



# 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)**

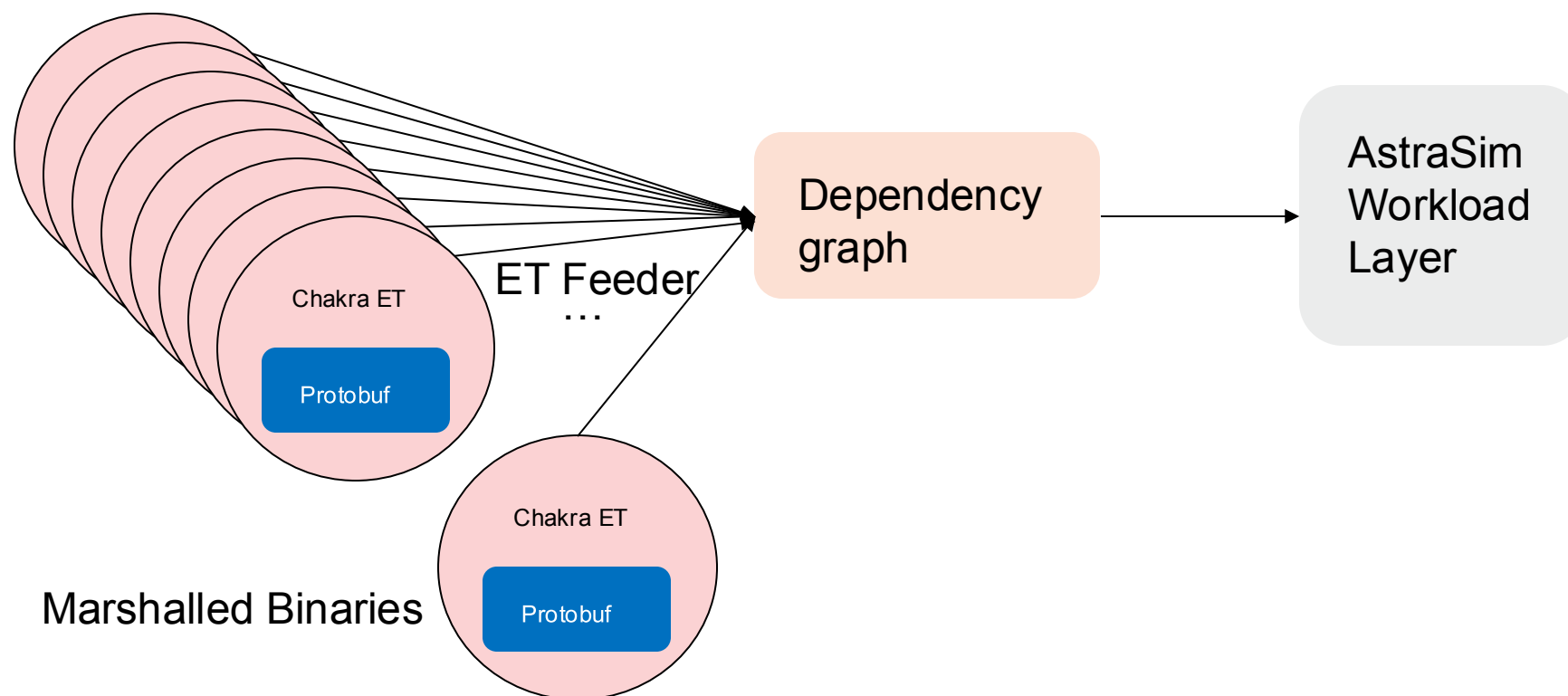
**AMD Research and Advanced Development**

