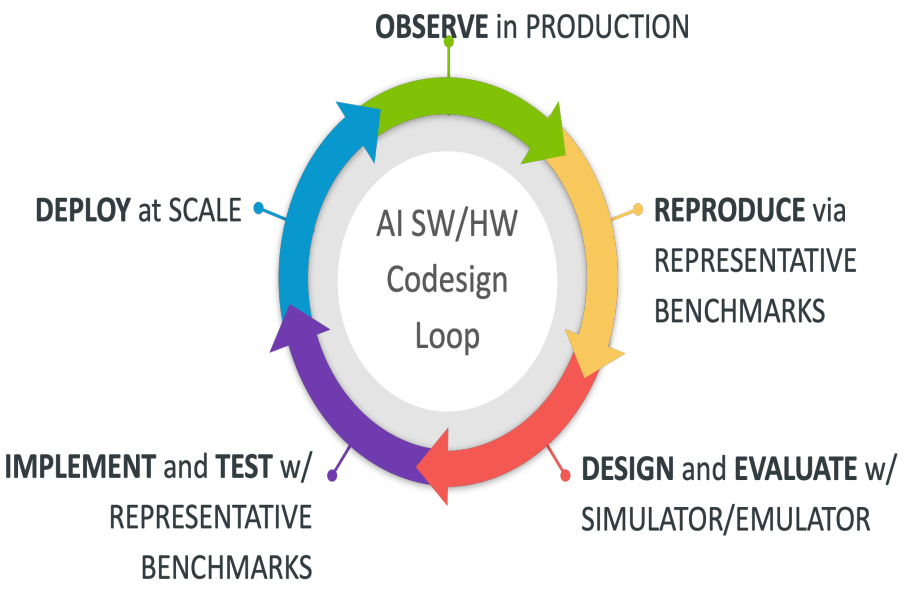
https://github.com/mlcommons/chakra

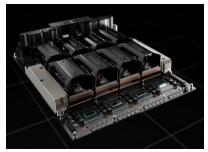
13th Annual MVAPICH
User Conference
August 18-20, 2025

