Georgia Tech School of Electrical and Computer Engineering
College of Engineering

http://synergy.ece.gatech.edu

ASTRA-SIM Description

Saeed Rashidi
Ph.D. Student, School of Electrical & Computer Engineering
Georgia Institute of Technology

saeed.rashidi@gatech.edu

Acknowledgments: Srinivas Sridharan (Meta), Sudarshan Srinivasan (Intel)

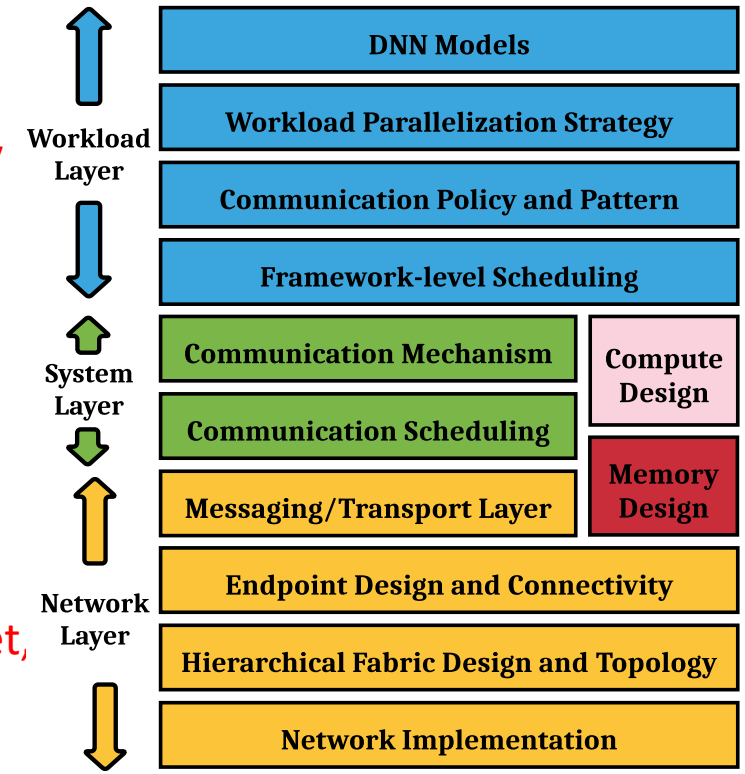
Overview

How to Model and Evaluate the Communication Effect

- It is a complex problem and can be viewed as three layers :
  - 1. Workload layer (the training loop):
    - Parallelism approach
    - Compute power
    - Communication size & type and dependency order
  - 2. System layer:
    - Collective communication algorithm
    - Chunk size, schedule of collectives
  - 3. Network layer:
    - Physical topology
    - Congestion control, communication protocol
    - Link BW, latency, buffers, routing algorithm

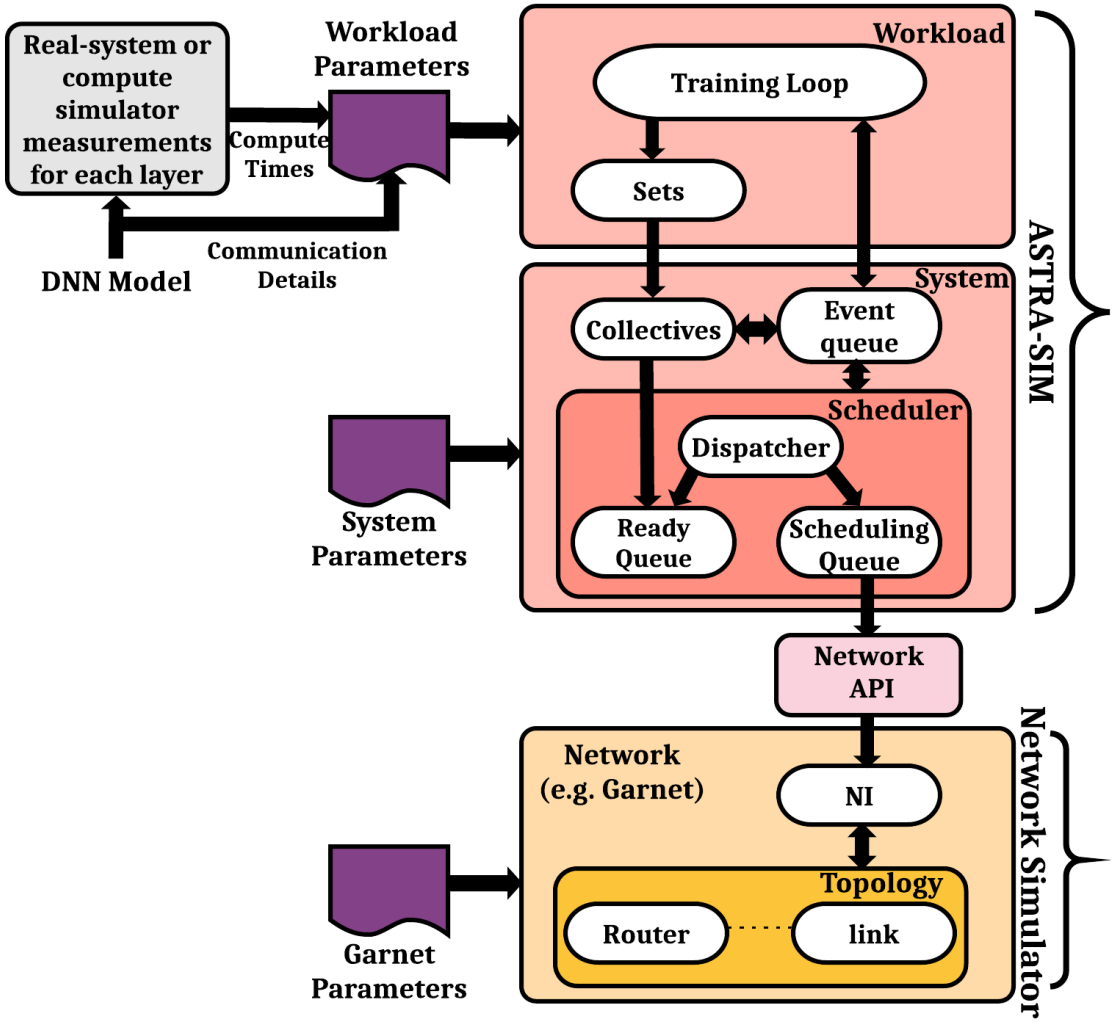
Not too many tools cover these aspects

Many tools In this area (e.g., Garnet, Analytical)



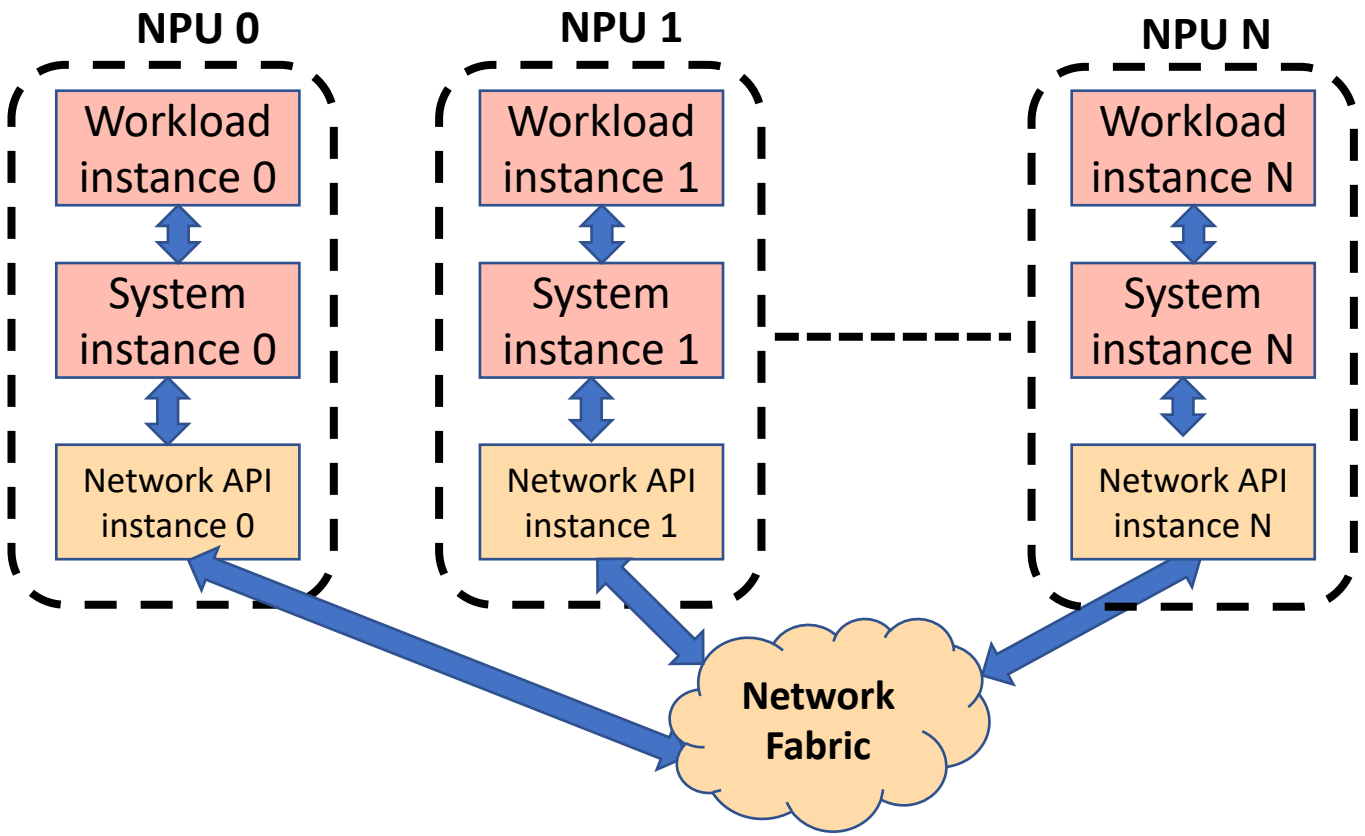
ASTRA-SIM Architecture

- Workload layer:
  - Supports Data-Parallel, Model-Parallel, Hybrid-Parallel training loops
  - Easy to add new arbitrary training loop
- System:
  - Ring based, Tree-based, AlltoAll based, and multi-phase collectives
  - Easy to add new collective communication
- Network:
  - Supports Analytical and GARNET Network simulator
  - Analytical:
    - Supports hierarchical topologies
    - Each level in hierarchy can be switch, ring, FC….
    - https://github.com/astra-sim/analytical/tree/develop
  - GARNET:
    - Supports switch-based and torus-based topologies
    - Supports credit-based flow control
    - https://github.com/georgia-tech-synergy-lab/gem5_astra/tree/reorgV2
  - Can add new topologies in both Analytical and GARNET



ASTRA-SIM Runtime Structure

- Each NPU is represented through separate instance of Workload, System, and Network API.
- Network API class is implemented by the network backend.



ASTRA-SIM Directory

rashidi1saeed Merge pull request #44 from srinivas212/master ...	f7e54a8 15 hours ago	**335** commits
.github/workflows	Added GitHub Actions (#30)	8 months ago
astra-sim	CSV Writer Updated	19 hours ago
build	gem5 is now compatible with new changes	18 hours ago
docs/images	updated	9 months ago
examples	run_multi example script bug fixed	18 hours ago
extern	Merge pull request #43 from astra-sim/saeed_astra_dev	15 hours ago
inputs	Update README.md	17 hours ago
scripts/workload_generator	-:TESTED:-	last month
test	Fix formatting using clang-format	9 months ago
.clang-format	Use PyTorch .clang-format	9 months ago
.clang-tidy	Added GitHub Actions (#30)	8 months ago
.gitignore	Added GitHub Actions (#30)	8 months ago
.gitmodules	Add scale sim v2 submodule	last month
CMakeLists.txt	CMAKELists updated	yesterday
CODEOWNERS	Modify CODEOWNERS	16 hours ago
LICENSE	Update LICENSE	9 months ago
README.md	Update README.md	9 months ago

Workload Layer

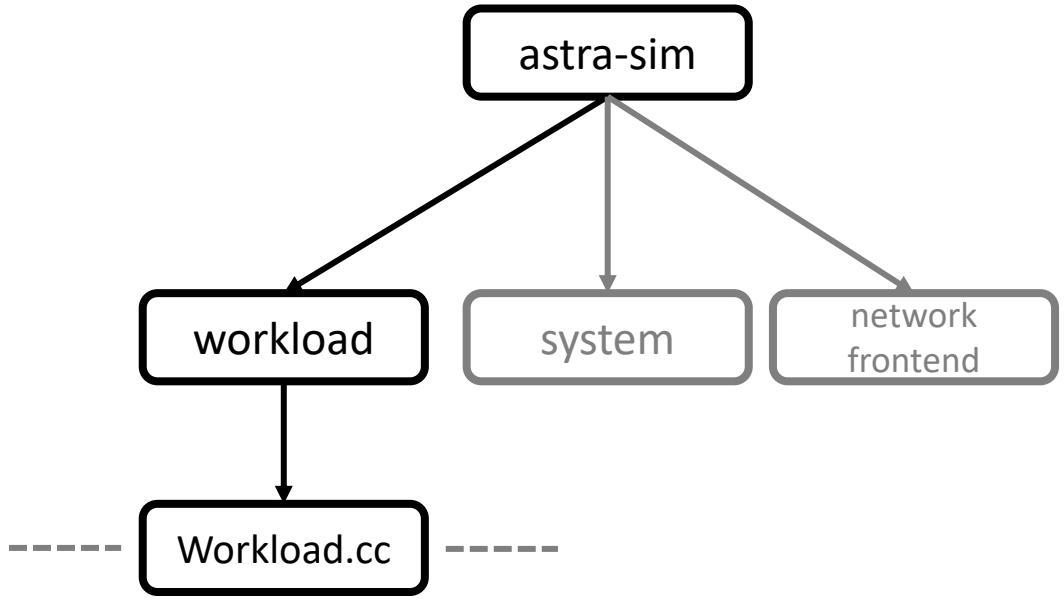
Workload Layer

- Currently, ASTRA-SIM supports 2 types of front-ends:
  - Text-based workload parser
  - New: Execution graph (EG) based workload parser
    - Please refer to the codebase for more info

We cover text-based front-end in this tutorial

Workload Layer Training Loop

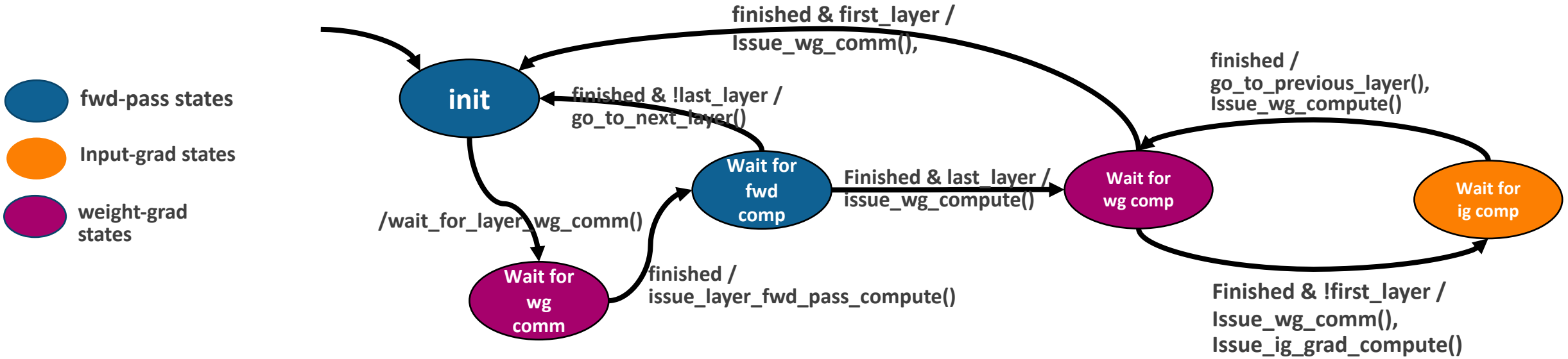
- Code Structure



Implements different training loops

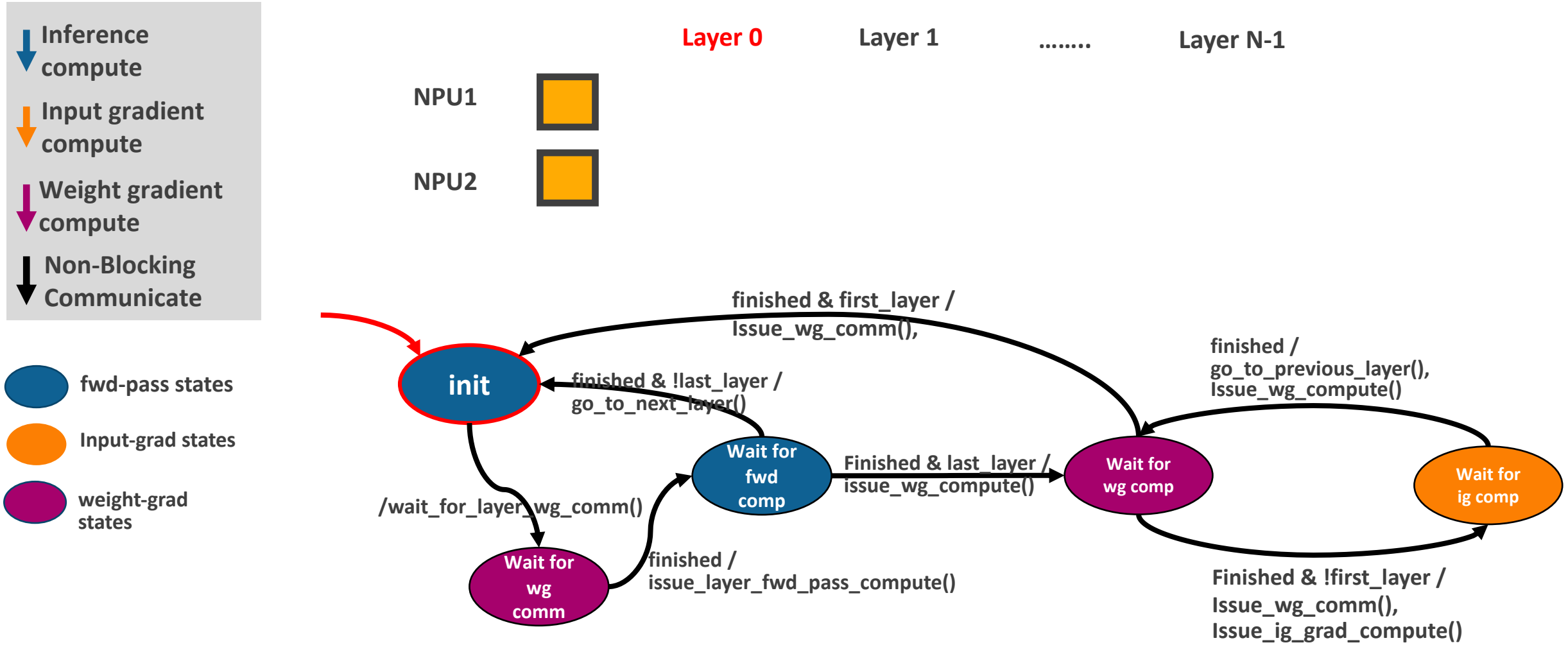
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



