Georgia Tech School of Electrical and Computer Engineering
College of Engineering

http://synergy.ece.gatech.edu

Conclusion and Next Steps

Taekyung Heo
Postdoctoral Fellow, School of ECE
Georgia Institute of Technology
taekyung@gatech.edu

Acknowledgments: William Won (GT), Srinivas Sridharan (Facebook), Sudarshan Srinivasan (Intel)

Motivation of this Tutorial

- Large model distributed training is an ongoing open-research area
- Many emerging supercomputing systems being designed specifically for this problem!
  - Cerebras CS2
  - Tesla Dojo
  - NVIDIA DGX + Mellanox SHARP switches
  - Intel Habana
  - IBM Blueconnect
  - Facebook Zion
  - …
- Co-design of algorithm and system offers high opportunities for speedup and efficiency

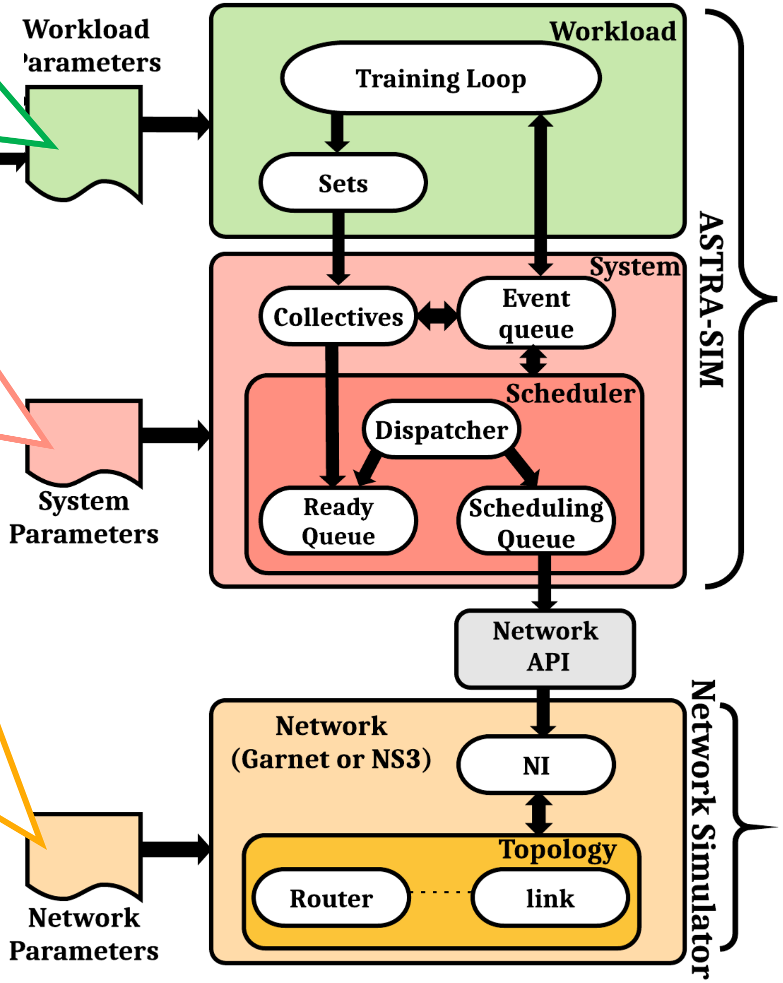
ASTRA-sim: Status and On-going Development

✓ *Released* ➢ *In progress*

- ✓ Supports Data-Parallel, Model-Parallel, Hybrid-Parallel training loops
- ✓ Extensible to more training loops
  - ➢ Graph-based input from PyTorch

- ✓ Ring based, Tree-based, AlltoAll based, and multi-phase collectives
- ✓ Variety of scheduling policies
- ✓ Compute times fed via offline system measurements or compute simulator

- ✓ Various topologies, flow-control, link bandwidth, congestion control
- ✓ Plug-and-play options
  - ✓ Analytical (roofline)
  - ➢ Analytical with congestion
  - ✓ Garnet (credit-based)
  - ➢ NS3 (TCP, RDMA)



Workload Parameters

Workload

Training Loop

Sets

System

Collectives

Event queue

Scheduler

Dispatcher

Ready Queue

Scheduling Queue

System Parameters

ASTRA-SIM

Network API

Network (Garnet or NS3)

