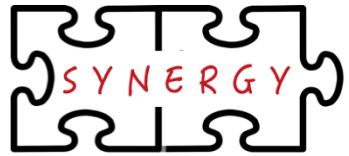


https://synergy.ece.gatech.edu

ASTRA-sim Tutorial
@HotI 2024
Aug 23, 2024

ASTRA-sim and Chakra Tutorial:
Demo

Will Won
Ph.D. Student
School CS, Georgia Institute of Technology
william.won@gatech.edu

ASTRA-sim Tutorial - Agenda

Time (PDT)	Topic	Presenter
3:00 – 3:30 pm	**Introduction to Distributed ML**	Tushar Krishna
3:30 – 3:45 pm	**Overview of Chakra and ASTRA-sim**	Tushar Krishna
3:45 – 4:35 pm	**Deeper Dive into Chakra and ASTRA-sim**	Will Won
	Workload, System, and Network Layers	
4:35 – 4:45 pm	**Demo**	Will Won
4:45 – 5:00 pm	**Closing Remarks**	Tushar Krishna

Tutorial Website

includes agenda, slides, ASTRA-sim installation instructions (via source + docker image)

https://astra-sim.github.io/tutorials/hoti-2024

Attention: Tutorial is being recorded

Outline

- **Demo 1: Getting Started with Chakra and ASTRA-sim2.0**
  - **Installing Chakra and ASTRA-sim2.0**
  - **Using Chakra ET Generation API**
  - **Executing ASTRA-sim2.0**
- Demo 2: Using ET Generation API Wrappers
  - Using Text-to-Chakra Converter (API Wrapper)
  - Executing End-to-End Simulation
  - Sneak Peek: Synthetic Transformer Trace Generator
- Demo 3: End-to-End Workload with Real System Collected ET
  - Collecting Traces from Real System
  - Preparing Chakra ET from Real System ET
  - Executing ASTRA-sim2.0 Simulation

Prerequisites

- ASTRA-sim project uses ***ssh protocol*** (instead of *https*) to clone repositories and submodules.
  - Please check you've set up your GitHub account with your ssh key.
  - *https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account*

- Since ASTRA-sim and Chakra requires *protobuf,* we strongly recommend using the **provided Docker environment** for execution.
  - We provide scripts to manually download and compile *protobuf*.
  - *https://astra-sim.github.io/tutorials/hoti-2024/installation*

Cloning ASTRA-sim

- We provide sandboxed Chakra/ASTRA-sim for tutorial purposes
  - Scripts are explained in slides

```
$ git clone git@github.com:astra-sim/tutorials.git
$ cd tutorials/hoti2024
$ ./clone_astra_sim.sh
```

clone_astra_sim.sh:

```
$ git clone git@github.com:astra-sim/astra-sim.git
$ cd astra-sim/
$ git checkout tags/tutorial-hoti2024
$ git submodule update --init --recursive
```

Launching Execution Environment (Docker)

- Download Docker Image

```
$ docker pull astrasim/tutorial-hoti2024
```

- Start a Docker Container with the cloned ASTRA-sim attached to it

```
$ docker run -it \
      -v $(pwd):/app/hoti2024 \
      astrasim/tutorials-hoti2024
[docker]$ cd hoti2024
```

Link current directory to Docker container

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 .
```

Compile ASTRA-sim

- Compile ASTRA-sim with the analytical network backend

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

compile_astra_sim.sh:

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

Creating Simple Chakra ET – Using API

- Chakra offers ET Generation API
  - For manual design and implementation of arbitrary chakra ETs

generate_all_reduce.py:

```
(...)
node = ChakraNode()
node.id = 1
node.name = "All-Reduce"
node.type = COMM_COLL_NODE
node.attr.append(ChakraAttr(name="comm_type", int64_val=ALL_REDUCE))
node.attr.append(ChakraAttr(name="comm_size", uint64_val=1_048_576))
encode_message(et, node)
(...)
```

Create Chakra node

Store Chakra ET file

Creating Simple Chakra ET – Microbenchmark

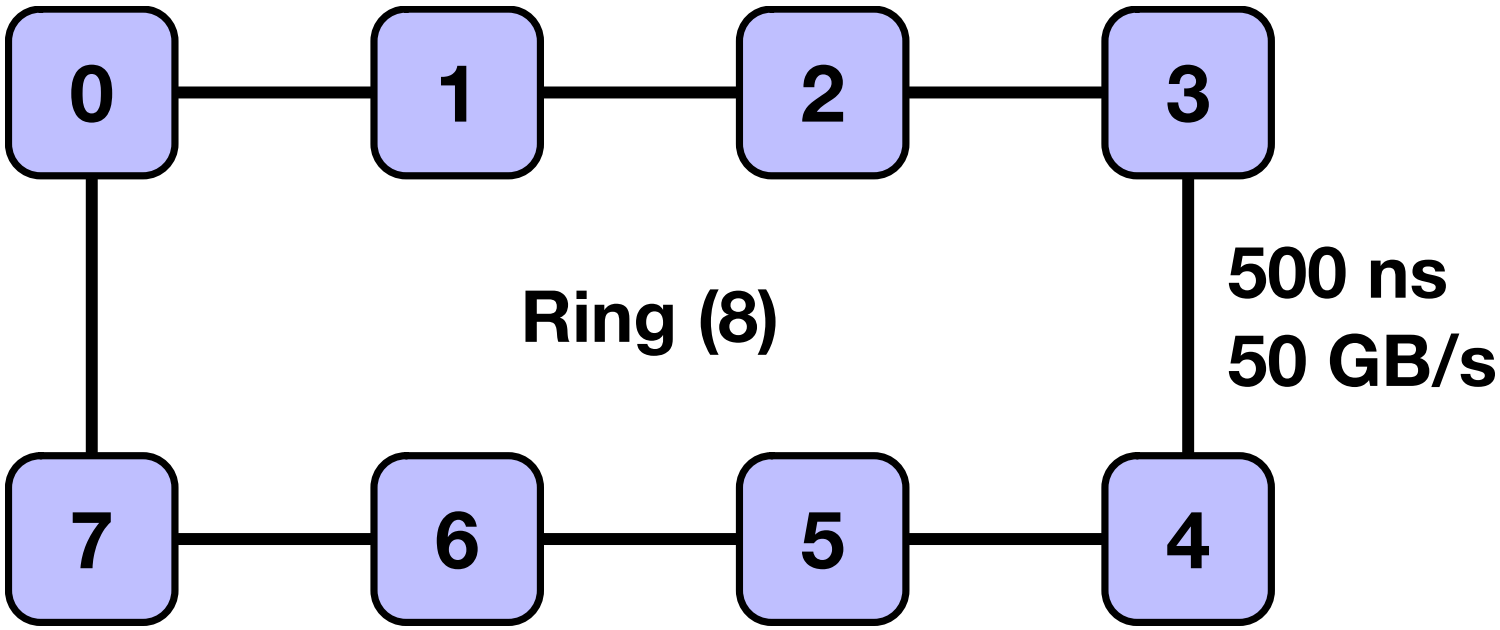
- 1 MB All-Reduce among 8 NPUs

