Chakra: Advancing Benchmarking and Codesign with Standardized Workload Traces

Taekyung Heo†*, Srinivas Sridharan†, Louis Feng, Zhaodong Wang, Matt Bergeron, Wenyin Fu, Shengbao Zheng, Brian Coutinho, Saeed Rashidi*, Changhai Man*, Tushar Krishna*

[1]Georgia Institute of Technology, [2] Hewlett Packard Enterprise, [3]Meta, [4]Intel

Paper

Background

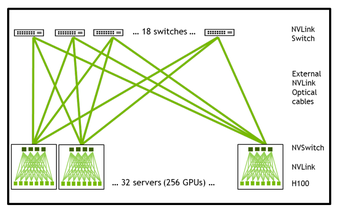
Increasing ML model sizes

Increasing dataset sizes

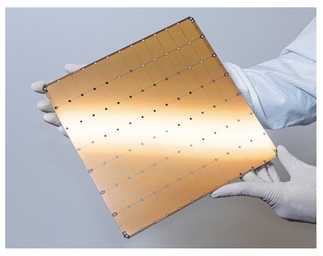
750GB texts

1.56 trillion words

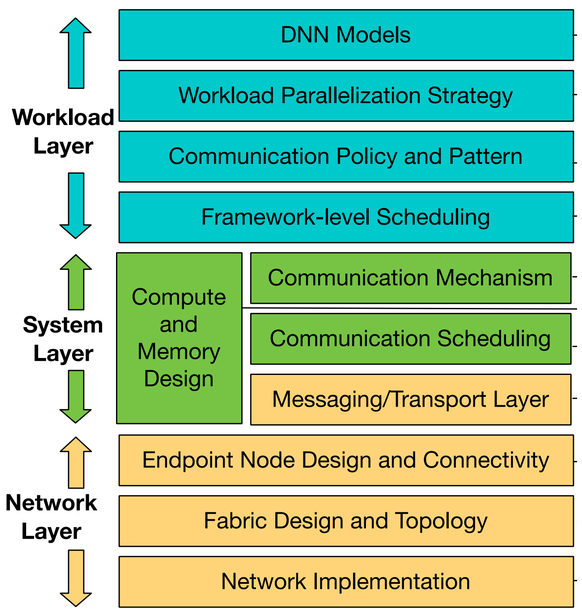
Distributed machine learning systems are required!



NVIDIA HGX H-100



Cerebras WSE-2



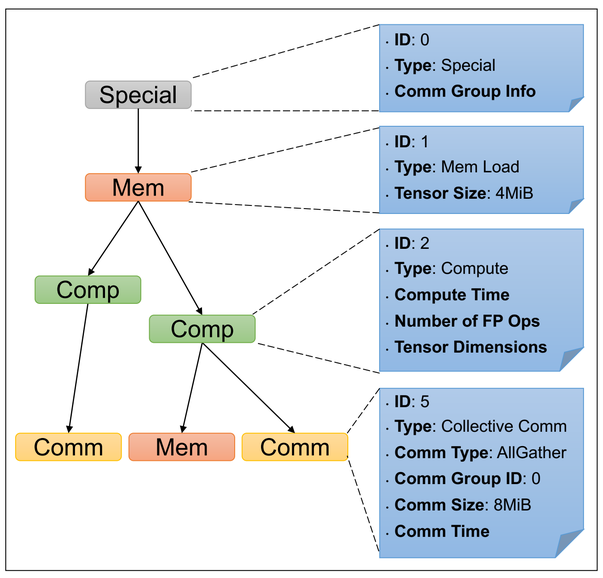
Motivation

- Lack of toolchains for performance modeling with real production execution traces
- Absence of a unified schema for exchanging execution traces
- Companies cannot share execution traces because of IP issues

Chakra

- Open-source tools for performance modeling and analyzing execution traces
- Common graph schema for exchanging execution traces
- Obfuscate proprietary AI model details

