https://astra-sim.github.io

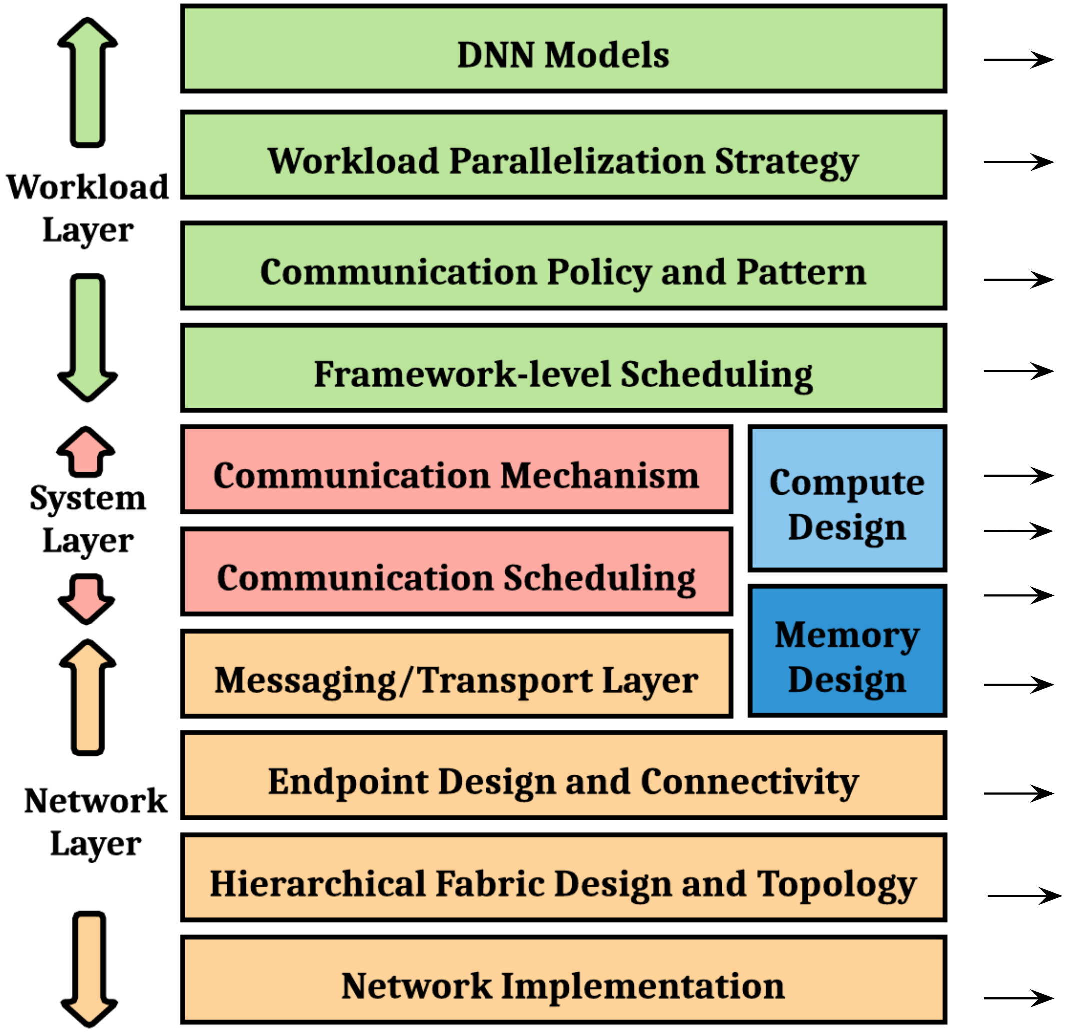
https://github.com/mlcommons/chakra

ASTRA-sim Tutorial
@MICRO 2024
November 3, 2024

ASTRA-sim and Chakra Tutorial: *Overview of Chakra and ASTRA-sim*

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

Challenge: Complex SW/HW Co-Design Space



Workload Layer

DNN Models → DLRM, ResNet-50, Transformer, GNMT

Workload Parallelization Strategy → Data, Model, Platform Agnostic Hybrid, Platform-aware Hybrid, Pipelined Parallelism

Communication Policy and Pattern → Distributed worker, Parameter Server

Framework-level Scheduling → Sync/Async, Blocking/Non-blocking

System Layer

Communication Mechanism → Topology-aware Collectives, Send/Recv, RPC

Compute Design → Dataflow, Microarchitecture, Flexibility, Sparsity Support

Communication Scheduling → LIFO, FIFO, Fusion

Memory Design

Messaging/Transport Layer → TCP, RDMA (+ GPUDirect RDMA)

Network Layer

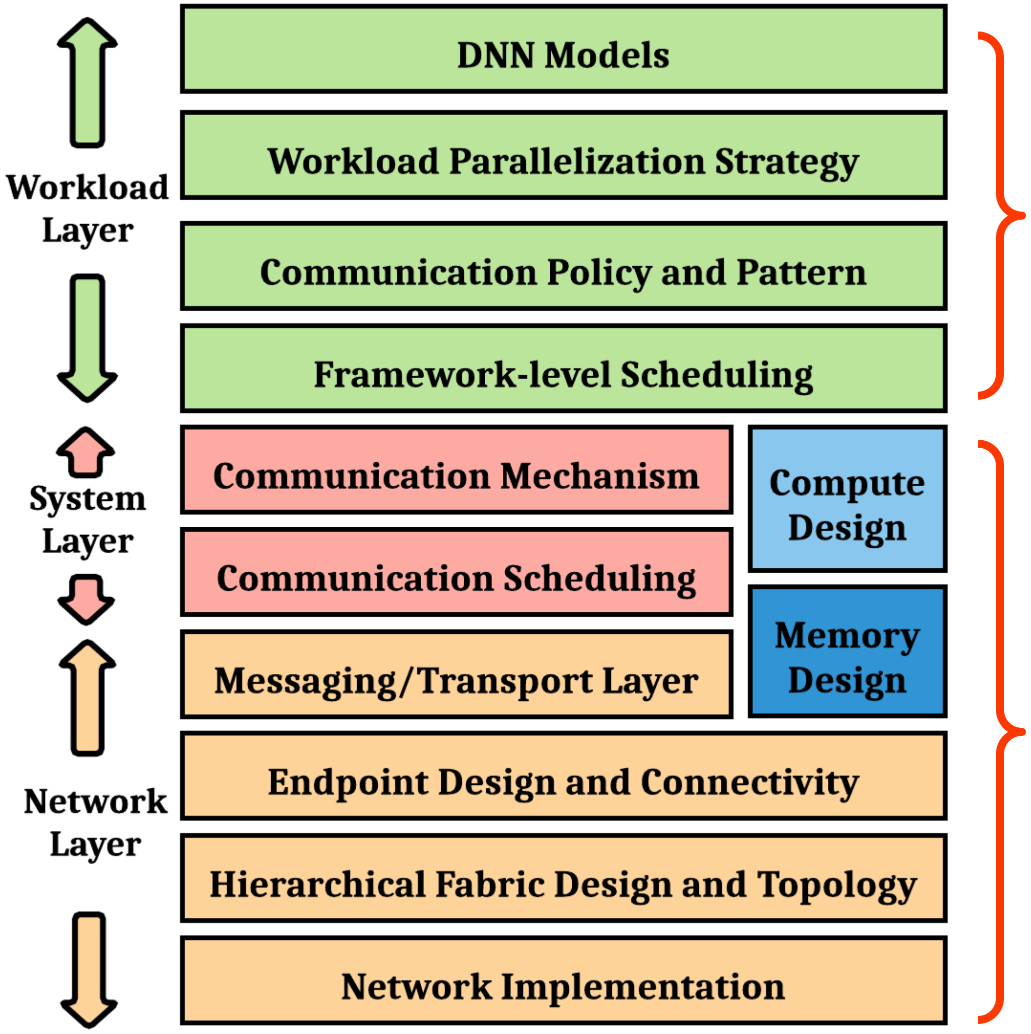
Endpoint Design and Connectivity → # links, BW per link, architecture (chip/package/board), NIC offload, compression

Hierarchical Fabric Design and Topology → Flat vs. Hierarchical, 2D/3D/4D Torus (TPU), Switch (DGX2), Fully-Connected, Hyper-Cube Mesh

Network Implementation → Buffering, Flow-control, Arbitration, Congestion Mgmt

How do we model this complex design-space?