Chakra – Standardized AI Workload Traces for Co-Design

Winston Liu, Dan Mihailescu
Keysight Technologies

Co-Design in Distributed AI Platforms

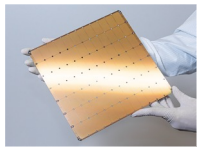




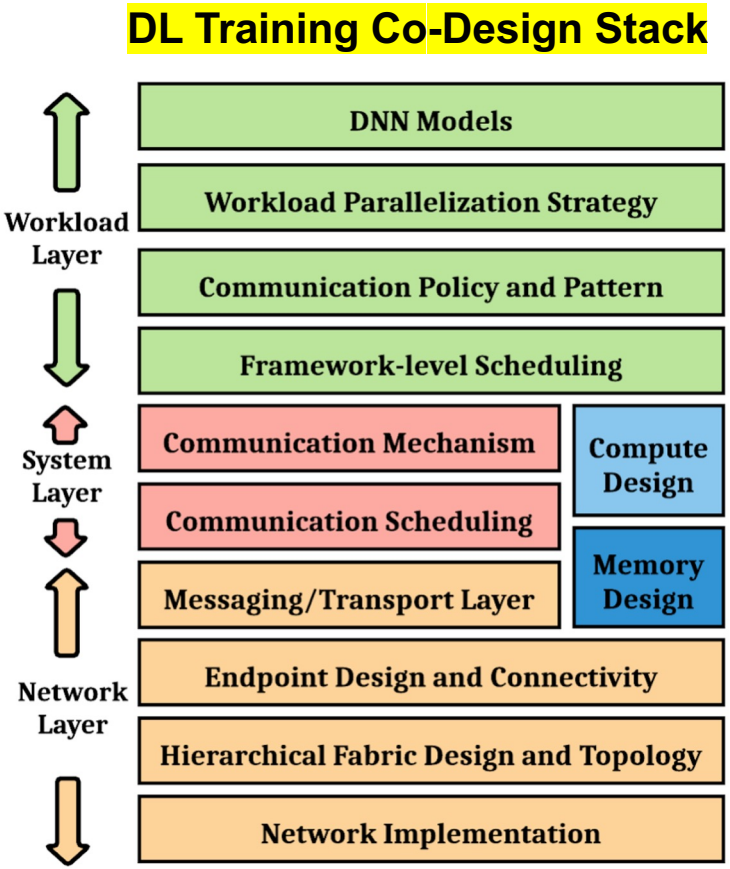
NVIDIA HGX-H100

Google TPUv4

AMD Instinct



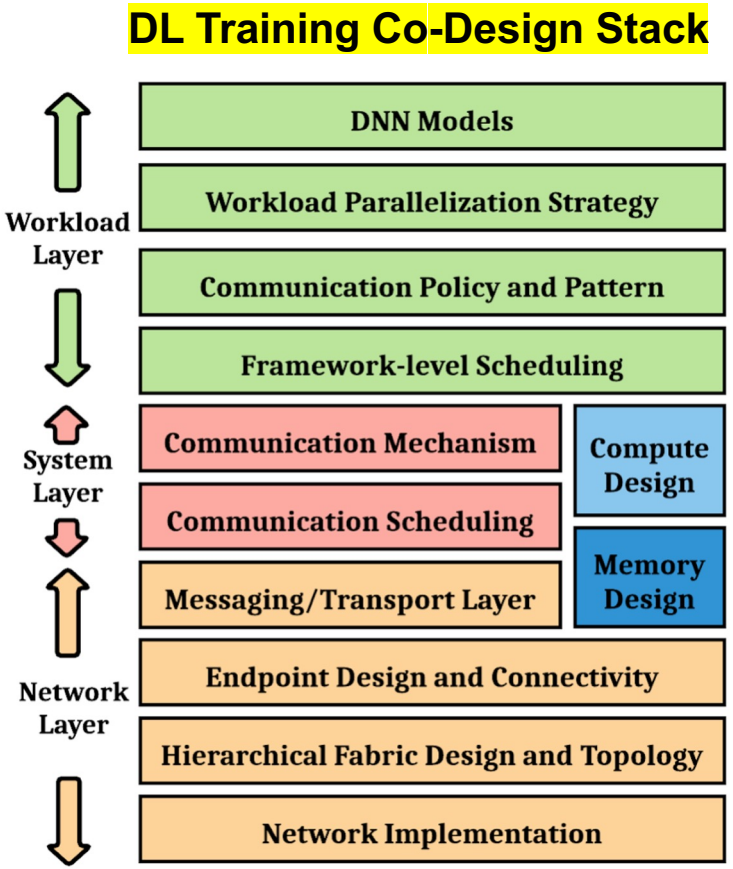
Cerebras CS-2



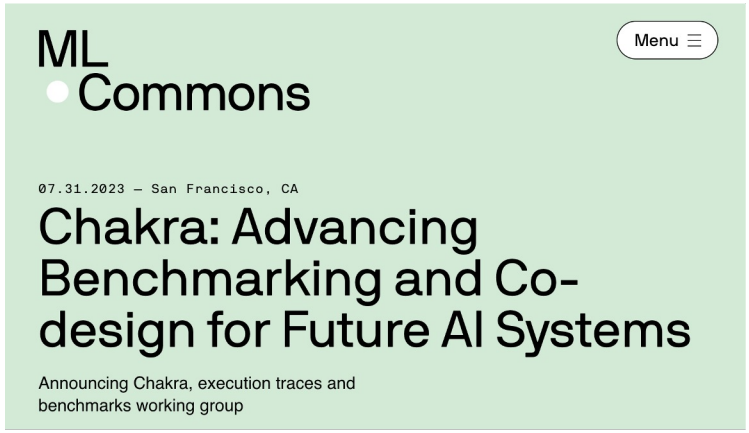
Challenges

- Limitations of full workload benchmarks
  1. High-cost of running full workload benchmarks
  2. Requires cross-domain full-stack expertise
  3. Difficult to isolate specific HW/SW bottlenecks
  4. Cannot keep up with the pace of AI innovation

- Limitations of ad-hoc benchmarks
  1. Incompatible format across tools and companies
  2. Exposure of proprietary AI model details
  3. Cannot to apply traces to different architectures/systems



Chakra is part of MLCommons!



