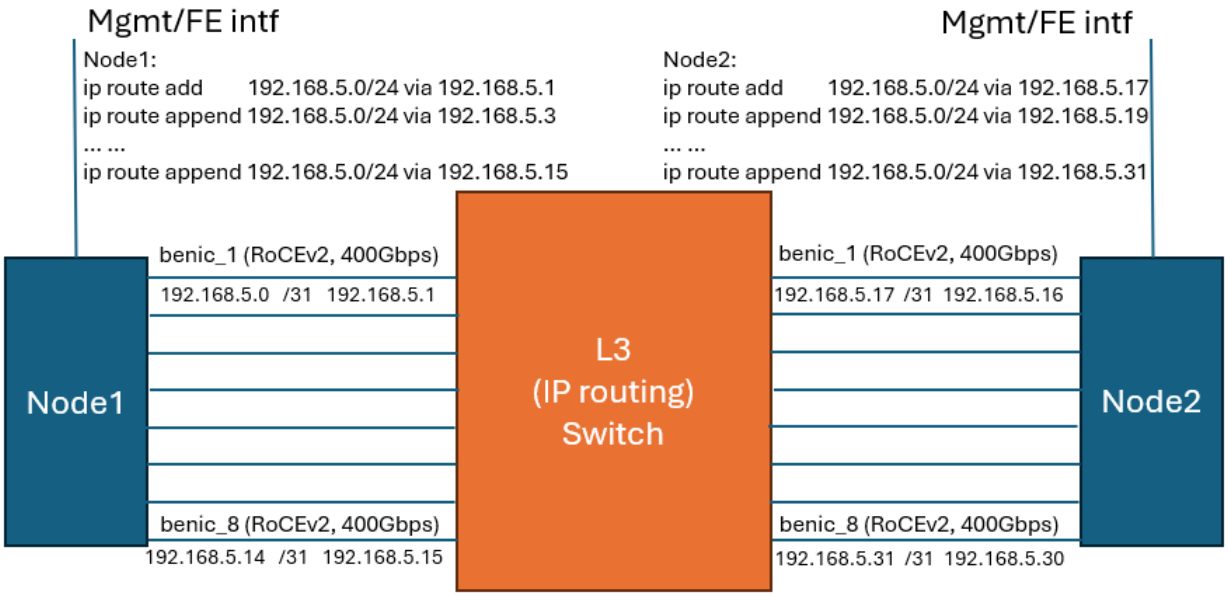
AINIC SGLANG PD mooncake Instruction

Topology (reference):

FE – Front End (for admin/control-plane traffic)
BE – Back End (for serving/data-plane traffic)



L3 Routing entries on Server Host (docker container run with Network=Host)

- On each server node, add 8 static route pointing to the common prefix (network address) from all BE interfaces of all to be interconnected server nodes, one per local BE interface as next hop. (Example above)
- IP connectivity test, from a given server node ping a BE interface IP of any other node supposed to be connected (in the same service pod), PING should come through. (If not check server firewall config, and switch L2/L3/FW config)

Docker Images

rocm/sgl-dev:v0.5.2-rocm700-mi35x-20250915-rc

AINIC driver updates

follow Pensando or field engineer instructions

When using docker, mount/map the updated driver into container:

MI350X (change PATH_TO_YOUR_IONIC_DRIVER to your driver path):

```
sudo docker run --rm -it \
  --network=host \
  --ulimit memlock=-1 \
  --cap-add=IPC_LOCK \
  --cap-add=SYS_PTRACE \
  --security-opt seccomp=unconfined \
  --privileged \
  --shm-size=32g \
  --group-add video \
  --group-add rdma \
  -e LD_LIBRARY_PATH=/opt/rocm/lib:/usr/local/lib \
  -v /sys:/sys \
  -v /vfs:/vfs \
  -v /apps:/apps \
  -v /dev/infiniband:/dev/infiniband \
  -v /sys/class/infiniband:/sys/class/infiniband:ro \
  -v /sys/class/net:/sys/class/net:ro \
  -v /sys/bus/pci:/sys/bus/pci:ro \
  -v /usr/lib/x86_64-linux-gnu/libibverbs/libionic-rdmav34.so:/usr/lib/x86_64-linux-gnu/libibverbs/libionic-rdmav34.so:ro \
  -v /etc/libibverbs.d:/etc/libibverbs.d:ro \
  -v /{PATH_TO_YOUR_IONIC_DRIVER}/ionic_driver/src/drivers-linux/rdma-
core/build/lib/libionic.so.1:/apps/shared/vijasrin/images/ainic_bundle_1.117.1-a-32/host_sw_pkg/ionic_driver/src/drivers-linux/rdma-
core/build/lib/libionic.so.1 \
  rocm/sgl-dev:v0.5.2-rocm700-mi35x-20250915-rc bash
```

