https://astra-sim.github.io

https://github.com/mlcommons/chakra

ASTRA-sim Tutorial
@MICRO 2024
Nov 3rd, 2024

ASTRA-Sim and Chakra Tutorial:
Demo – Supplementary: The NS3 Network Backend

Jinsun Yoo
Ph.D Student
School CS, Georgia Institute of Technology
jinsun@gatech.edu

Available Network Backends

- Network backends are maintained separately and imported as **submodule**.
- We currently have **2 network backends** which implement NetworkAPI

Backend	Purpose	Notable Feature
analytical/analytical	analytical equation-based simulation	fast simulation, hierarchical topologies
ns-3	inter-network simulation	large parallel GPU clusters

NS3 Simulator: https://github.com/astra-sim/astra-network-ns3/

- Based on study for Congestion Control on general-purpose RDMA network
  - Yuliang Li, et. al,. PCC: High Precision Congestion Control (SIGCOMM' 2019)
- Packet level, event-based simulator modelling network features such as:
  - RDMA, Congestion Control, Load Balancing (ECMP), etc
  - On arbitrarily defined topologies

Cloning Tutorial Repository

- We provide sandboxed Chakra/ASTRA-sim for tutorial purposes

```
$ git clone git@github.com:astra-sim/tutorials.git
$ cd tutorials/micro2024
$ git rev-parse --short HEAD
ead2d5d
$ ./clone_repos.sh
```

Launching Execution Environment (Docker)

- Download Docker Image

```
$ docker pull astrasim/tutorial-micro2024
```

- Start a Docker Container and **link current directory into**

```
$ docker run -i -t \
      -v $(pwd):/app/micro2024 \
      astrasim/tutorial-micro2024
[docker]$ cd ../micro2024/
```

Mount

Mounted directory

Install Chakra

- Install Chakra utilities that comes with ASTRA-sim

```
[docker]$ ./install_chakra.sh
```

install_chakra.sh:

```
$ cd astra-sim/extern/graph_frontend/chakra
$ pip3 install .
$ pip3 install --upgrade protobuf
```

Compile ASTRA-sim

- Compile ASTRA-sim with analytical & ns3 network backend
  - Different network backend = different binaries

```
[docker]$ ./compile_astra_sim.sh
          ./compile_astra_sim_ns3.sh
```

compile_astra_sim.sh:

```
$ cd astra-sim/build/astra_analytical
$ ./build.sh
```

compile_astra_sim_ns3.sh:

```
$ cd astra-sim
$ ./build/astra_ns3/build.sh
```

Running Simulation: 1D Ring across Fat-Tree

- Execute ASTRA-sim Simulation on NS3

```
[docker]$ cd astra-sim-demo/demo3; ./run_demo3-1.sh
```