- **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

MLCommons Chakra Working Group Members

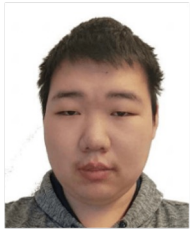
Georgia Tech


Tushar Krishna


William Won


Joongun Park


Jinsun Yoo


Changhai Man

NVIDIA


Srinivas Sridharan


Taekyung Heo


Brian Coutinho

KEYSIGHT TECHNOLOGIES


Winston Liu


Dan Mihailescu

Andy Balogh

Meta


Saeed Rashidi


Sheng Fu

AMD


Brad Beckmann


Vinay Ramakrishnaiah

MARVELL

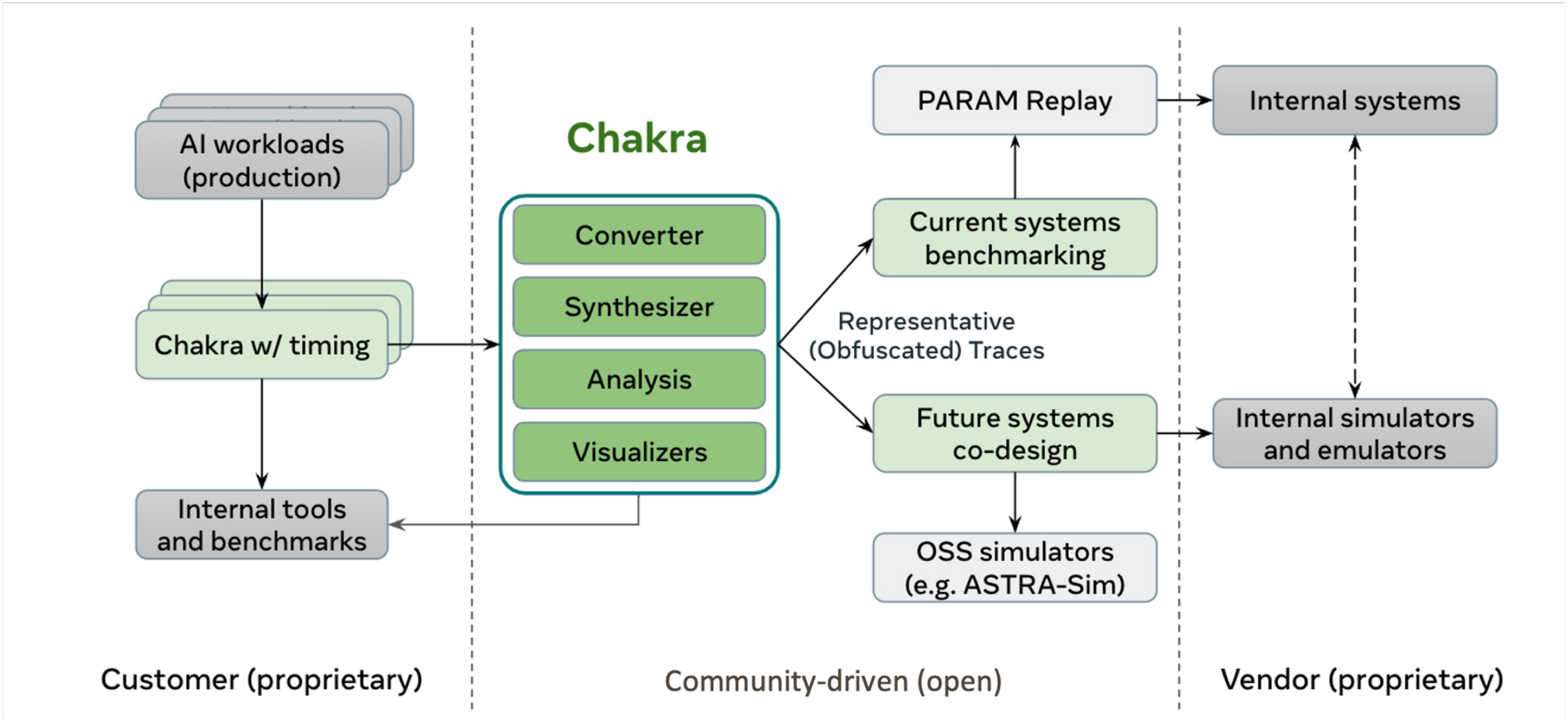
+

intel

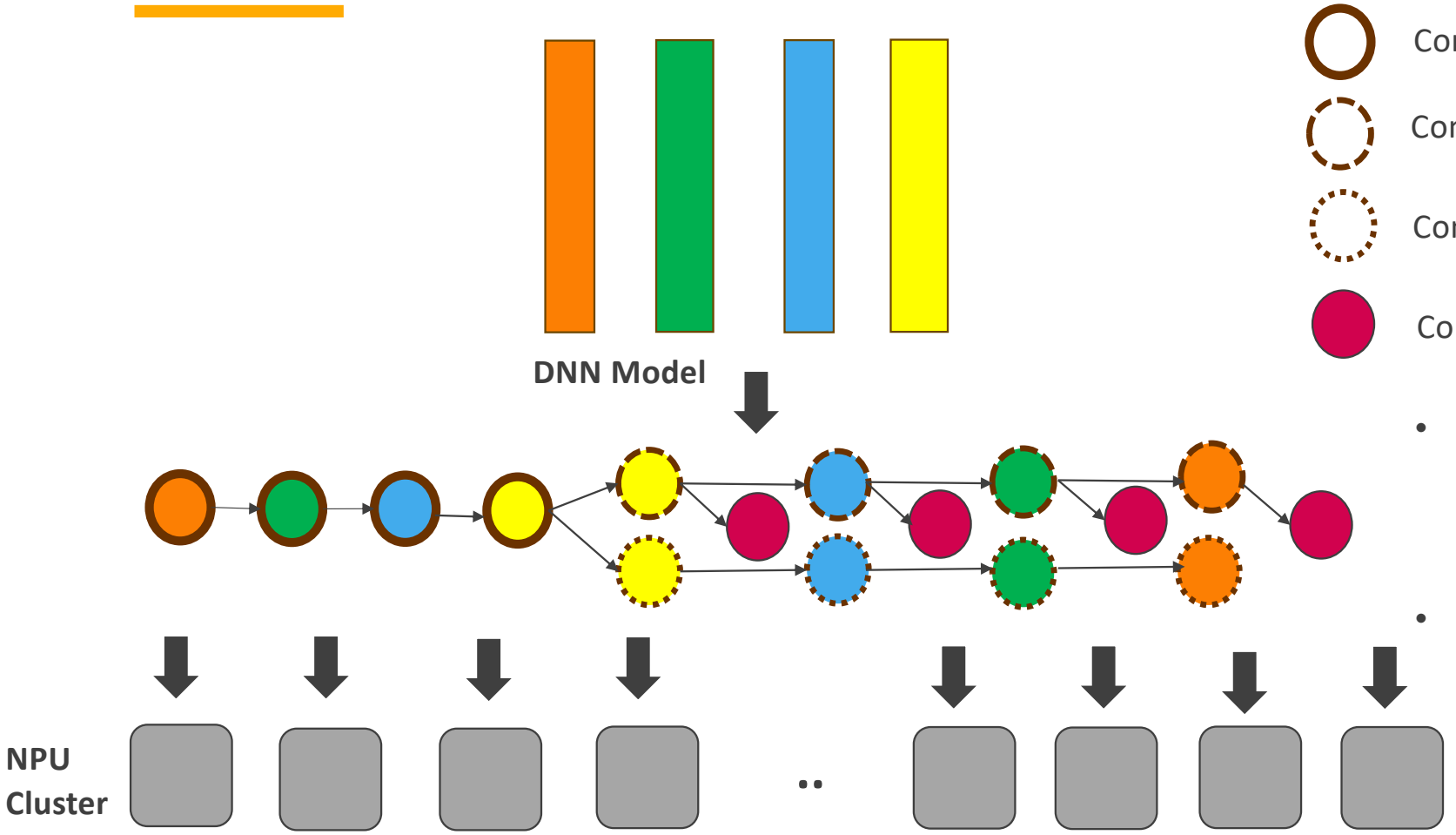
Hewlett Packard Enterprise

SCALA COMPUTING

Chakra Ecosystem and End-to-End Flow



Chakra Execution Trace