Execution Trace Schema



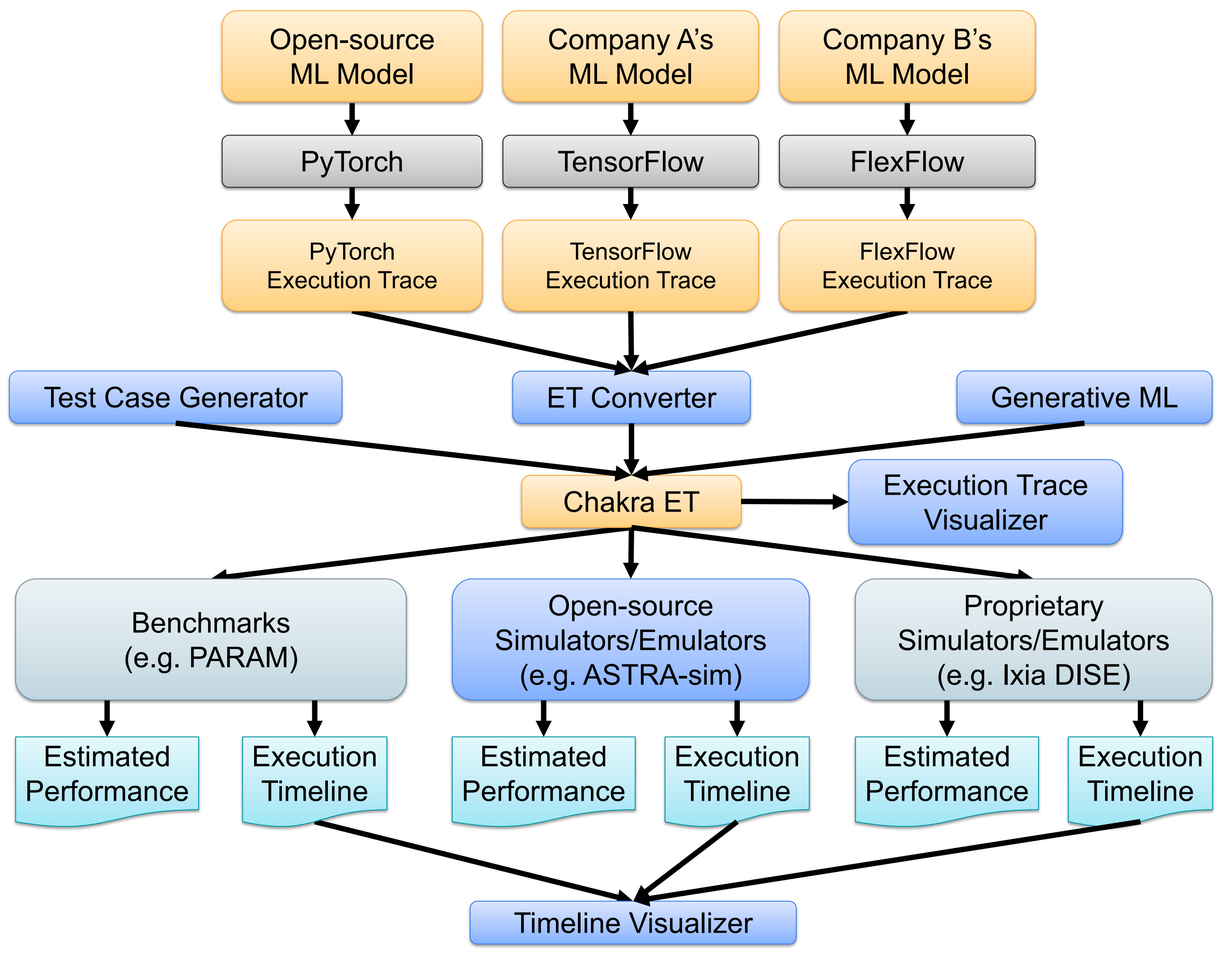
Field Name	Data Type	Description
id	required uint64	Node ID
name	required string	Node name
type	required enum	Node type
parent	repeated uint64	Parent node IDs
attribute	repeated AttributeProto	Any additional fields

Table 1: Chakra node schema.

Field Name	Data Type	Description
name	required string	Name of this attribute
type	required enum	Type of this attribute
doc_string	required string	Description of this attribute
f	optional float	Float value
i	optional int64	Integer value
s	optional string	String value
floats	repeated float	Repeated float values
ints	repeated int64	Repeated integer values
strings	repeated string	Repeated string values

Table 2: Chakra AttributeProto schema.