Introducing Chakra and ASTRA-sim



Workload Layer

DNN Models

Workload Parallelization Strategy

Communication Policy and Pattern

Framework-level Scheduling

System Layer

Communication Mechanism

Communication Scheduling

Compute Design

Memory Design

Network Layer

Messaging/Transport Layer

Endpoint Design and Connectivity

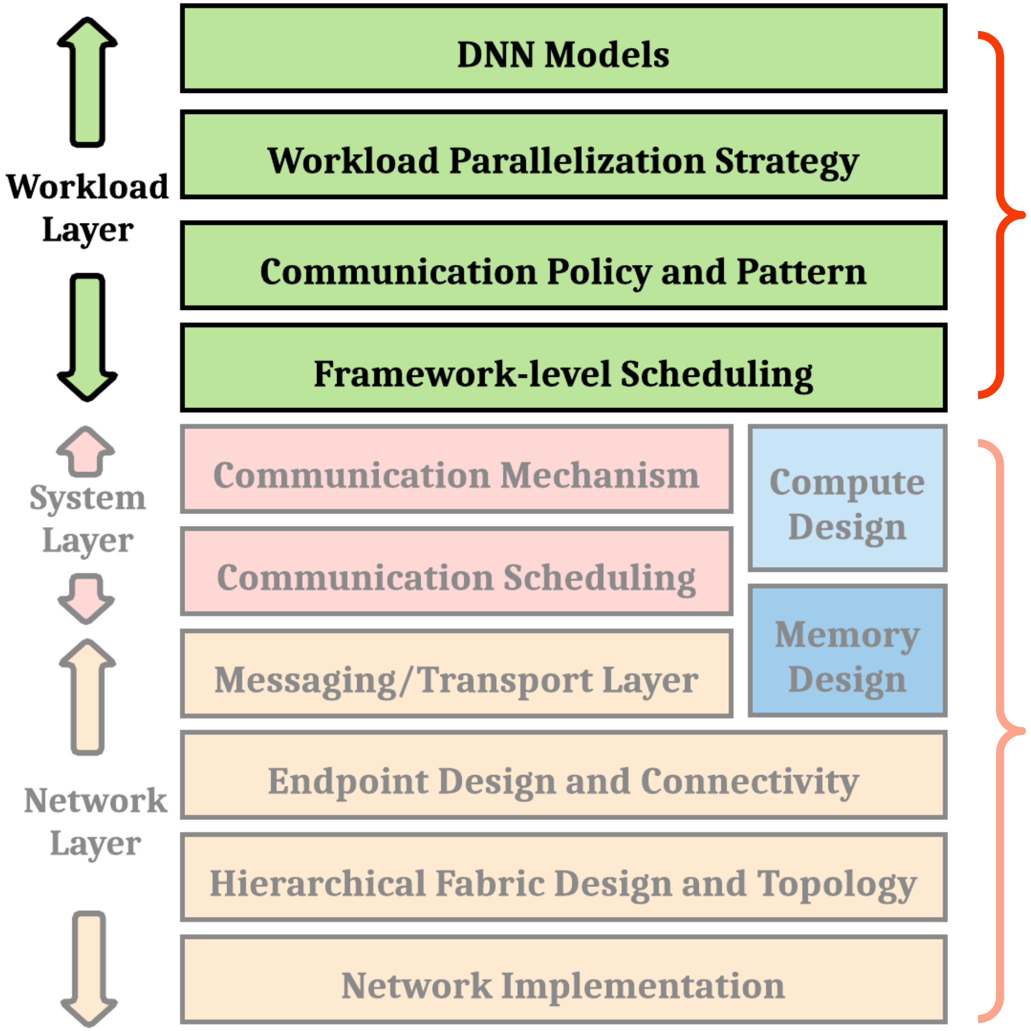
Hierarchical Fabric Design and Topology

Network Implementation

Chakra Execution Trace: an open graph-based representation of AI/ML workload execution

ASTRA-sim: Distributed AI system simulator

Introducing Chakra and ASTRA-sim



Workload Layer

DNN Models

Workload Parallelization Strategy

Communication Policy and Pattern

Framework-level Scheduling

System Layer

Communication Mechanism

Communication Scheduling

Compute Design

Memory Design

Messaging/Transport Layer

Network Layer

Endpoint Design and Connectivity

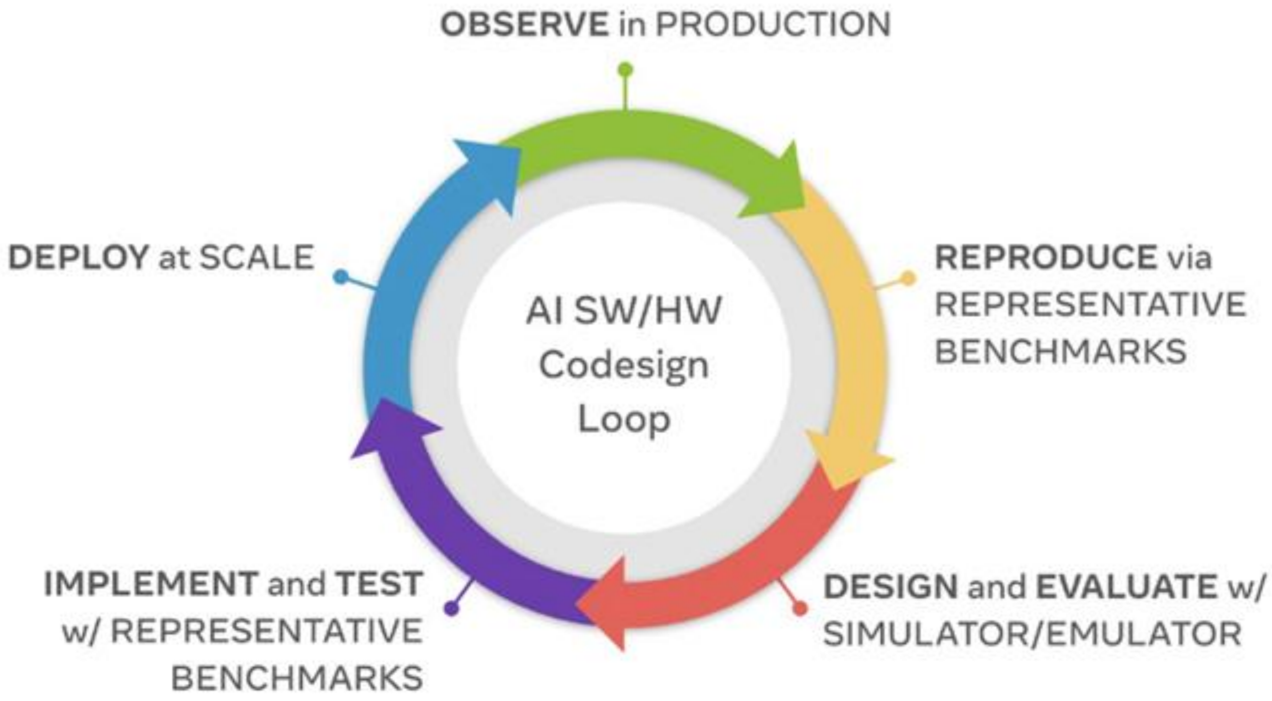
Hierarchical Fabric Design and Topology

Network Implementation

Chakra Execution Trace: an open graph-based representation of AI/ML workload execution