**AMD**   
together we advance\_

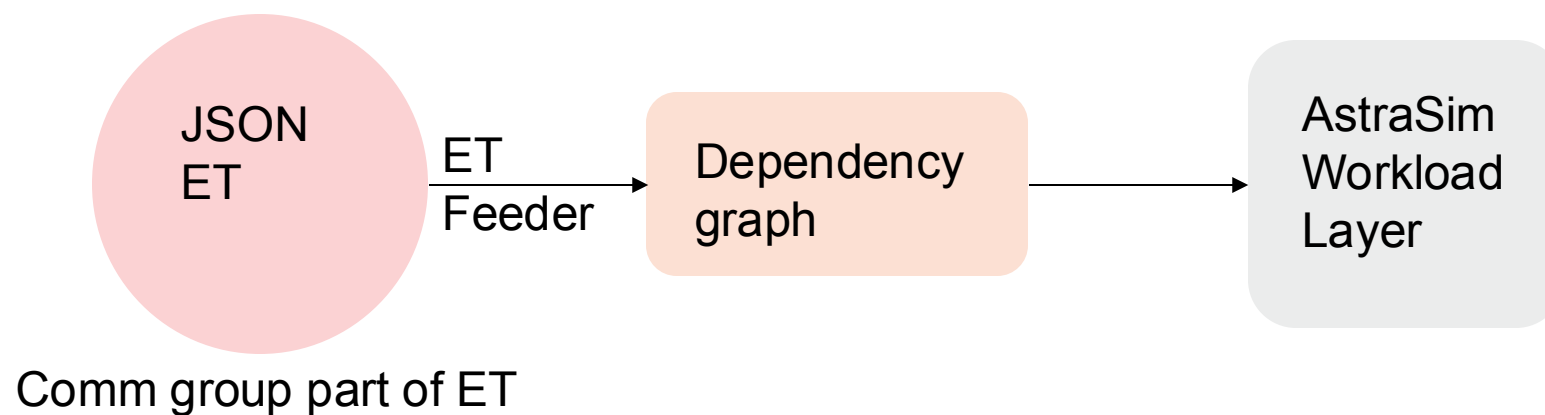
# Outline

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

# Chakra Execution Traces (Protobuf format)



# Chakra Execution Traces (JSON format)

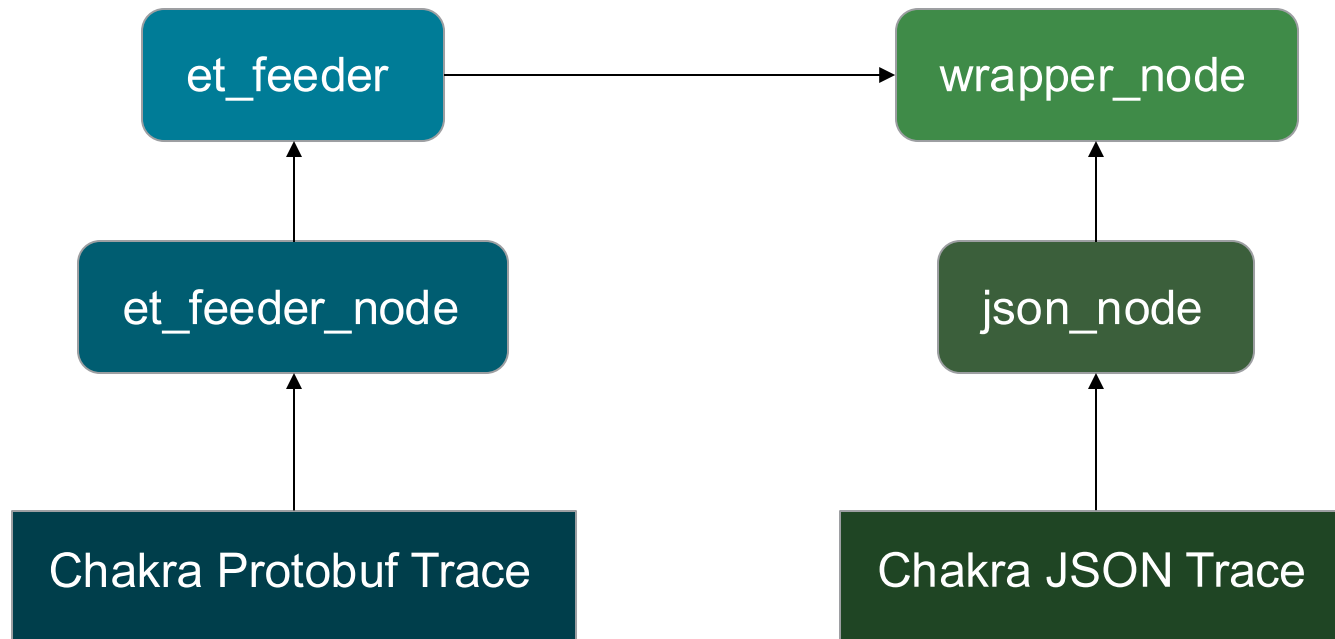


# Example JSON ET

```
1  {
2    "workload_name": "llama13b",
3    "ngpus": 1024,
4    "socs": ["<List of NPUs>"],
5    "{other global metadata}": "<metatdata>",
6    "communicators": [
7      {
8        "group_id": 1,
9        "involvedNPUs": [0, 1, 2, 3]
10     },
11     {
12       "group_id": 2,
13       "involvedNPUs": [4, 5, 6, 7]
14     }
15   ],
16   "workload_graph": [
17     {
18       "ID": 0,
19       "Name": "linear_allgather",
20       "NodeType": "Communication",
21       "data_deps": [],
22       "comm_type": "AllGatherV",
23       "CommSizeOut": [
24         [
25           [10, 20, 30, 40],
26           [23, 33, 43, 53],
27           [10, 20, 30, 40],
28           [23, 33, 43, 53]