NI

Topology

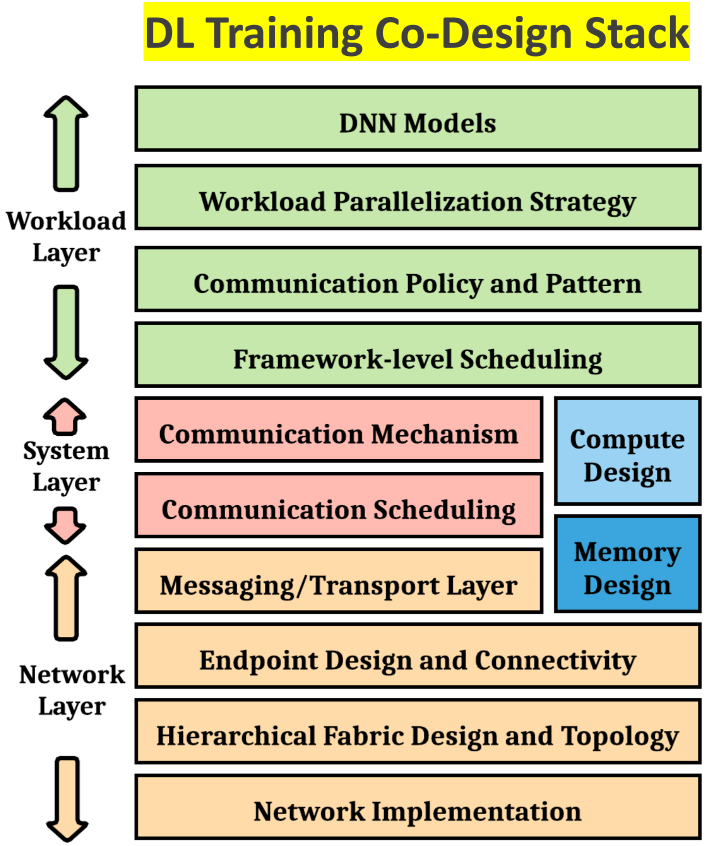
Router

link

Network Parameters

Network Simulator

http://github.com/astra-sim/astra-sim



DL Training Co-Design Stack

Workload Layer

DNN Models

Workload Parallelization Strategy

Communication Policy and Pattern

Framework-level Scheduling

System Layer

Communication Mechanism

Communication Scheduling

Messaging/Transport Layer

Compute Design

Memory Design

Network Layer

Endpoint Design and Connectivity

Hierarchical Fabric Design and Topology

Network Implementation

*S. Rashidi et al.,"**ASTRA-SIM: Enabling SW/HW Co-Design Exploration for Distributed DL Training Platforms**", ISPASS 2020*

*S. Rashidi, et al., "**Scalable Distributed Training of Recommendation Models: An ASTRA-SIM + NS3 case-study with TCP/IP transport**", Hot Interconnects 2020*

