP MoE

nohup vllm serve /data/mixtral/ --port 8000 --tensor-parallel-size 8 --max-num-batched-tokens 8192 --gpu-memory-utilization 0.9 --enforce-eager --no-enable-prefix-caching > mixtral_server.log 2>&1 &

vllm bench serve --backend vllm --model /data/mixtral/ --dataset-name random --num-prompts 4096 --random-input-len 256 --random-output-len 256

TP attention + EP MoE

nohup vllm serve /data/mixtral/ --port 8000 --tensor-parallel-size 8 --enable-expert-parallel --max-num-batched-tokens 8192 --gpu-memory-utilization 0.9 --enforce-eager --no-enable-prefix-caching > mixtral_server.log 2>&1 &

vllm bench serve --backend vllm --model /data/mixtral/ --dataset-name random --num-prompts 4096 --random-input-len 256 --random-output-len 256

vLLM AITER Flags:
sudo sh -c \\'echo 0 > /proc/sys/kernel/numa_balancing
export VLLM_USE_AITER=1
export VLLM_USE_AITER_MOE=1
export VLLM_USE_AITER_2STAGE_MOE=1

Profiling:

GPU Counters:

```
cuda_start = torch.cuda.Event(enable_timing=True)
cuda_end = torch.cuda.Event(enable_timing=True)
cuda_start.record()

{code_to_profile}

cuda_end.record()
torch.cuda.synchronize()
torch.distributed.barrier()

latency = cuda_start.elapsed_time(cuda_end)
```

Torch Profiler:

torch.profiler — PyTorch 2.9 documentation

```
with torch.profiler.profile(
   activities=[
      torch.profiler.ProfilerActivity.CPU,
      torch.profiler.ProfilerActivity.CUDA,
   ]
) as p:

   {code_to_profile}

print(p.key_averages().table(sort_by="self_cuda_time_total", row_limit=-1))
```