Comp: Fwd Pass

Comp: Weight Grad

Comp: Input Grad

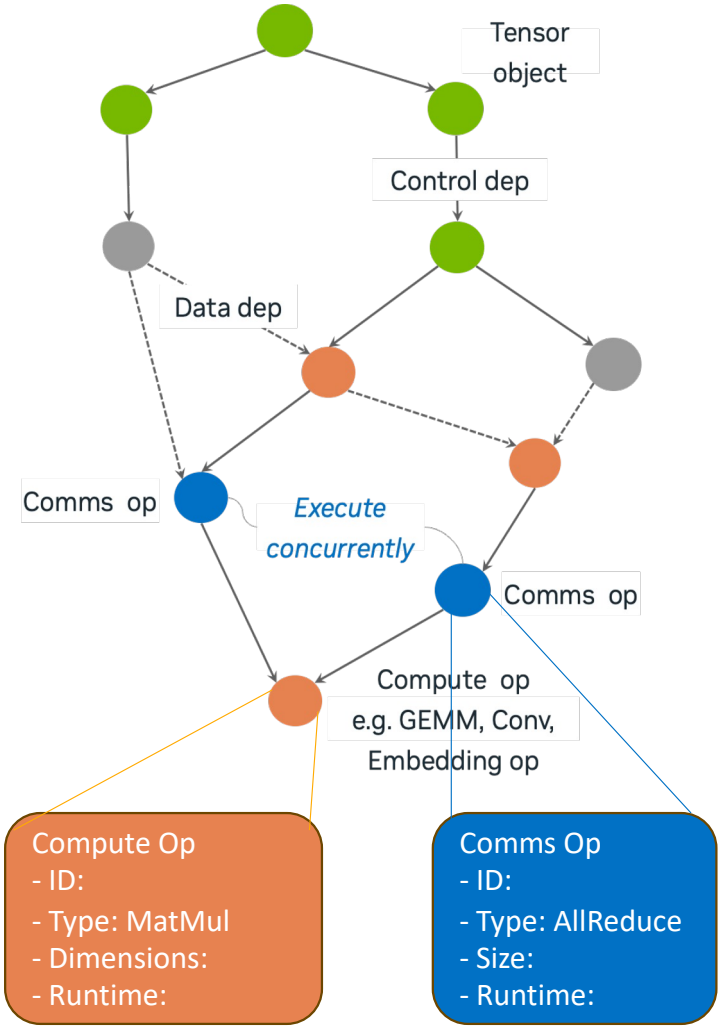
Comm: All Reduce

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

Chakra Execution Trace

- **Extensible and standardized graph format to represent AI workloads**
  - **Nodes:** primitive operators and tensor objects with attributes and timing
  - **Edges:** data and control dependency

- **Benefits**
  - Isolate comms and compute operators
  - Operator, dependencies, and timing for replay, simulation, and analysis
  - Flexible to represent both workloads and collective implementations
  - Graph transformations to obscure sensitive IP