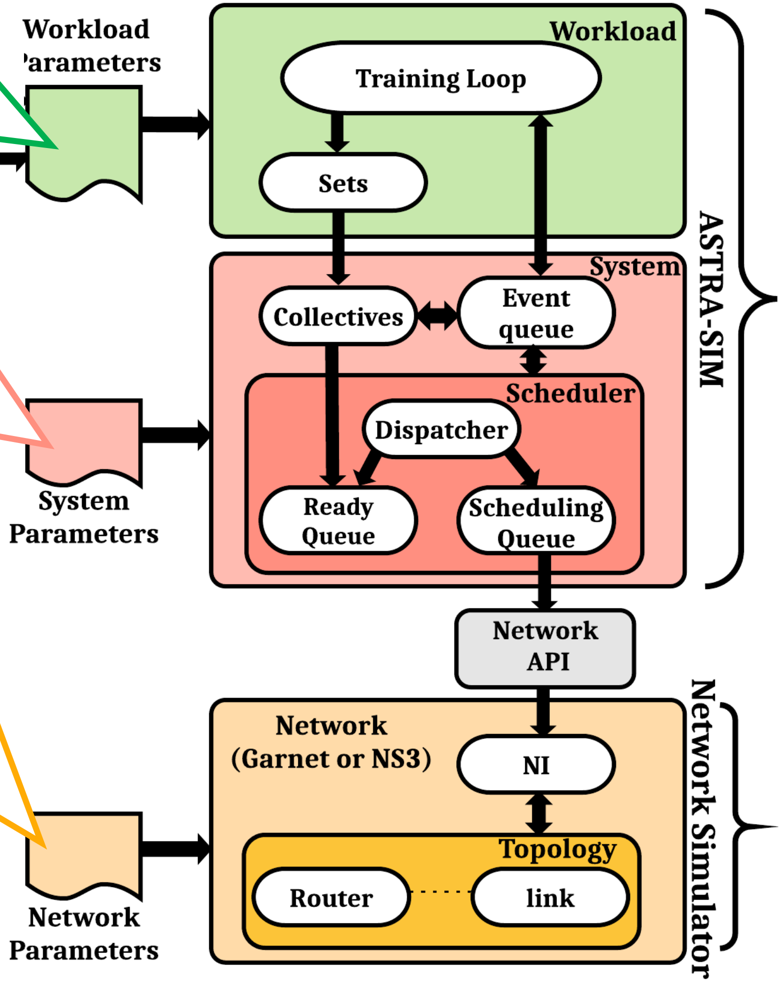
ASTRA-sim: Status and On-going Development

✓ *Released* ➢ *In progress*

- ✓ Supports Data-Parallel, Model-Parallel, Hybrid-Parallel training loops
- ✓ Extensible to more training loops
  - ➢ Graph-based input from PyTorch

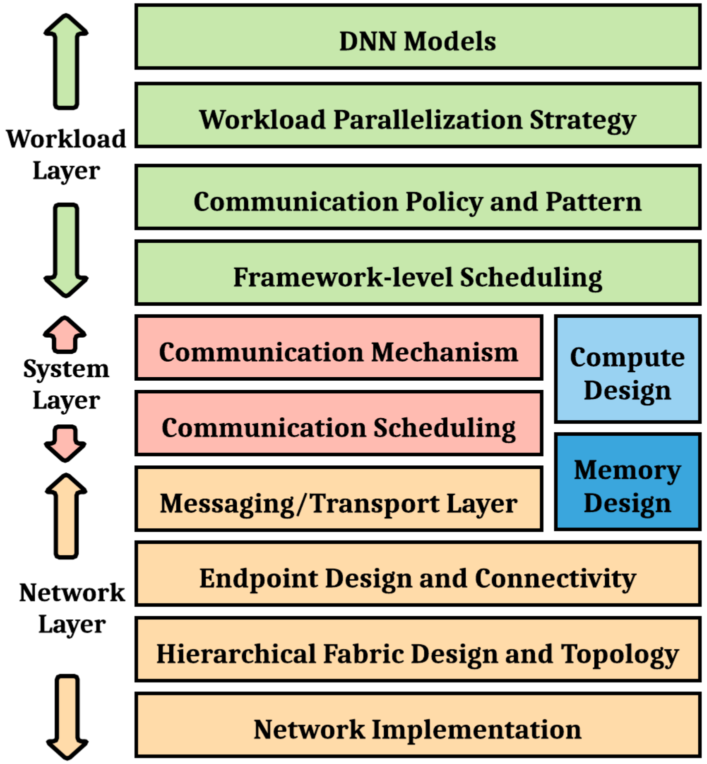
- ✓ Ring based, Tree-based, AlltoAll based, and multi-phase collectives
- ✓ Variety of scheduling policies
- ✓ Compute times fed via offline system measurements or compute simulator

- ✓ Various topologies, flow-control, link bandwidth, congestion control
- ✓ Plug-and-play options
  - ✓ Analytical (roofline)
  - ➢ Analytical with congestion
  - ✓ Garnet (credit-based)
  - ➢ NS3 (TCP, RDMA)



http://github.com/astra-sim/astra-sim

DL Training Co-Design Stack


DNN Models
Workload Parallelization Strategy
Communication Policy and Pattern
Framework-level Scheduling
Workload Layer
Communication Mechanism
Communication Scheduling
Compute Design
Memory Design
System Layer
Messaging/Transport Layer
Endpoint Design and Connectivity
Hierarchical Fabric Design and Topology
Network Implementation
Network Layer