```
[docker]$ cd demo1
[docker]$ python3 generate_all_reduce.py
```

Generated Chakra ET Files

Network Setup

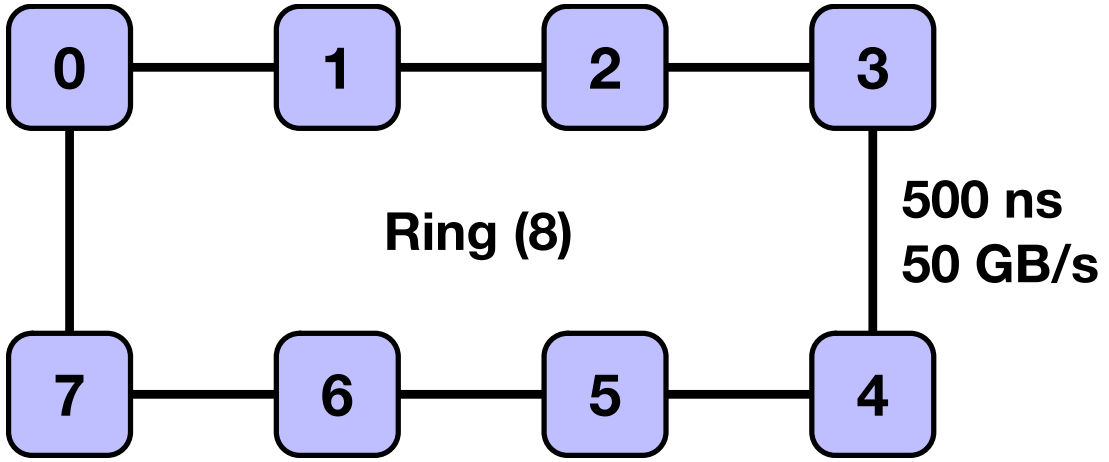
- **Ring** topology with **8 NPUs**
- **500 ns** (latency), **50 GB/s** (bandwidth)



Network Setup

demo1/inputs/Ring_8.yml:

```
topology: [ Ring ]
npus_count: [ 8 ]
bandwidth: [ 50.0 ]  # GB/s
latency: [ 500.0 ]  # ns
```



System Setup

demo1/inputs/Ring_sys.json

```
"scheduling-policy": "LIFO",
"endpoint-delay": 10,
"active-chunks-per-dimension": 1,
"preferred-dataset-splits": 4,
"all-reduce-implementation": ["ring"],
"all-gather-implementation": ["ring"],
"reduce-scatter-implementation": ["ring"],
"all-to-all-implementation": ["ring"],
"collective-optimization": "localBWAware",
"local-mem-bw": 50,
"boost-mode": 0
```

4 chunks per collective

Ring algorithm

Running Simulation

- Execute ASTRA-sim Simulation

```
[docker]$ ./run_demo1-1.sh
```

run_demo1-1.sh:

```
${ASTRA_SIM} \
    --workload-configuration=./allreduce/allreduce \
    --system-configuration=./inputs/Ring_sys.json \
    --network-configuration=./inputs/Ring_8.yml \
```

Demo 1-1: Simulation Result

- 1 MB All-Reduce time: 117.780 μs

```
sys[0] finished, 117780 cycles
sys[1] finished, 117780 cycles
sys[2] finished, 117780 cycles
sys[3] finished, 117780 cycles
sys[4] finished, 117780 cycles
sys[5] finished, 117780 cycles
sys[6] finished, 117780 cycles
sys[7] finished, 117780 cycles
```

Simulation Result

Demo 1-2: A Little Bit More

- Using APIs to create 3 nodes with dependencies

demo1/generate_3nodes.py:

```
(...)
node1 = ChakraNode()

node2 = ChakraNode()
node2.data_deps.append(node1.id)

node3 = ChakraNode()
node3.data_deps.append(node1.id)
(...)
```

Assign dependency

Assign dependency

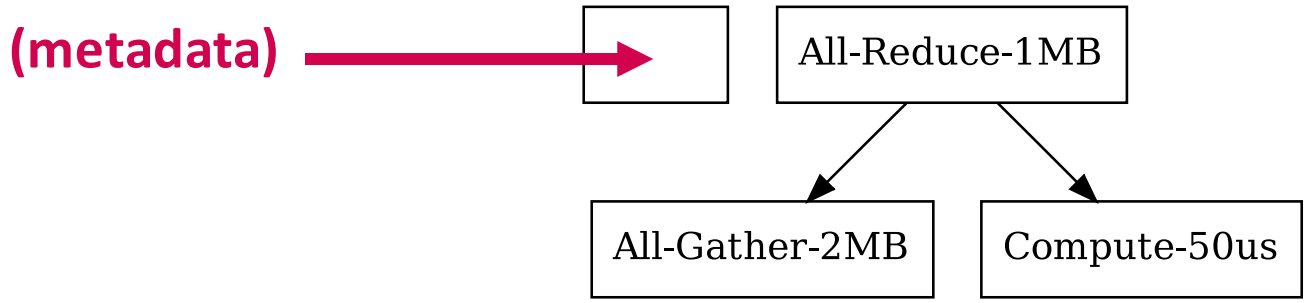
Visualize Generated Chakra ET

- Using Chakra's ET Visualizer