29         ],
30         [
31           [10, 20, 30, 40],
32           [23, 33, 43, 53],
33           [10, 20, 30, 40],
34           [23, 33, 43, 53]
35         ]
36       ],
37       "CommTime": [[41770.0], [41770.0]],
38       "CommGroups": [1, 2],
39       "Device": "NIC"
40     },
41     {
42       "ID": 1,
43       "Name": "linear_gemm",
44       "NodeType": "Compute",
45       "data_deps": [0],
46       "comm_type": "AllGather",
47       "ComputeTime": [[97610.0], [97610.0], [97610.0], [97610.0], [97610.0], [97610.0], [97610.0], [97610.0]],
48       "M": 16384,
49       "N": 3840,
50       "K": 5120,
51       "Batch": 1,
52       "InputType": 2,
53       "OutputType": 2,
54       "Device": "GPU"

```

# Backward Compatibility

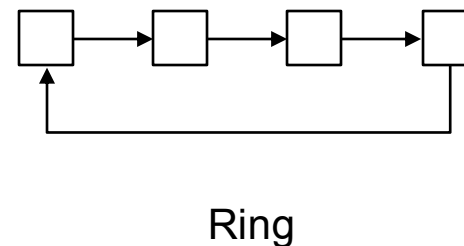
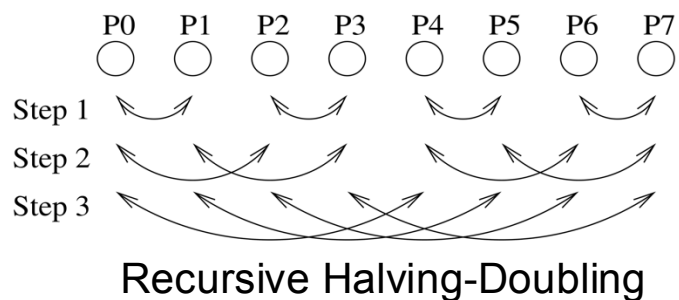


# Outline

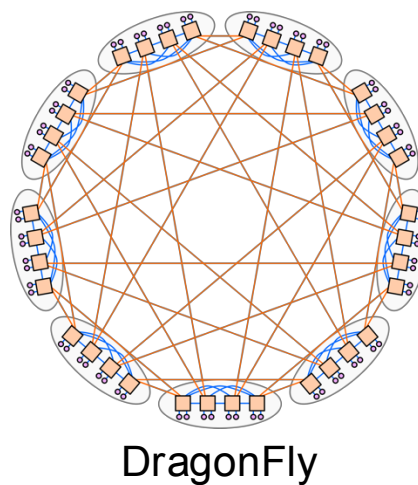
- JSON Chakra Traces
- **Custom Collective Plans**
- **Logging and Visualization**