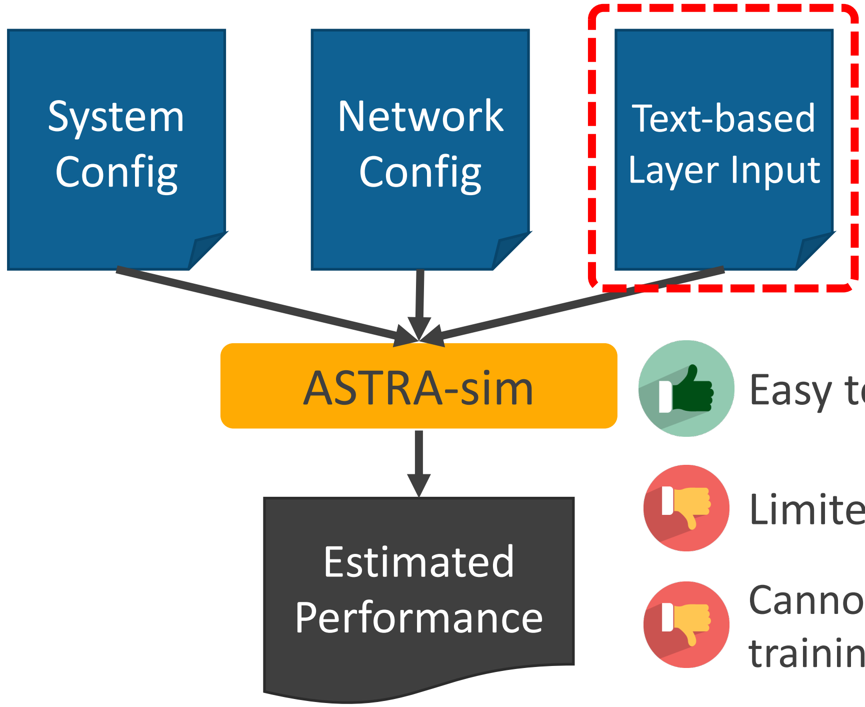
S. Rashidi et al.,"***ASTRA-SIM: Enabling SW/HW Co-Design Exploration for Distributed DL Training Platforms***", ISPASS 2020

S. Rashidi, et al., "***Scalable Distributed Training of Recommendation Models: An ASTRA-SIM + NS3 case-study with TCP/IP transport***", Hot Interconnects 2020

Execution-graph-based Workload Layer

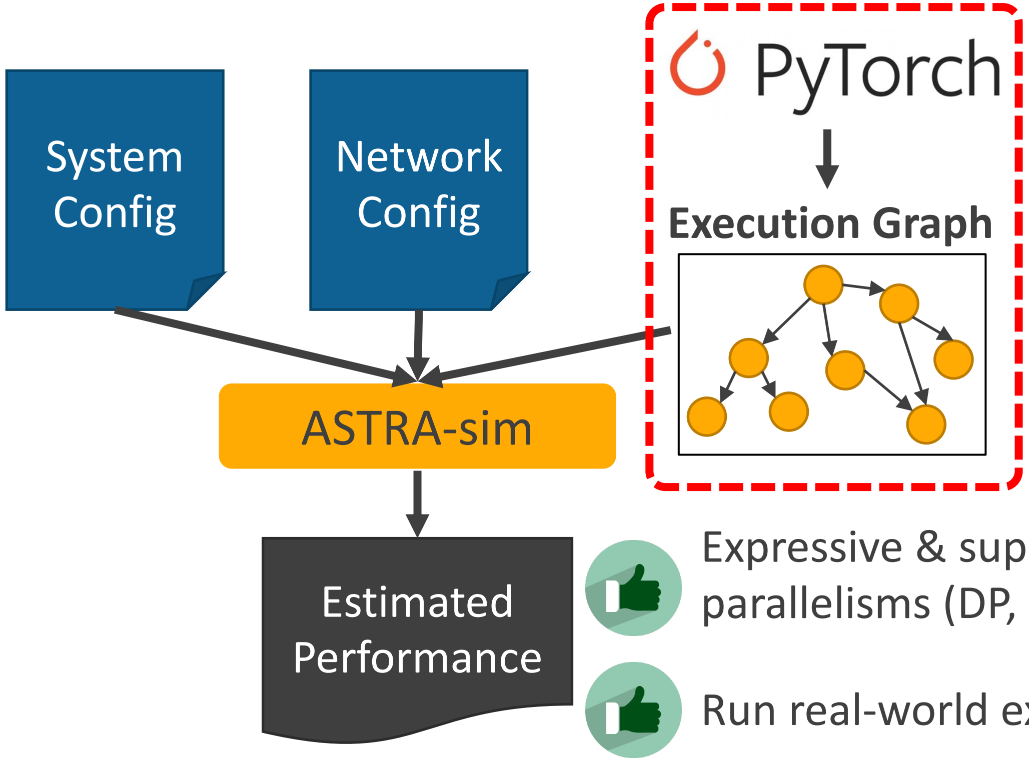
- **Limitation**: ASTRA-sim cannot model complex training loops (pipeline)
- **Solution**: Run ASTRA-sim with execution graphs