ASTRA-sim: Distributed AI system simulator

Chakra: Motivation

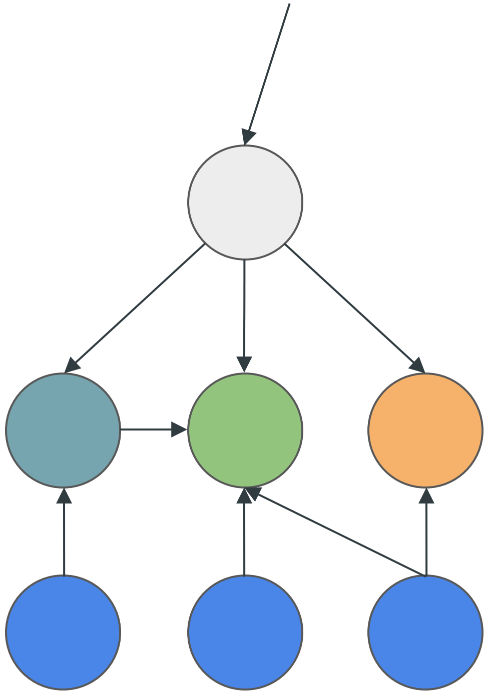

OBSERVE in PRODUCTION
REPRODUCE via REPRESENTATIVE BENCHMARKS
DESIGN and EVALUATE w/ SIMULATOR/EMULATOR
IMPLEMENT and TEST w/ REPRESENTATIVE BENCHMARKS
DEPLOY at SCALE
AI SW/HW Codesign Loop

Motivation

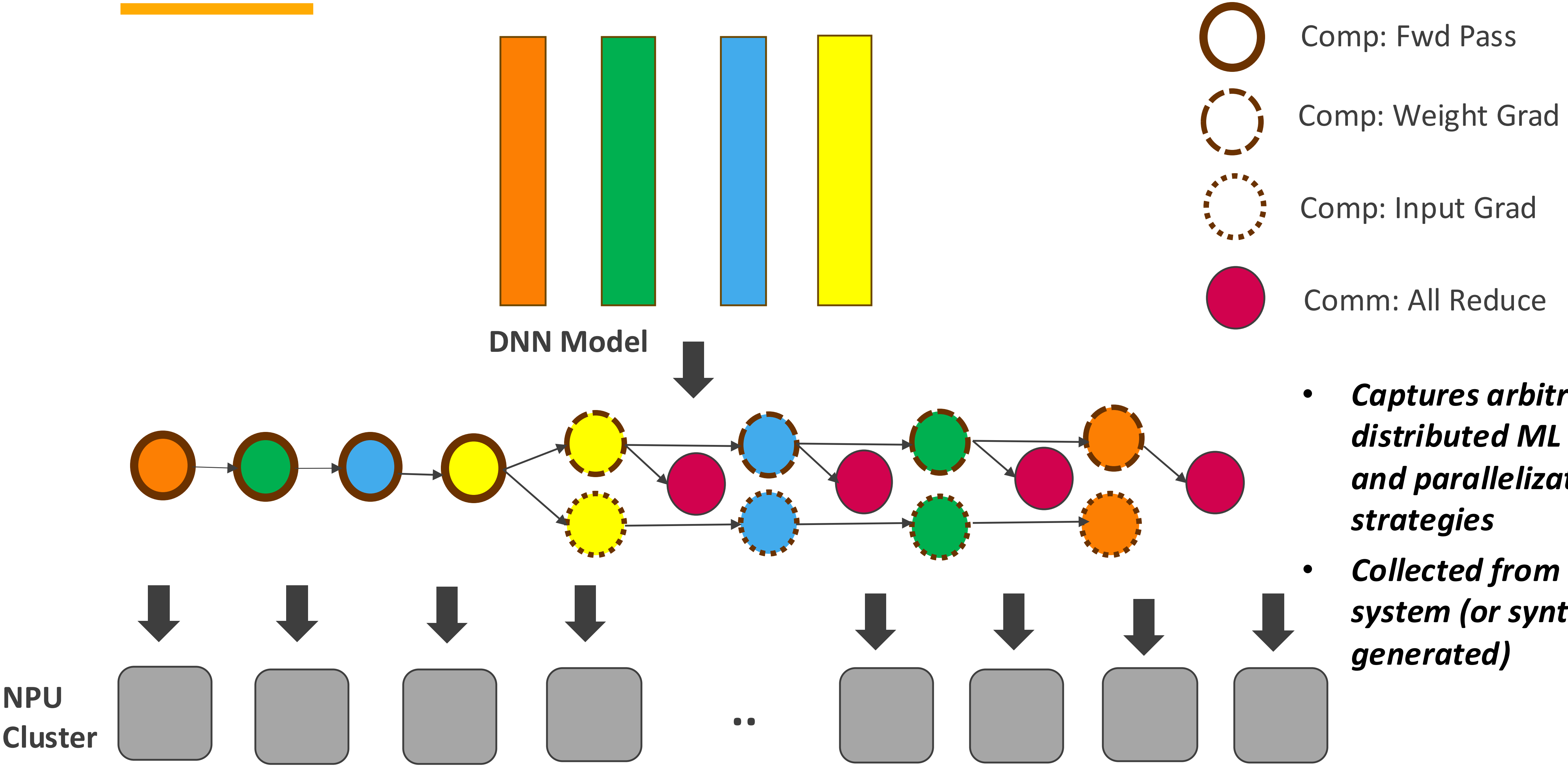
- High-cost of running full workload benchmarks
- Requires cross-domain full-stack expertise
- Difficult to isolate specific HW/SW bottlenecks
- Difficult to isolate compute, memory, network behavior
- Cannot keep up with the pace of AI innovation
- Hard to obfuscate proprietary AI model details
- Hard to reproduce without support infrastructure

Chakra Execution Traces

- **Hierarchical DAG**
- **Nodes**
  - Primitive operators: compute, comms, memory
  - Tensor objects: shape, size, device (local/remote)
  - Timing and resource constraints
- **Edges**
  - Data dependency
  - Control dependency (e.g. call stack)
- **Higher-level abstractions (e.g., components)**
  - Comprises of other components or primitive ops