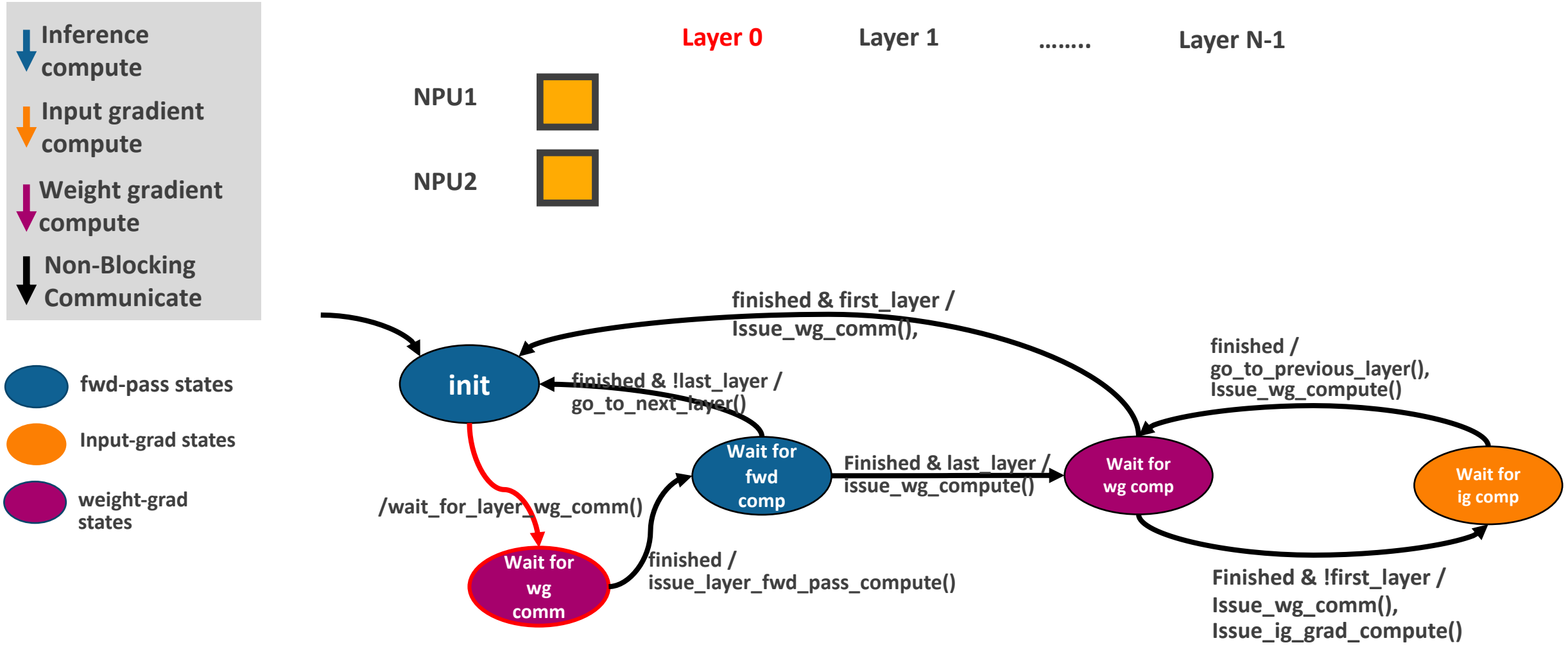
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



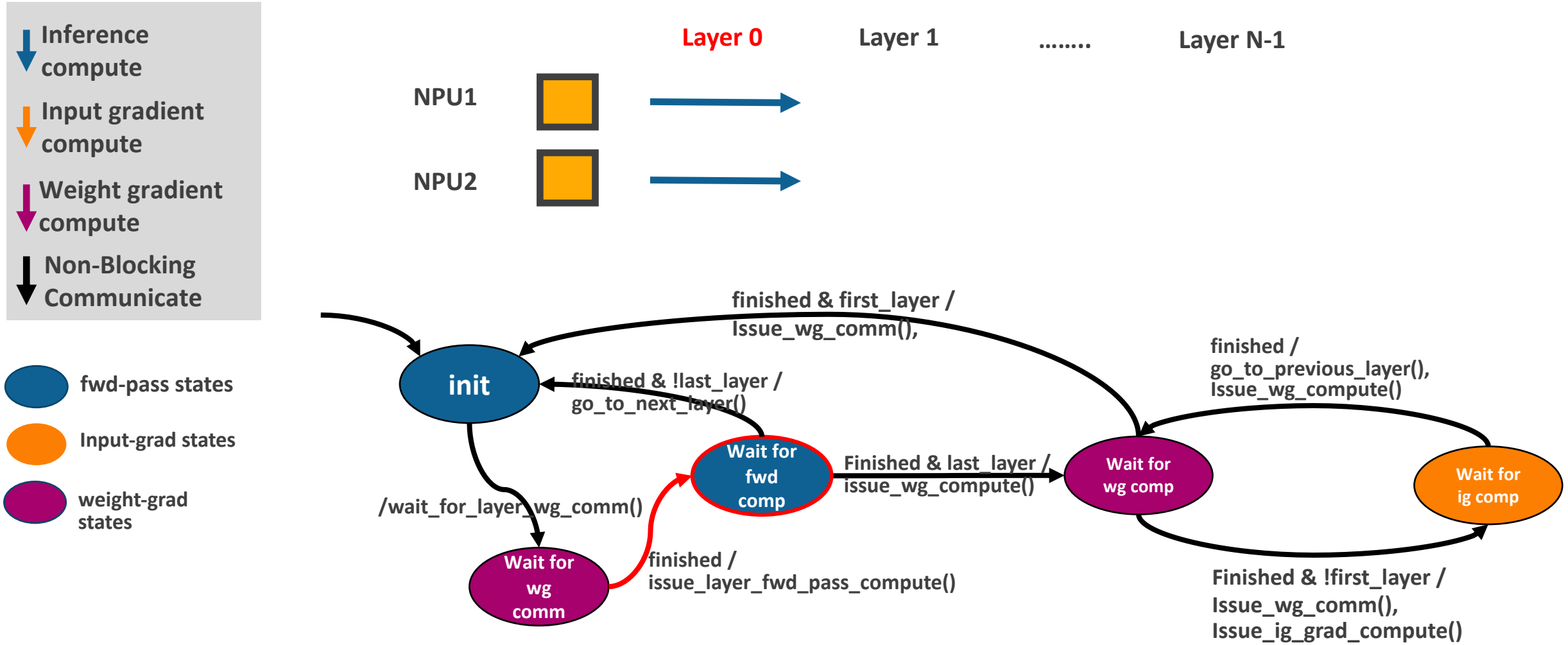
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



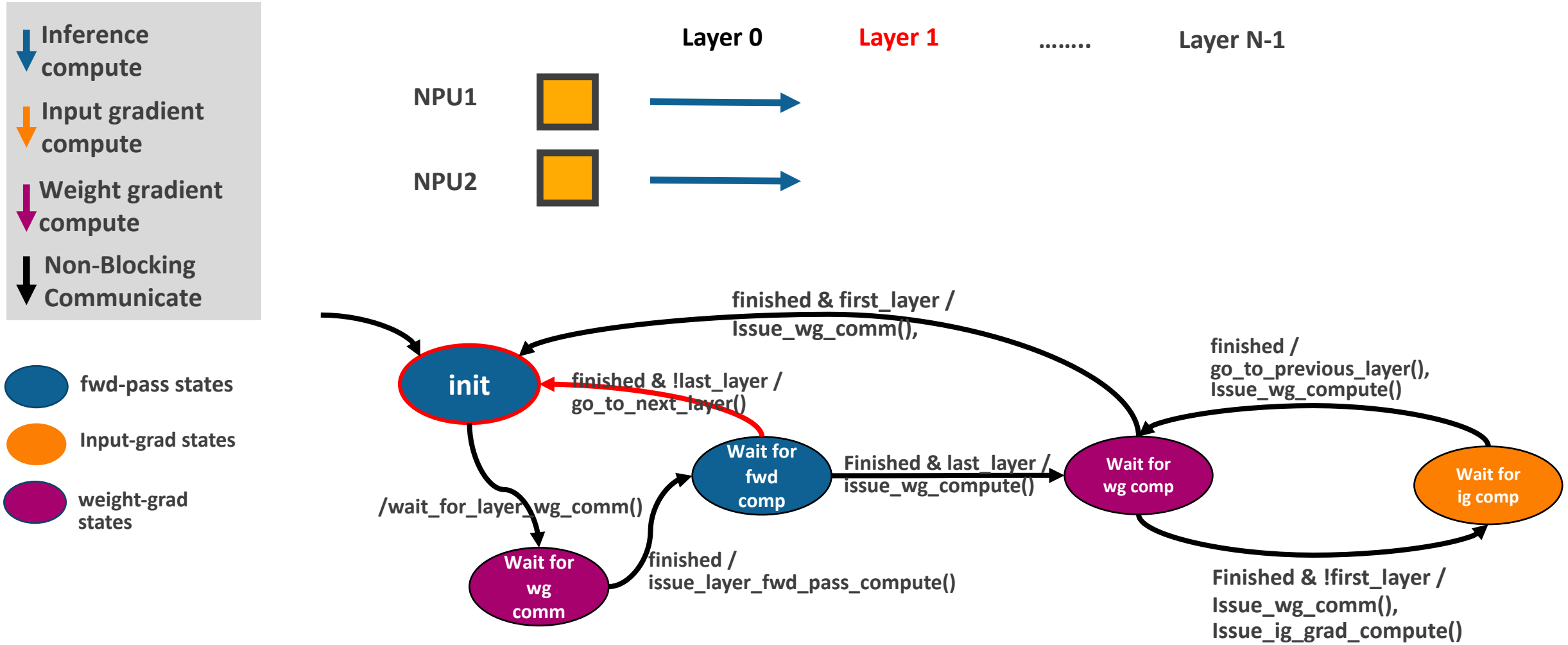
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



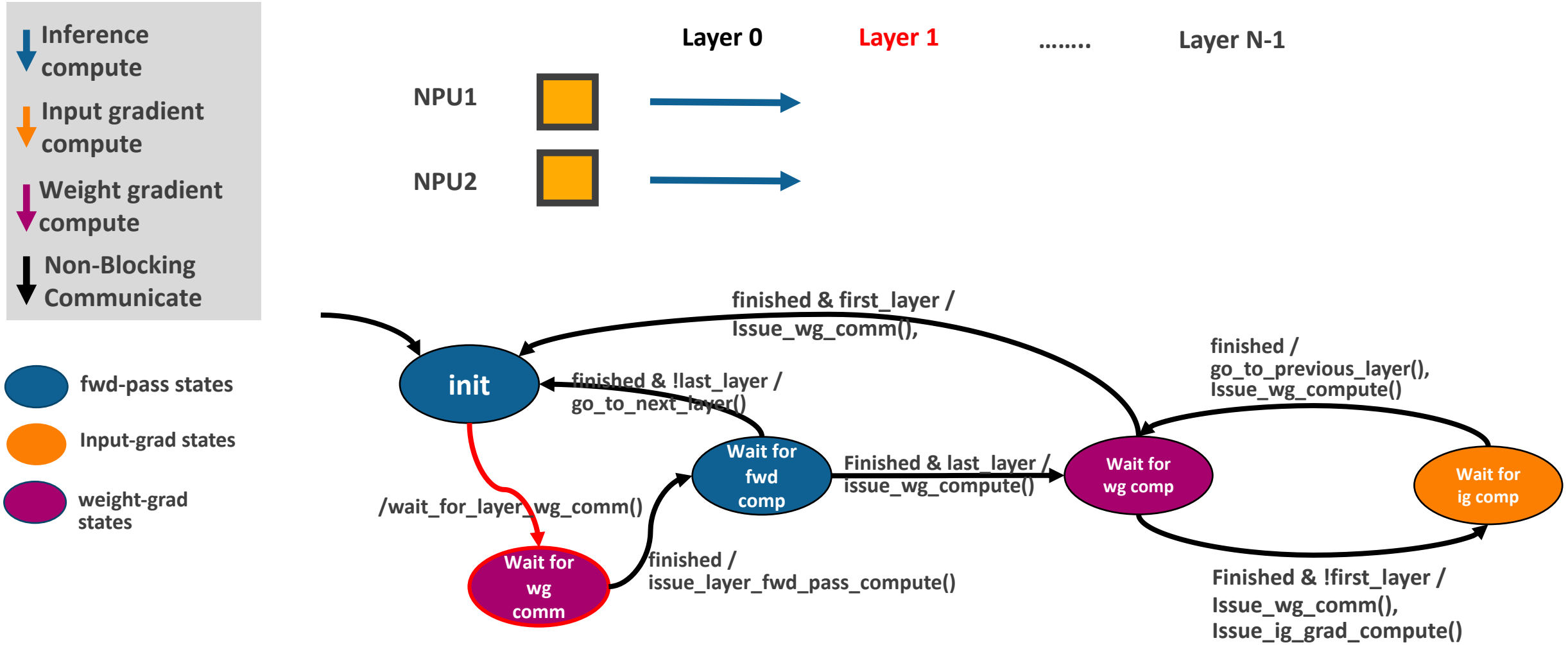
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



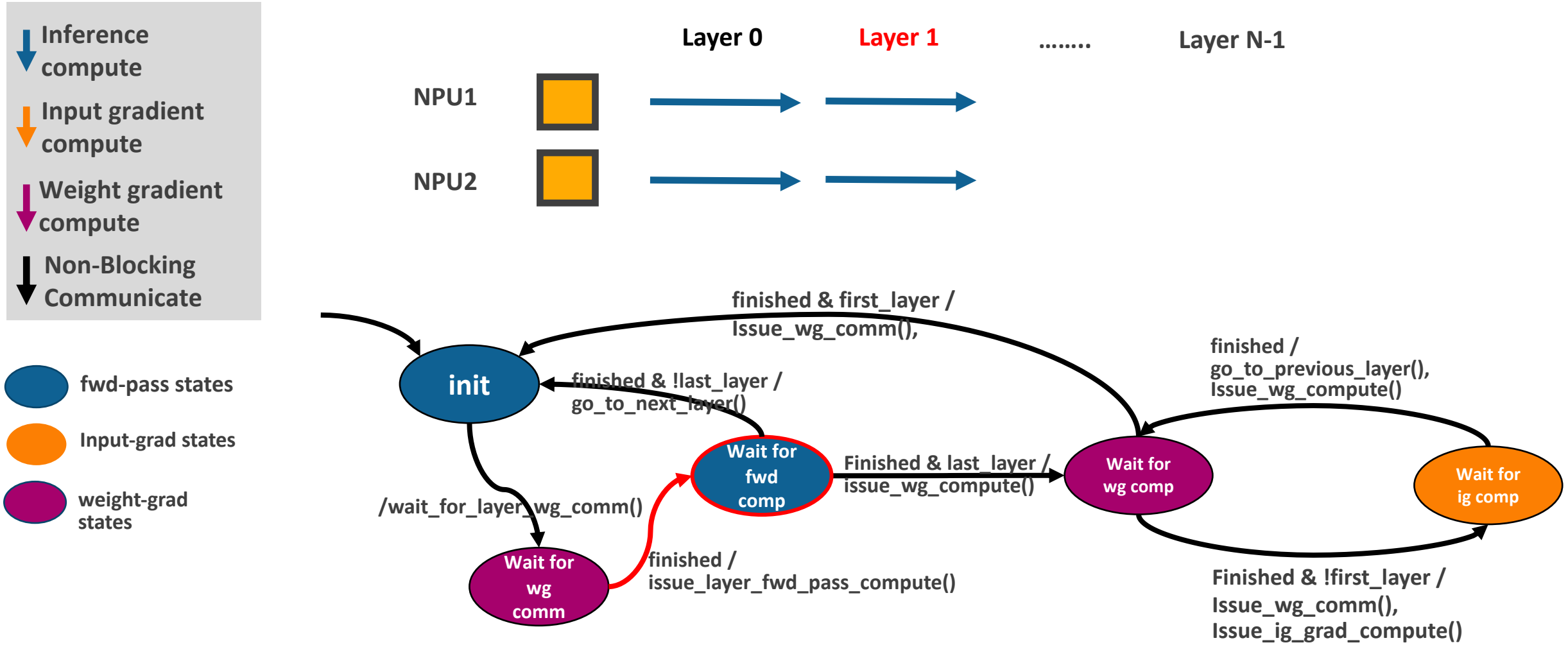
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



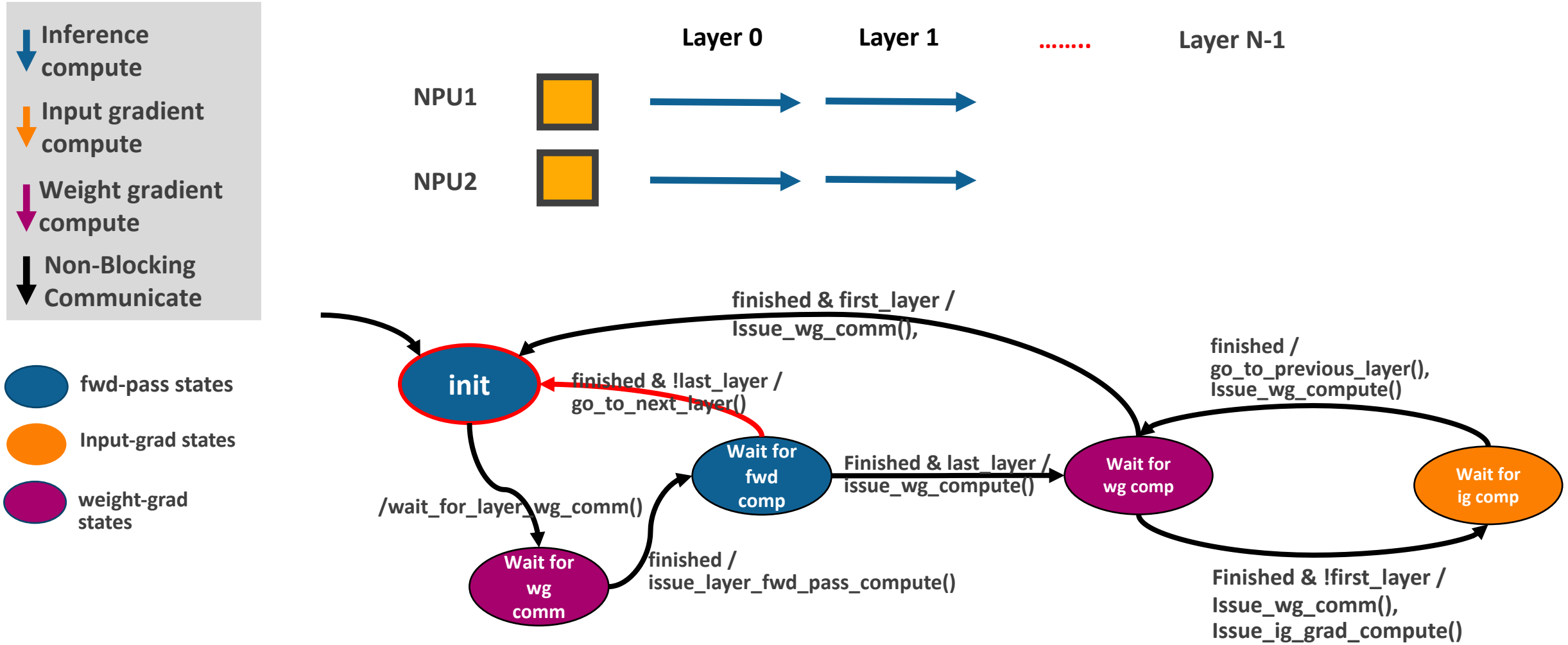
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



