ASTRA-sim New Features

ASTRA-sim and Chakra Tutorial at MICRO '24
Nov 3, 2024

Vinay Ramakrishnaiah (Vinay.Ramakrishnaiah@amd.com)
William Won (William.Won@amd.com)
Ruchi Shah (Ruchi.Shah@amd.com)

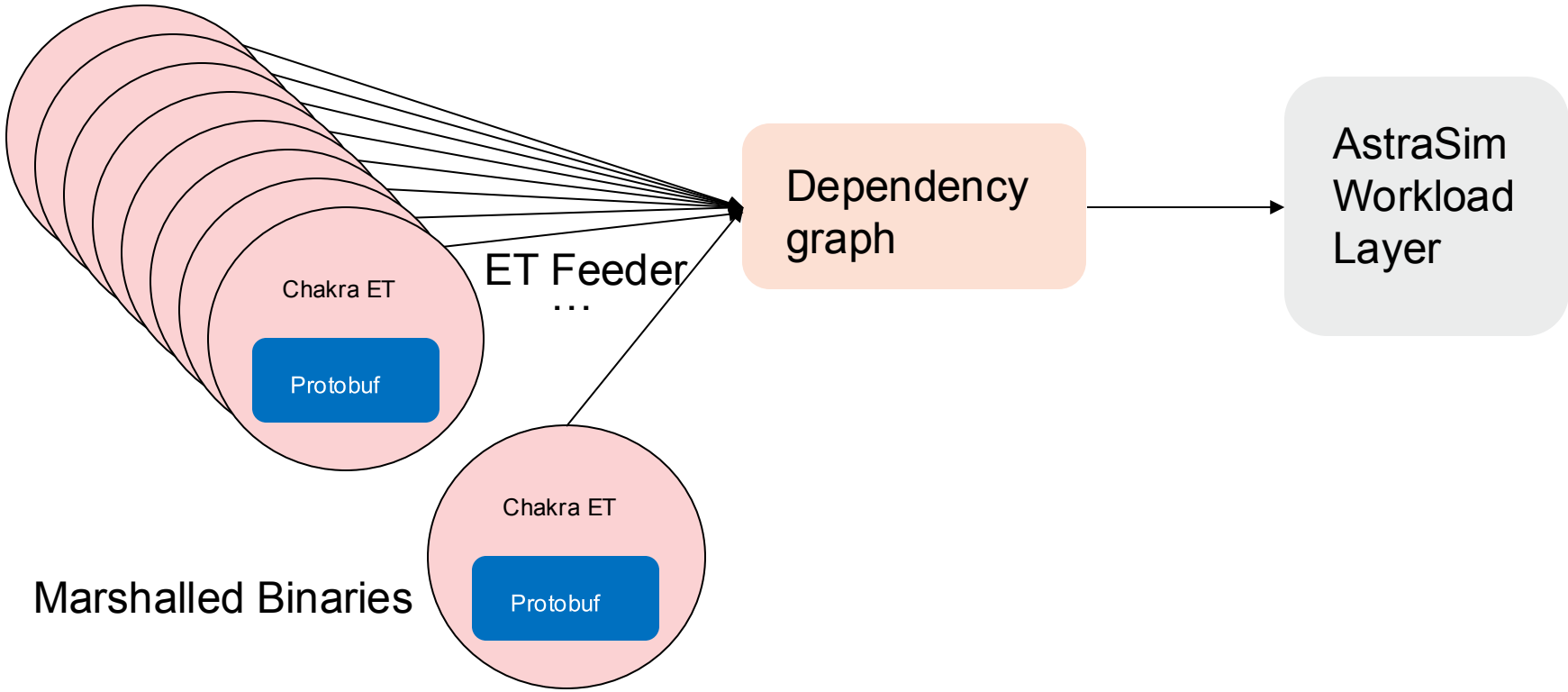
AMD Research and Advanced Development

AMD
together we advance_

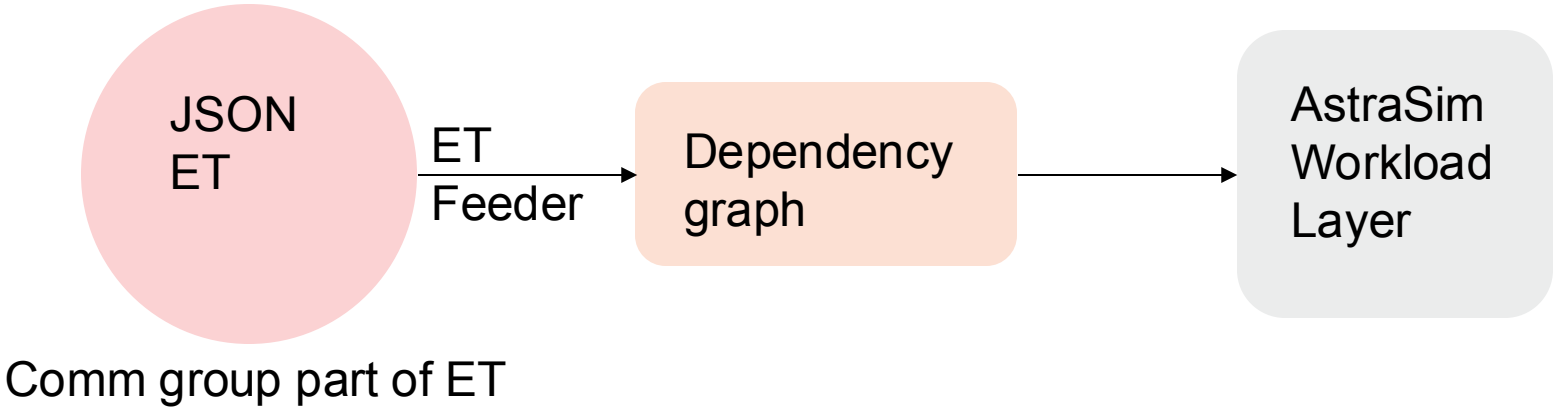
Outline

- JSON Chakra Traces
- Custom Collective Plans
- Logging and Visualization

Chakra Execution Traces (Protobuf format)



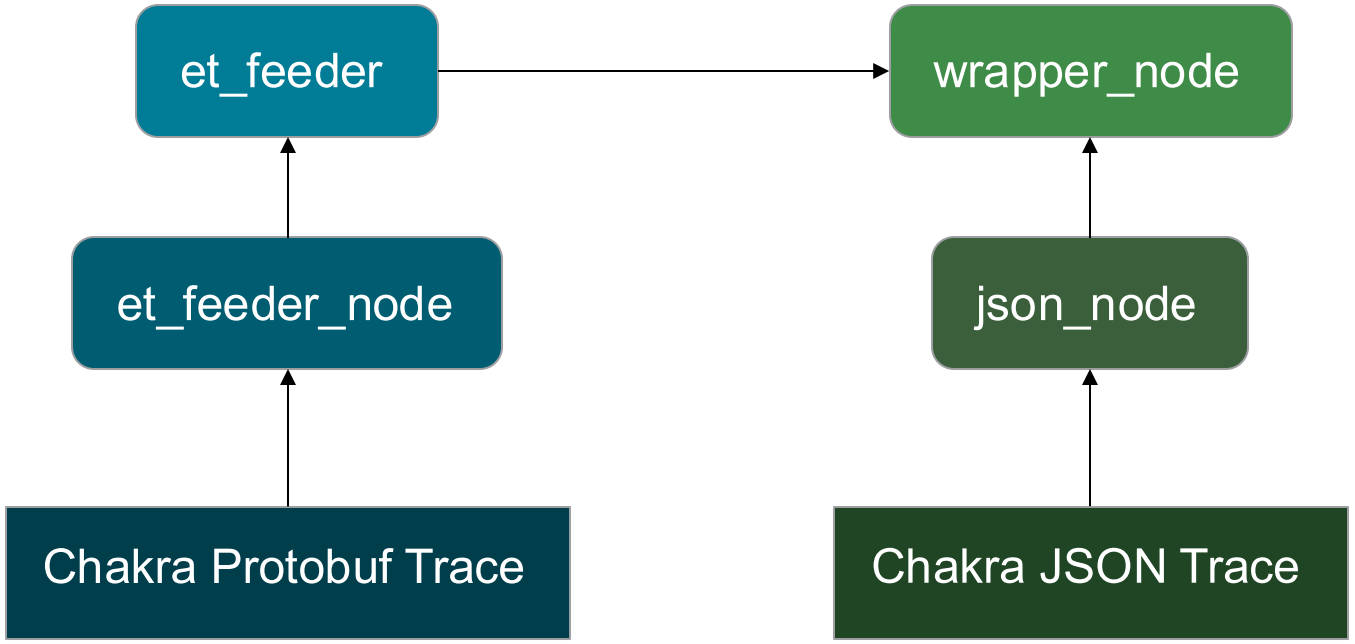
Chakra Execution Traces (JSON format)



Example JSON ET


```
{
    "workload_name": "llama13b",
    "ngpus": 1024,
    "socs": ["<List of NPUs>"],
    "{other global metadata}": "<metatdata>",
    "communicators": [
        {
            "group_id": 1,
            "involvedNPUs": [0, 1, 2, 3]
        },
        {
            "group_id": 2,
            "involvedNPUs": [4, 5, 6, 7]
        }
    ],
    "workload_graph": [
        {
            "ID": 0,
            "Name": "linear_allgather",
            "NodeType": "Communication",
            "data_deps": [],
            "comm_type": "AllGatherV",
            "CommSizeOut": [
                [
                    [10, 20, 30, 40],
                    [23, 33, 43, 53],
                    [10, 20, 30, 40],
                    [23, 33, 43, 53]
                ],
                [
                    [10, 20, 30, 40],
                    [23, 33, 43, 53],
                    [10, 20, 30, 40],
                    [23, 33, 43, 53]
                ]
            ],
            "CommTime": [[41770.0], [41770.0]],
            "CommGroups": [1, 2],
            "Device": "NIC"
        },
        {
            "ID": 1,
            "Name": "linear_gemm",
            "NodeType": "Compute",
            "data_deps": [0],
            "comm_type": "AllGather",
            "ComputeTime": [[97610.0], [97610.0], [97610.0], [97610.0], [97610.0], [97610.0], [97610.0], [97610.0]],
            "M": 16384,
            "N": 3840,
            "K": 5120,
            "Batch": 1,
            "InputType": 2,
            "OutputType": 2,
            "Device": "GPU"
```