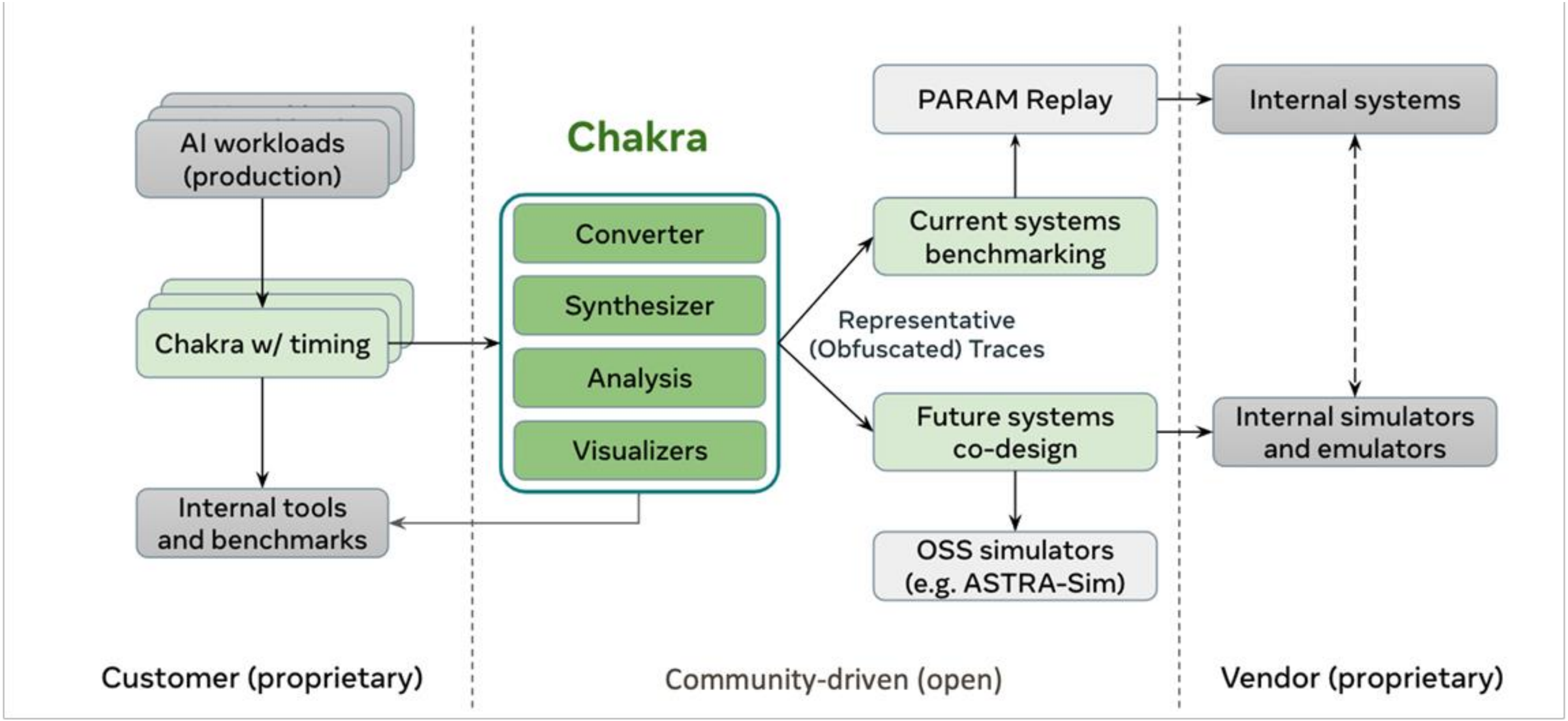


Chakra Execution Traces

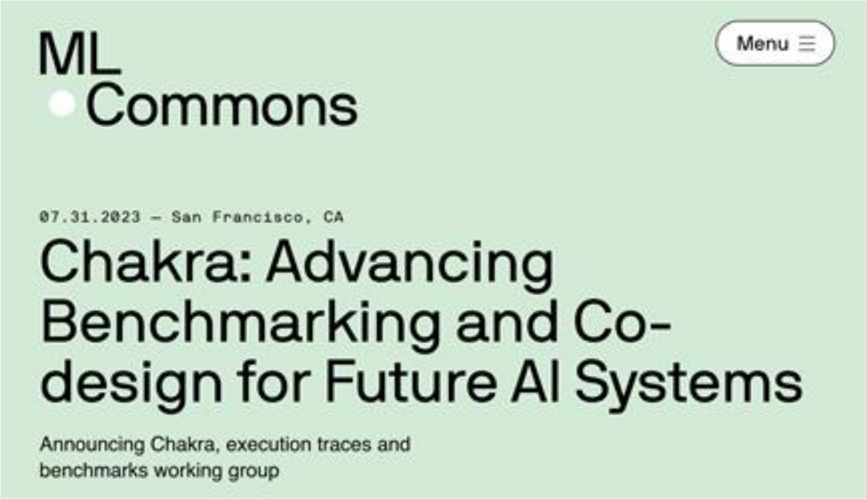


- ***Captures arbitrary distributed ML workload and parallelization strategies***
- ***Collected from real-system (or synthetically generated)***

Chakra Ecosystem and End-to-End Flow


AI workloads (production)
Chakra
Converter
Synthesizer
Analysis
Visualizers
PARAM Replay
Internal systems
Current systems benchmarking
Chakra w/ timing
Representative (Obfuscated) Traces
Future systems co-design
Internal simulators and emulators
Internal tools and benchmarks
OSS simulators (e.g. ASTRA-Sim)
Customer (proprietary)
Community-driven (open)
Vendor (proprietary)

Chakra is now part of MLCommons!



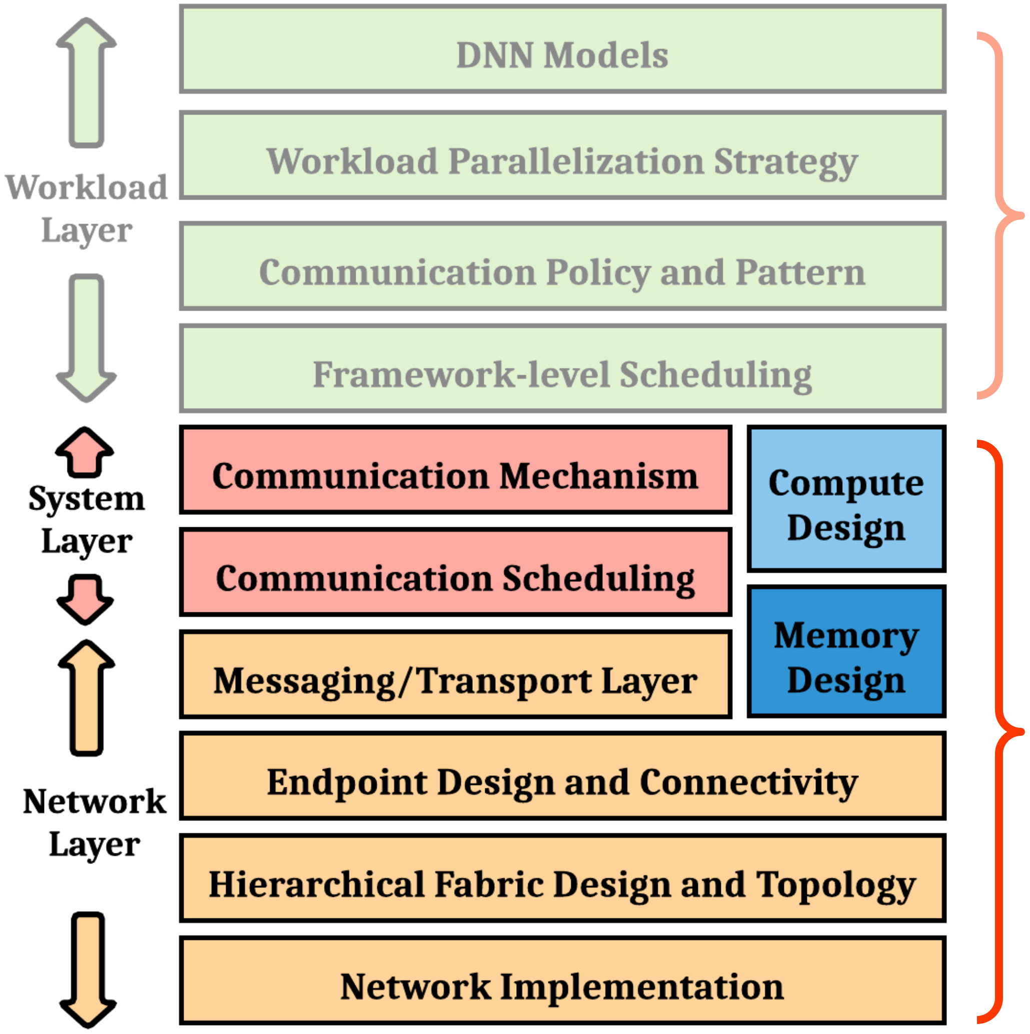
07.31.2023 – San Francisco, CA

Chakra: Advancing Benchmarking and Co-design for Future AI Systems

Announcing Chakra, execution traces and benchmarks working group

- **Build consensus on Execution Trace methodology**
  - Enable easier sharing between hyperscaler/cloud and vendors (with/without NDA)
  - Vendors can focus on different components (compute/memory/network)
  - Enable faster ramp-up for startups and academia
- **Shared engineering effort towards open/vibrant ecosystem**
  - Trace collection and synthesis
  - Support tools and downstream enablement
- **Benchmark suite definition and supervision**
  - Single workload and datacenter-scale benchmark scoring
  - Future workload projection