# Necessity of Custom Collective Plans

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



- Distinct datacenter-scale **topology options**



*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

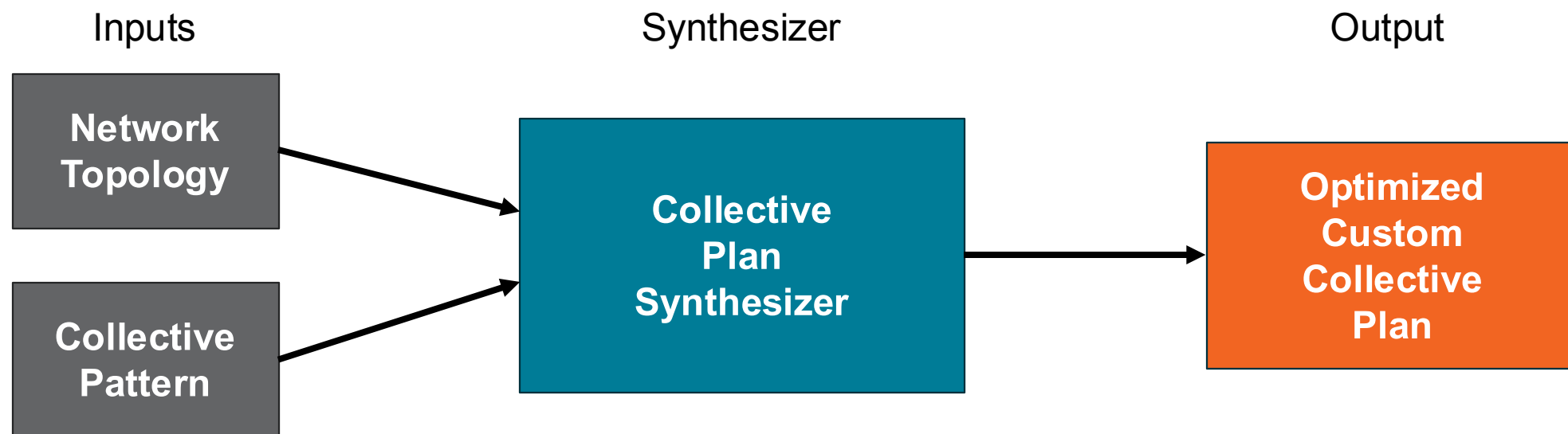
        c = c.copy(next_rank, Buffer.output, index, sendtb=channel, recvtb=channel, ch=channel)
```

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)

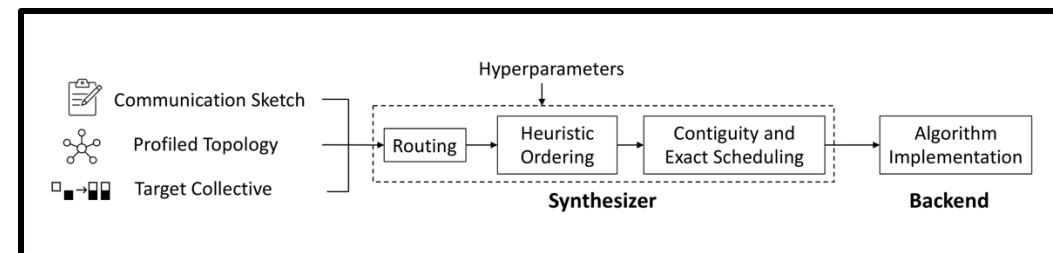
    next_rank = (index + step + 1) % size
    channel = index%channels