Backward Compatibility

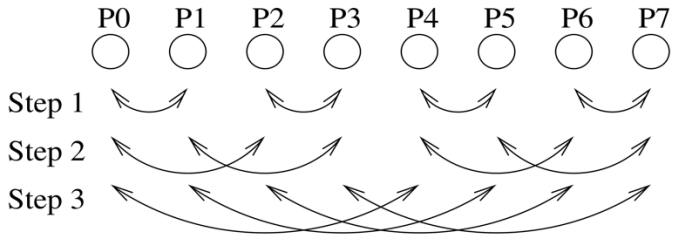


Outline

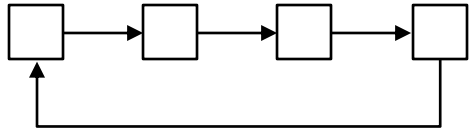
- JSON Chakra Traces
- Custom Collective Plans
- Logging and Visualization

Necessity of Custom Collective Plans

- **Latency-optimal** and **bandwidth-optimal** collective algorithm changes

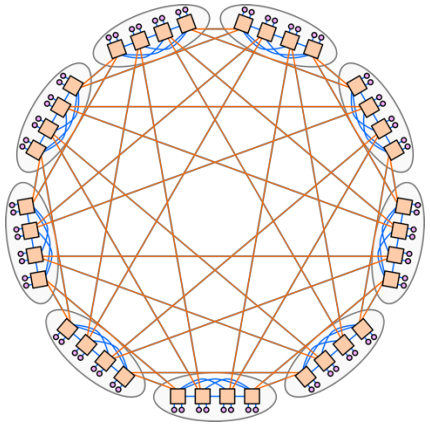


Recursive Halving-Doubling



Ring

- Distinct datacenter-scale **topology options**



DragonFly

R. Thakur et al., "Optimization of Collective Communication Operations in MPICH," in IJHPCA '05
J. Kim et al., "Technology-Driven, Highly-Scalable Dragonfly Topology," in ISCA '08

Custom Collective Plans

- **Domain Specific Language (DSL)** to describe custom collective plans
  - e.g., MSCCLang (ASPLOS '23)

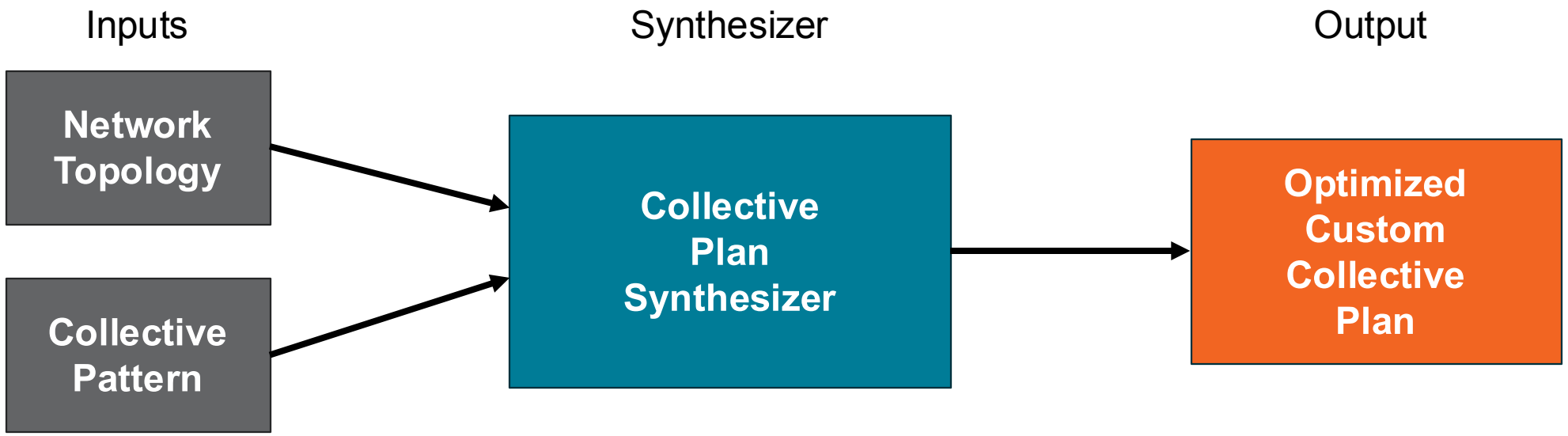
```
for step in range(0, size-1):
    for index in range(0, size):
        rank = (index + step) % size
        c = chunk(rank, Buffer.output, index)

        next_rank = (index + step + 1) % size
        channel = index % channels

        c = c.copy(next_rank, Buffer.output, index, sendtb=channel, recvtb=channel, ch=channel)
```