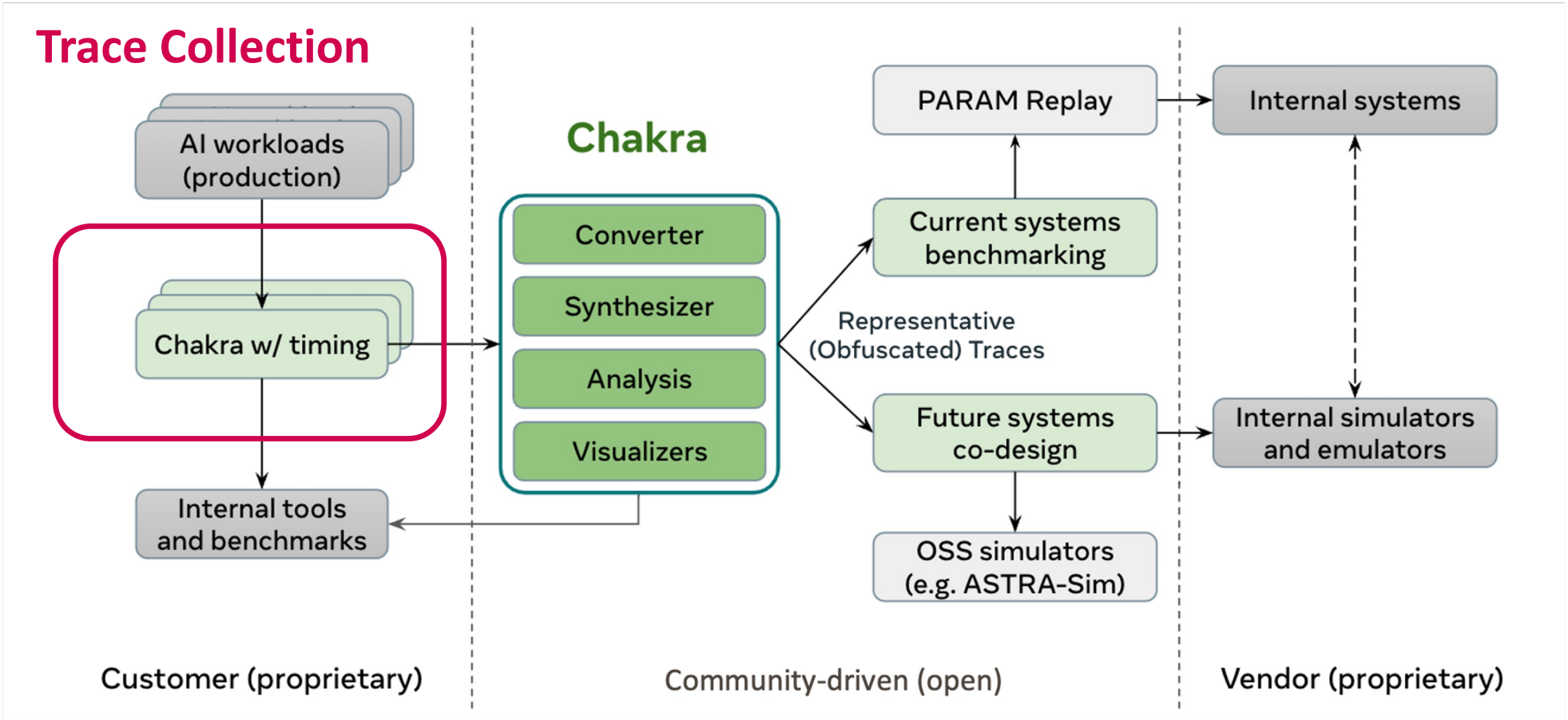
Chakra Execution Trace Schema

chakra / schema / protobuf / **et_def.proto**

```
message Node {
  uint64 id = 1;
  string name = 2;
  NodeType type = 3;

  // Control and data dependencies
  repeated uint64 ctrl_deps = 4;
  repeated uint64 data_deps = 5;

  // Timing information
  uint64 start_time_micros = 6;
  uint64 duration_micros = 7;

  IOInfo inputs = 8;
  IOInfo outputs = 9;
  repeated AttributeProto attr = 10;
}
```

```
enum NodeType {
  INVALID_NODE = 0;
  METADATA_NODE = 1;
  MEM_LOAD_NODE = 2;
  MEM_STORE_NODE = 3;
  COMP_NODE = 4;
  COMM_SEND_NODE = 5;
  COMM_RECV_NODE = 6;
  COMM_COLL_NODE = 7;
}

enum CollectiveCommType {
  ALL_REDUCE = 0;
  REDUCE = 1;
  ALL_GATHER = 2;
  GATHER = 3;
  SCATTER = 4;
  BROADCAST = 5;
  ALL_TO_ALL = 6;
  REDUCE_SCATTER = 7;
  REDUCE_SCATTER_BLOCK = 8;
  BARRIER = 9;
}
```

Components of Chakra Ecosystem



Chakra Trace Collection Flow

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

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

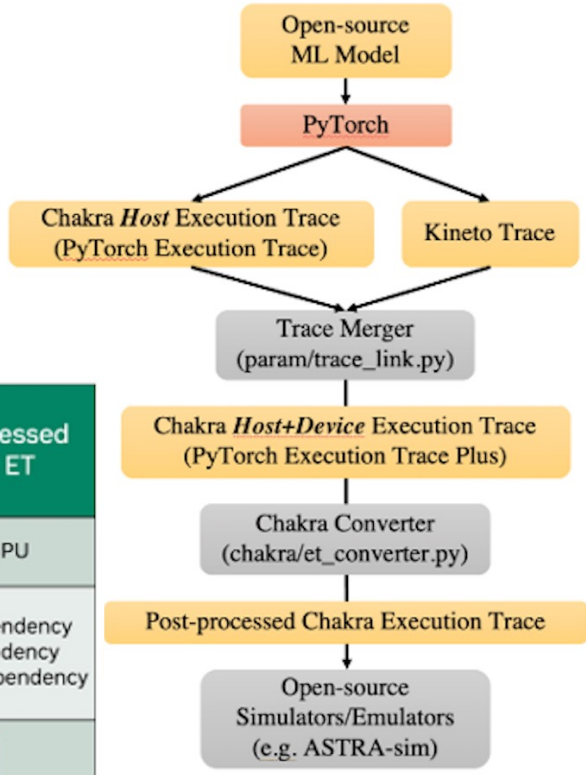
Code modifications

```
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_
) 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}")
```

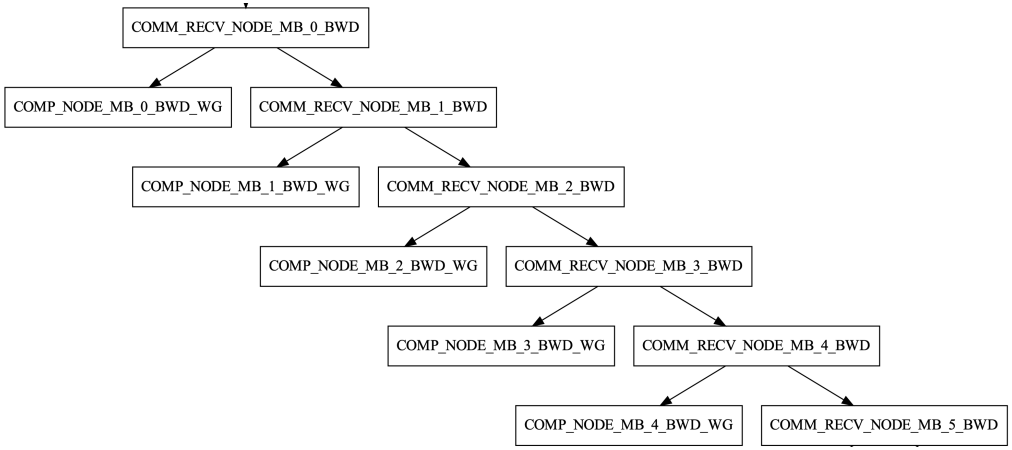
	PyTorch Chakra Host Execution Trace	Kineto Trace	Chakra Host+Device ET	Post-processed Chakra ET
Encoded Operators	CPU	CPU & GPU	CPU & GPU	CPU & GPU
Encoded Dependencies	Control dependency Data dependency	No explicit dependencies	Control dependency Data dependency	Control dependency Data dependency Simulation dependency
Input/Output Values, Shapes, Types	✓	✗	✓	✓
Duration	✓	✓	✓	✓
GPU Kernel	✗	✓	✓	✓