Current ASTRA-sim



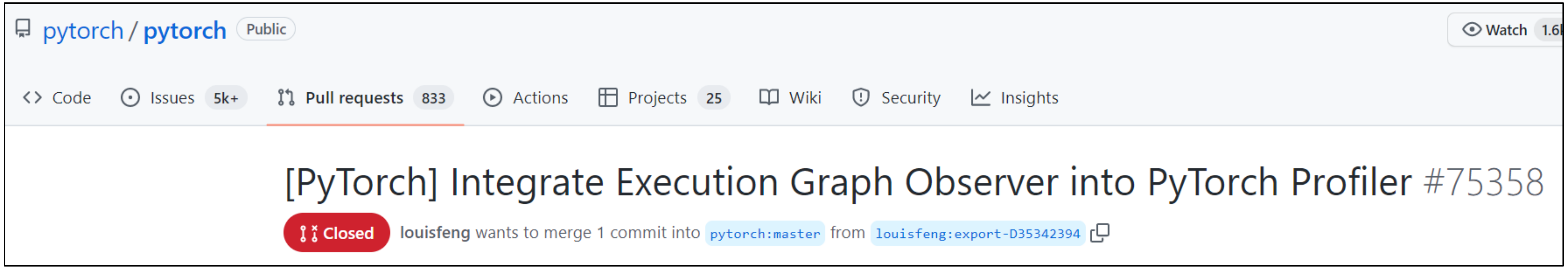
- 👍 Easy to write
- 👎 Limited expressiveness
- 👎 Cannot run complex training loops (pipeline)

Execution-graph-based ASTRA-sim



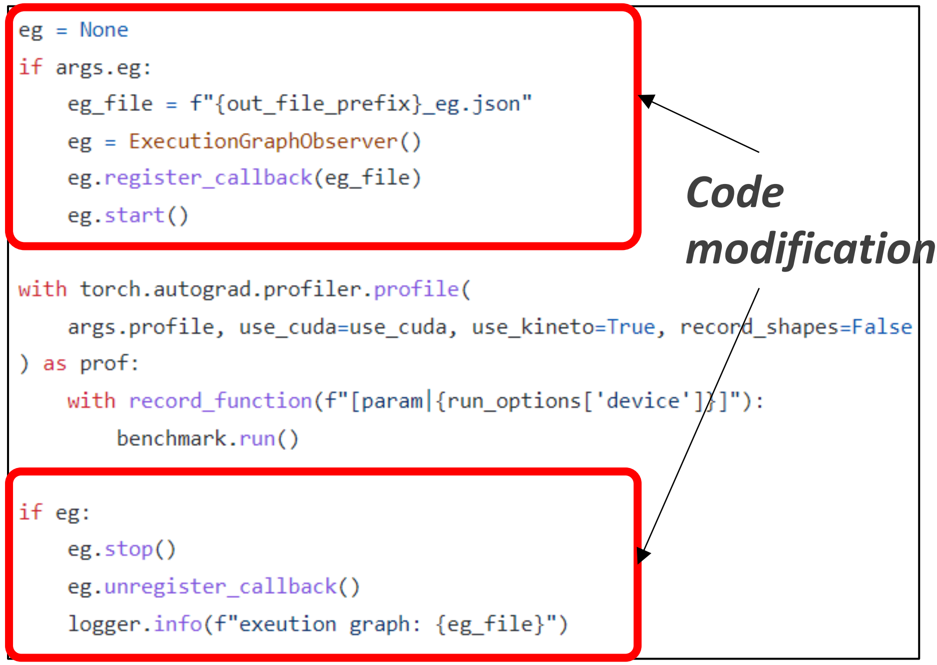
- 👍 Expressive & supports all parallelisms (DP, MP, PP)
- 👍 Run real-world exec traces