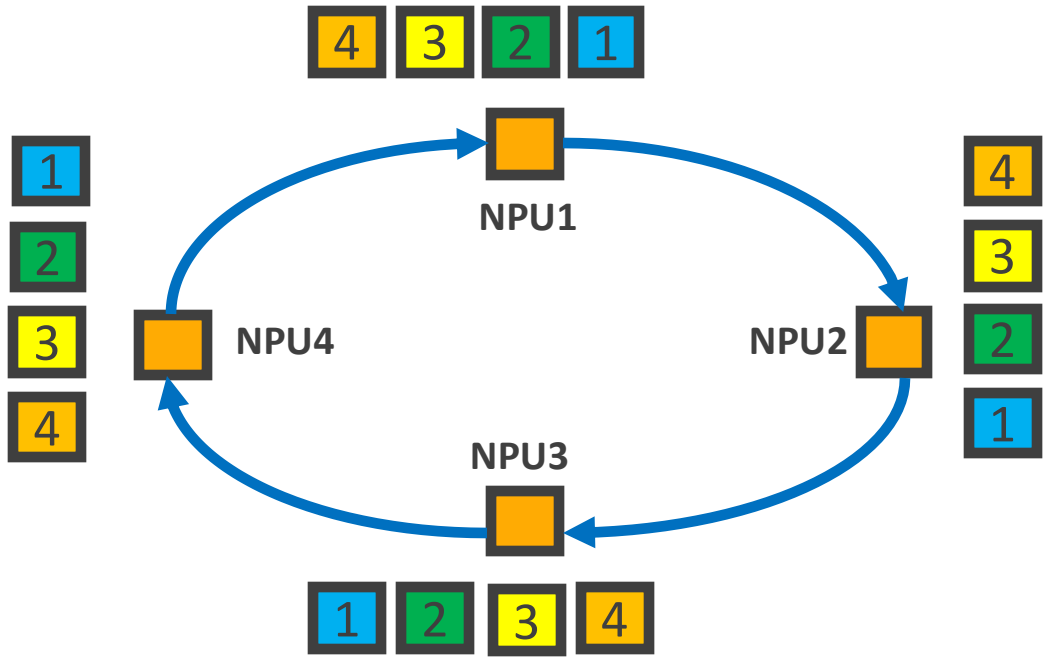
run_demo3-1.sh:

```
cd ${NS3_DIR}
./ns3.42-AstraSimNetwork-default\
    --workload-configuration=./allreduce_1D/allreduce_128 \
    --system-configuration=./inputs/Ring_sys.json \
    --network-configuration=../../../ns-3/scratch/config.txt \
    --logical-topology=./inputs/128nodes_1D.json
```

Note, For NS-3, we have a new option, '**logical-topology**'

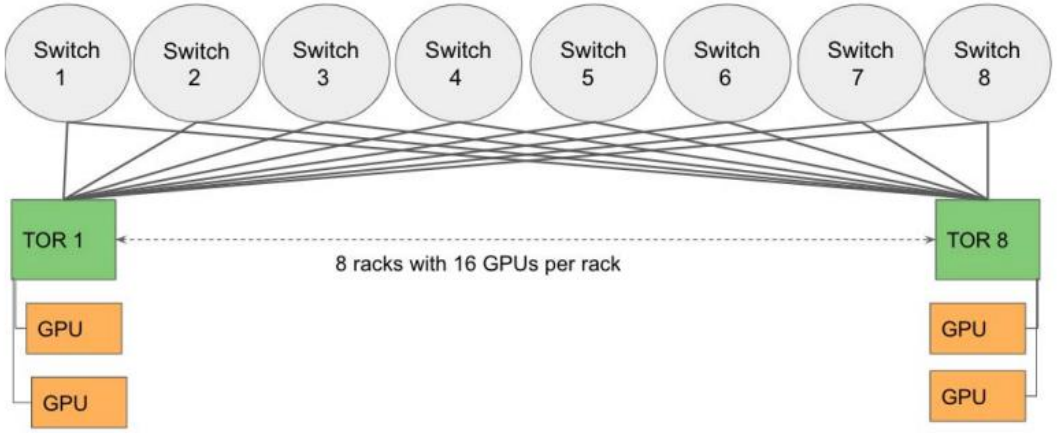
Logical Topology v. Physical Topology

Logical Topology



Which NPUs will NPU X communicate with?

Physical Topology



Actual connectivity between wires, switches, etc.

Logical Topology v. Physical Topology

Logical Topology

demo3/inputs/128_nodes_1D.json:

```
{
    "logical-dims": ["128"]
}
```

demo3/inputs/128_nodes_2D.json:

```
{
    "logical-dims": ["8", "16"]
}
```

Network(ns-3) configuration

demo3/…/config.txt:

```
TOPOLOGY_FILE \
../8_nodes_1_switch.txt
```

Physical Topology

demo3/…/8_nodes_1_switch.txt:

```
9 1 8
8
8 0 400Gbps 0.0005ms 0
8 1 400Gbps 0.0005ms 0
8 2 400Gbps 0.0005ms 0
8 3 400Gbps 0.0005ms 0
…
```