```
[docker]$ python3 generate_3nodes.py
[docker]$ ./visualize_3nodes.sh
```

demo1/visualize_3nodes.sh:

```
python3 -m chakra.et_visualizer.et_visualizer \
    --input_filename=${SCRIPT_DIR}/3nodes/3nodes.0.et \
    --output_filename=${SCRIPT_DIR}/3nodes.0.pdf
```



Visualization Result

Running Simulation

- Execute ASTRA-sim Simulation

```
[docker]$ ./run_demo1-2.sh
```

```
sys[4] finished, 405520 cycles
sys[5] finished, 405520 cycles
sys[6] finished, 405520 cycles
sys[7] finished, 405520 cycles
sys[0] finished, 405520 cycles
sys[1] finished, 405520 cycles
sys[2] finished, 405520 cycles
sys[3] finished, 405520 cycles
```

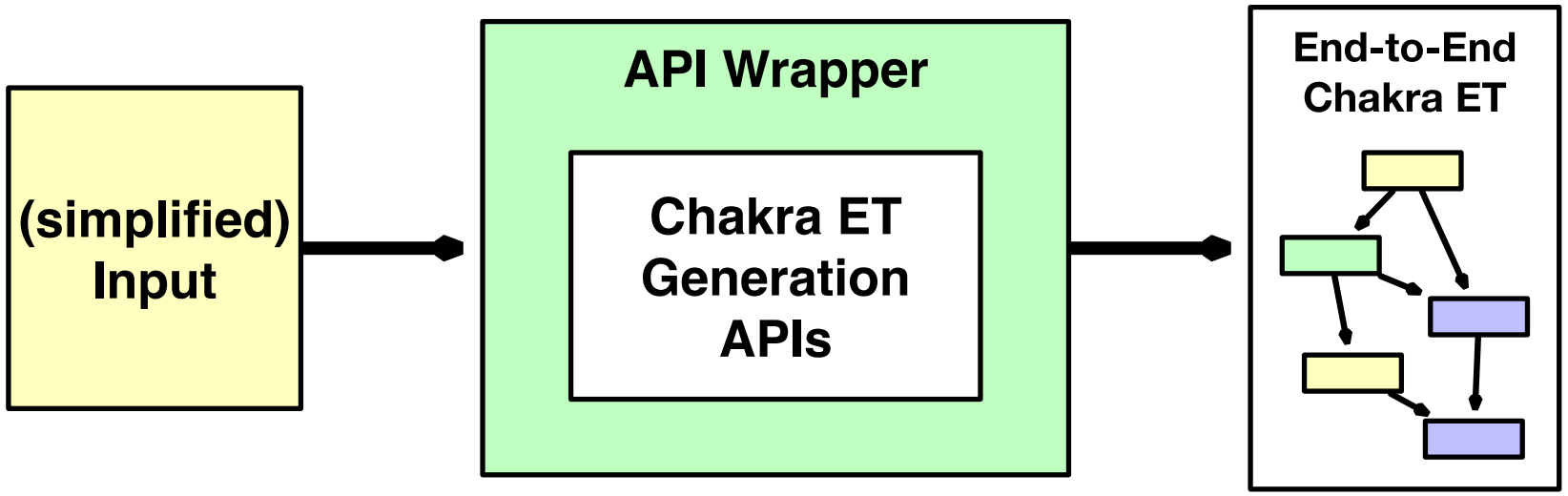
Simulation Result: 405.520 µs

Outline

- Demo 1: Getting Started with Chakra and ASTRA-sim2.0
  - Installing Chakra and ASTRA-sim2.0
  - Using Chakra ET Generation API
  - Executing ASTRA-sim2.0
- **Demo 2: Using ET Generation API Wrappers**
  - **Using Text-to-Chakra Converter (API Wrapper)**
  - **Executing End-to-End Simulation**
  - **Sneak Peek: Synthetic Transformer Trace Generator**
- Demo 3: End-to-End Workload with Real System Collected ET
  - Collecting Traces from Real System
  - Preparing Chakra ET from Real System ET
  - Executing ASTRA-sim2.0 Simulation

API Wrappers

- Manually representing all Chakra nodes could be demanding
- API Wrappers to **semi-automate** end-to-end workload ET generation



Text-to-Chakra Wrapper

- **ASTRA-sim1.0's text-based** end-to-end workload representation

demo2/workload_text.txt