Example MSCCLang Representation: Ring All-Gather

M. Cowan et al., "MSCCLang: Microsoft Collective Communication Language," in ASPLOS '23

Custom Collective Plans (cont'd)

- Autonomous Collective Plan **Synthesizer**
  - e.g., TACCL (NSDI '23), ForestColl (arXiv '24), TACOS (MICRO '24)



A. Shah et al., "TACCL: Guiding Collective Algorithm Synthesis using Communication Sketches," in NSDI '23
L. Zhao et al., "ForestColl: Efficient Collective Communications on Heterogeneous Network Fabrics," in arXiv:2402.06787 [cs.NI]
W. Won et al., "TACOS: Topology-Aware Collective Algorithm Synthesizer for Distributed Machine Learning," in MICRO '24

Custom Collective Plan Representation

- **Two-sided** Collective Plan Representation: Markup (i.e., **XML** format)
  - Called *MSCCL-IR*, defined in MSCCLang (ASPLOS '23)

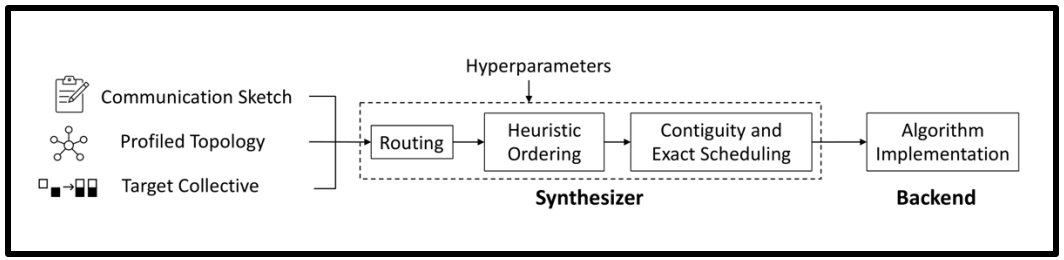
- **One-sided** Collective Plan Representation: **MSCCL++** program (work in progress)

M. Cowan et al., "MSCCLang: Microsoft Collective Communication Language," in ASPLOS '23
MSCCL++: https://github.com/microsoft/mscclpp

Custom Collective Plan Representation (cont'd)

```
for step in range(0, size-1):
  for index in range(0, size):
    rank = (index + step) % size
    c = chunk(rank, Buffer.output, index)

    next_rank = (index + step + 1) % size
    channel = index%channels

    c = c.copy(next_rank, Buffer.output, index, sendtb=channel, recvtb=channel, ch=channel)
```

MSCCLang DSL



TACCL Synthesizer

lowering

synthesize

Custom Plan (in XML)

M. Cowan et al., "MSCCLang: Microsoft Collective Communication Language," in ASPLOS '23
A. Shah et al., "TACCL: Guiding Collective Algorithm Synthesis using Communication Sketches," in NSDI '23

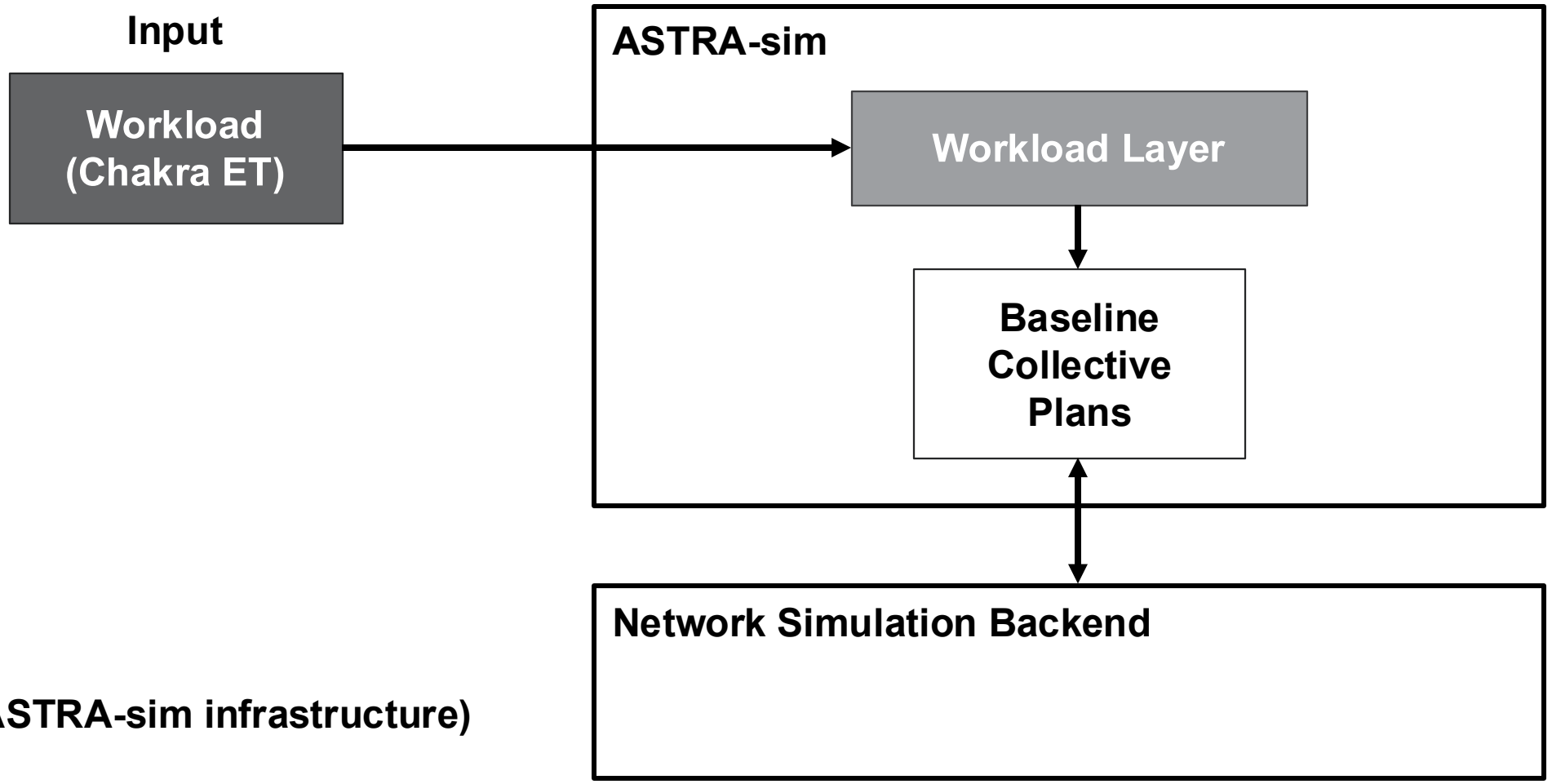
Custom Collective Plan in XML Representation