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

```

MSCCLang DSL

lowering



TACCL Synthesizer

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)

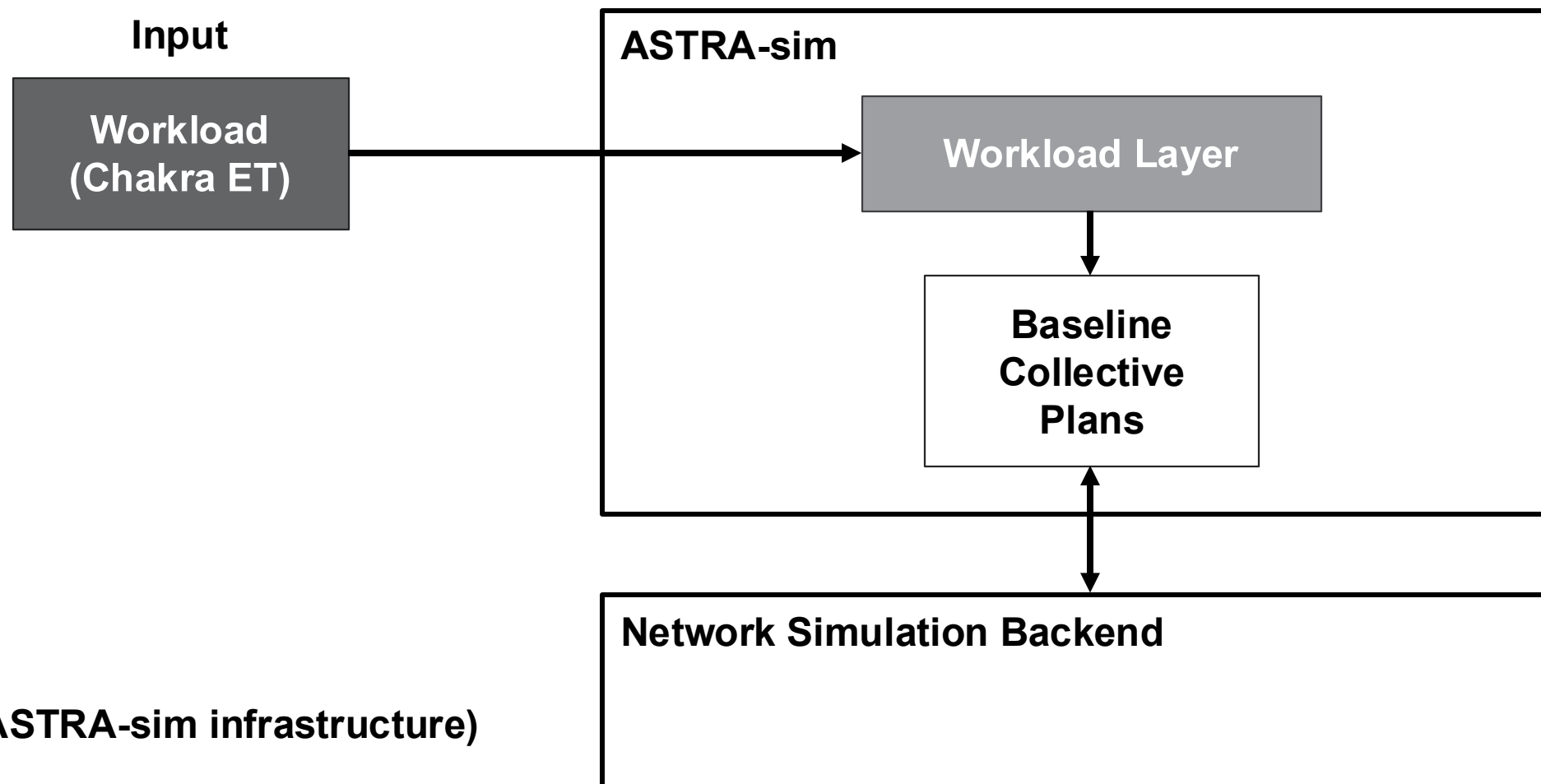
```

<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" rcv="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" rcv="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" rcv="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" rcv="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>