```
MODEL
6
layer_64_1_mlp0 -1 32291 ALLGATHER 37632 32291 ALLREDUCE 37632 12864 NONE 0 3229
layer_64_1_mlp1 -1 7488 ALLGATHER 65536 7488 ALLREDUCE 65536 3648 NONE 0 749
layer_64_1_mlp2 -1 7488 ALLGATHER 65536 7488 ALLREDUCE 65536 3456 NONE 0 749
layer_64_1_mlp3 -1 14144 ALLGATHER 147456 14144 ALLREDUCE 147456 10368 NONE 0 1414
layer_64_1_mlp4 -1 7488 ALLGATHER 65536 7488 ALLREDUCE 65536 3648 NONE 0 749
layer_64_2_mlp5 -1 9984 ALLGATHER 65536 9984 ALLREDUCE 65536 3456 NONE 0 998
```

parallelization strategy

#layers

per-layer information

Text-to-Chakra Wrapper

- **Converts** text-based workload **into Chakra ET**

chakra/et_converter/text2chakra_converter.py

```
# forward pass
for idx, layer in enumerate(layers):
    fwd_comp_node = self.get_comp_node(
            layer.name, "FWD",
            layer.fwd_comp_time)
    if idx != 0:
        self.add_parent(fwd_comp_node, layers[idx-1].fwd_comm_node)
    if layer.bwd_wg_comp_node != None:
        self.add_parent(fwd_comp_node, layer.bwd_wg_comp_node)
    layer.fwd_comp_node = fwd_comp_node
    encode_message(g, fwd_comp_node)
```

Use APIs to create Chakra node

Use APIs to assign dependencies

Using Text-to-Chakra Wrapper

- Run Text-to-Chakra Wrapper

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

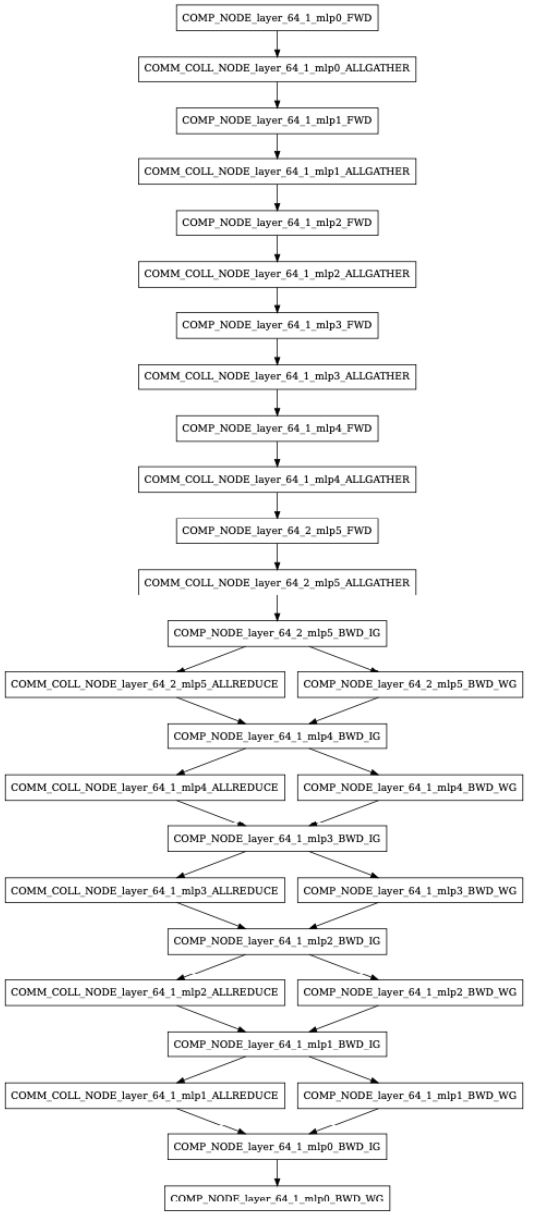
demo2/convert.sh:

```
python3 -m chakra.et_converter.et_converter \
    --input_type="Text" \
    --input_filename=./workload_text.txt" \
    --output_filename=./workload_chakra/workload_chakra" \
    --num_npus=8
```

Using Text-to-Chakra Wrapper

- Visualization Result

```
[docker]$ ./visualize.py
```



Visualization Result

Using Text-to-Chakra Wrapper

- Simulation Result

```
[docker]$ ./run_demo2.py
```

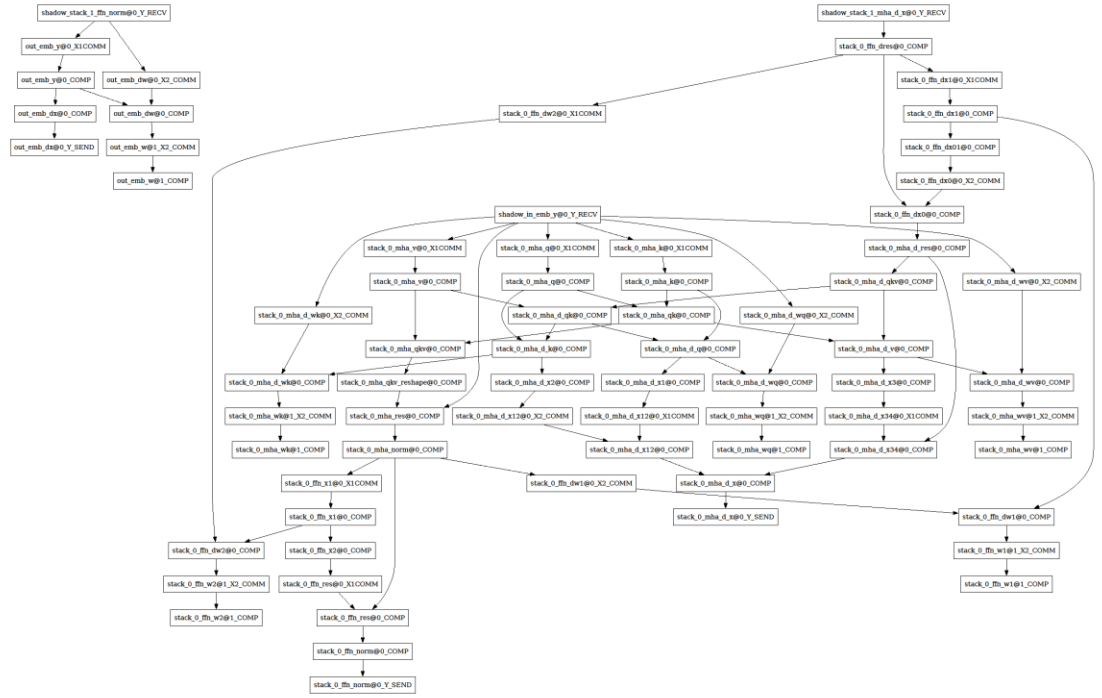
```
sys[0] finished, 195206000 cycles
sys[1] finished, 195206000 cycles
sys[2] finished, 195206000 cycles
sys[3] finished, 195206000 cycles
sys[4] finished, 195206000 cycles
sys[5] finished, 195206000 cycles
sys[6] finished, 195206000 cycles
sys[7] finished, 195206000 cycles
```

Simulation Result: 195.206 ms

Sneak Peek: Synthetic Chakra ET Generator

- 4D Parallelism for Transformer-based LLMs

	A	B	C	D	E	F	G	H	I	J	K
1	id	require_grads	x1	x2	op_type	op_attr	x1_shape	x1_hidden	x2_shape	x2_hidden	grad_of
2	x	N			T		Batch/dp, Seq/sp, Din/mp	1			
3	w	Y			T		Din/(dp*sp), Dout/mp	1			
4	y	N	x	w	M	bsm,mn->bsn	Batch/dp, Seq/sp, Din	1	Din, Dout/mp	1	
5	dy	N			T		Batch/dp, Seq/sp, Dout/mp	1			y
6	dw	N	dy	x	M	bsn,bsm->mn	Batch/dp, Seq/sp, Dout/mp	1	Batch/dp, Seq/sp, Din	1	w
7	dx	N	dy	w	M	bsn,mn->bsm	Batch/dp, Seq/sp, Dout/mp	1	Din, Dout/mp	1	x
8											



Work in Progress!

Slide courtesy: Changhai Man <cman8@gatech.edu>

Outline

- Demo 1: Getting Started with Chakra and ASTRA-sim2.0
  - Installing Chakra and ASTRA-sim2.0
  - Using Chakra ET Generator
  - Executing ASTRA-sim2.0
- Demo 2: End-to-End Workload with Synthetic ET Generator
  - Installing Synthetic Chakra ET Generator
  - Running Synthetic ET Generator
  - Executing ASTRA-sim2.0 Simulation
- **Demo 3: End-to-End Workload with Real System Collected ET**
  - **Collecting Traces from Real System**
  - **Preparing Chakra ET from Real System ET**
  - **Executing ASTRA-sim2.0 Simulation**

Chakra ET Collection

- Profile/Collect Real System Trace from PyTorch