- Encapsulates all two-sided communication information for every GPU/workgroup (threadblock)

```
<algo name="allgather_allpairs" proto="LL" nchannels="1" nchunksperloop="4" ngpus="4">
  <gpu id="0" i_chunks="0" o_chunks="4" s_chunks="0">
    <tb id="0" send="1" recv="1" chan="0">
      <step s="0" type="s" srcbuf="o" srcoff="0" dstbuf="o" dstoff="0" cnt="1" depid="-1" deps="-1" hasdep="0"/>
      <step s="1" type="r" srcbuf="o" srcoff="1" dstbuf="o" dstoff="1" cnt="1" depid="-1" deps="-1" hasdep="0"/>
    </tb>
    <tb id="1" send="2" recv="2" chan="0">
      <step s="0" type="s" srcbuf="o" srcoff="0" dstbuf="o" dstoff="0" cnt="1" depid="-1" deps="-1" hasdep="0"/>
      <step s="1" type="r" srcbuf="o" srcoff="2" dstbuf="o" dstoff="2" cnt="1" depid="-1" deps="-1" hasdep="0"/>
    </tb>
  </gpu>
  <gpu id="1" i_chunks="0" o_chunks="4" s_chunks="0">
    <tb id="0" send="0" recv="0" chan="0">
      <step s="0" type="s" srcbuf="o" srcoff="1" dstbuf="o" dstoff="1" cnt="1" depid="-1" deps="-1" hasdep="0"/>
      <step s="1" type="r" srcbuf="o" srcoff="0" dstbuf="o" dstoff="0" cnt="1" depid="-1" deps="-1" hasdep="0"/>
    </tb>
    <tb id="1" send="2" recv="2" chan="0">
      <step s="0" type="s" srcbuf="o" srcoff="1" dstbuf="o" dstoff="1" cnt="1" depid="-1" deps="-1" hasdep="0"/>
      <step s="1" type="r" srcbuf="o" srcoff="2" dstbuf="o" dstoff="2" cnt="1" depid="-1" deps="-1" hasdep="0"/>
    </tb>
  </gpu>
```