Logical Topology v. Physical Topology

In Analytical backend, logical dimensions automatically matches physical dimension

In NS-3, we decouple the logical dimension and the physical dimension

- e.g. We could run a 1D Ring AllReduce across all 128 nodes in a physical Fat-Tree topology
  And compare with a 2D Ring AllReduce

Running Simulation: 2D Ring across Fat-Tree

- Execute ASTRA-sim Simulation on NS3

```
[docker]$ ./run_demo3-2.sh
```

run_demo3-3.sh:

```
cd ${NS3_DIR}
./ns3.42-AstraSimNetwork-default\
    --workload-configuration=./allreduce_2D/allreduce_128 \
    --system-configuration=./inputs/Ring_Ring_sys.json \
    --network-configuration=../../../ns-3/scratch/config_clos.txt \
    --logical-topology=./inputs/128nodes_2D.json
```

Must use relative directory in script for network config

Physical Topology Setup

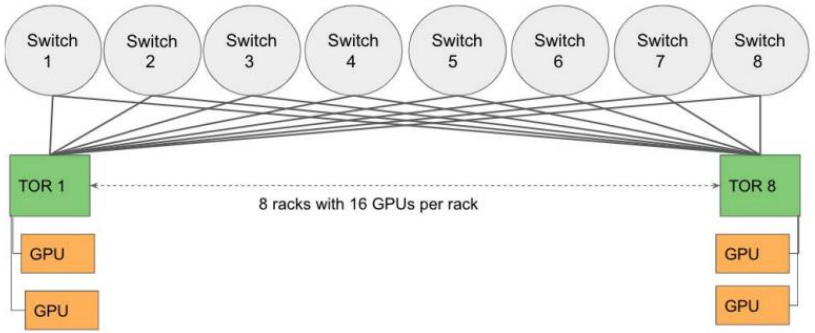
demo3/inputs/128_nodes_16_switch.txt:

Total # node(NPU) + Switch **#Switches** **#Links**

List of Switch ID

List of
- **Endpoint ID 1**
- **Endpoint ID 2**
- **Bandwidth**
- **Latency**
- **Error Rate**

```
144 16 192
128 129 130 …  142 143
0 128 200Gbps 0.005ms 0
1 128 200Gbps 0.005ms 0
2 128 200Gbps 0.005ms 0
3 128 200Gbps 0.005ms 0
128 136 200Gbps 0.0125ms 0
128 137 200Gbps 0.0125ms 0
128 138 200Gbps 0.0125ms 0
```



Physical Topology Setup

demo3/inputs/8_nodes_1_switch.txt:

Total # node(NPU) + Switch **#Switches** **#Links**

```
9 1 8
8
8 0 400Gbps 0.0005ms 0
8 1 400Gbps 0.0005ms 0
8 2 400Gbps 0.0005ms 0
8 3 400Gbps 0.0005ms 0
8 4 400Gbps 0.0005ms 0
8 5 400Gbps 0.0005ms 0
8 6 400Gbps 0.0005ms 0
8 7 400Gbps 0.0005ms 0
```

List of Switch ID

List of
- **Endpoint ID 1**
- **Endpoint ID 2**
- **Bandwidth**
- **Latency**
- **Error Rate**

Running Simulation: 1D Ring across Switch

- Execute ASTRA-sim Simulation on NS3

```
[docker]$ cd astra-sim-demo/demo3; ./run_demo3-3.sh
```

run_demo3-3.sh:

```
cd ${NS3_DIR}
./ns3.42-AstraSimNetwork-default\
    --workload-configuration=./allreduce_1D/allreduce_8 \
    --system-configuration=./inputs/Ring_sys.json \
    --network-configuration=../../../ns-3/scratch/config.txt \
    --logical-topology=./inputs/8nodes_1D.json
```