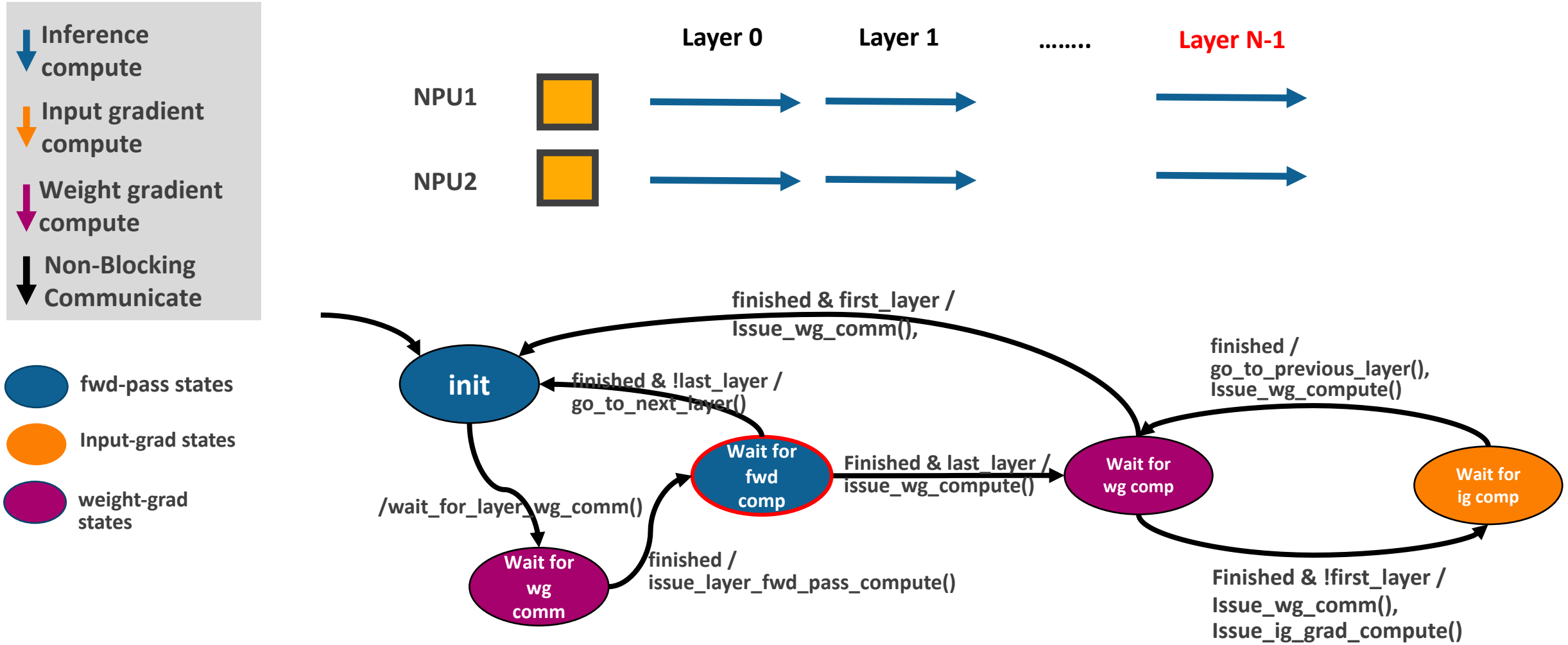
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



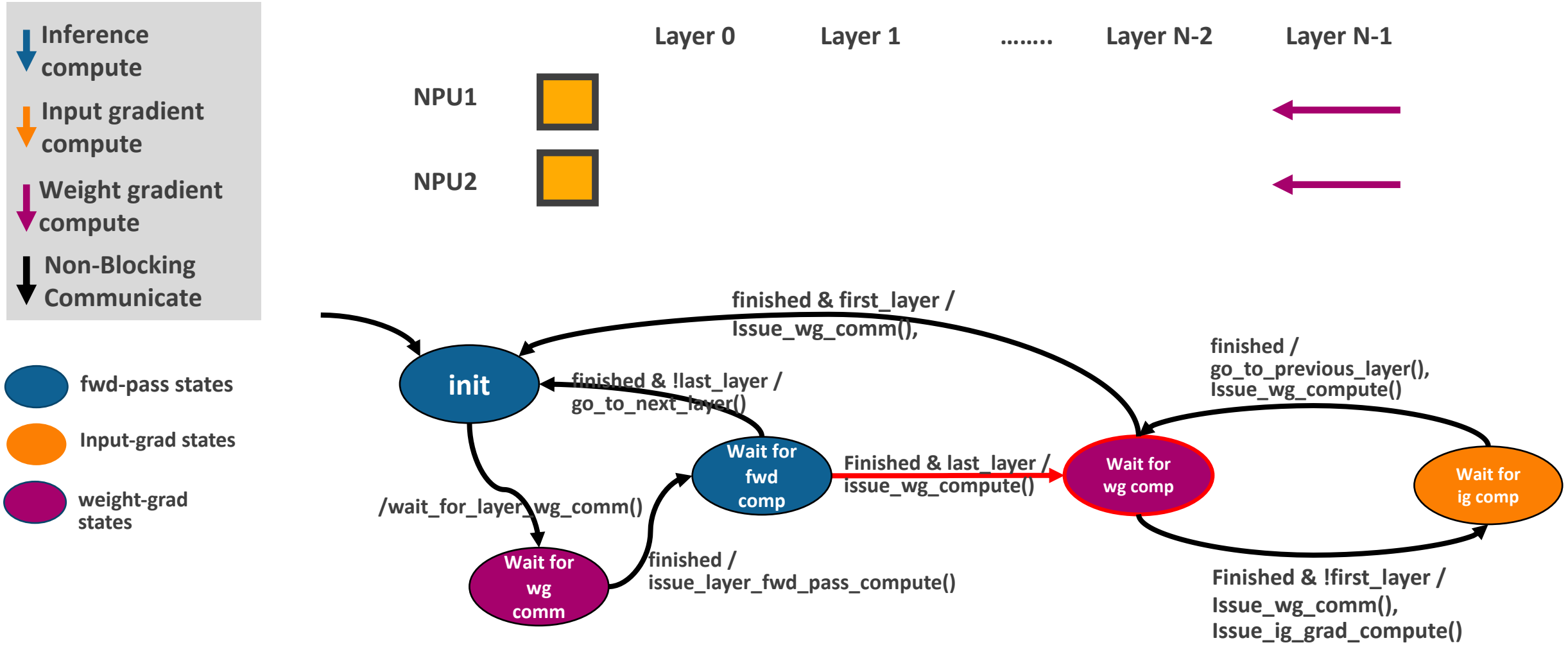
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



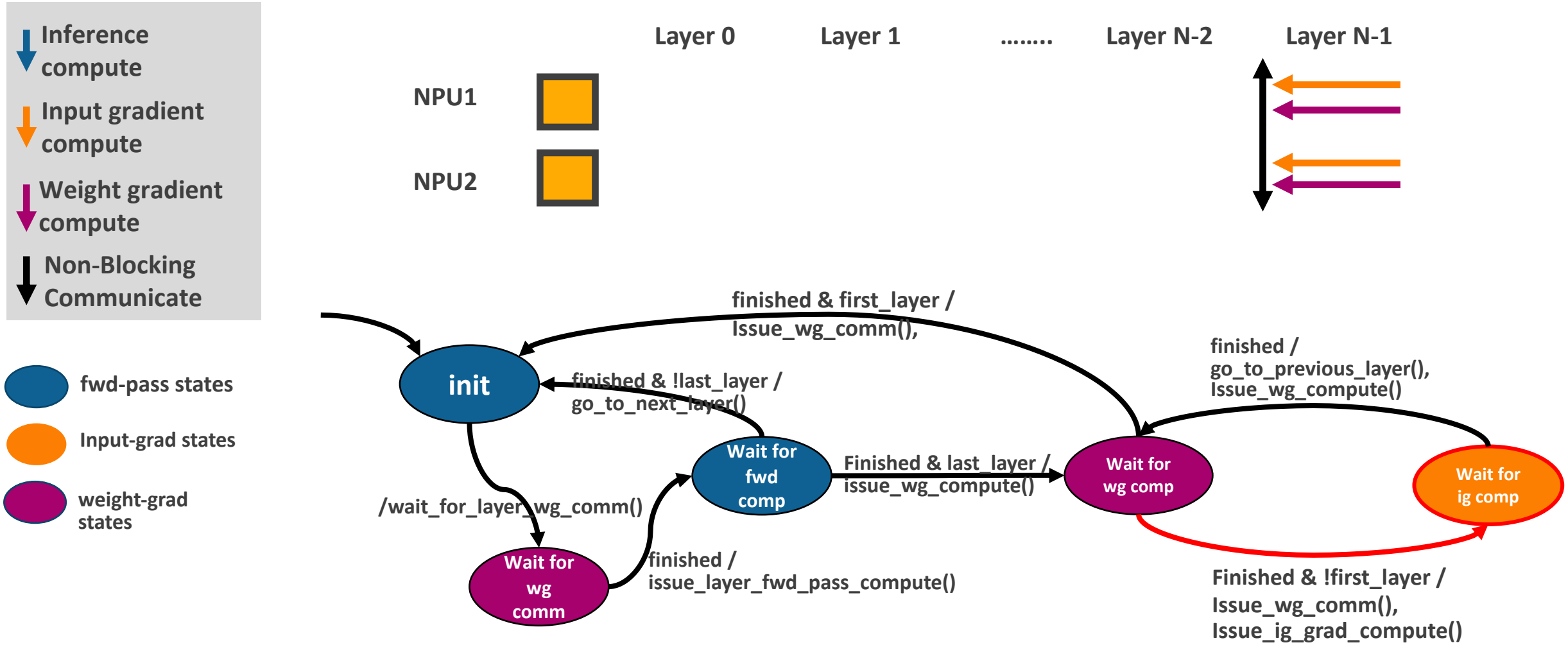
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



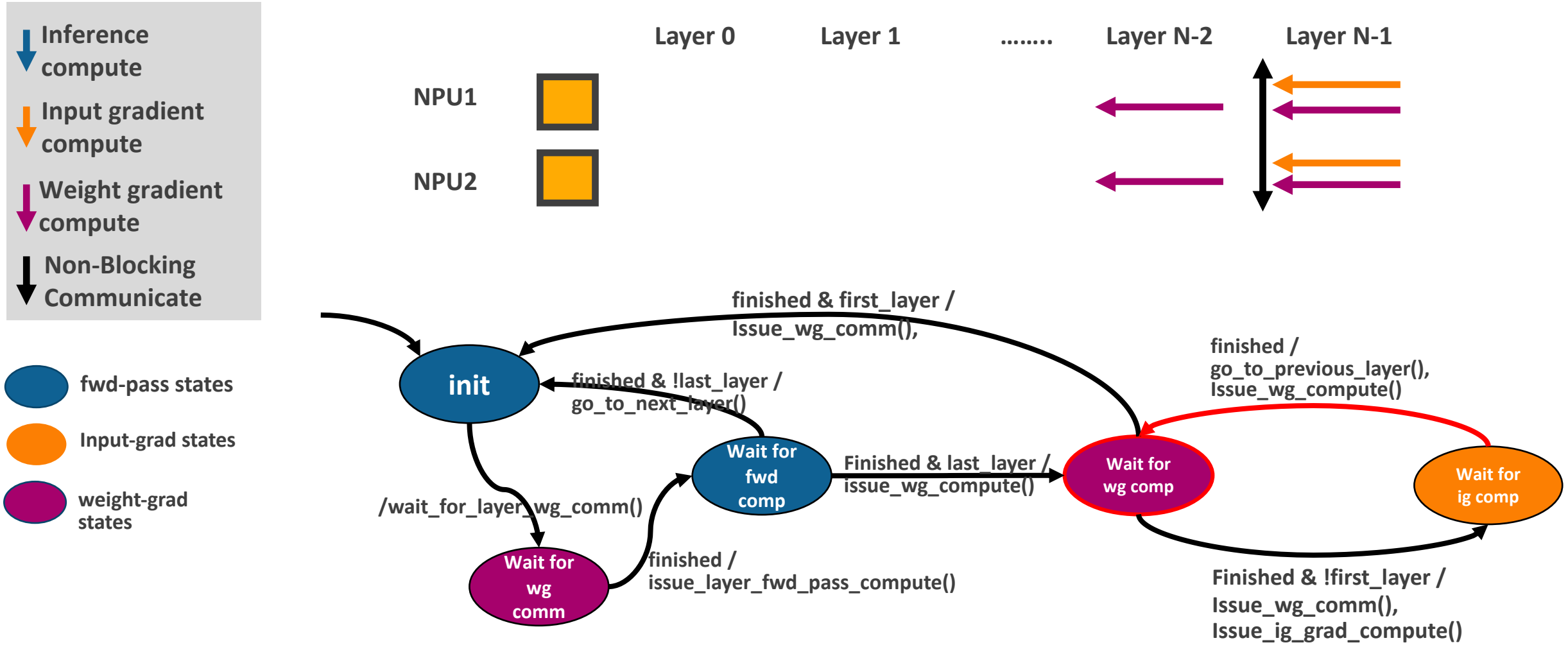
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



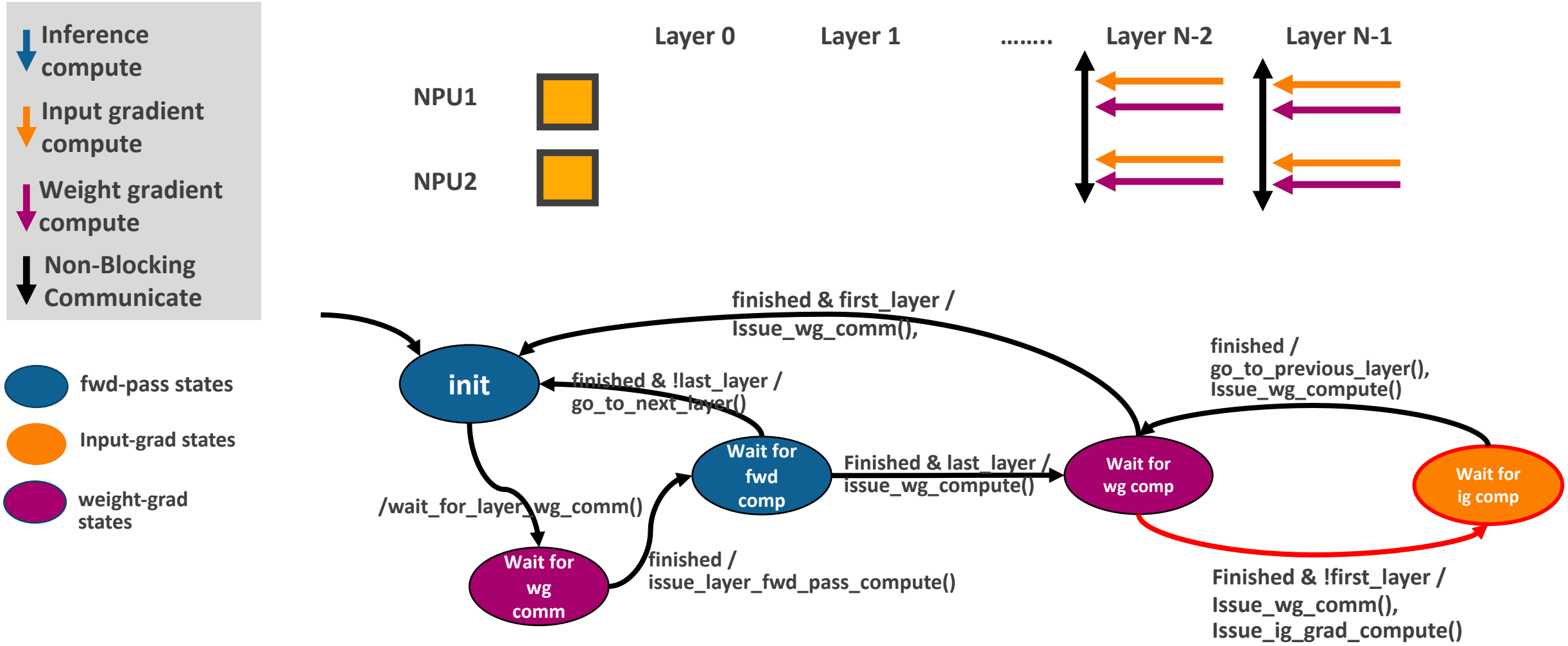
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



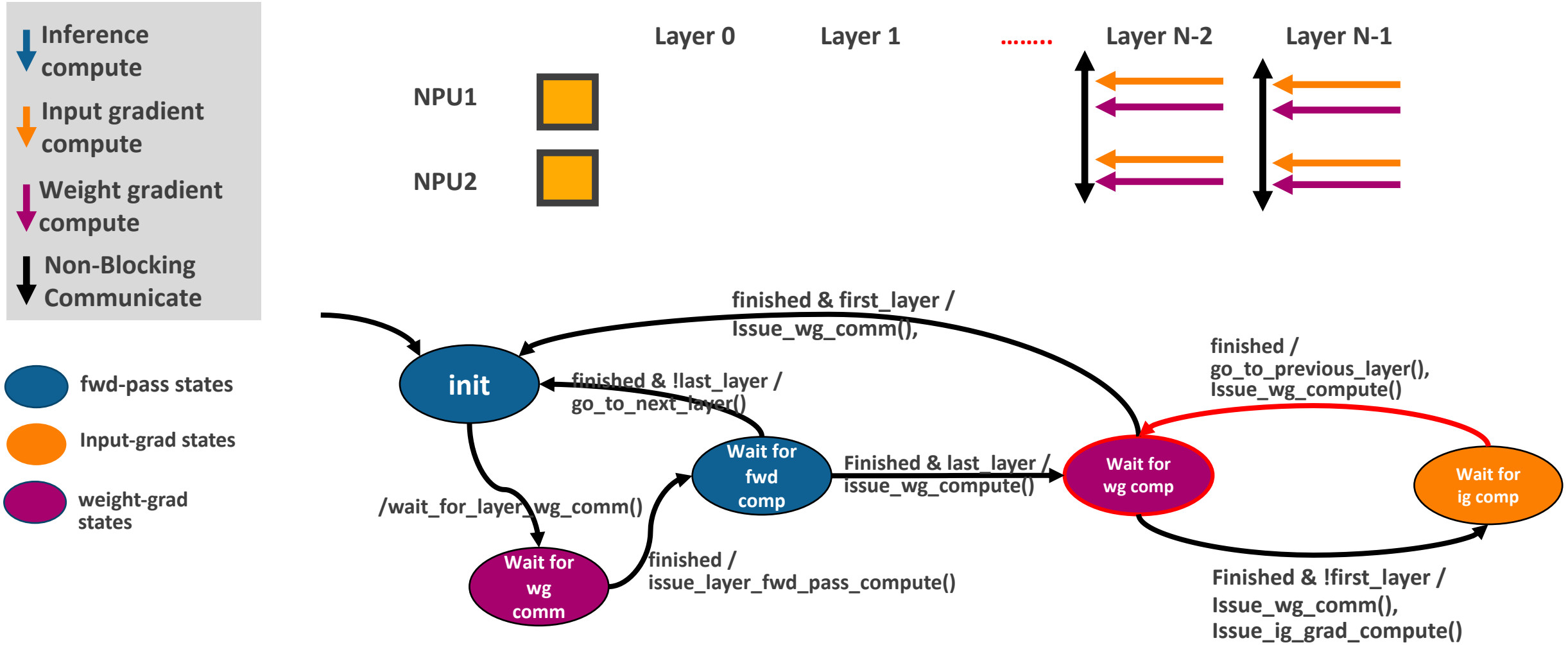
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