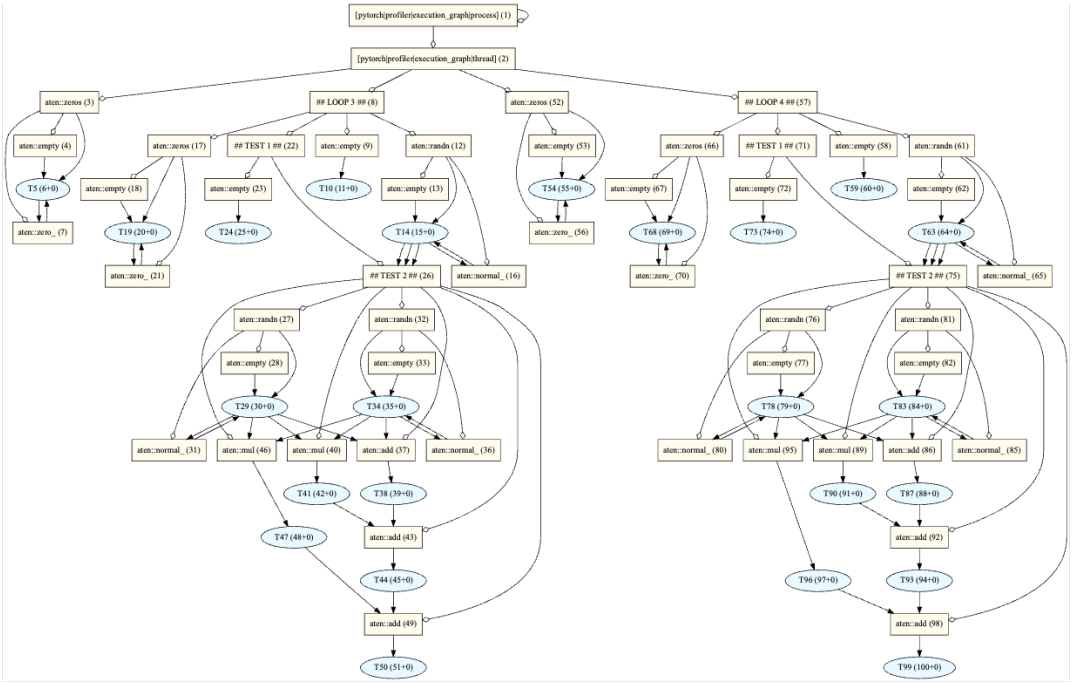
Trace Merger

- **Link Pytorch Ops with Kineto ops**
  - **Encode durations**
  - **Encode additional metadata**
- **Add GPU operators**