Custom Plan Simulation in ASTRA-sim

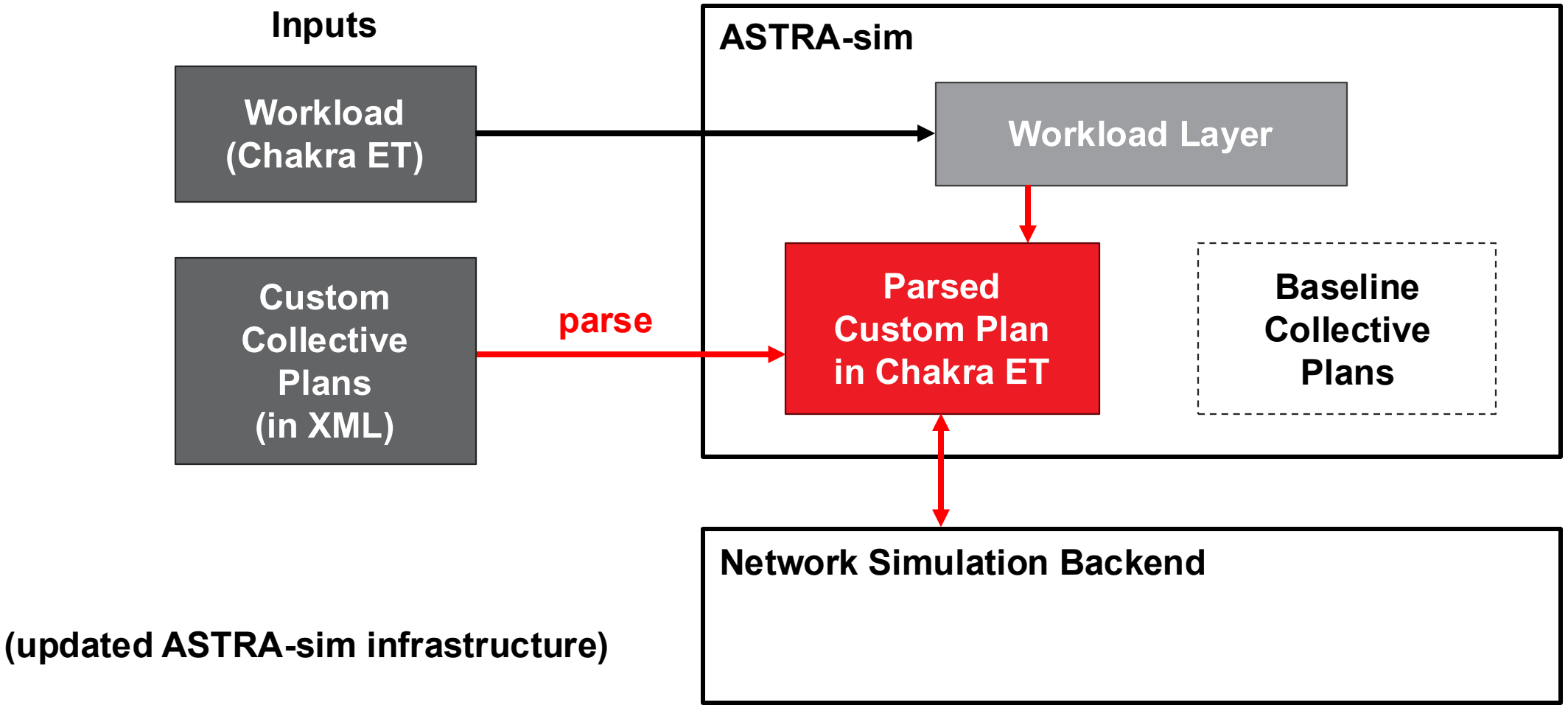
- Objective: Update ASTRA-sim to enable the **simulation of custom collective plans in XML**



(current ASTRA-sim infrastructure)

Custom Plan Simulation in ASTRA-sim

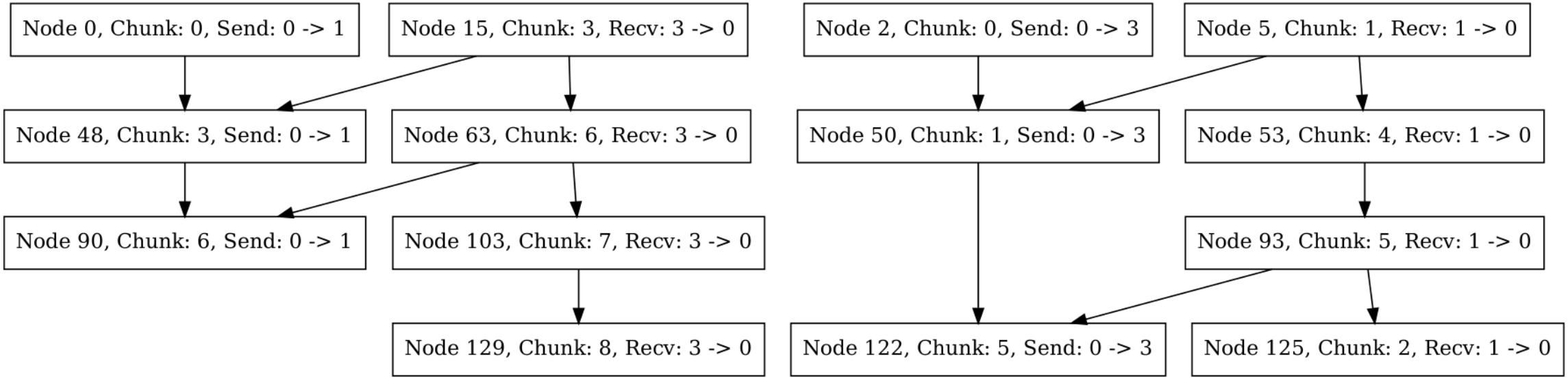
- Objective: Update ASTRA-sim to enable the **simulation of custom collective plans in XML**



(updated ASTRA-sim infrastructure)

Parsed Custom Plan

- Example TACOS custom collective plan, **parsed into Chakra ET**
  - Can be directly consumed by ASTRA-sim to **substitute the baseline collective plan**