Chakra Ecosystem



Test Case Gen

```python
def one_comp_node(num_npus: int, runtime: int) -> None
    for npu_id in range(num_npus):
        output_filename = f"one_comp_node.{npu_id}.et"
        with open(output_filename, "wb") as et:
            node = get_node("COMP_NODE", COMP_NODE)
            attr = get_runtime_attr(runtime)
            node.attribute.append(attr)
            encode_message(et, node)
```

ML Commons

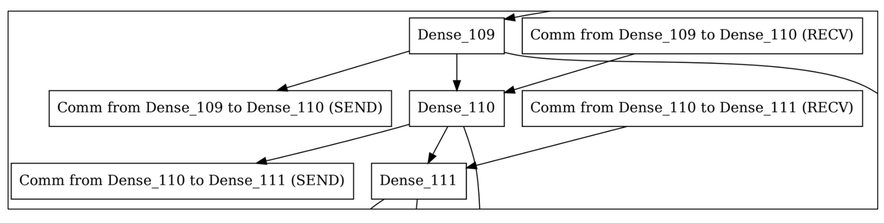
- Chakra has a working group in MLCommons.
- Please feel free to join!

Trace Collection

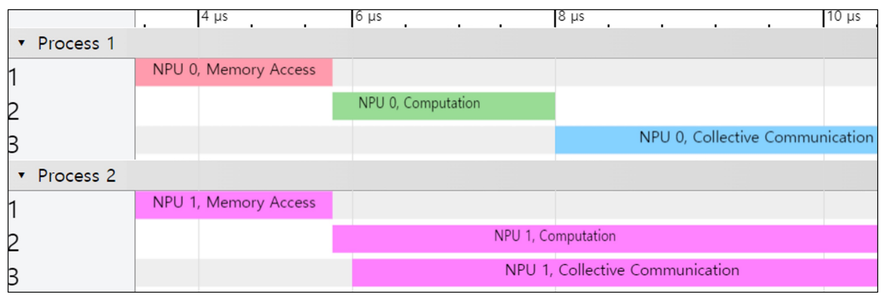
```python
eg = None
if args.eg:
    eg_file = f"{out_file_prefix}_eg.json"
    eg = ExecutionGraphObserver()
    eg.register_callback(eg_file)
    eg.start()
```

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

Execution Trace Visualizer



Execution Timeline Visualizer



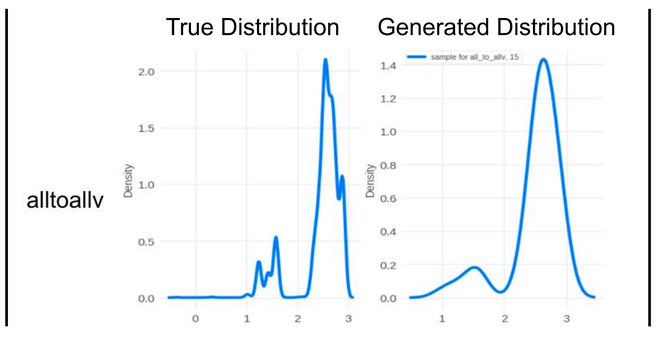
Execution Trace Feeder

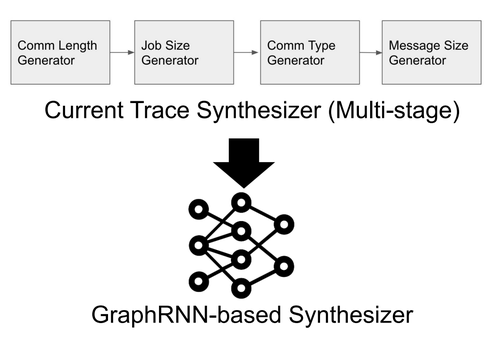
```cpp
class GraphFeeder {
public:
  GraphFeeder(std::string filename) {};
  virtual ~GraphFeeder() {};

  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;
};
```

Execution Trace Synthesizer

Goals

1. Obfuscate details of production workloads
2. Generate representative traces





Paper

GitHub