Introducing Chakra and ASTRA-sim


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

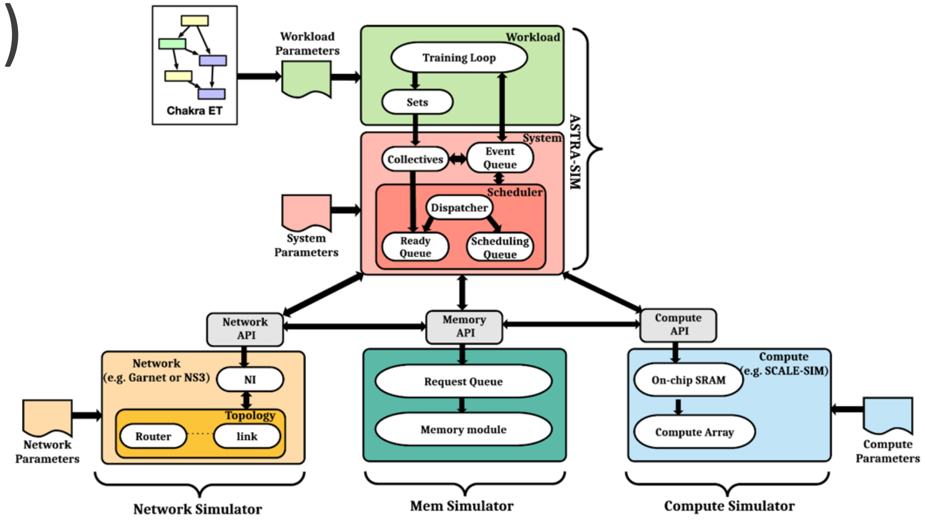
Chakra Execution Trace: an open graph-based representation of AI/ML workload execution

ASTRA-sim: Distributed AI system simulator

ASTRA-sim: Design Principles

A **framework** to model/simulate/emulate AI systems with varying degrees of fidelity.

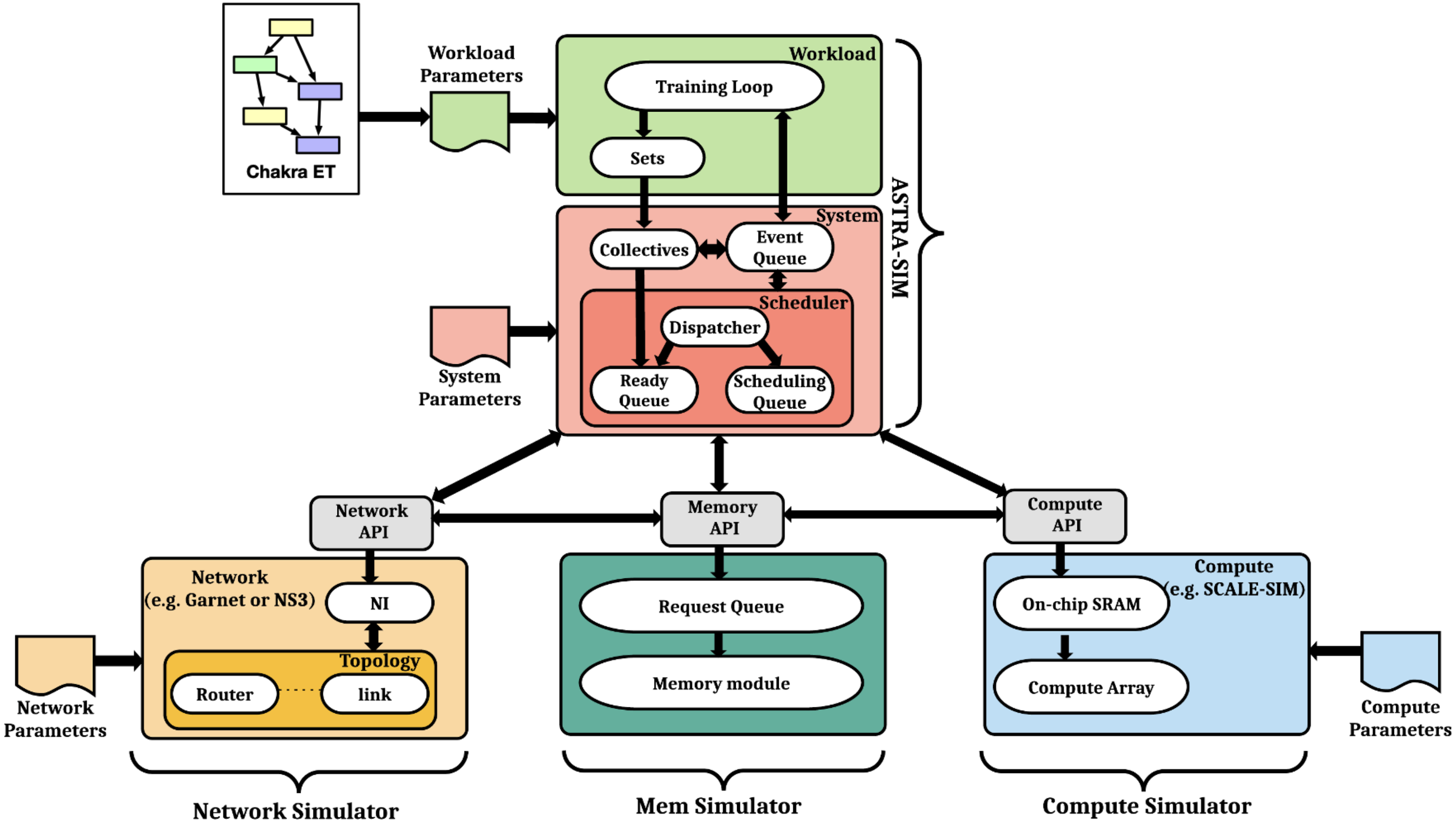
Key enabler: APIs for plugging in diverse external tools (i.e., composable simulators)