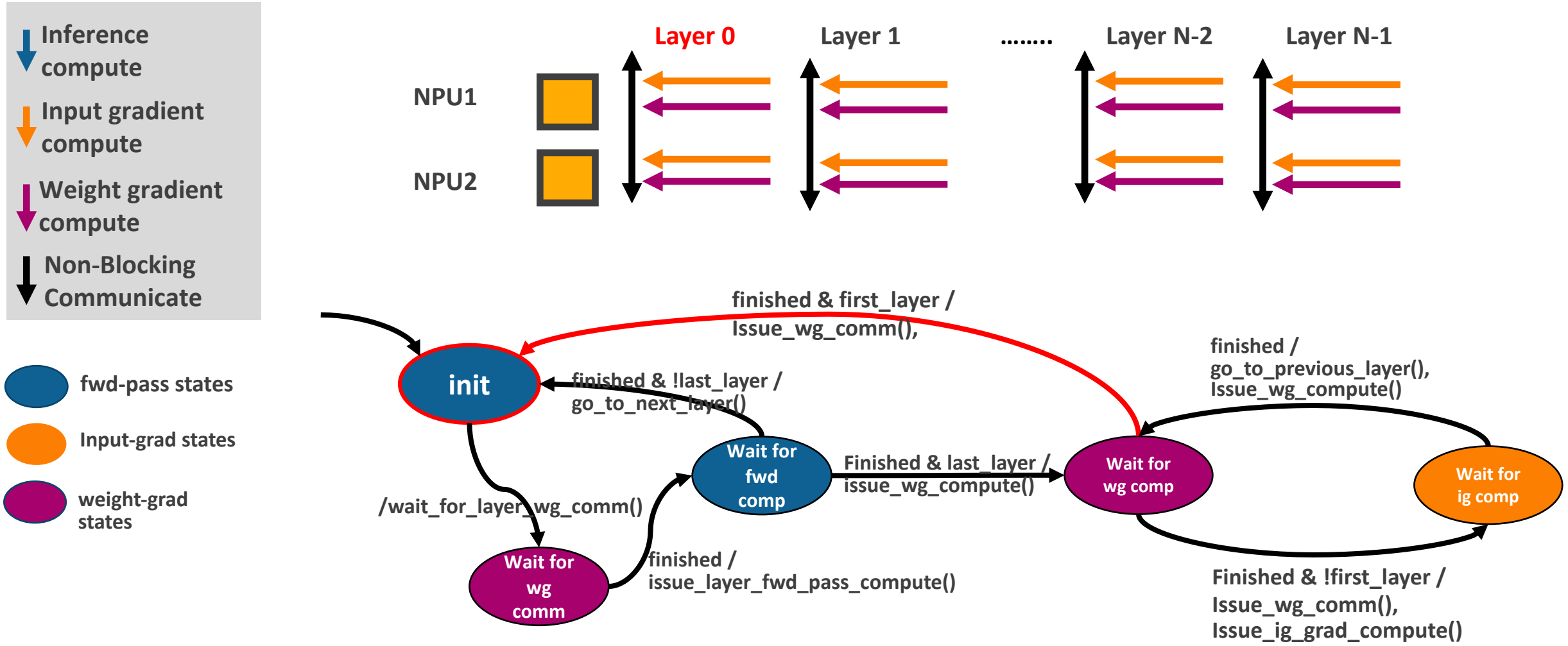
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



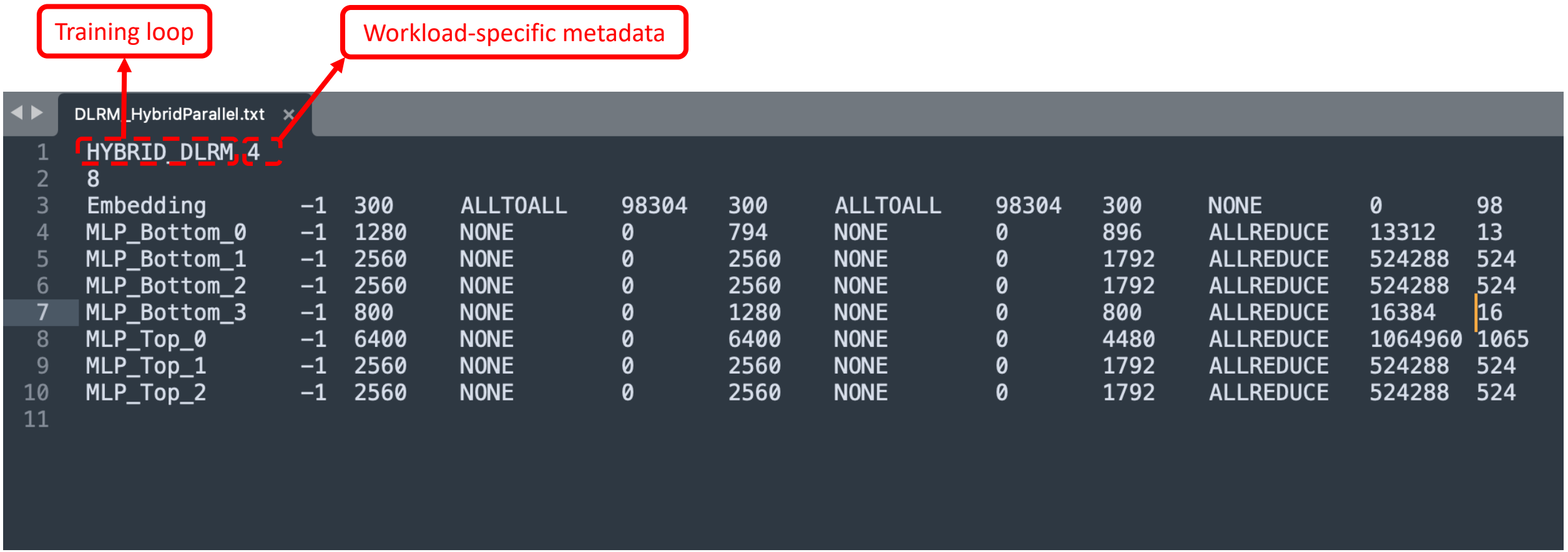
Text-Based Workload Training Loop: Data-Parallel Example

- Different training loops can be captured using state machines.



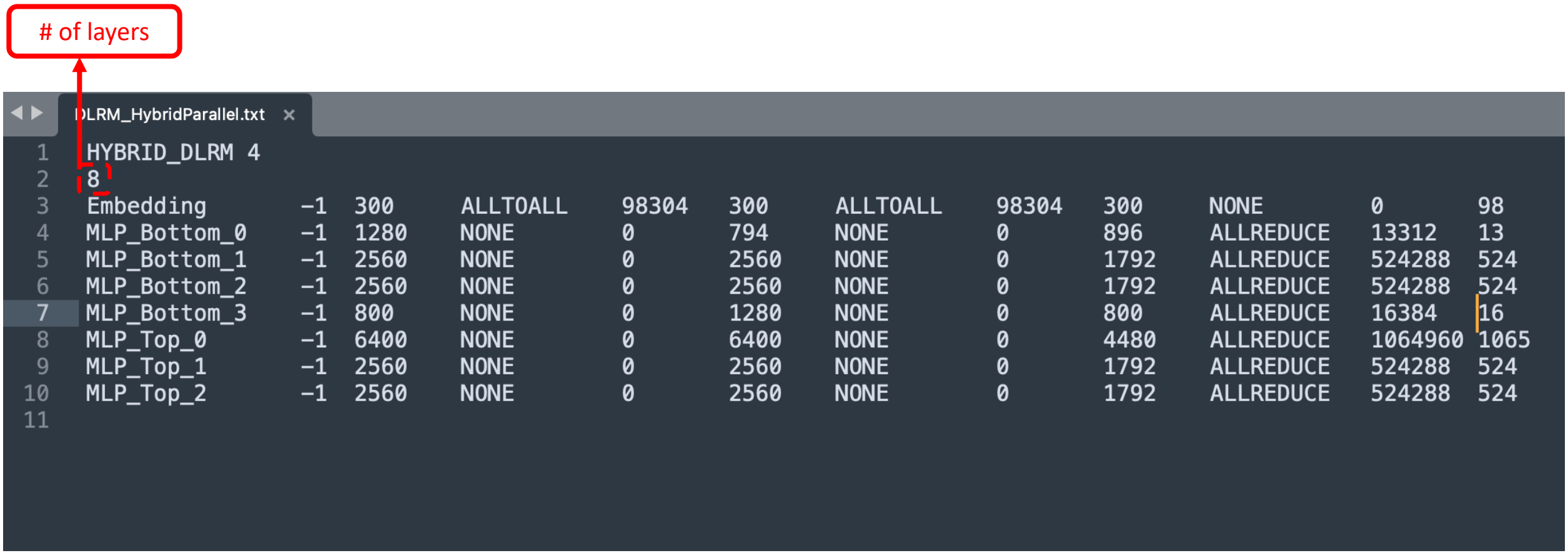
Text-Based Workload Input

- Captures the static structure of the workload (training loop).



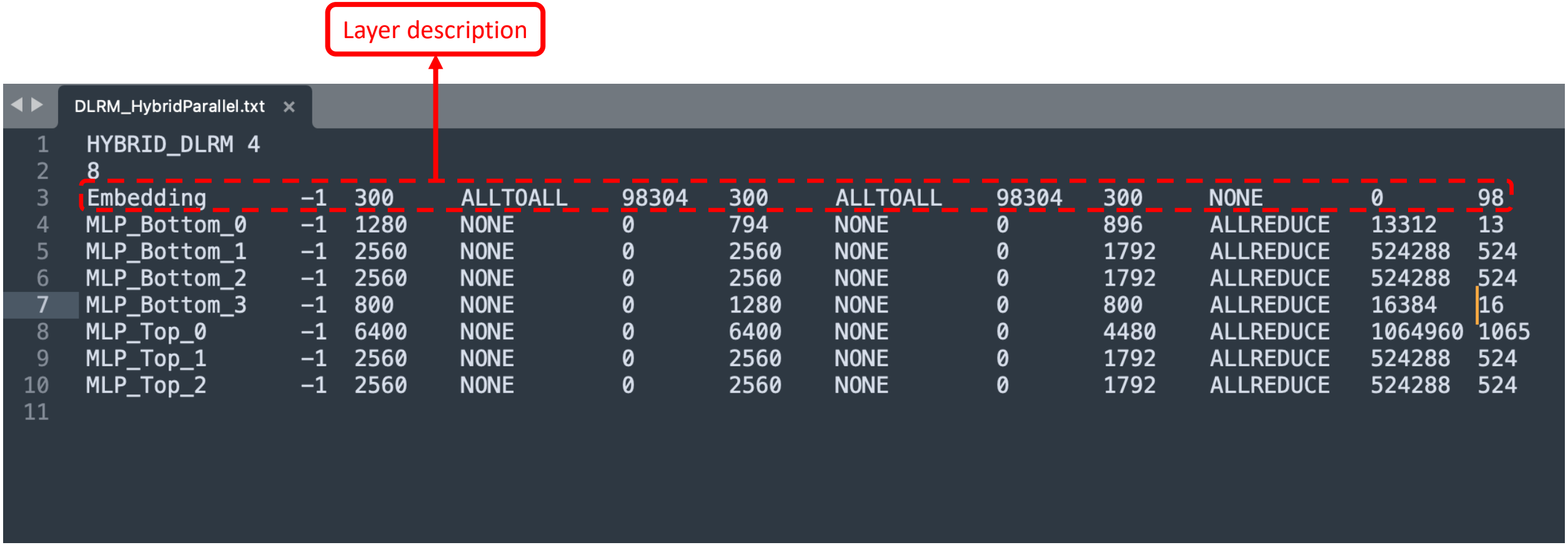
Text-Based Workload Input

- Captures the static structure of the workload (training loop).



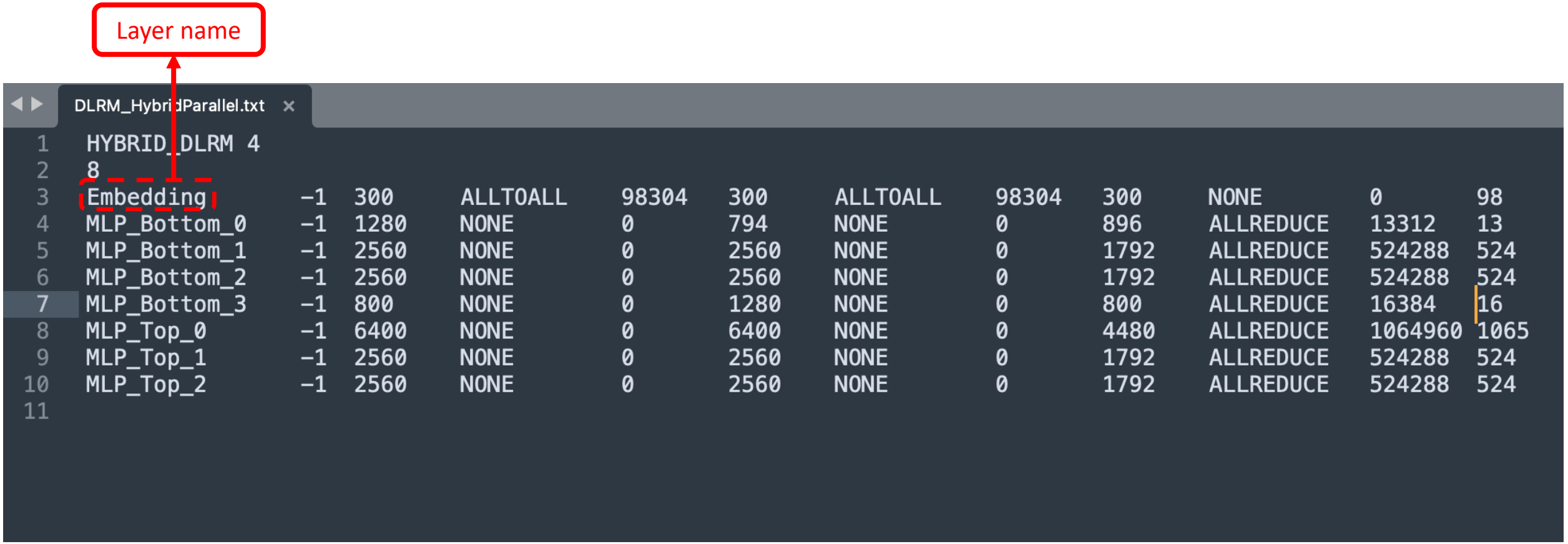
Text-Based Workload Input

- Captures the static structure of the workload (training loop).



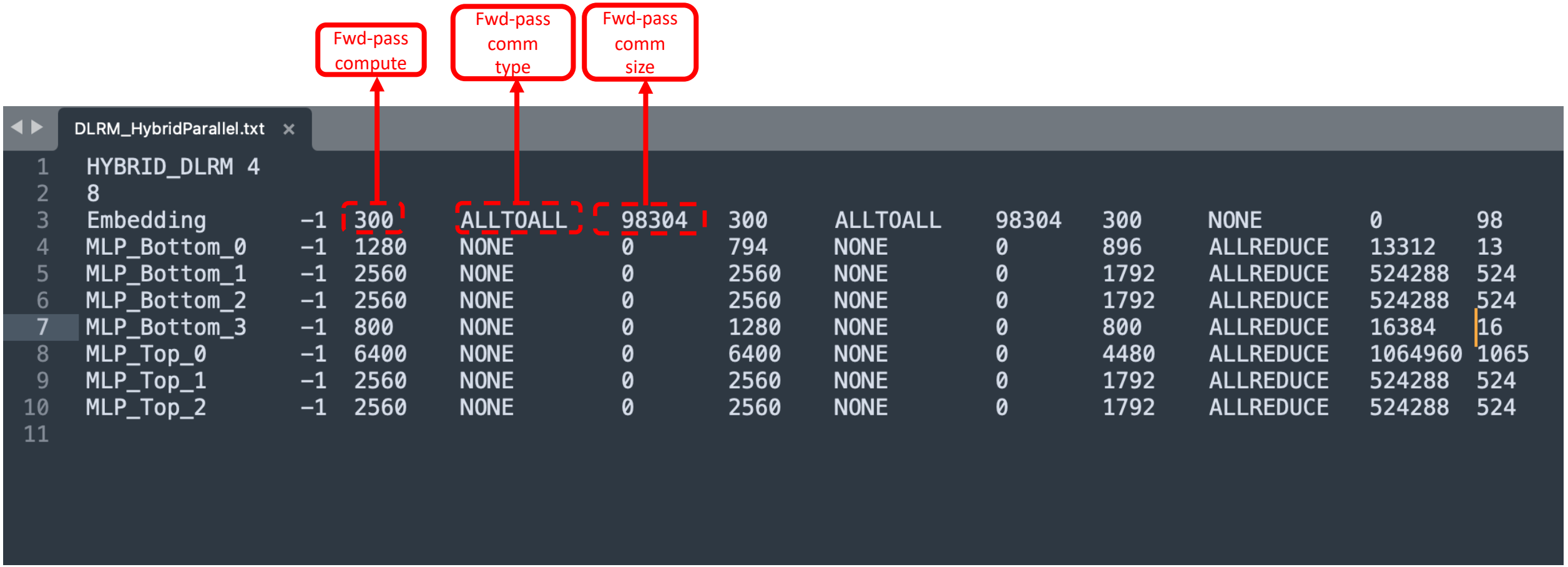
Text-Based Workload Input

- Captures the static structure of the workload (training loop).



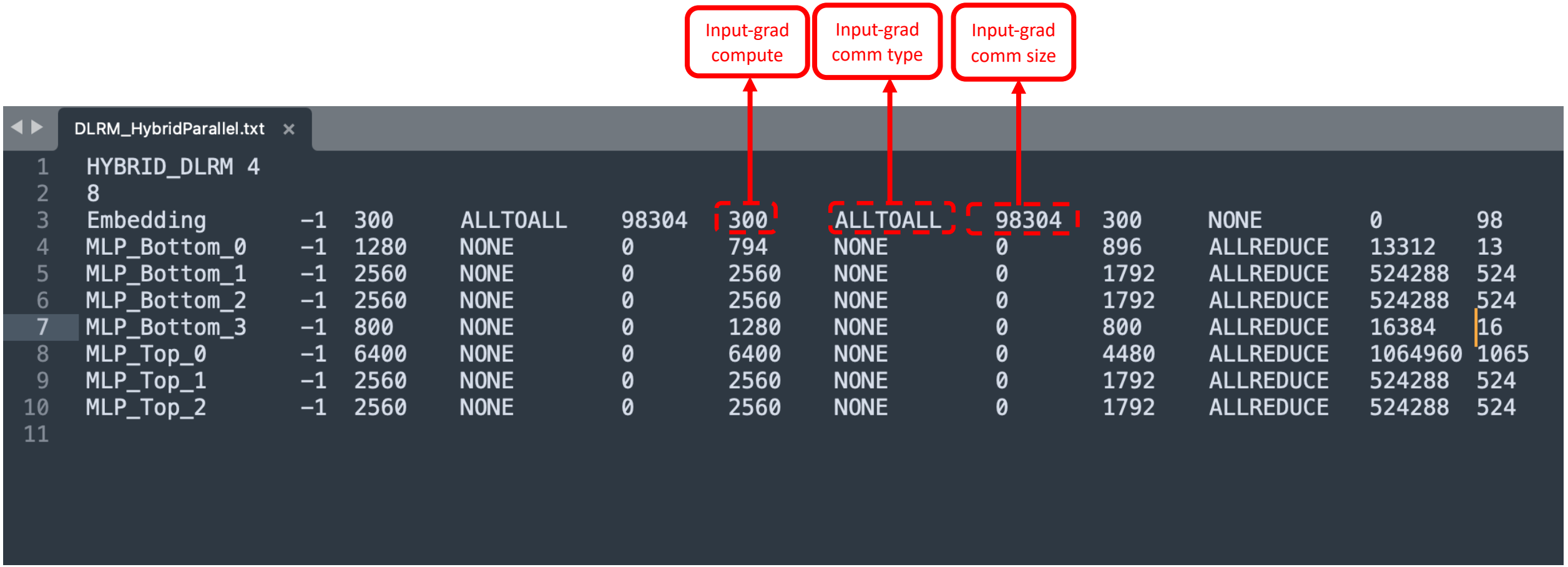
Text-Based Workload Input

- Captures the static structure of the workload (training loop).