```

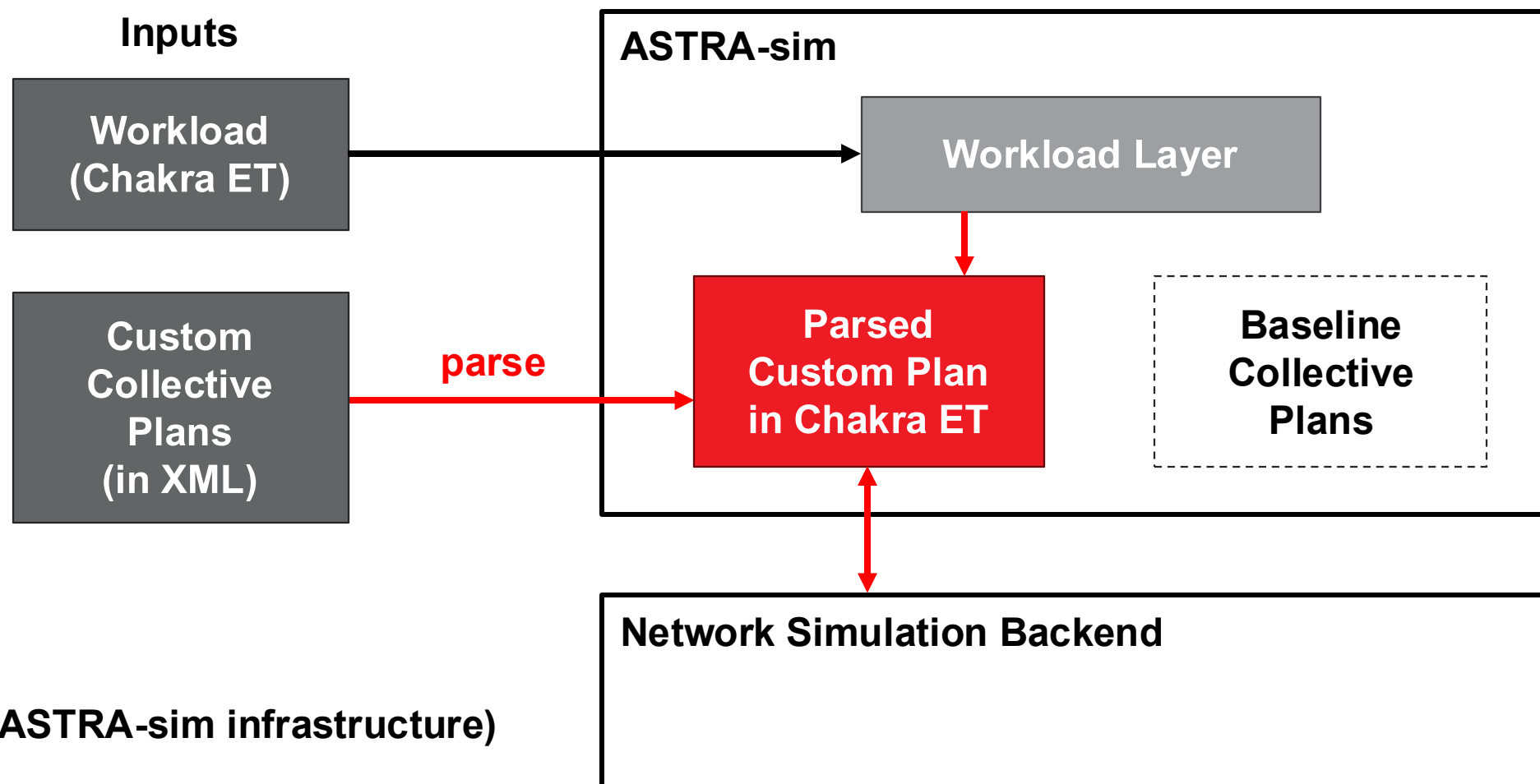
# Custom Plan Simulation in ASTRA-sim

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



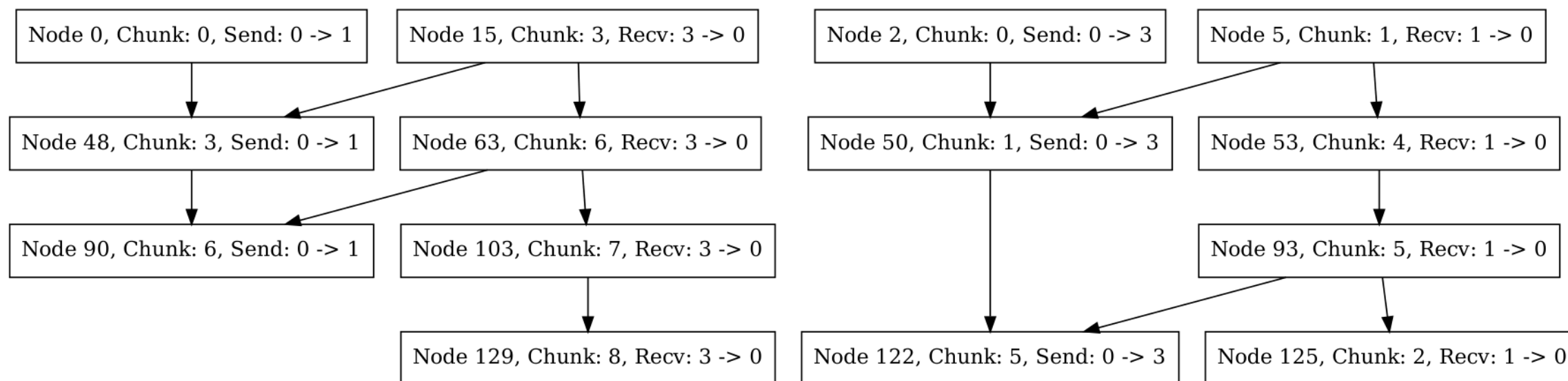
# Custom Plan Simulation in ASTRA-sim

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



# 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

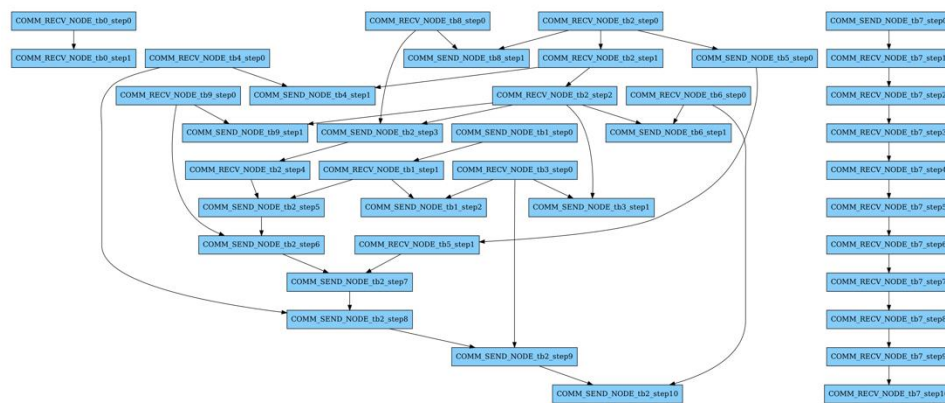
```

<algo name="taccl" nchannels="1" (...)>
  <gpu id="0" i_chunks="1" o_chunks="32" s_chunks="0">
    <tb id="0" send="-1" rcv="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>
    (...)
  </gpu>
</algo>