Execution-graph-based Workload Layer



[PyTorch] Integrate Execution Graph Observer into PyTorch Profiler #75358

Closed louisfeng wants to merge 1 commit into `pytorch:master` from `louisfeng:export-D35342394`



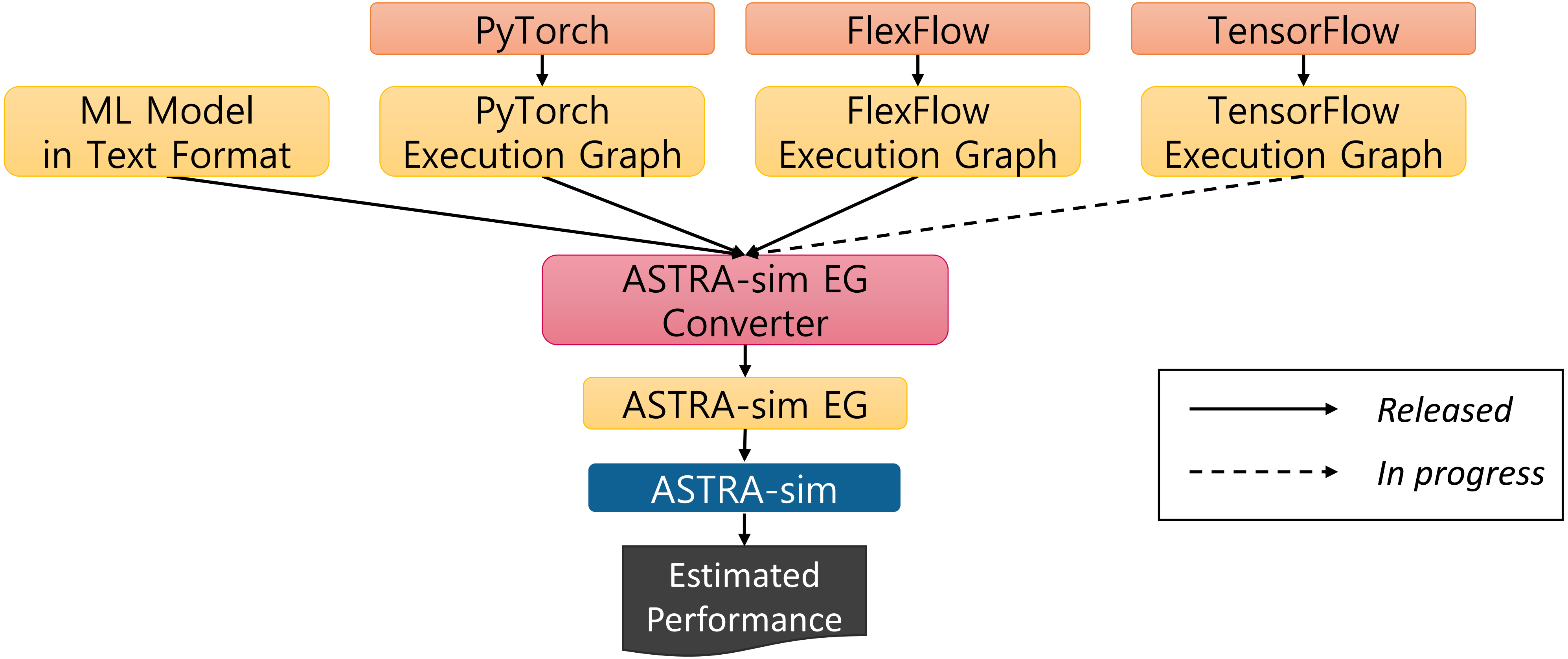
```
eg = None
if args.eg:
    eg_file = f"{out_file_prefix}_eg.json"
    eg = ExecutionGraphObserver()
    eg.register_callback(eg_file)
    eg.start()

with torch.autograd.profiler.profile(
    args.profile, use_cuda=use_cuda, use_kineto=True, record_shapes=False
) as prof:
    with record_function(f"[param|{run_options['device']}]"):
        benchmark.run()

if eg:
    eg.stop()
    eg.unregister_callback()
    logger.info(f"exeution graph: {eg_file}")
```