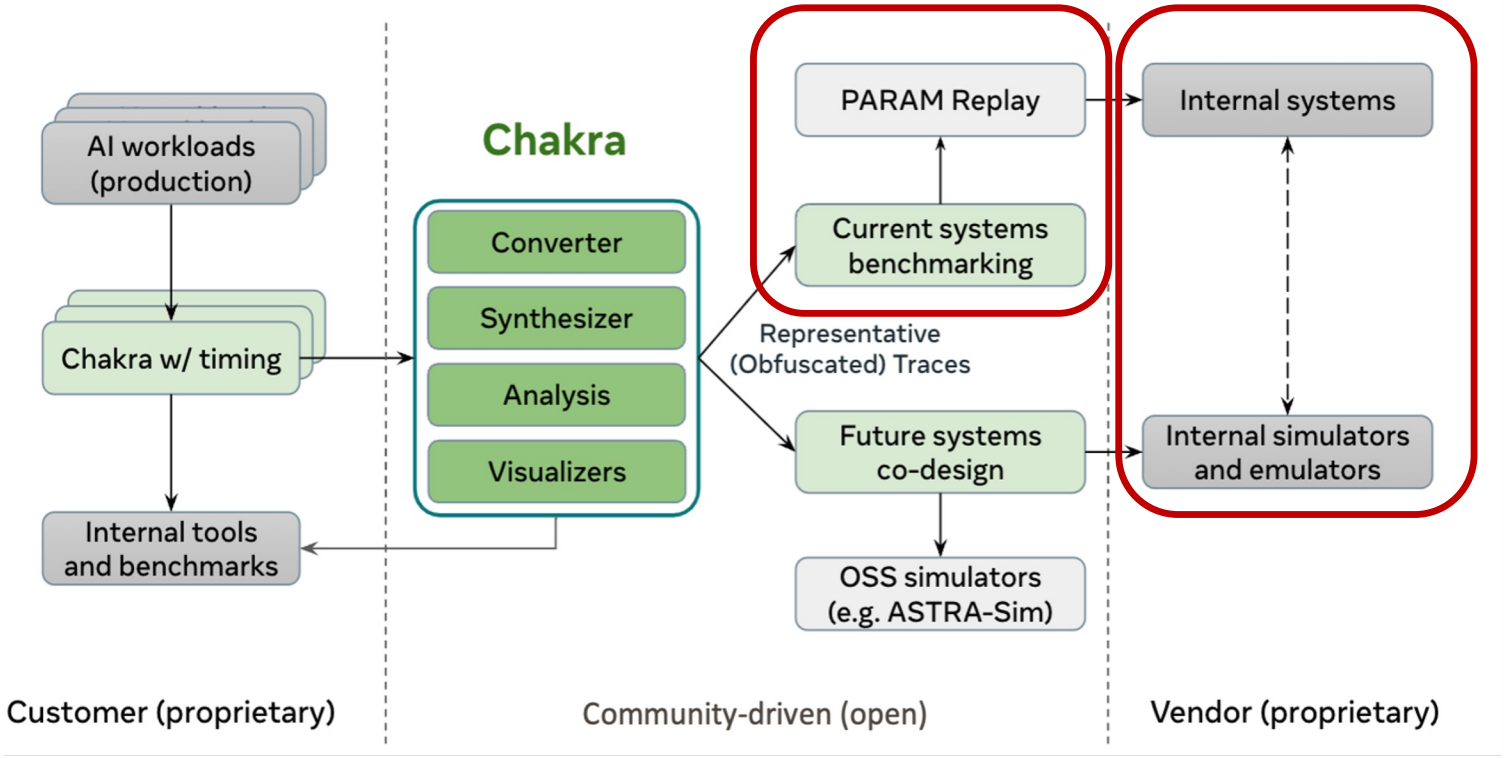


Example Chakra Traces





Components of Chakra Ecosystem

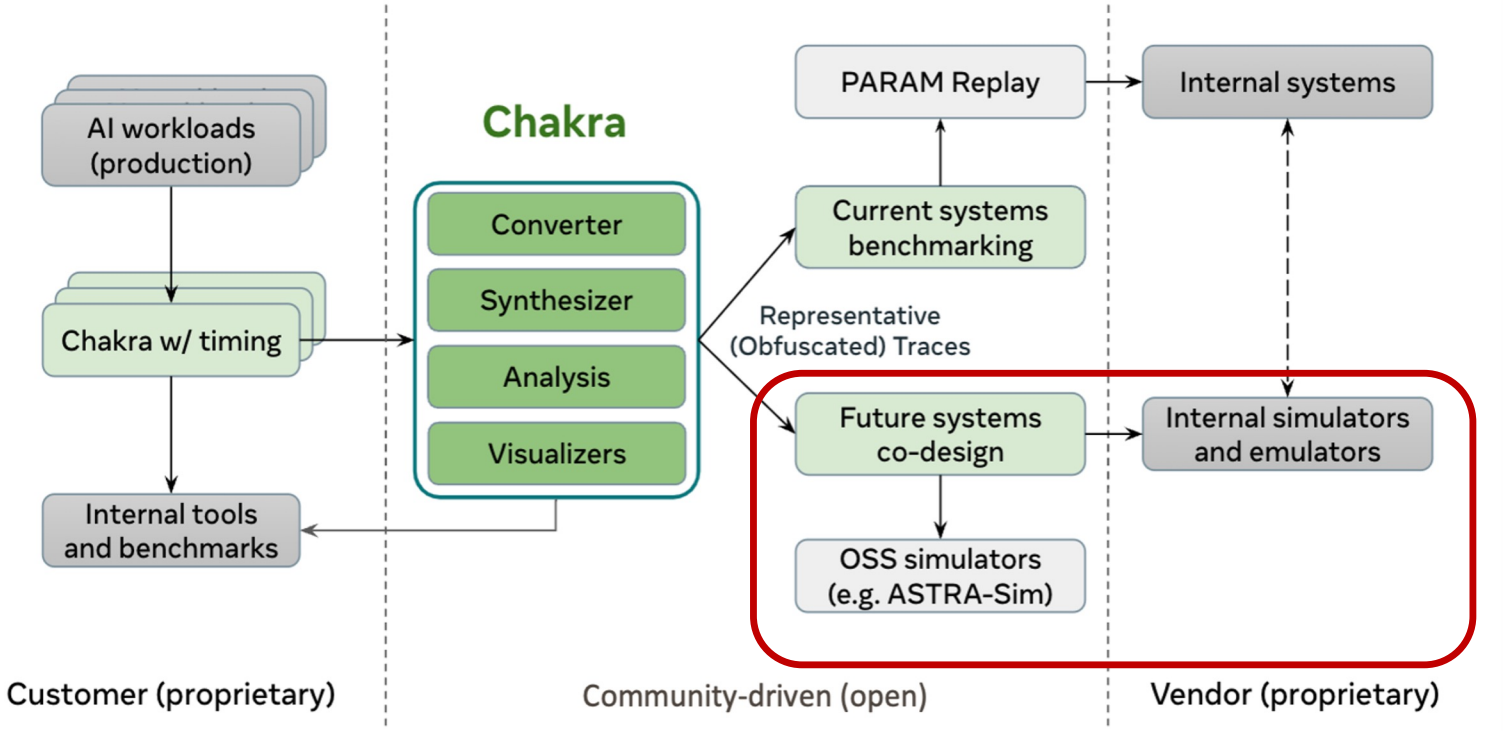


Replay on same / similar system

- Enable production teams to quickly reproduce a bug, perf regression, or benchmark on real systems
  - E.g., Mystique (ISCA 2023)
- Enable commercially supported emulators in pre-production labs

Details:
https://github.com/facebookresearch/param
https://www.keysight.com/us/en/cmp/kai.html

Components of Chakra Ecosystem



Simulation / Emulation

- Execute *some* of the operators
- Project performance of workloads on future systems to enable co-design