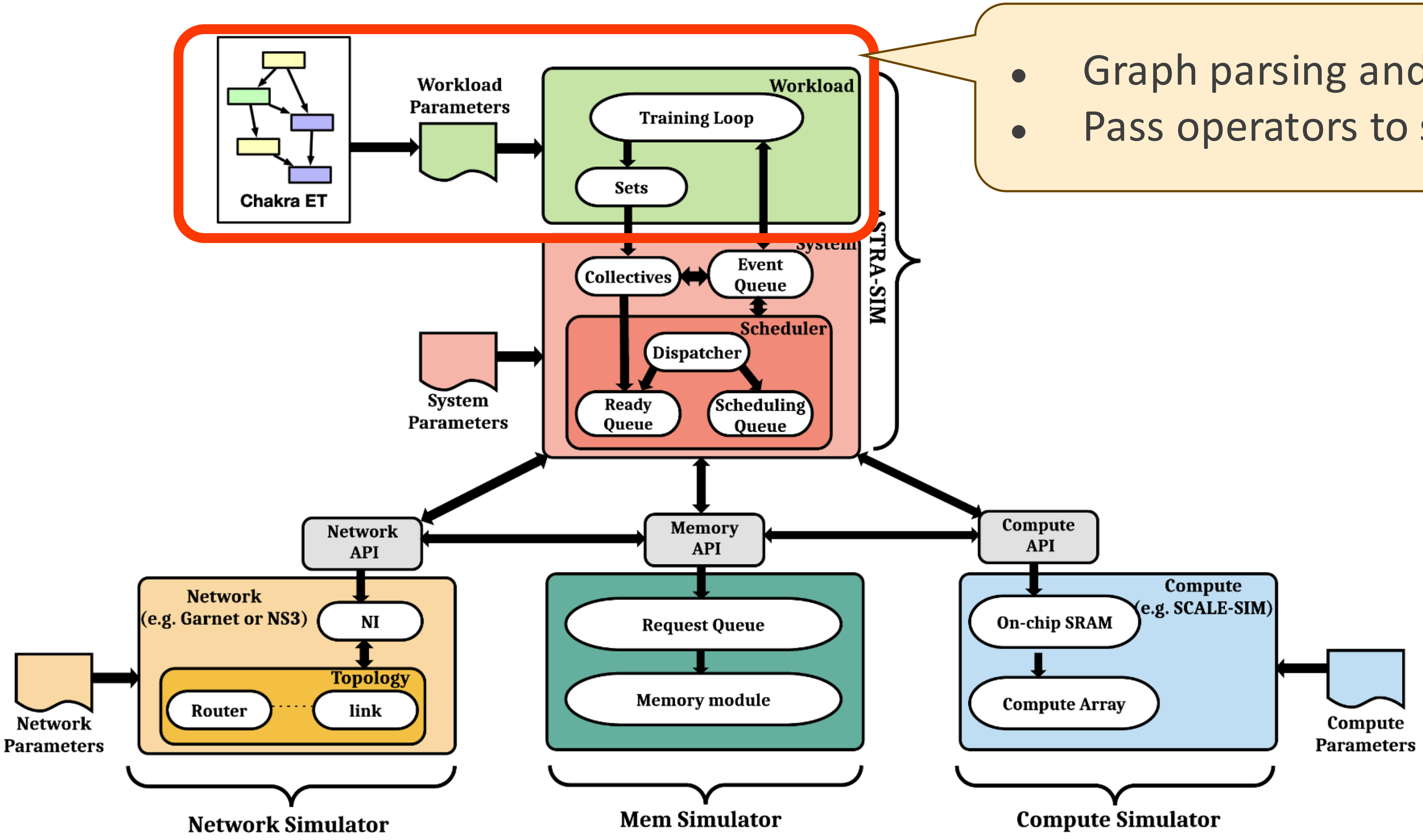
Reference Implementation: http://github.com/astra-sim/astra-sim

Website: https://astra-sim.github.io/

ASTRA-sim

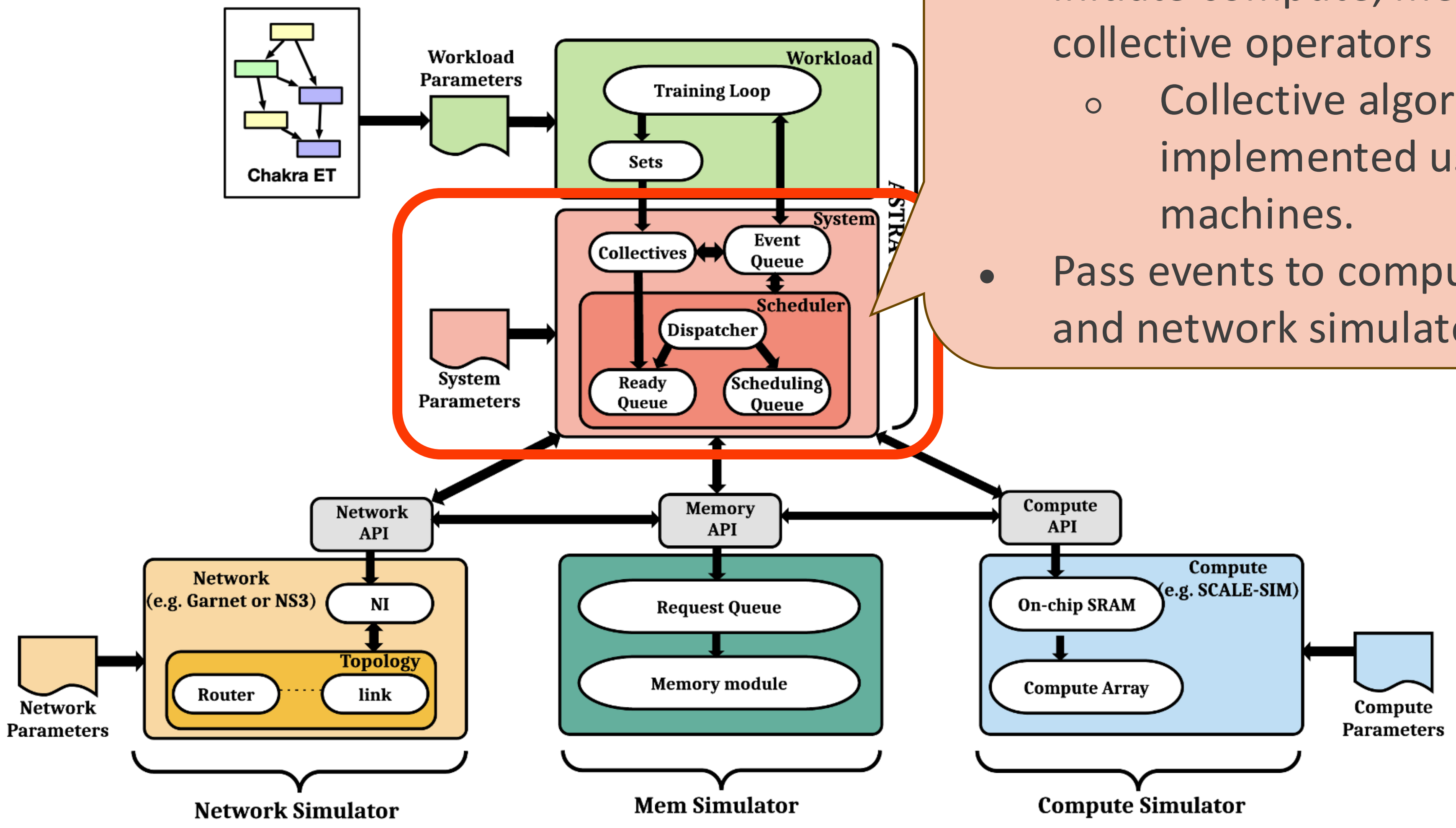

Chakra ET
Workload Parameters
Workload
Training Loop
Sets
System
Collectives
Event Queue
Scheduler
Dispatcher
Ready Queue
Scheduling Queue
ASTRA-SIM
System Parameters
Network API
Memory API
Compute API
Network (e.g. Garnet or NS3)
NI
Topology
Router
link
Network Parameters
Request Queue
Memory module
Compute (e.g. SCALE-SIM)
On-chip SRAM
Compute Array
Compute Parameters
Network Simulator
Mem Simulator
Compute Simulator