Code modifications

- PyTorch supports collecting execution graphs
- Requires less than 10 lines of codes

Execution-graph-based Workload Layer

- We provide a converter to support text input files and any other execution graphs (EGs)



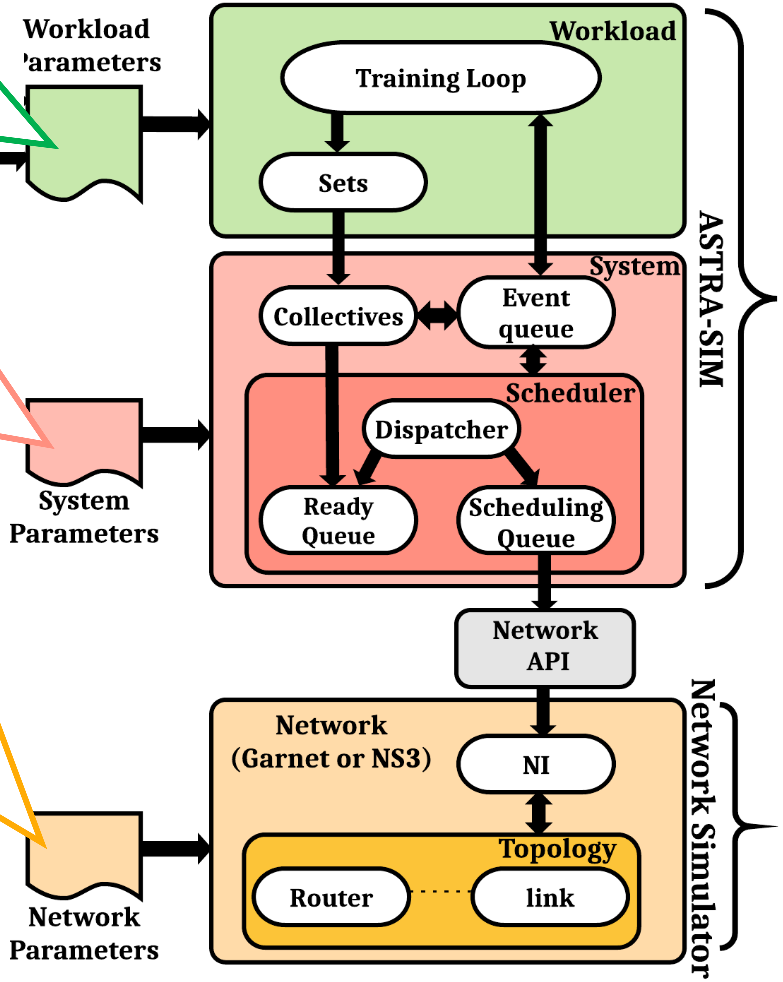
ASTRA-sim: Status and On-going Development

✓ *Released* ➢ *In progress*

- ✓ Supports Data-Parallel, Model-Parallel, Hybrid-Parallel training loops
- ✓ Extensible to more training loops
  - ➢ Graph-based input from PyTorch