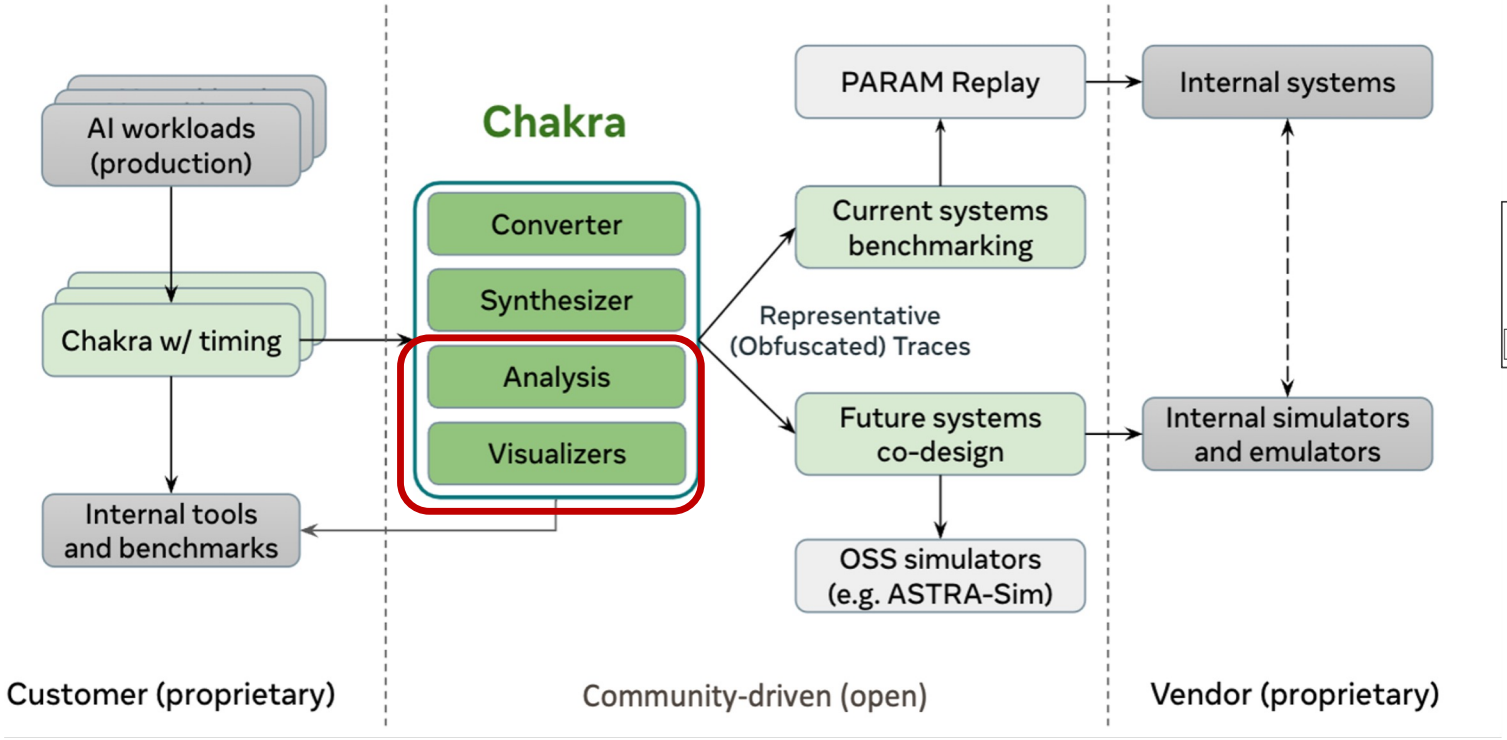
Chakra ET Feeder

```
class GraphFeeder {
public:
  GraphFeeder(std::string filename) {};
  virtual ~GraphFeeder() {};

  virtual void add_node(GraphNode* node) = 0;
  virtual void remove_node(NodeId node_id) = 0;
  virtual bool has_nodes_to_issue() = 0;
  virtual GraphNode* get_next_issuable_node() = 0;
  virtual void push_back_issuable_node(NodeId node_id) = 0;
  virtual GraphNode* lookup_node(NodeId node_id) = 0;
  virtual void free_children_nodes(NodeId node_id) = 0;
};
```

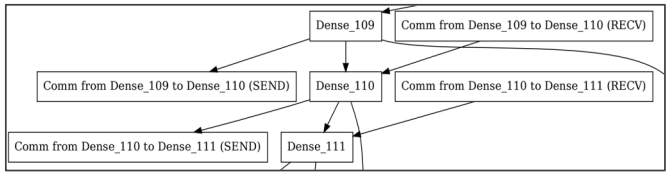
https://astra-sim.github.io/tutorials/micro-2024
https://www.keysight.com/us/en/cmp/kai.html

Components of Chakra Ecosystem

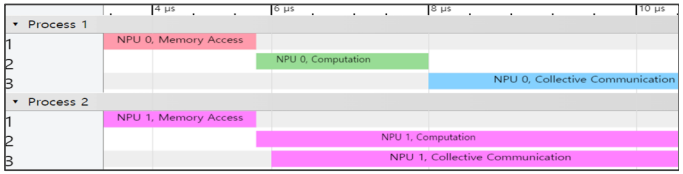
Supporting Tools



ET Visualizer



Timeline Visualizer



Summary

- **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

- **Resources**

  - **[Github]** https://github.com/mlcommons/chakra
  - **[Wiki]** https://github.com/mlcommons/chakra/wiki
  - **[Chakra Concept Paper]** https://arxiv.org/abs/2305.14516

Thanks!

- **Participation**

  - **[Working Group]** https://mlcommons.org/en/groups/research-chakratracebench/
  - **[Bi-Weekly Meeting]:** Monday from 11:05-12:00PM Pacific.