Instructions to run 1P1D (for simplicity, using topology above)

- Proxy command (run at where request sent, maybe standalone node, or P, D node)

```
set -x
# change this to your prefill and decode ip, you will set them up in the prefill command and decode command.
IP_FIRST_PREFILL=192.168.5.0
IP_FIRST_DECODE=192.168.5.16

echo "run load balance"
python3 -m sglang_router.launch_router --pd-disaggregation --prefill http://${IP_FIRST_PREFILL}:30025 \
            --decode http://${IP_FIRST_DECODE}:30025 --host 0.0.0.0 --port 30028
```

- P(refill) command (run at P node)

```
MODEL_PATH=$1
# Use this ip in your proxy launch
host_ip=192.168.5.0
export LD_LIBRARY_PATH=/opt/rocm/lib:/usr/local/lib
export SGLANG_USE_AITER=1
# Your --disaggregation-ib-device should be the rdma you want to use
python -m sglang.launch_server --model-path $MODEL_PATH \
        --disaggregation-mode prefill --port 30025 --disaggregation-ib-device ionic_0,ionic_1  \
        --host ${host_ip} --tp 8 --disable-radix-cache --trust-remote-code \
        --disable-cuda-graph \
        --enable-two-batch-overlap --attention-backend aiter --log-level debug
```

- D(ecode) command (run at D node)

```
MODEL_PATH=$1
# Use this ip in your proxy launch
host_ip=192.168.5.16
export LD_LIBRARY_PATH=/opt/rocm/lib:/usr/local/lib
export SGLANG_USE_AITER=1
# Your --disaggregation-ib-device should be the rdma you want to use
python -m sglang.launch_server --model-path $MODEL_PATH \
    --disaggregation-mode decode --port 30025 --disaggregation-ib-device ionic_0,ionic_1 \
      --host ${host_ip} --tp 8  --disable-radix-cache --trust-remote-code \
      --disable-cuda-graph  --log-level debug \
        --enable-two-batch-overlap --mem-fraction-static 0.8 --attention-backend aiter
```

- Accuracy check (run on same place as Proxy command)

```
python3 /sgl-workspace/sglang/benchmark/gsm8k/bench_sglang.py --num-questions 1000 --parallel 100 --host
http://0.0.0.0 --port 30028
```

- Performance check (run on same place as Proxy command)

```
#!/bin/bash
set -euo pipefail
set -x
echo "running bench"
num_prompt=128
input_sizes=(1000 2000 3000)
output_sizes=(600 800 1000)
request_rates=(0.1 0.2 0.5 1 2 4)
for input_size in "${input_sizes[@]}"; do
 for output_size in "${output_sizes[@]}"; do
  for request_rate in "${request_rates[@]}"; do
   echo "num_prompt=${num_prompt}, input_size=${input_size}, output_size=${output_size}, request_rate=${request_rate}"
   python3 -m sglang.bench_serving \
    --model /apps/data/models/DSR1/ \
    --dataset-name random \
    --host 0.0.0.0 \
    --port 30028 \
    --num-prompts "${num_prompt}" \
    --random-input-len "${input_size}" \
    --random-output-len "${output_size}" \
    --random-range-ratio 1 \
    --request-rate "${request_rate}" \
    --extra-request-body '{"chat_template_kwargs":{"enable_thinking":false}}'
  done
 done
done
```