ASTRA-sim: Workload Layer



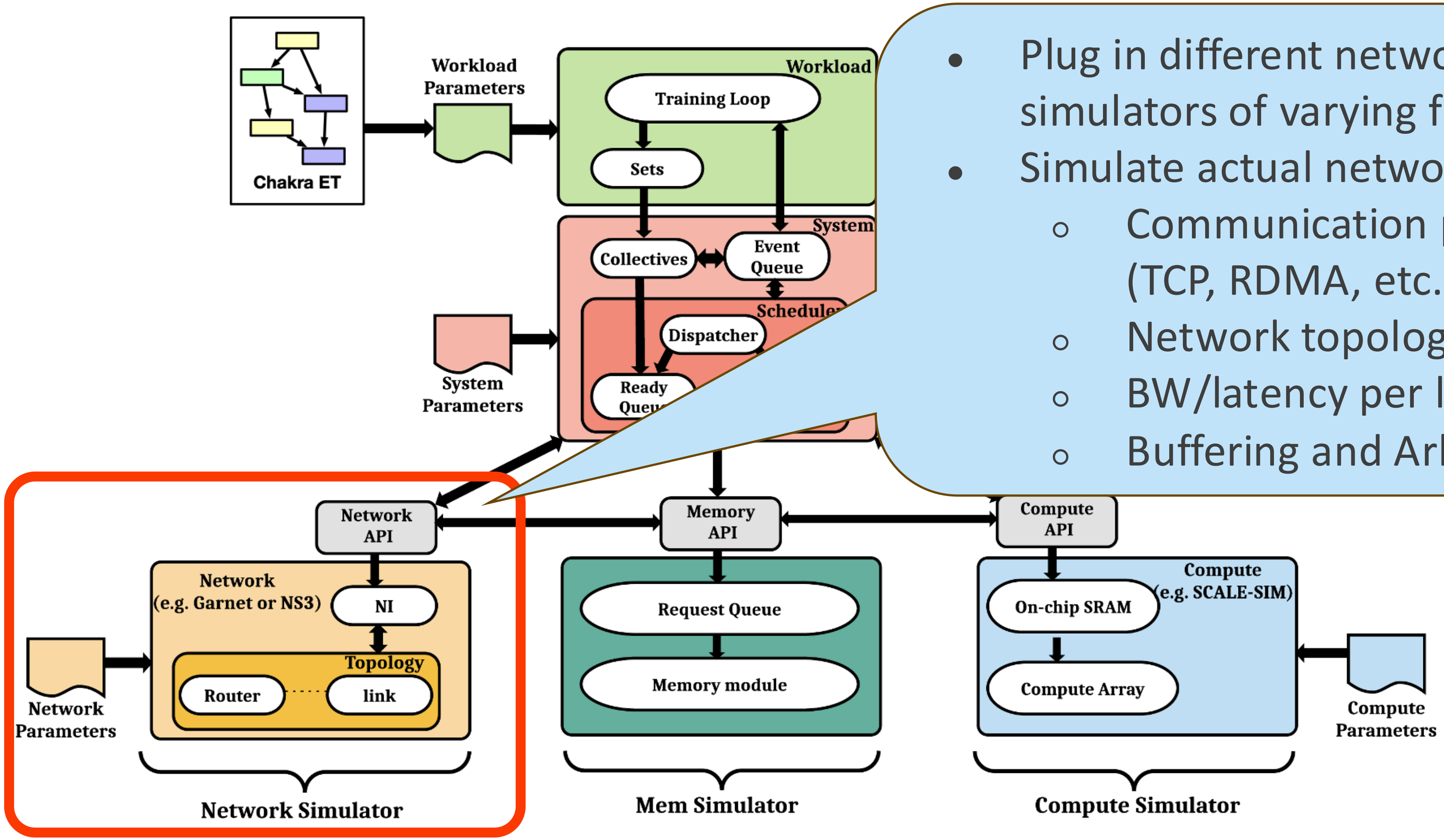
- Graph parsing and execution
- Pass operators to system layer

ASTRA-sim: System Layer



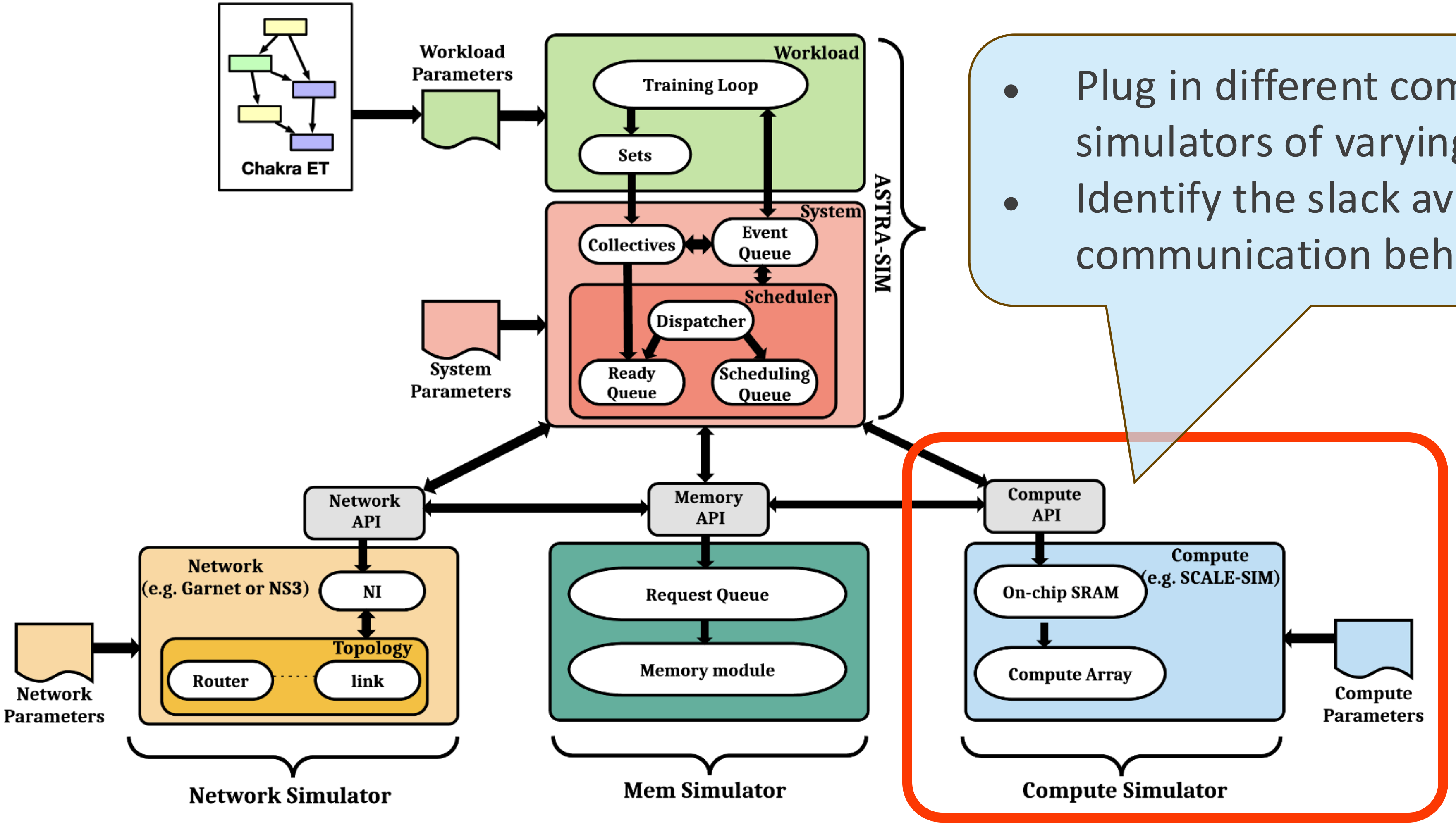
- Initiate compute, memory and collective operators
  - Collective algorithms are implemented using state machines.
- Pass events to compute, memory and network simulators