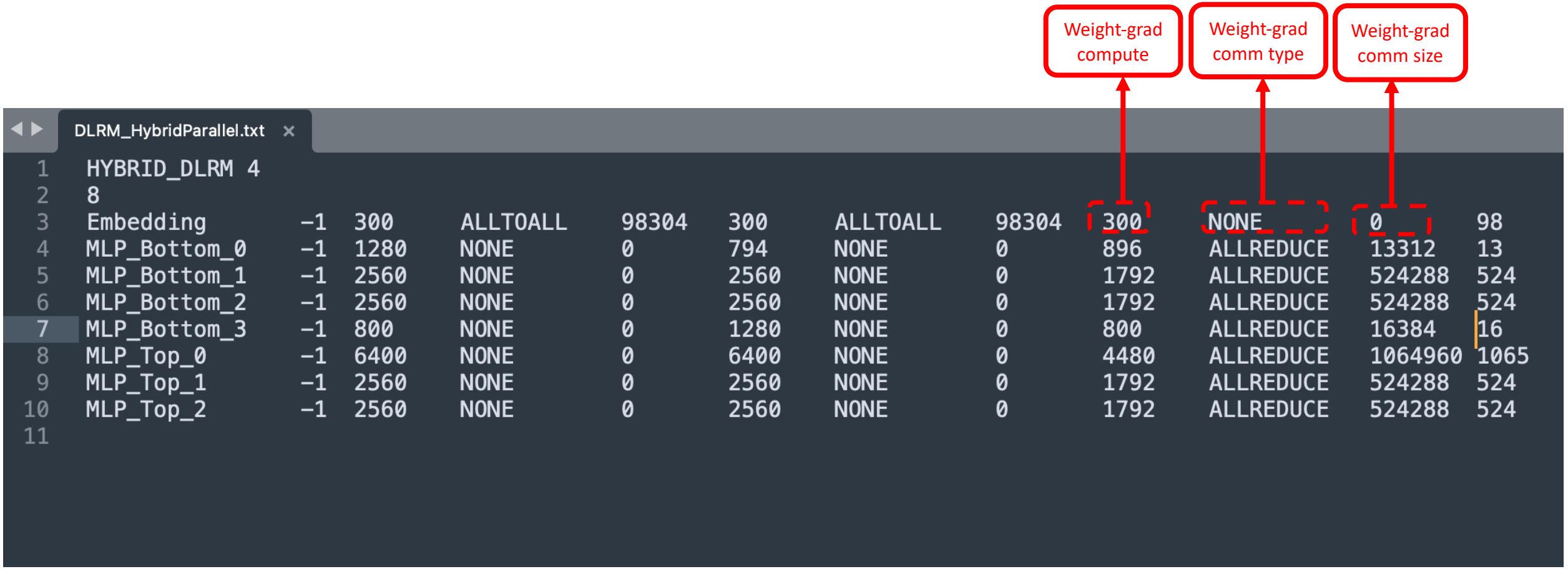
Text-Based Workload Input

- Captures the static structure of the workload (training loop).



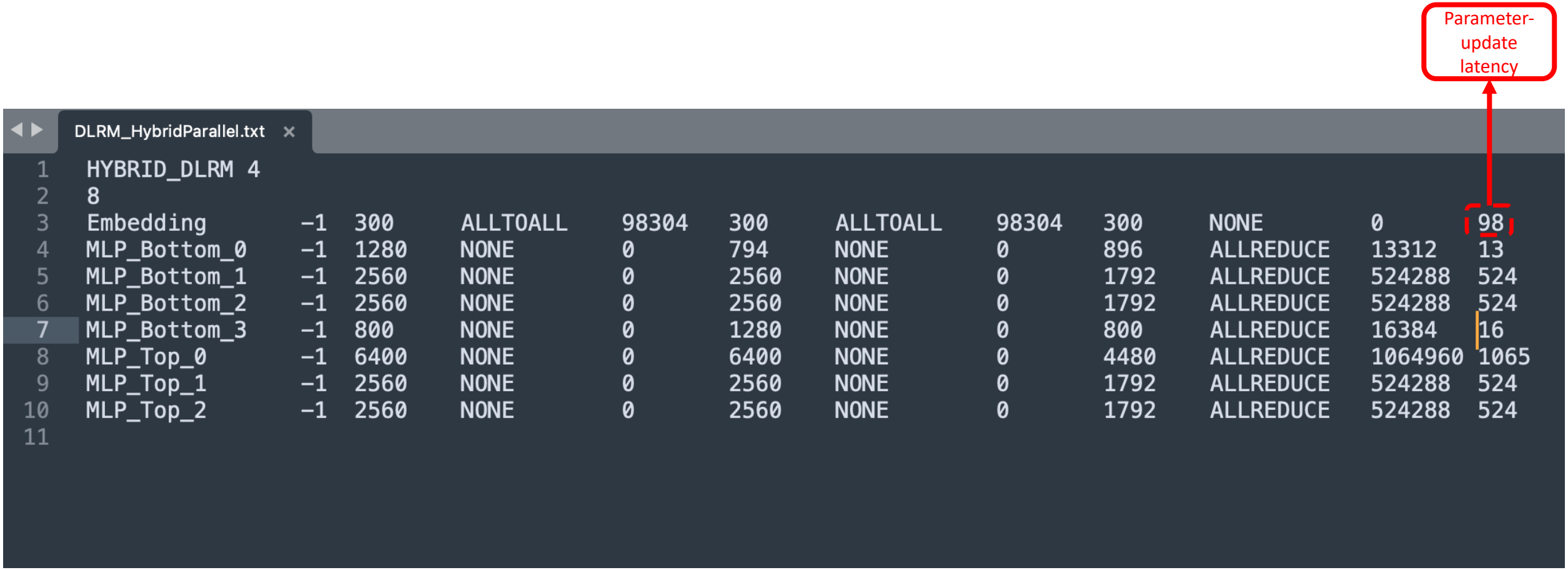
Text-Based Workload Input

- Captures the static structure of the workload (training loop).



Text-Based Workload Input

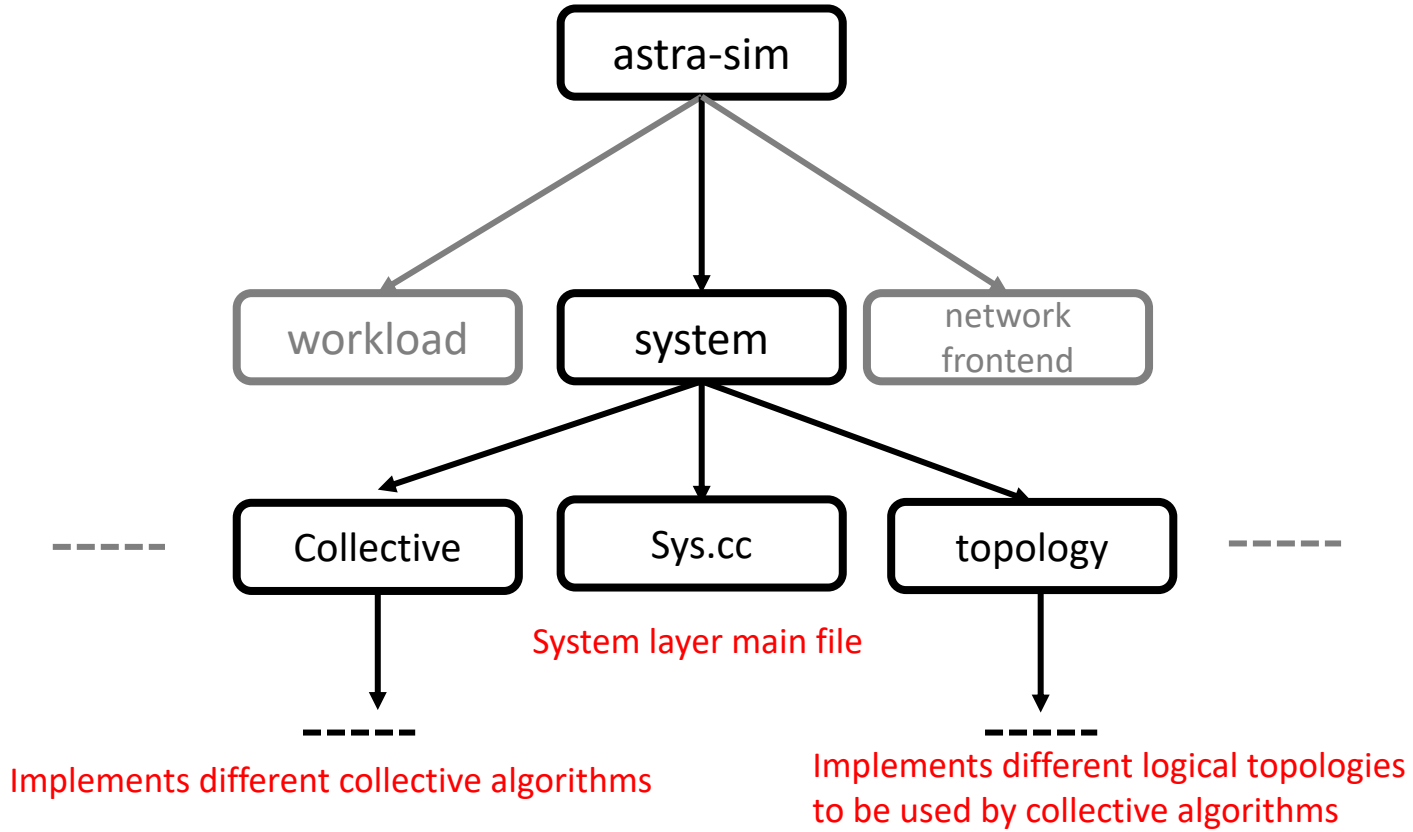
- Captures the static structure of the workload (training loop).



System Layer

System Layer Collective Implementation

- Code Structure

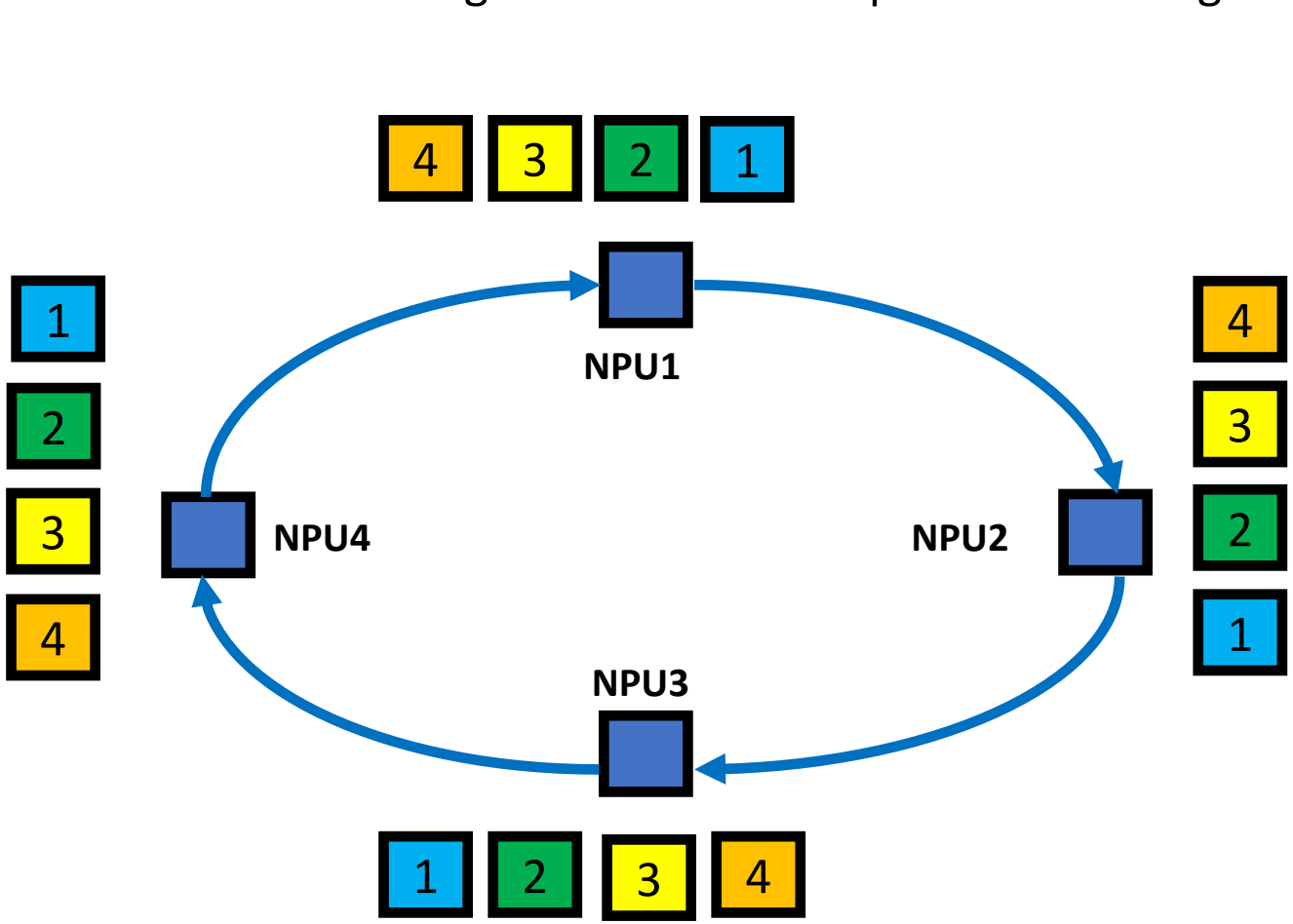


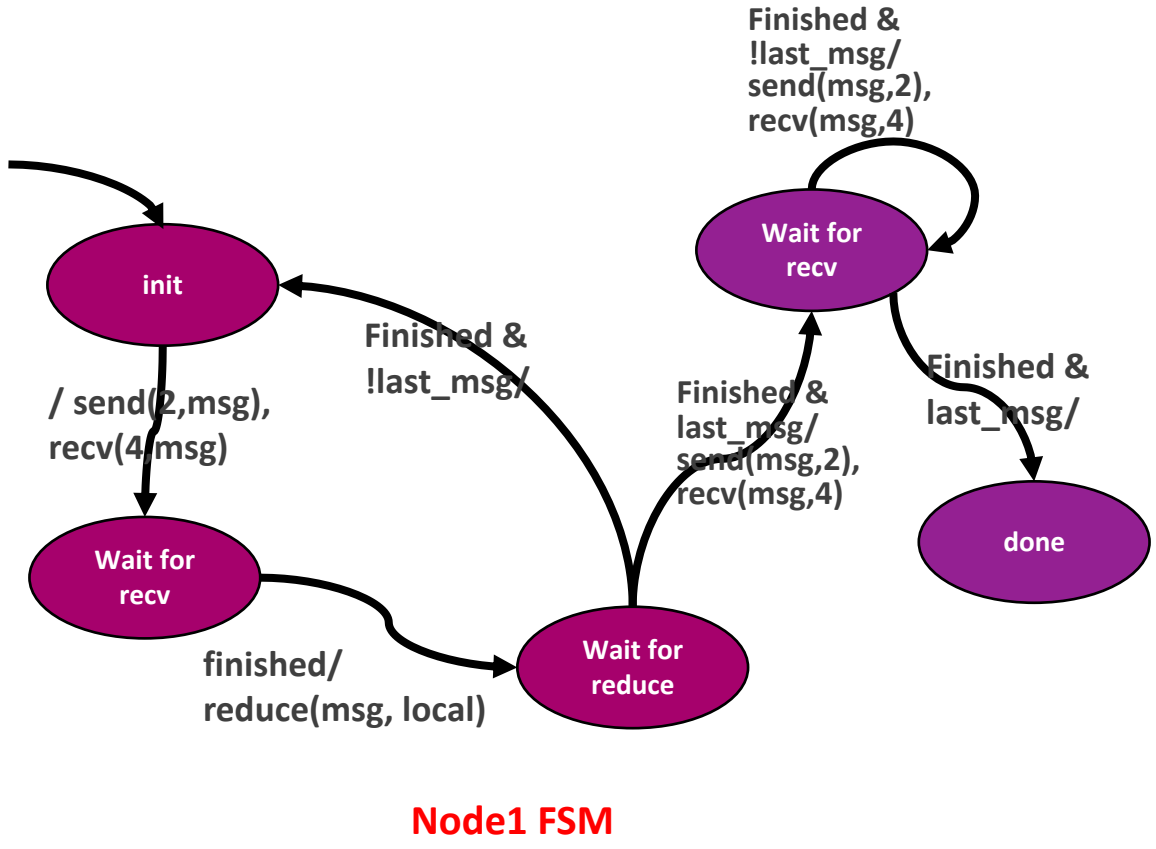
System Layer Collective Implementation

- Each collective algorithm works based on a logical topology.
- Logical topologies are implemented in “system/topology/*” and instantiated in sys.cc.
- Collective algorithms are implemented in “system/topology/*” and instantiated in sys.cc.
- Collective algorithms can be implemented using **state machines**.