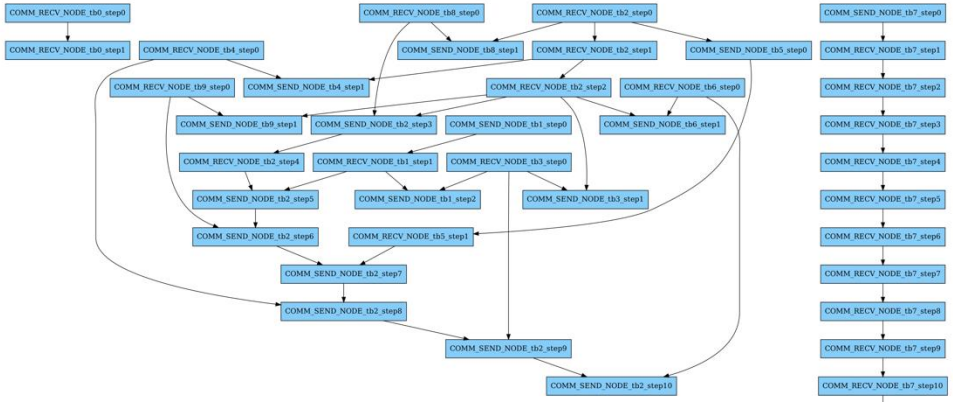
W. Won et al., "TACOS: Topology-Aware Collective Algorithm Synthesizer for Distributed Machine Learning," in MICRO '24

ASTRA-sim Simulation Result

```
<algo name="taccl" nchannels="1" (...)>
  <gpu id="0" i_chunks="1" o_chunks="32" s_chunks="0">
    <tb id="0" send="-1" recv="1" chan="0">
      <step s="0" type="r" srcbuf="o" srcoff="28" dstbuf="o" dstoff="28" cnt="1" depid="-1" deps="-1" hasdep="0"/>
      <step s="1" type="r" srcbuf="o" srcoff="29" dstbuf="o" dstoff="29" cnt="1" depid="-1" deps="-1" hasdep="0"/>
    </tb>
    (...)
```

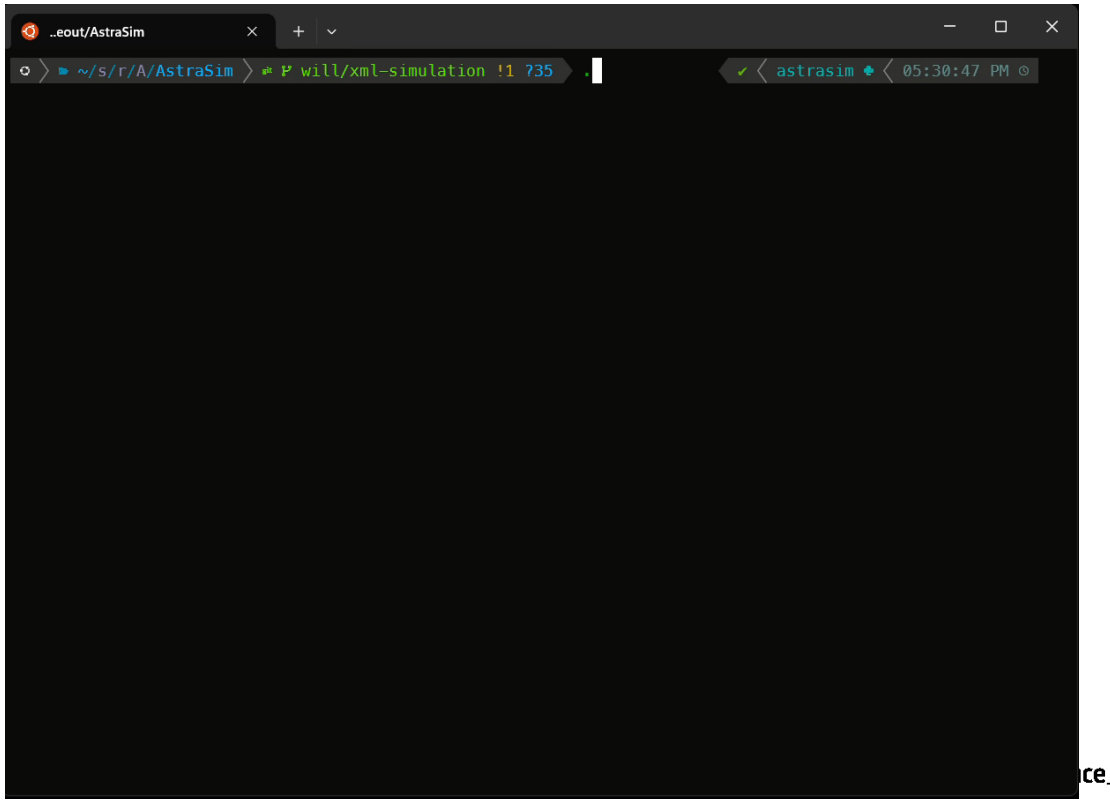
TACCL-Synthesized Custom Plan, in XML Representation