ASTRA-sim: Network Layer



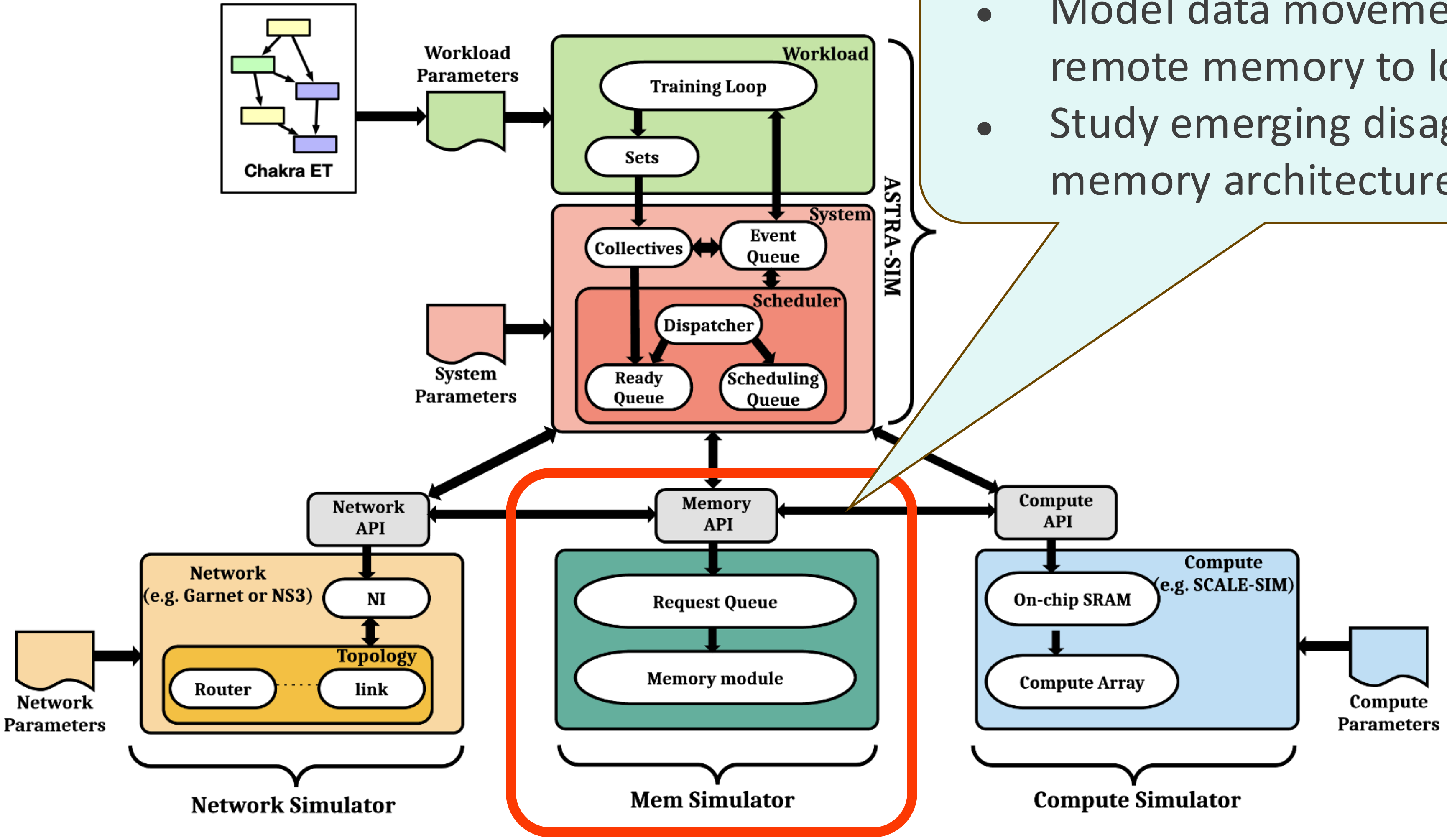
- Plug in different network models / simulators of varying fidelity
- Simulate actual network behavior
  - Communication protocols (TCP, RDMA, etc.)
  - Network topology
  - BW/latency per link
  - Buffering and Arbitration

ASTRA-sim: Compute Layer



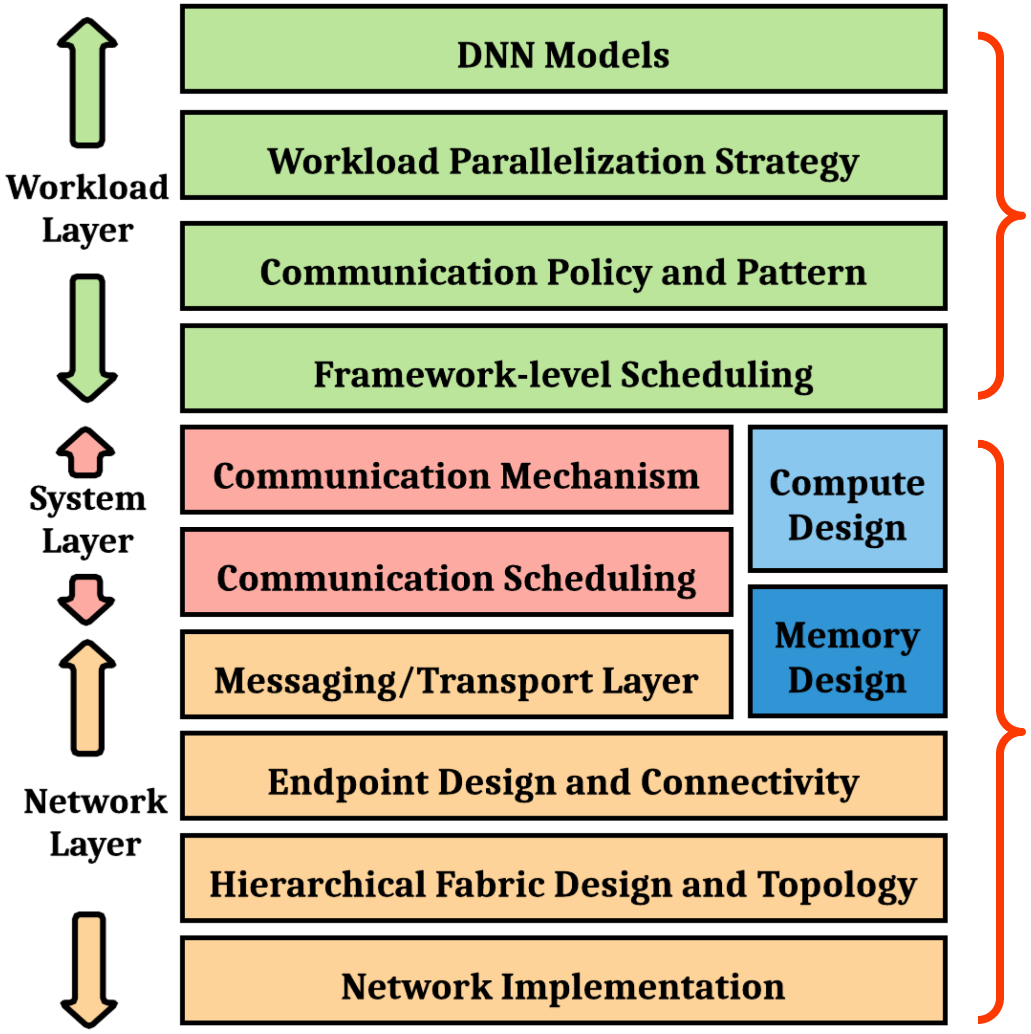
- Plug in different compute models / simulators of varying fidelity
- Identify the slack available to hide communication behind compute

ASTRA-sim: Memory Layer



- Model data movement between remote memory to local memory
- Study emerging disaggregated memory architectures (e.g., CXL)

Introducing Chakra and ASTRA-sim


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

Chakra Execution Trace: an open graph-based representation of AI/ML workload execution

- enables isolation and optimization of compute, memory, communication behavior
- an ecosystem for benchmarking, performance analysis, and performance projection

ASTRA-sim: Distributed AI system simulator

- effectively models various aspects of distributed training
- allows mix-and-match of performance models for compute, memory and network (API-based)