System Layer Collective Implementation

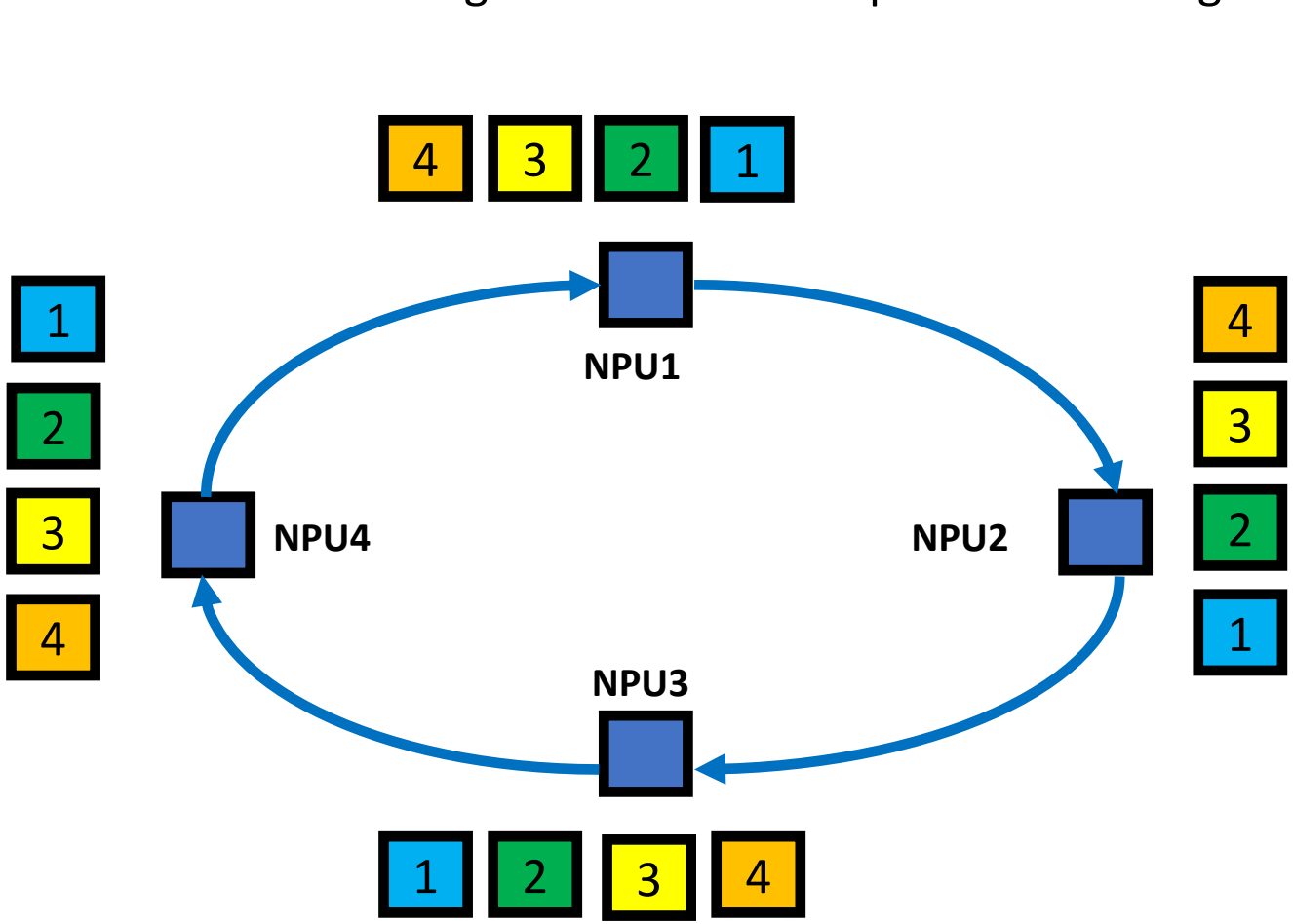
- Collective algorithms can be implemented using **state machines**.

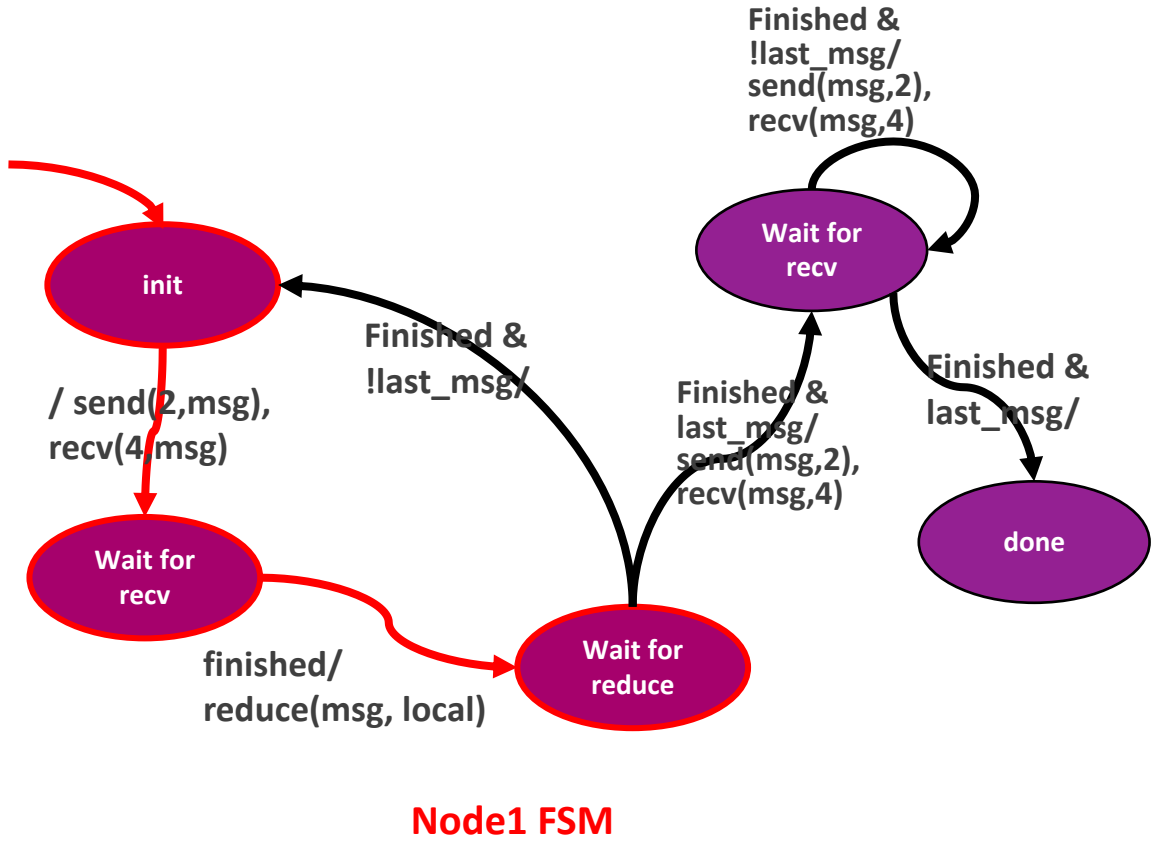




System Layer Collective Implementation

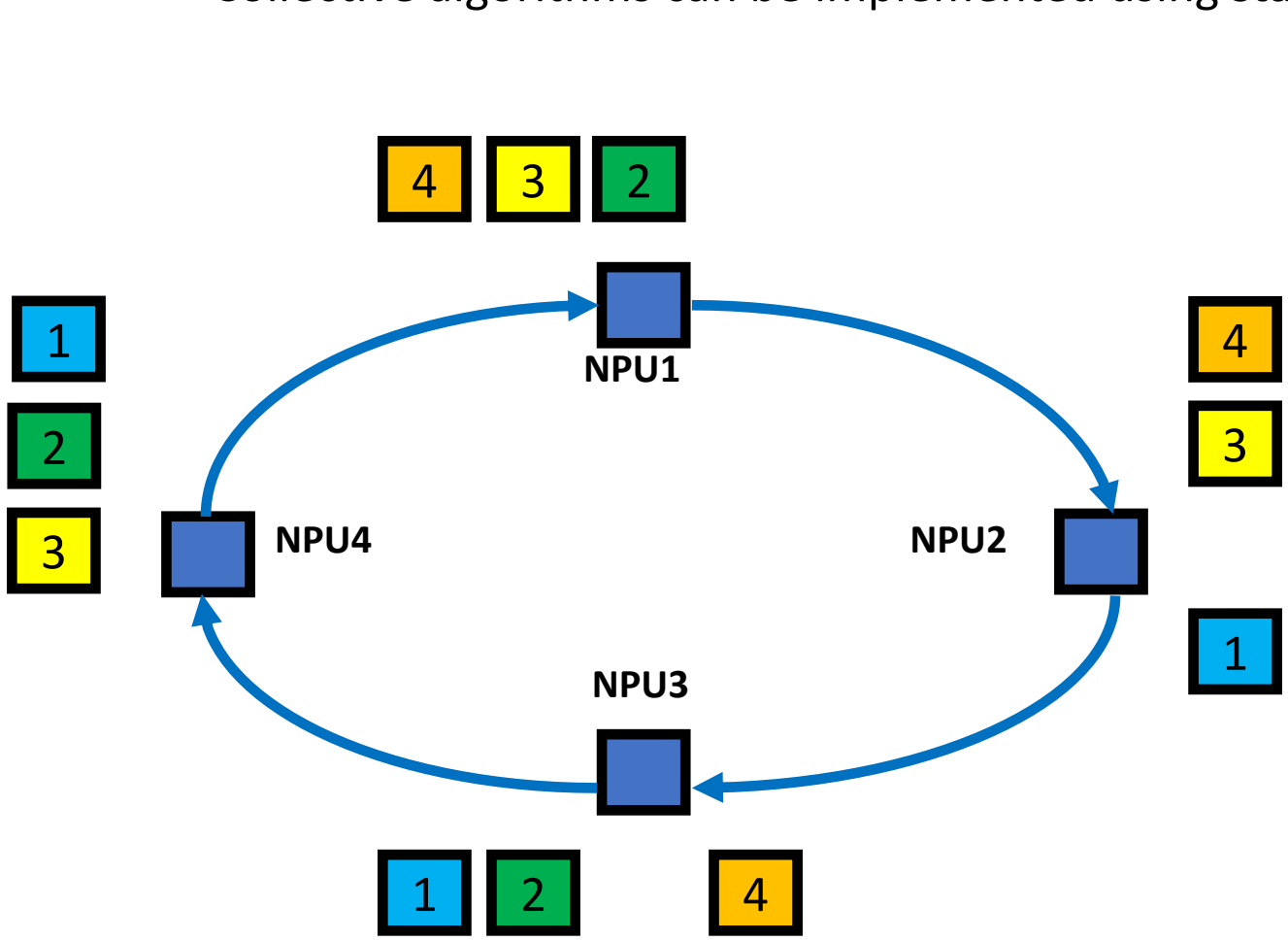
- Collective algorithms can be implemented using **state machines**.

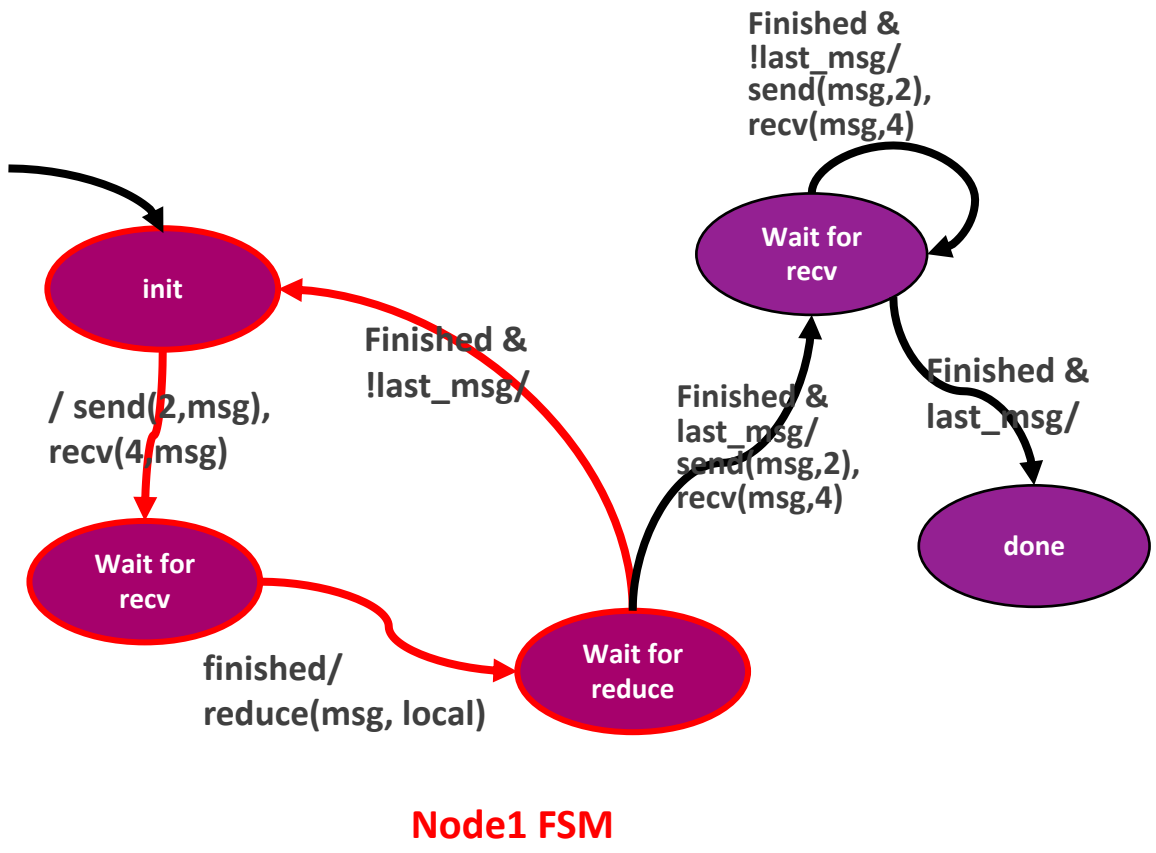




System Layer Collective Implementation

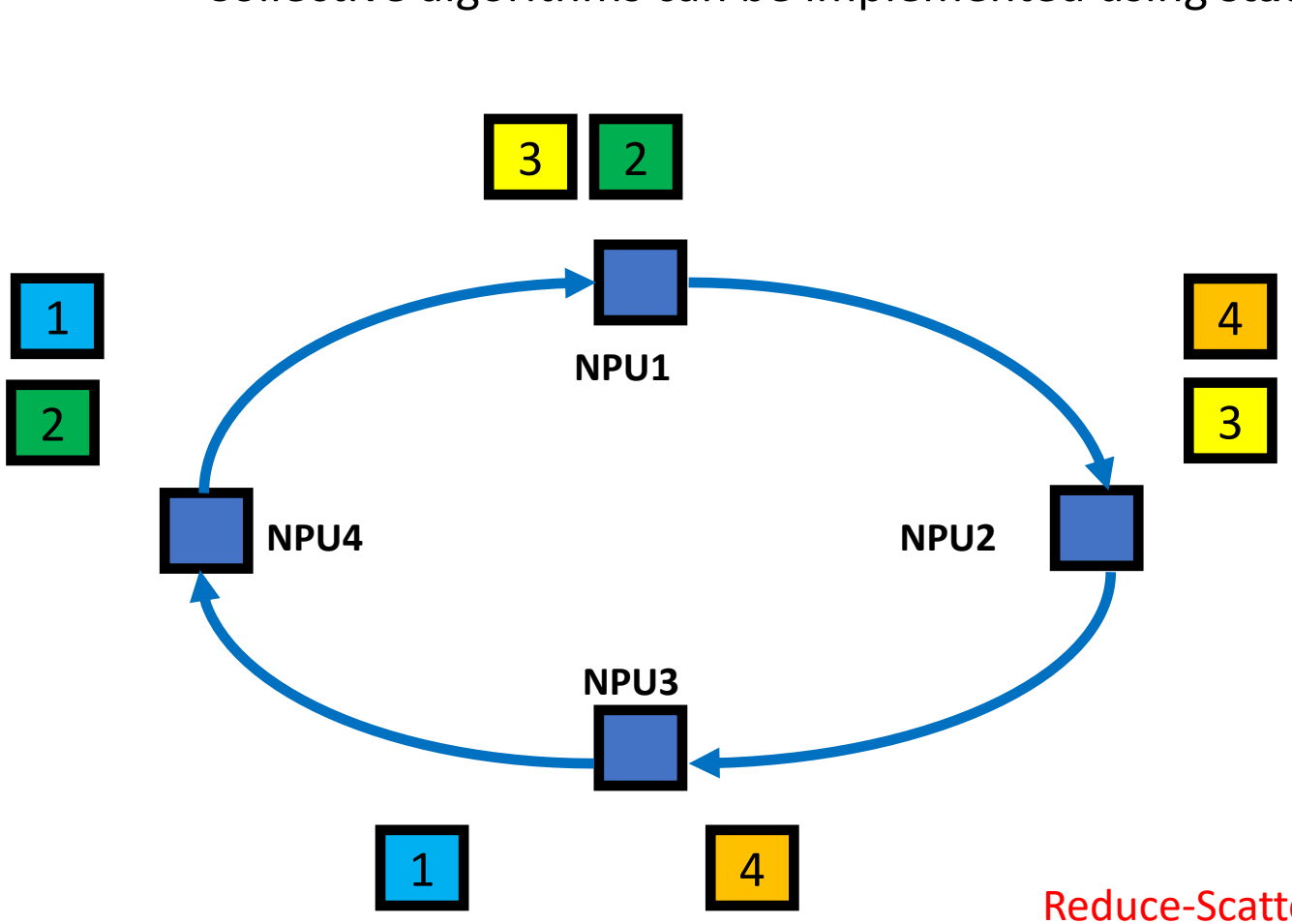
- Collective algorithms can be implemented using **state machines**.

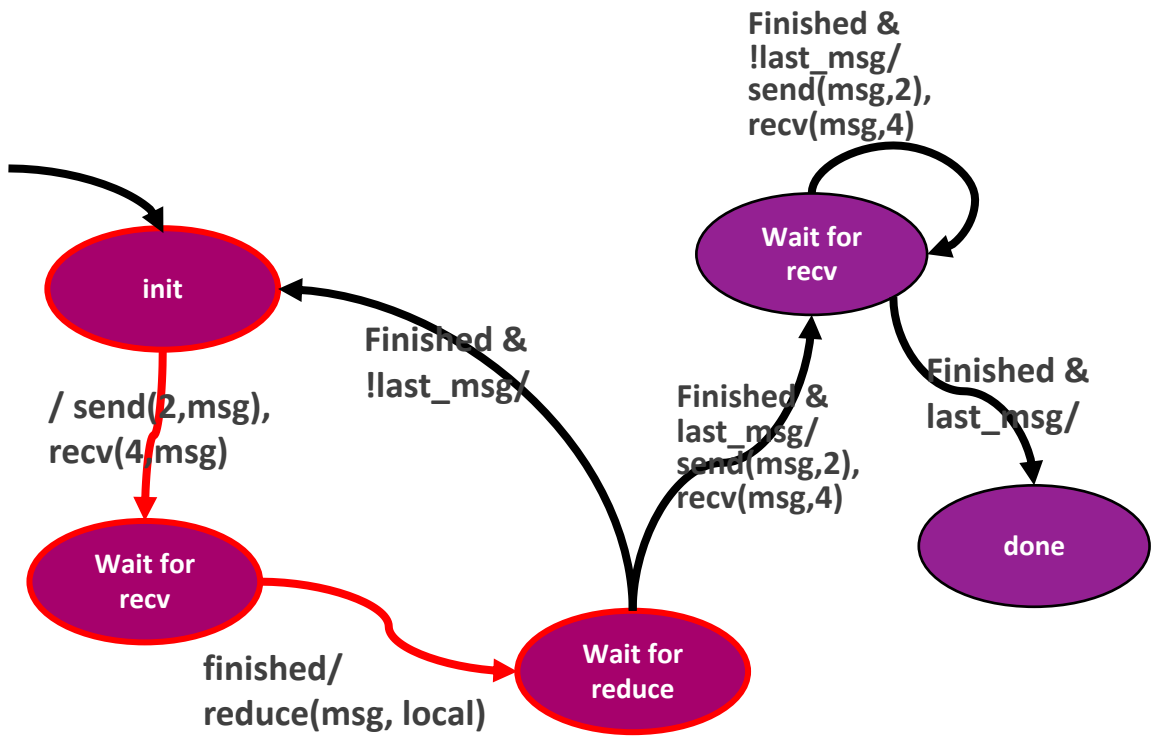




System Layer Collective Implementation

- Collective algorithms can be implemented using **state machines**.