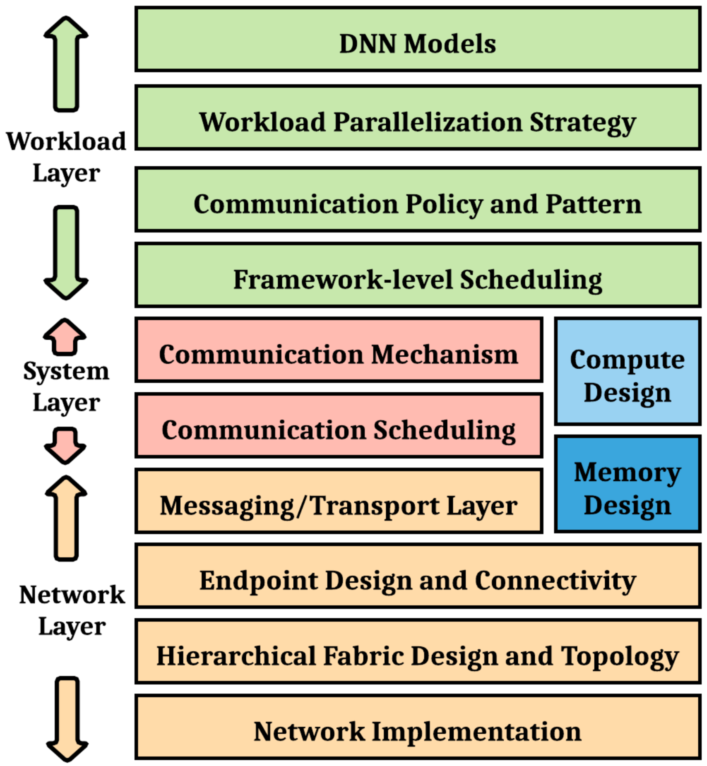
- ✓ Ring based, Tree-based, AlltoAll based, and multi-phase collectives
- ✓ Variety of scheduling policies
- ✓ Compute times fed via offline system measurements or compute simulator

- ✓ Various topologies, flow-control, link bandwidth, congestion control
- ✓ Plug-and-play options
  - ✓ Analytical (roofline)
  - ➢ Analytical with congestion
  - ✓ Garnet (credit-based)
  - ➢ NS3 (TCP, RDMA)



http://github.com/astra-sim/astra-sim

DL Training Co-Design Stack



Workload
Layer
System
Layer
Network
Layer
DNN Models
Workload Parallelization Strategy
Communication Policy and Pattern
Framework-level Scheduling
Communication Mechanism
Communication Scheduling
Messaging/Transport Layer
Compute
Design
Memory
Design
Endpoint Design and Connectivity
Hierarchical Fabric Design and Topology
Network Implementation

*S. Rashidi et al.,"**ASTRA-SIM: Enabling SW/HW Co-Design Exploration for Distributed DL Training Platforms**", ISPASS 2020*

*S. Rashidi, et al., "**Scalable Distributed Training of Recommendation Models: An ASTRA-SIM + NS3 case-study with TCP/IP transport**", Hot Interconnects 2020*

Network Backends

✓ Released

- **Analytical:**
  - Fastest backend.
  - Models a rich set of hierarchical networks.
  - Accurate for congestion-less topology/comm patterns.
- **Garnet:**
  - Credit-based flow control modeling.
  - Most accurate for NOCs and chiplet-based interconnects.

➤ In progress

- **Analytical + Congestion Modeling:**
  - Same as Analytical but performs message-level congestion modeling.
  - Expected to model patterns with congestions with 10-20% error rate.
- **NS3:**
  - Models RDMA over converged ethernet (RoCE) comm protocol.
  - Supports several congestion control schemes (DCQCN, HPCC, Timely, etc.).

*Tarannum Khan, Saeed Rashidi, Srinivas Sridharan, Pallavi Shurpali, Aditya Akella and Tushar Krishna, ”**Impact of RoCE Congestion Control Policies on Distributed Training of DNNs**”. HOTI 2022.*
https://arxiv.org/abs/2207.10898

Contribution and Participation

- **The new features will be released soon!**
  - Please reach out to us for early access

- **ASTRA-sim is open-source!**
  - Feel free to raise GitHub issues and contribute via pull-requests

Organization Team

Contact any/all four of us if any questions

Presenters

Tushar Krishna
Associate Professor, School of ECE
Georgia Institute of Technology
tushar@ece.gatech.edu

Saeed Rashidi
PhD Student, School of ECE
Georgia Institute of Technology
saeed.rashidi@ece.gatech.edu

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

Taekyung Heo
Postdoctoral Fellow, School of ECE
Georgia Institute of Technology
taekyung@gatech.edu

Collaborators

Srinivas Sridharan
Research Scientist, Meta

Sudarshan Srinivasan
Research Scientist, Intel

Thank you!