parse



Parsed Representation in Chakra ET

used by ASTRA-sim

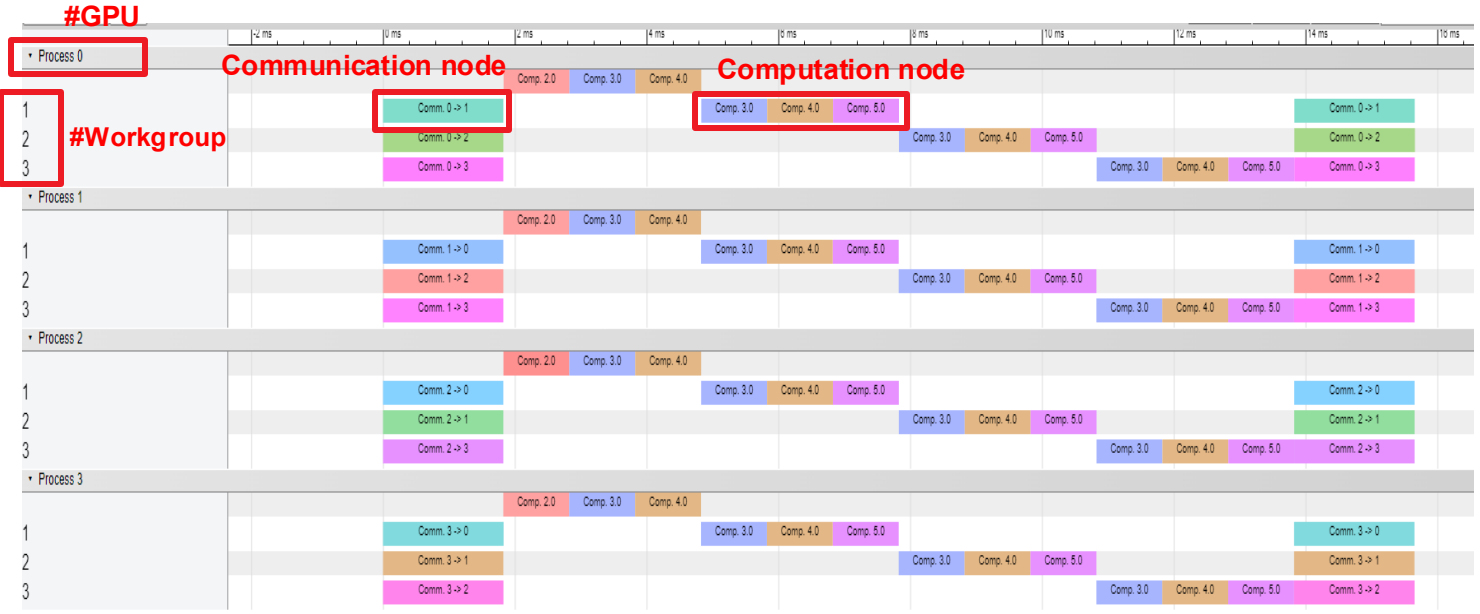


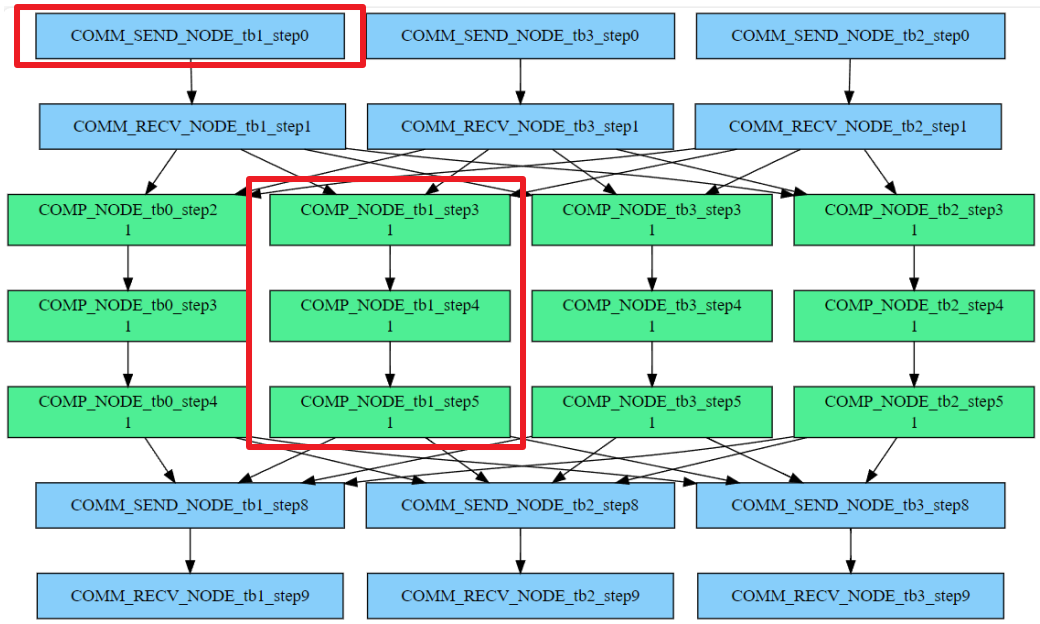
Outline

- JSON Chakra Traces
- Custom Collective Plans
- Logging and Visualization

Logging and Visualization

- Lightweight script to generate a unified timeline of logged events
- During every run, user can choose to enable logging for various events
- Below is an example of the ASTRA-sim timeline result for small four-GPU setup





Disclaimers and Attributions


AMD