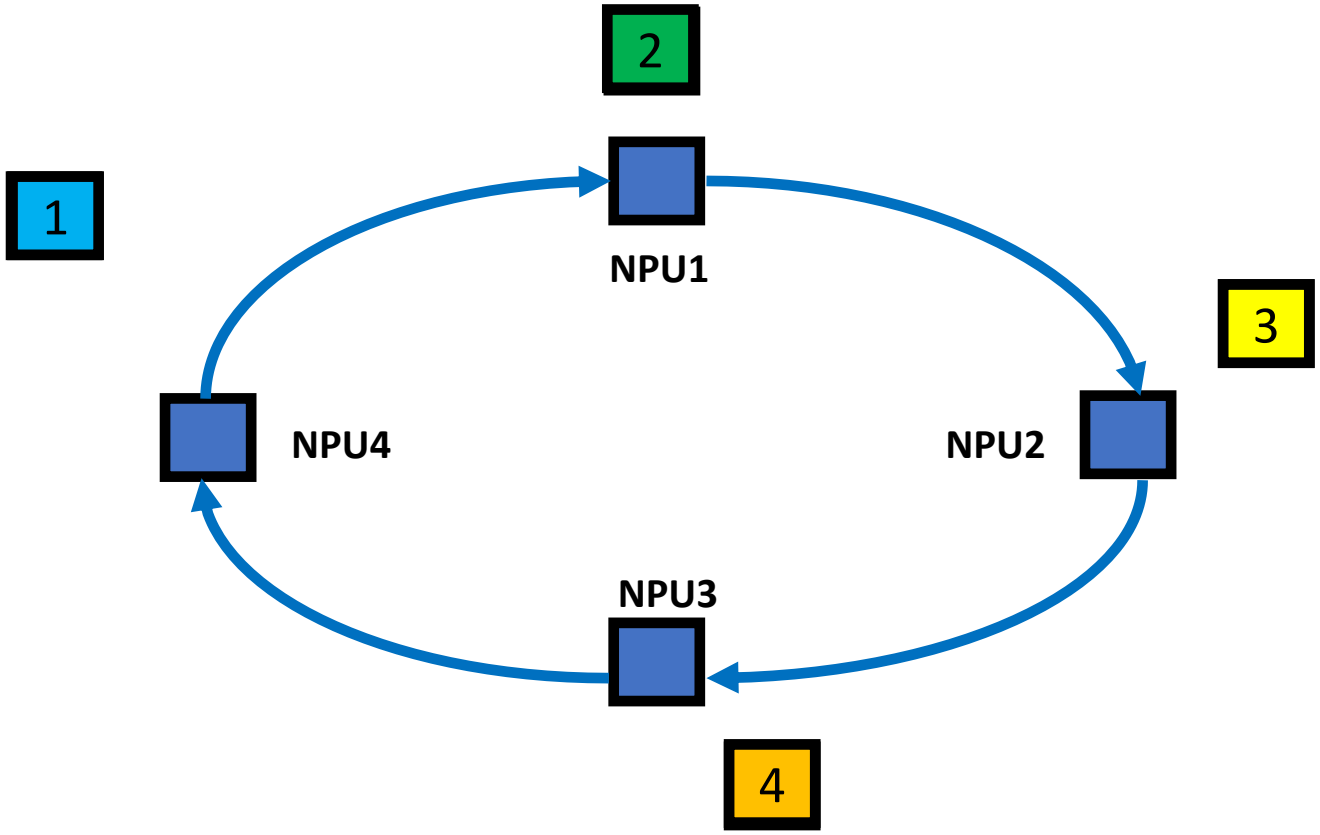


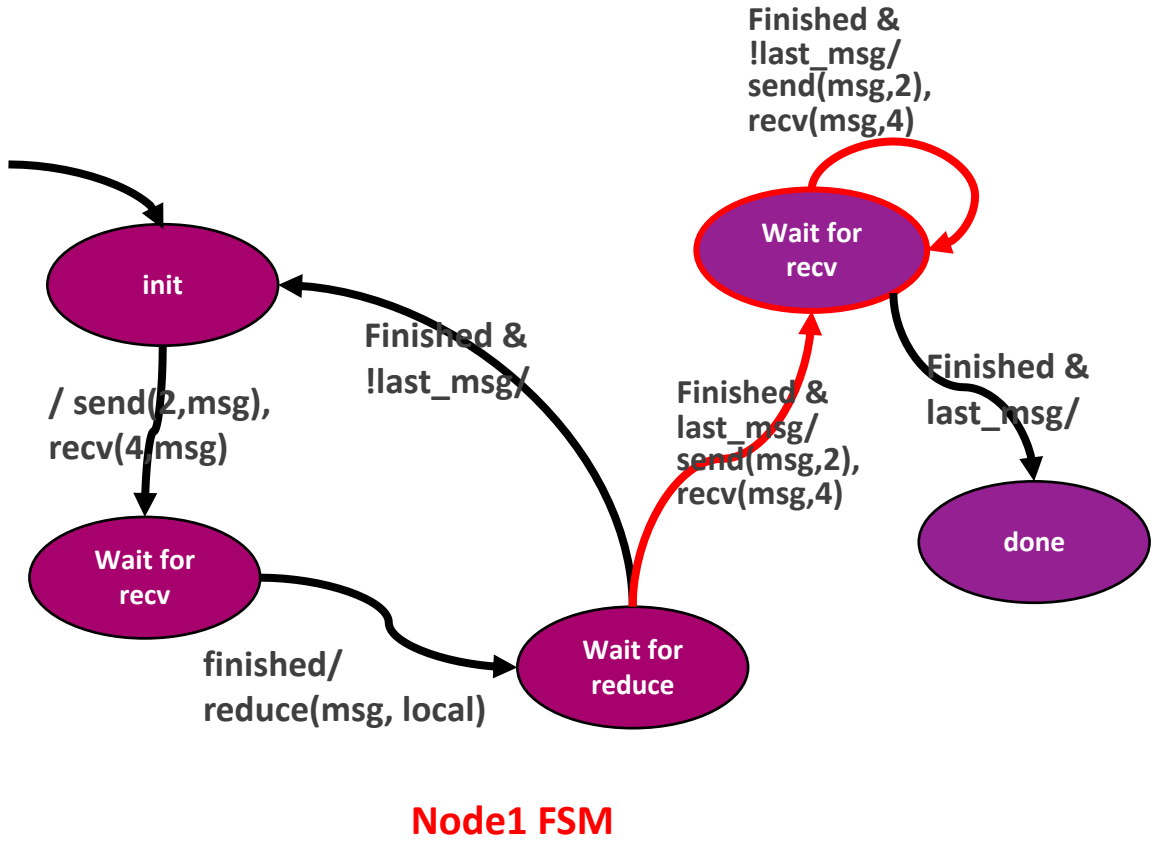
Node1 FSM

Reduce-Scatter done!

System Layer Collective Implementation

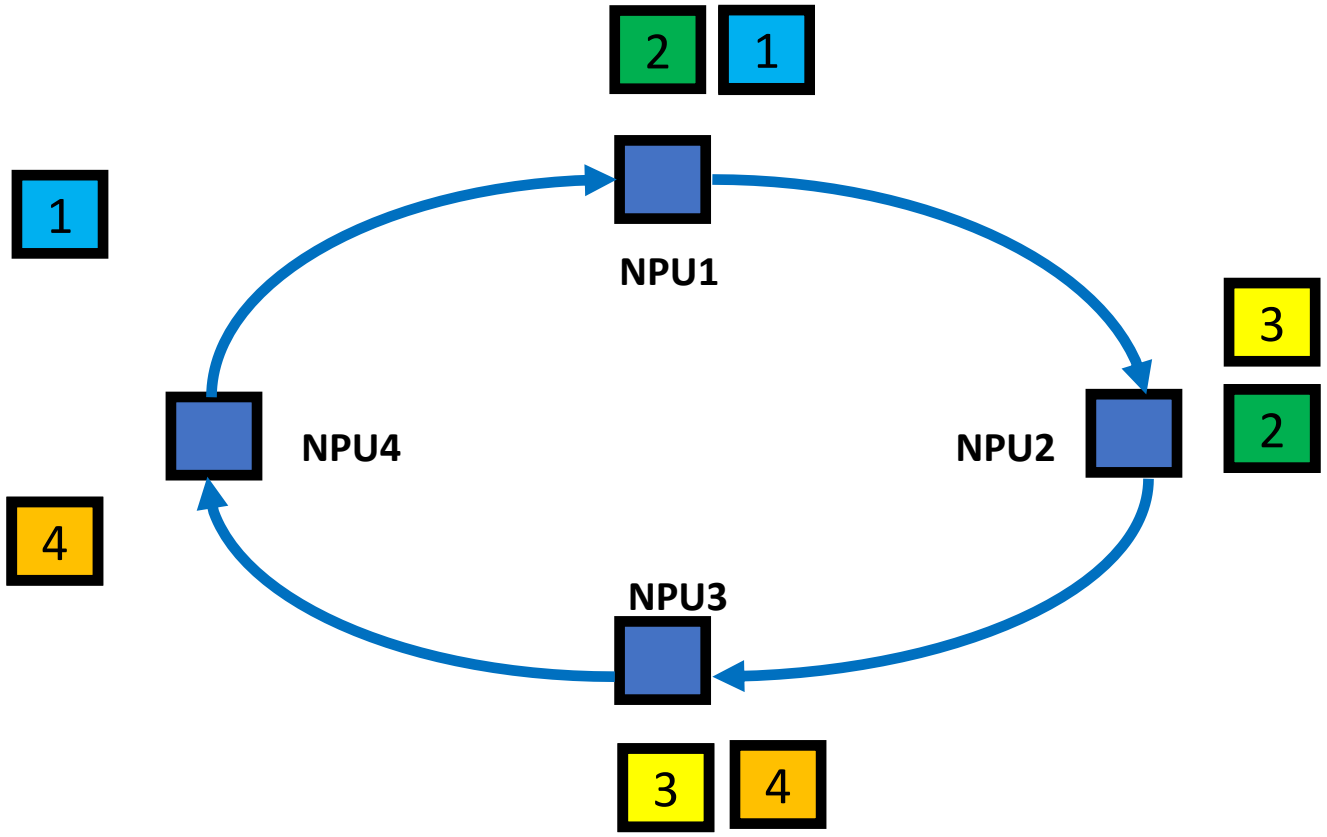
- Collective algorithms can be implemented using **state machines**.

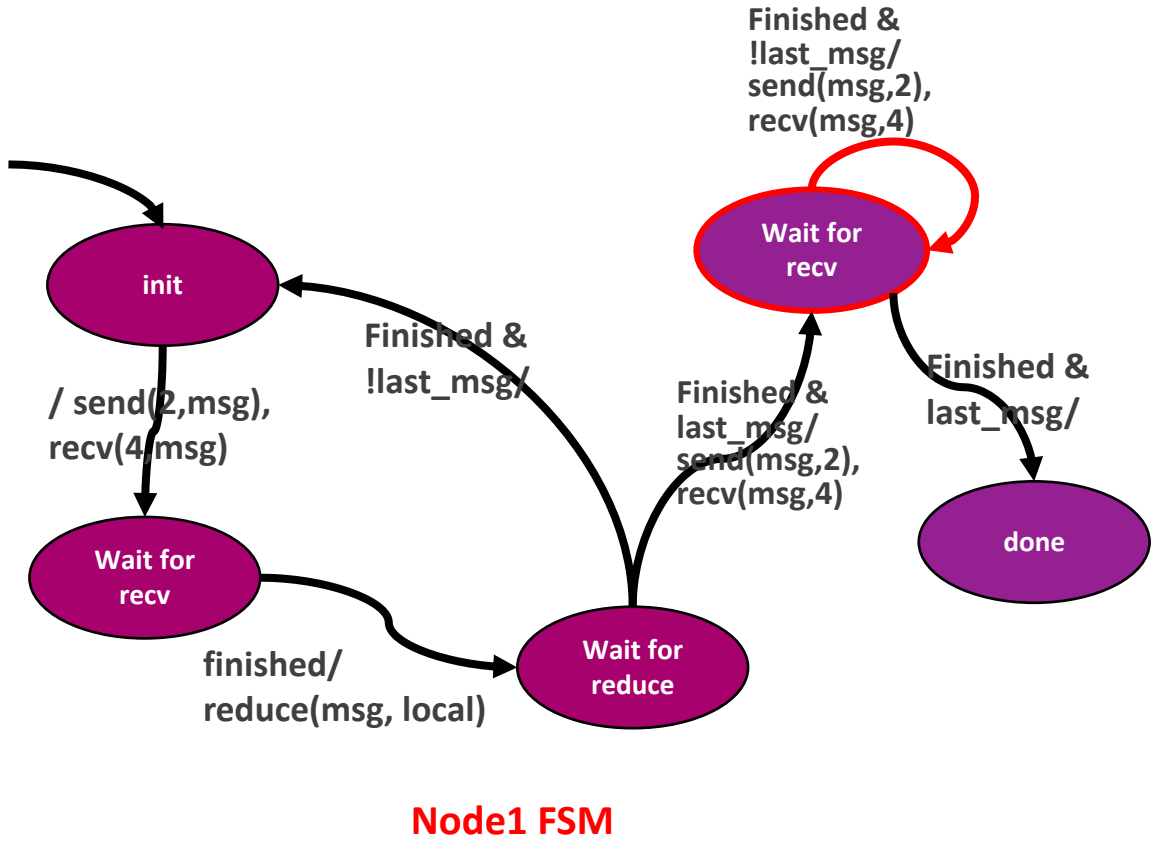




System Layer Collective Implementation

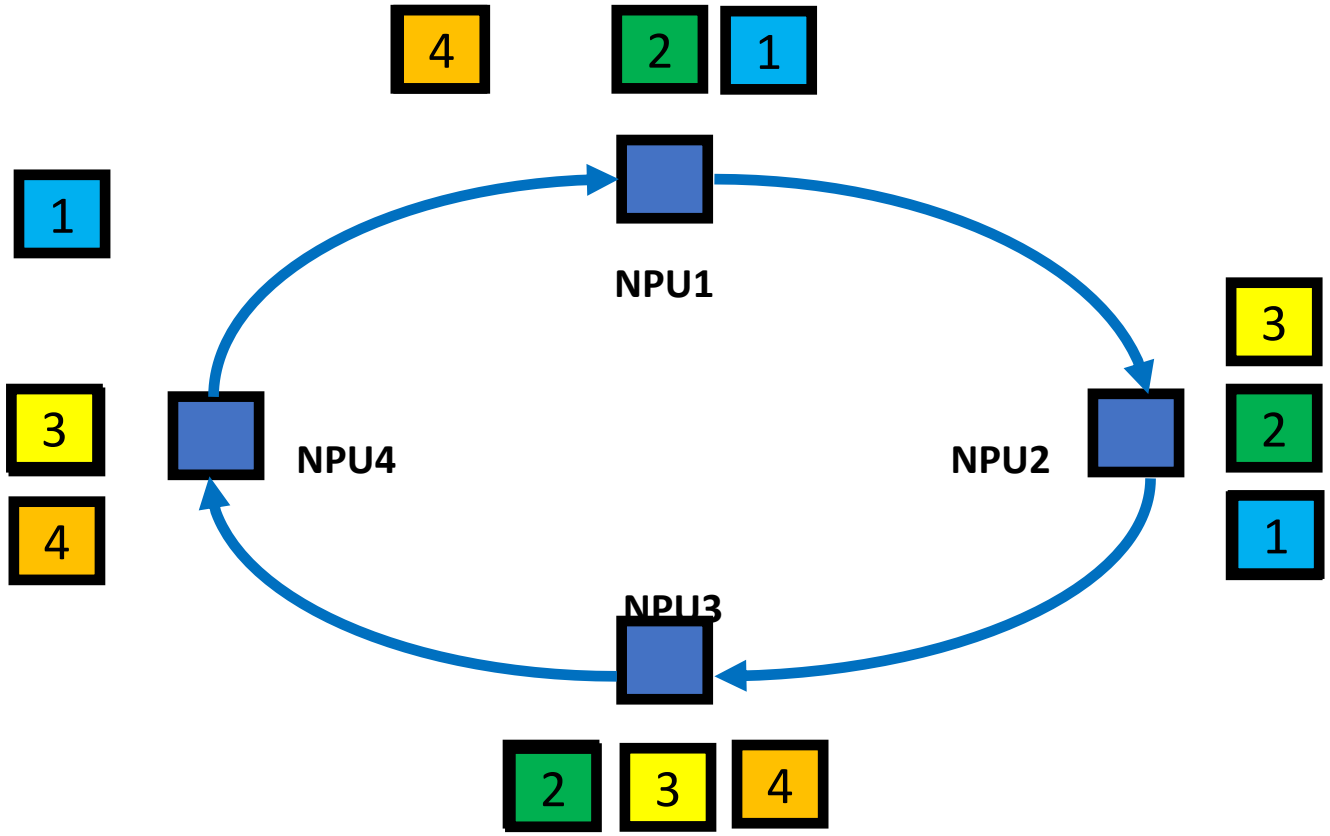
- Collective algorithms can be implemented using **state machines**.

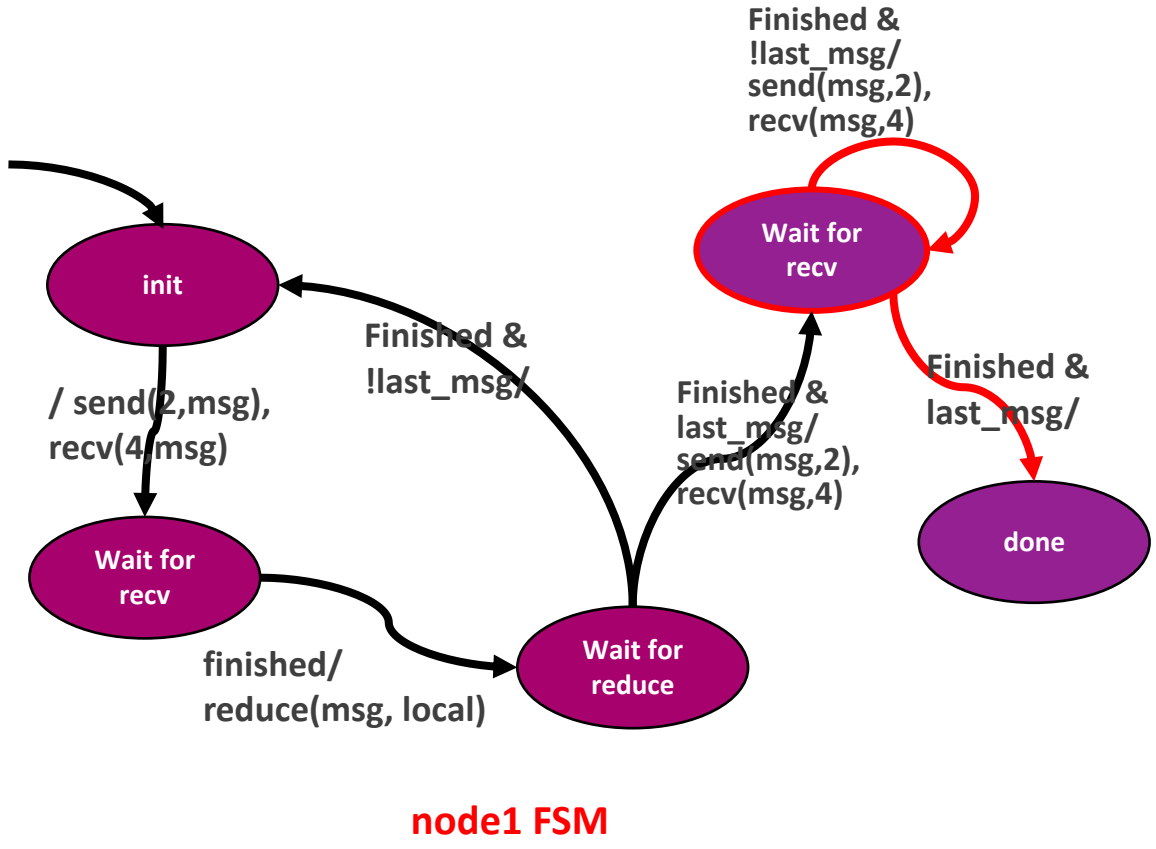




System Layer Collective Implementation

- Collective algorithms can be implemented using **state machines**.