```

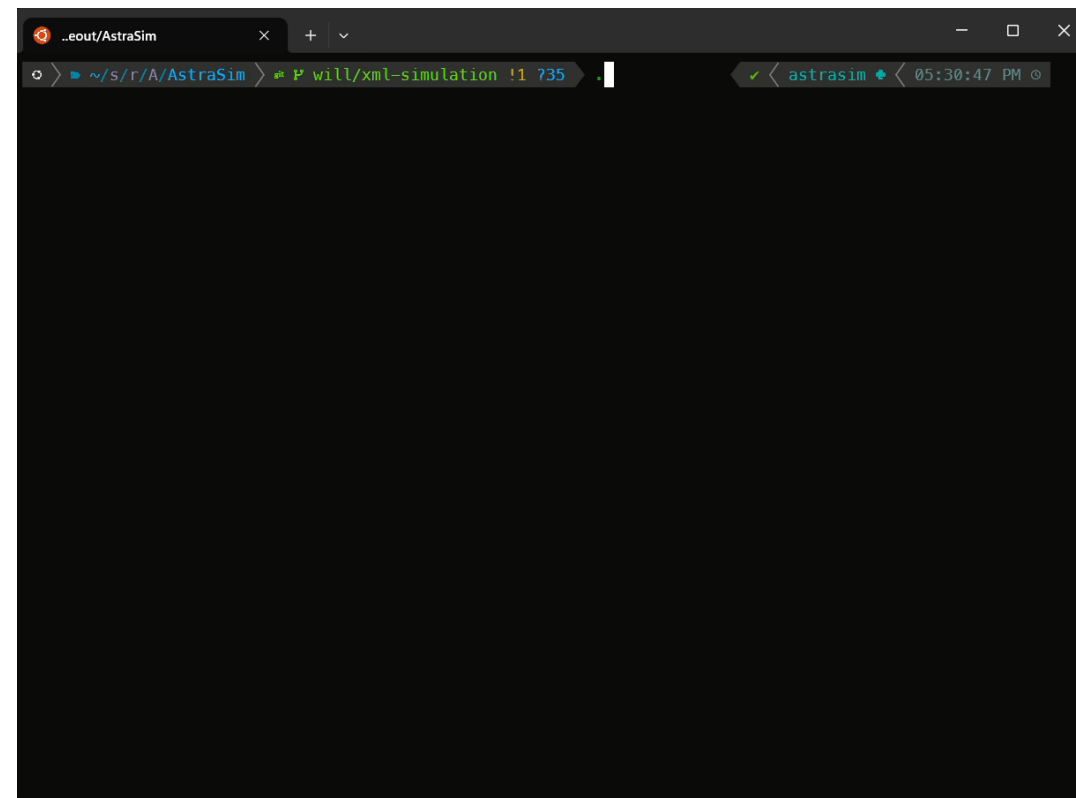
TACCL-Synthesized Custom Plan, in XML Representation