```
et = ExecutionGraphObserver()
et.register_callback("et_file.json")
et.start()

# run PyTorch model

et.stop()
et.unregister_callback()
```

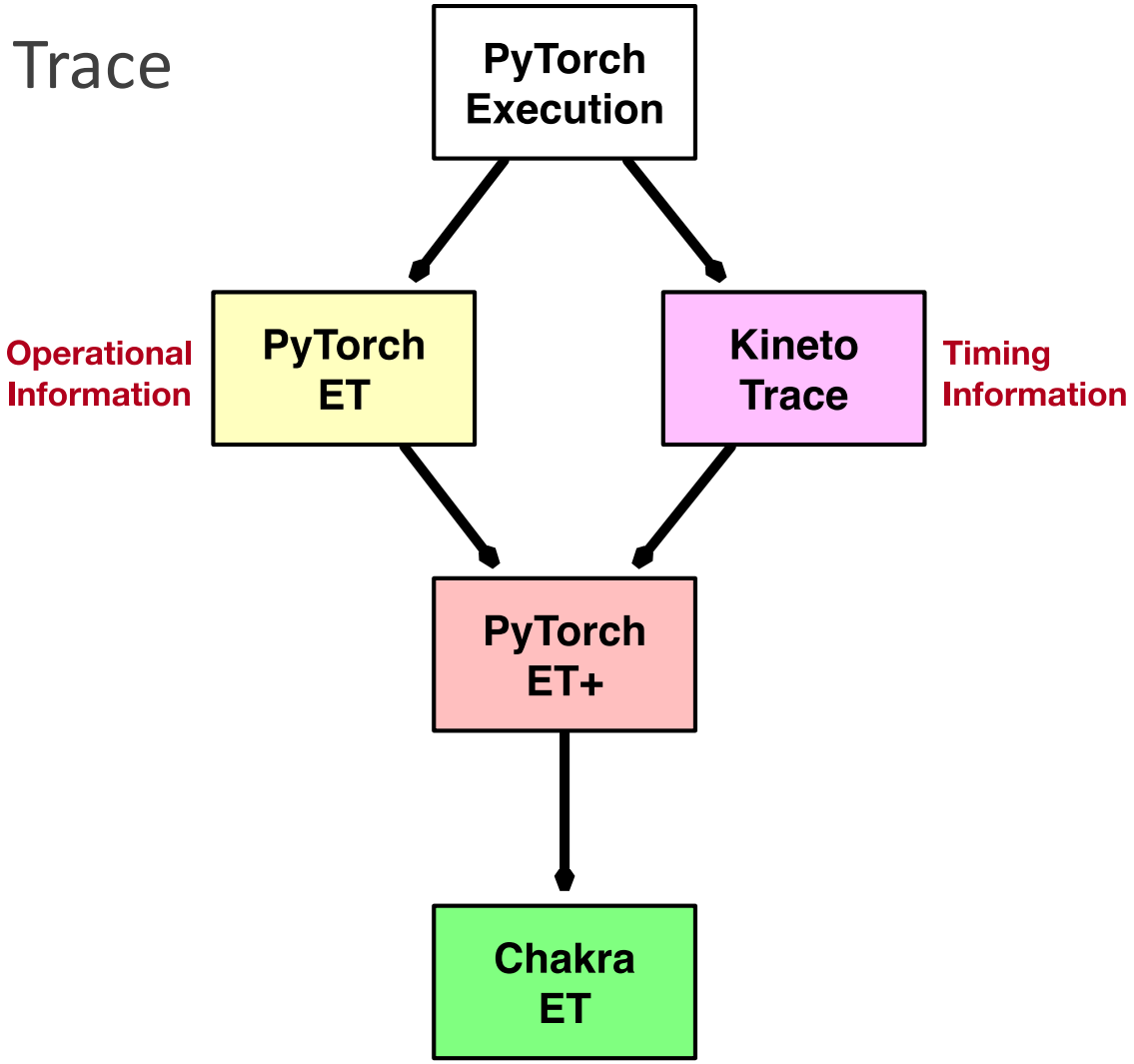
Start ET collection

Run model

Stop collection

PyTorch Trace: Flow

- PyTorch saves: **PyTorch ET** and **Kineto** Trace
- Merge them into: **PyTorch ET+**
- Convert PyTorch ET+ into **Chakra ET**



Merge PyTorch Traces

- Merge PyTorch ET and Kineto into **PyTorch ET+**

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

demo3/merge.sh:

```
python3 -m chakra.trace_link.trace_link \
    --chakra-host-trace=pytorch_traces/pytorch_et_0.json \
    --chakra-device-trace=pytorch_traces/kineto_0.json \
    --output-file=etplus_traces/etplus_0.json
```

ResNet-50, 2-GPU, CIFAR-10 Dataset

Trace collected and shared by: Joongun Park <jpark3234@gatech.edu>

Convert PyTorch ET+ into Chakra

- Convert PyTorch ET+ into Chakra ET

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

demo3/convert.sh:

```
python3 -m chakra.et_converter.et_converter \
    --input_type="PyTorch" \
    --input_filename="${SCRIPT_DIR}/etplus_traces/etplus_0.json" \
    --output_filename="${SCRIPT_DIR}/chakra_traces/et.0.et"
```

ResNet-50, 2-GPU, CIFAR-10 Dataset

Trace collected and shared by: Joongun Park <jpark3234@gatech.edu>

Using Text-to-Chakra Wrapper

- Simulation Result

```
[docker]$ ./run_demo3.py
```

```
sys[0] finished, 581291000 cycles
sys[1] finished, 585752000 cycles
```

Simulation Result: 58.752 ms