parse



Parsed Representation in Chakra ET

used by  
ASTRA-sim

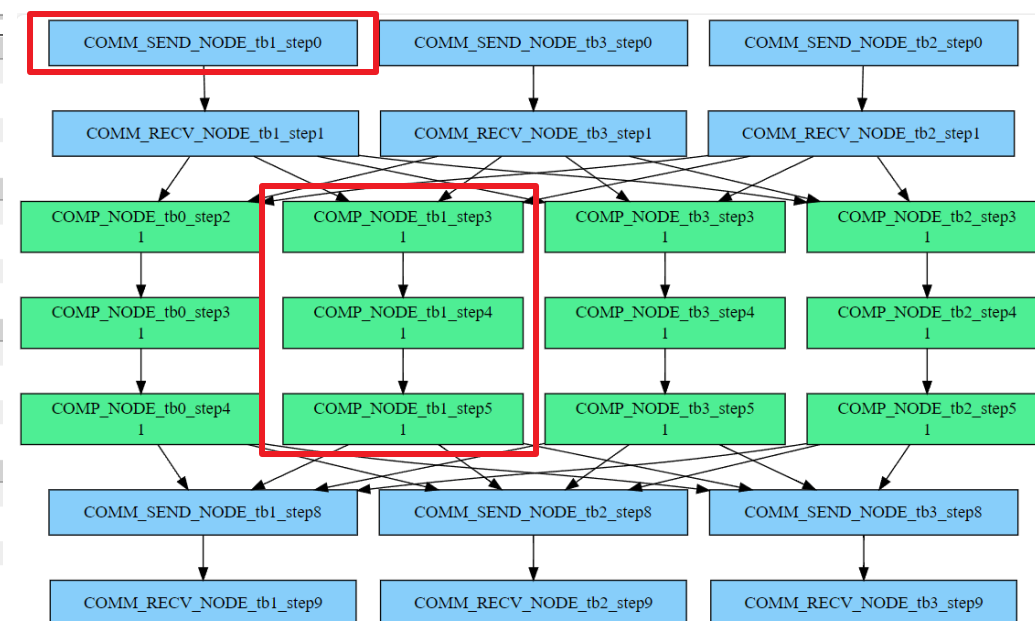
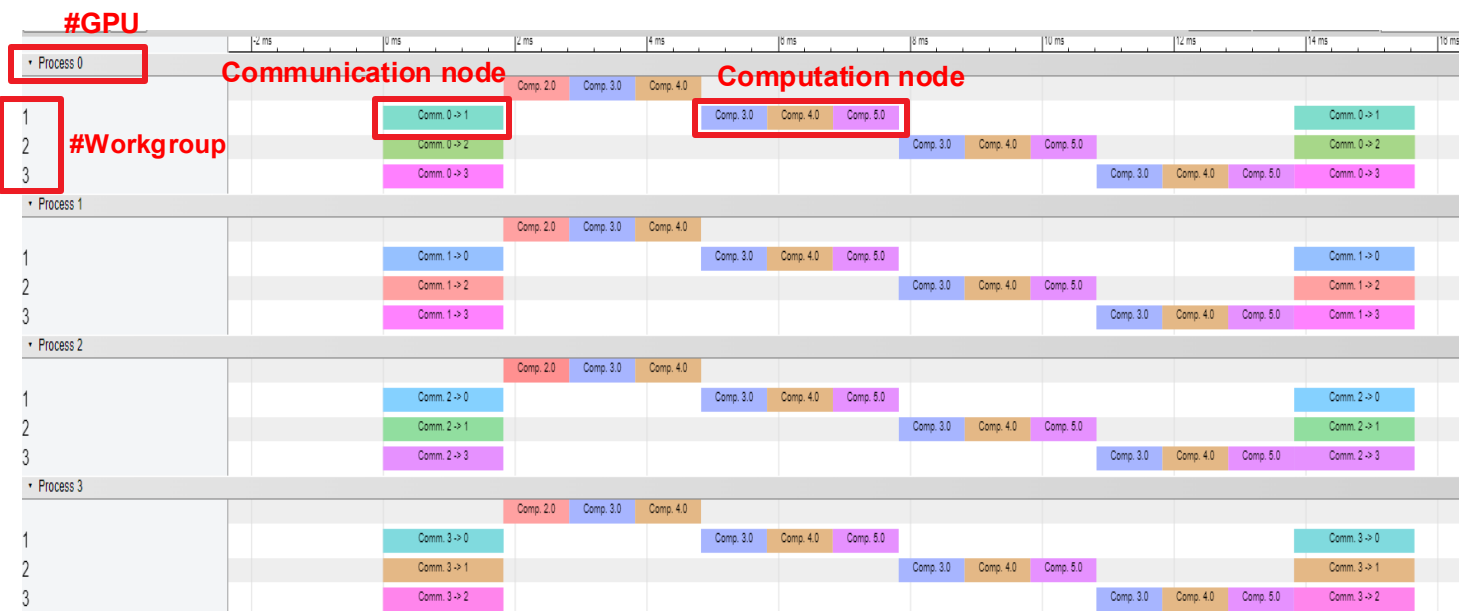


# Outline

- 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

## DISCLAIMER

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content thereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED 'AS IS.' AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Services and Deliverables and all other AMD products and technologies referenced in this presentation are commercial services or commercial products or commercial computer software or COTS products as defined in FAR 2.101 (Definitions).

Third-party content is licensed to you directly by the third party that owns the content and is not licensed to you by AMD. ALL LINKED THIRD-PARTY CONTENT IS PROVIDED "AS IS" WITHOUT A WARRANTY OF ANY KIND. USE OF SUCH THIRD-PARTY CONTENT IS DONE AT YOUR SOLE DISCRETION AND UNDER NO CIRCUMSTANCES WILL AMD BE LIABLE TO YOU FOR ANY THIRD-PARTY CONTENT. YOU ASSUME ALL RISK AND ARE SOLELY RESPONSIBLE FOR ANY DAMAGES THAT MAY ARISE FROM YOUR USE OF THIRD-PARTY CONTENT.

## ATTRIBUTION

©2024 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

**AMD** 