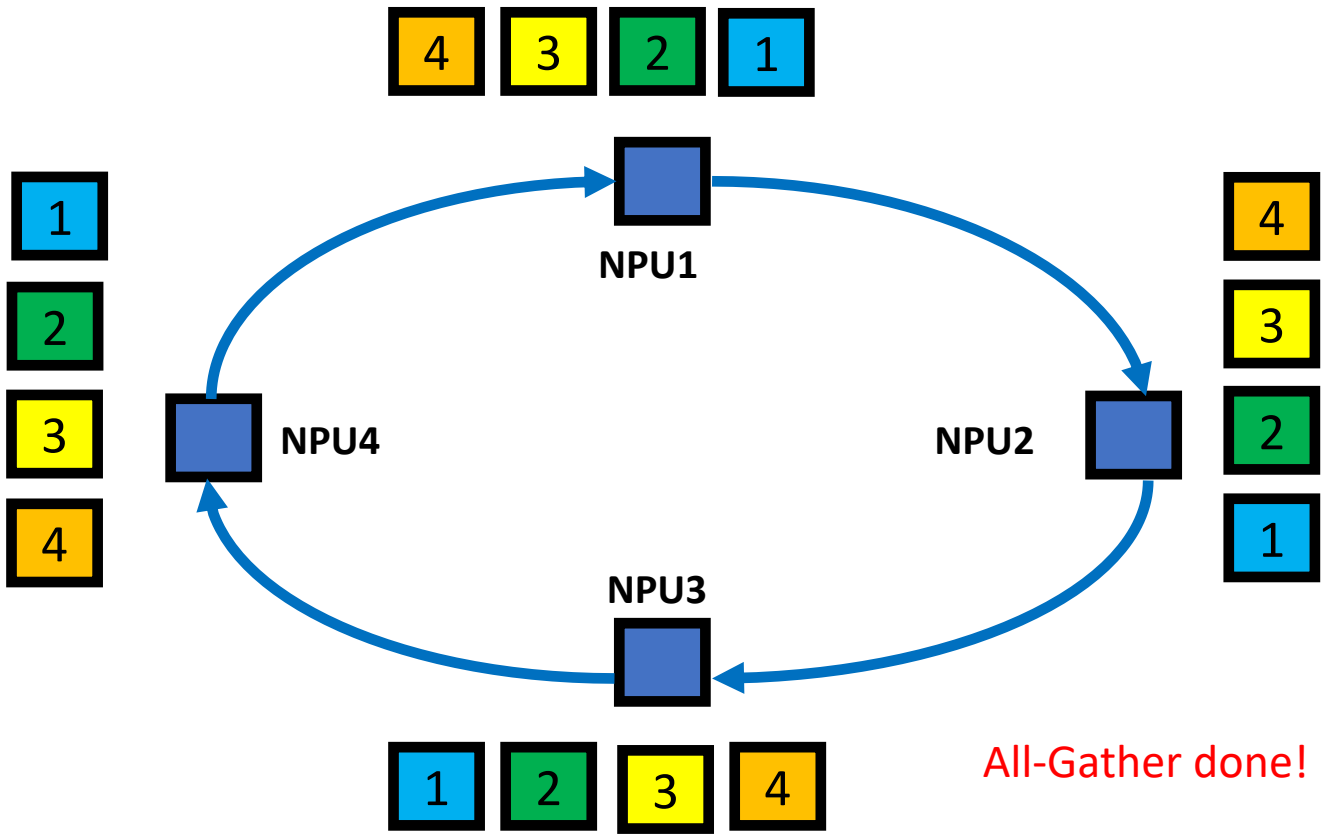




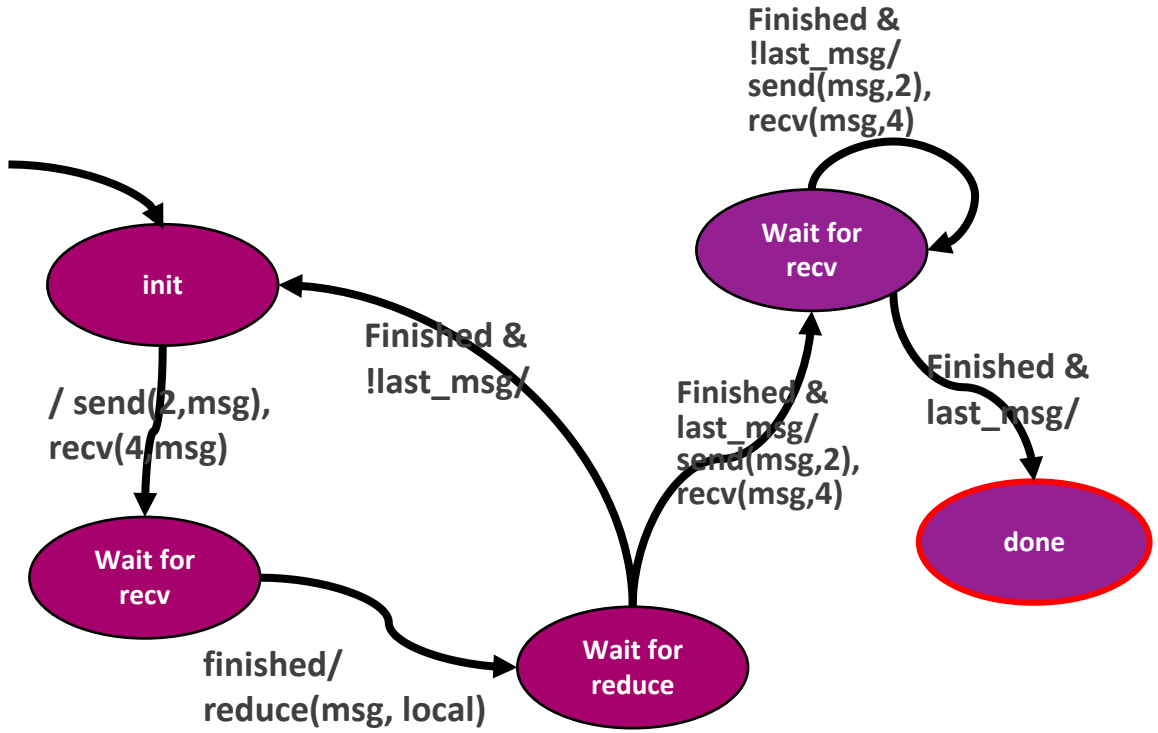
node1 FSM

System Layer Collective Implementation

- Collective algorithms can be implemented using **state machines**.



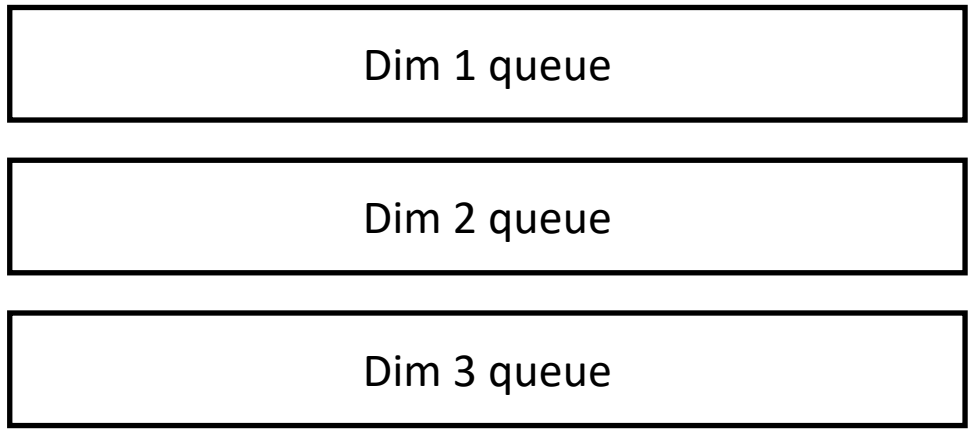
All-Gather done!

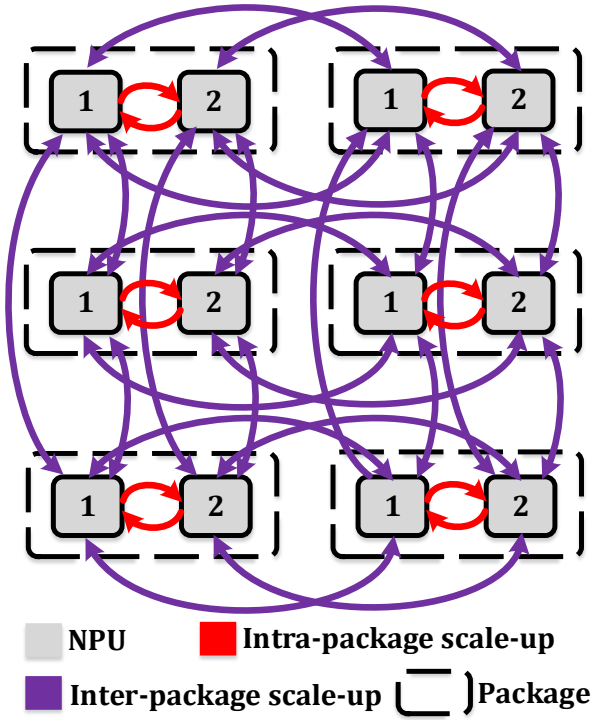


node1 FSM

Collective Scheduler

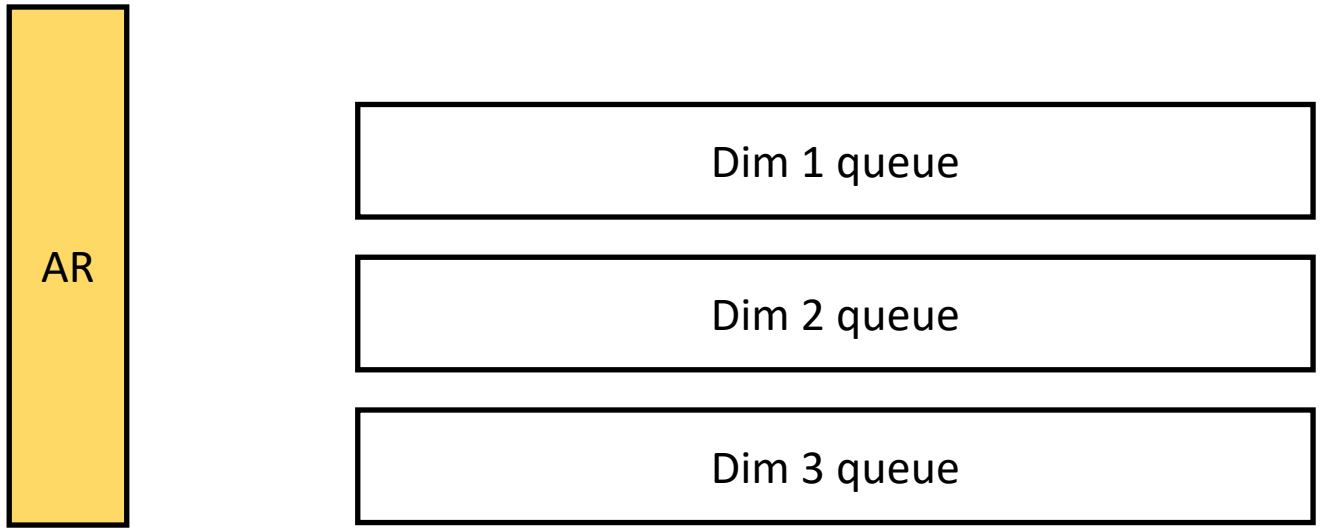
- There are one/multiple queue(s) per each physical network dimension.
- A collective is broken into multiple chunks and inserted into the first queue.
- Queues process chunks in-order.



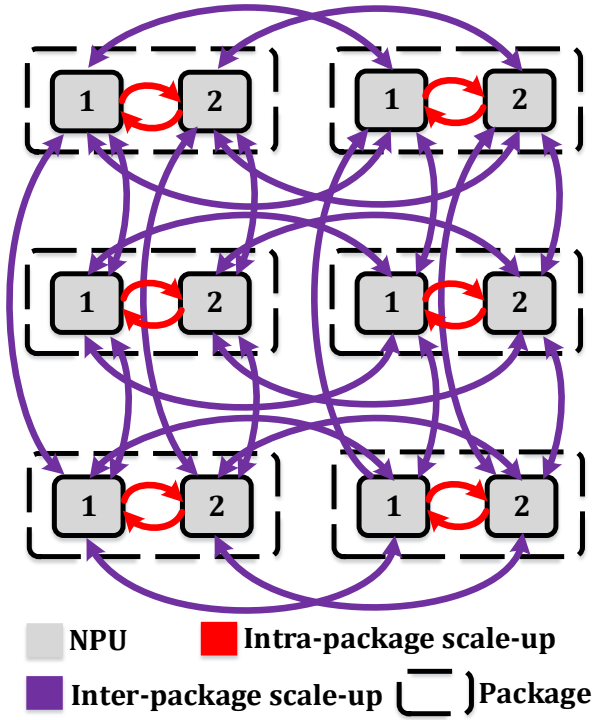


Collective Scheduler

- There are one/multiple queue(s) per each physical network dimension.
- A collective is broken into multiple chunks and inserted into the first queue.
- Queues process chunks in-order.

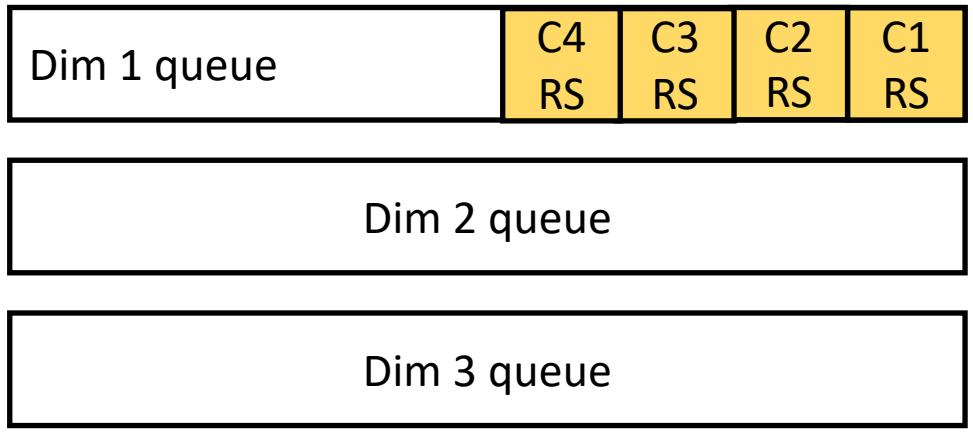


AR: All-Reduce
RS: Reduce-Scatter
AG: All-Gather

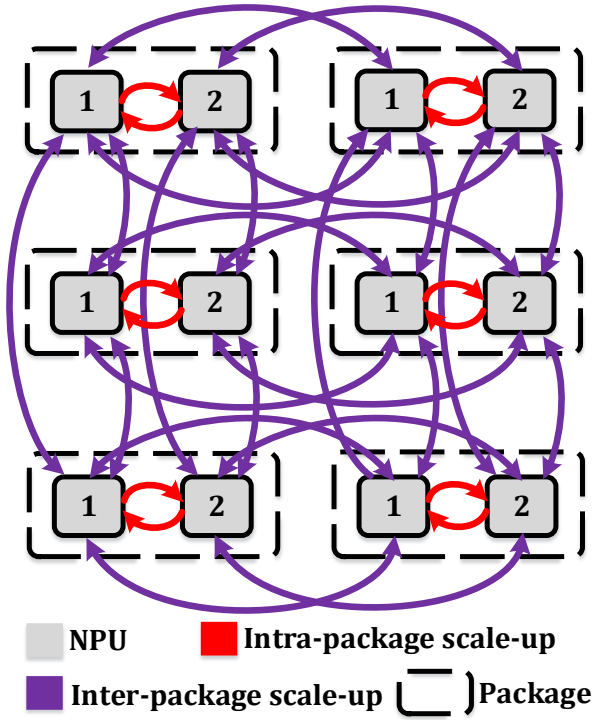


Collective Scheduler

- There are one/multiple queue(s) per each physical network dimension.
- A collective is broken into multiple chunks and inserted into the first queue.
- Queues process chunks in-order.

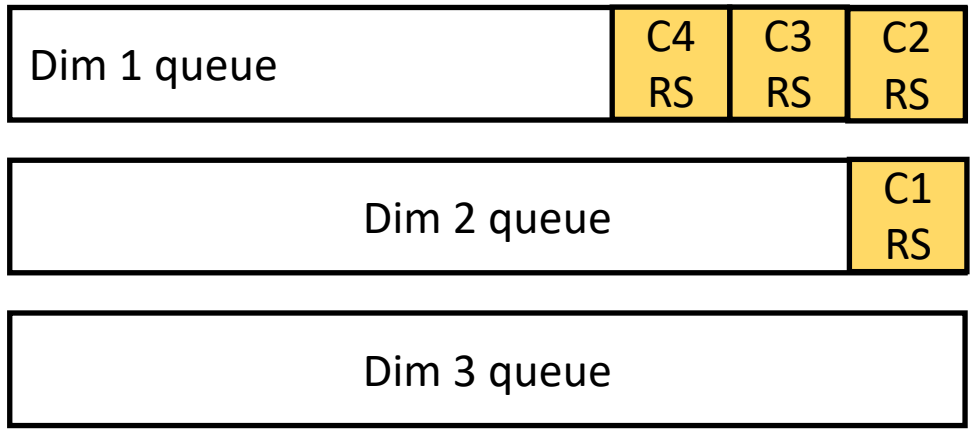


AR: All-Reduce
RS: Reduce-Scatter
AG: All-Gather

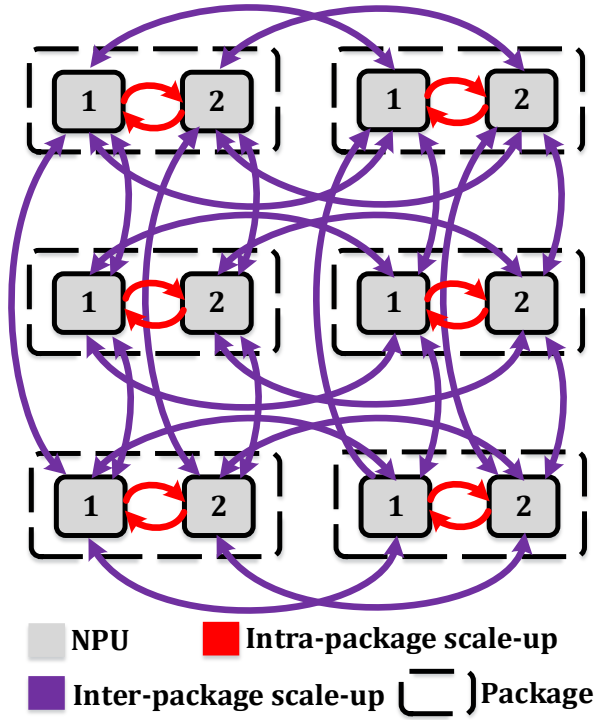


Collective Scheduler

- There are one/multiple queue(s) per each physical network dimension.
- A collective is broken into multiple chunks and inserted into the first queue.
- Queues process chunks in-order.

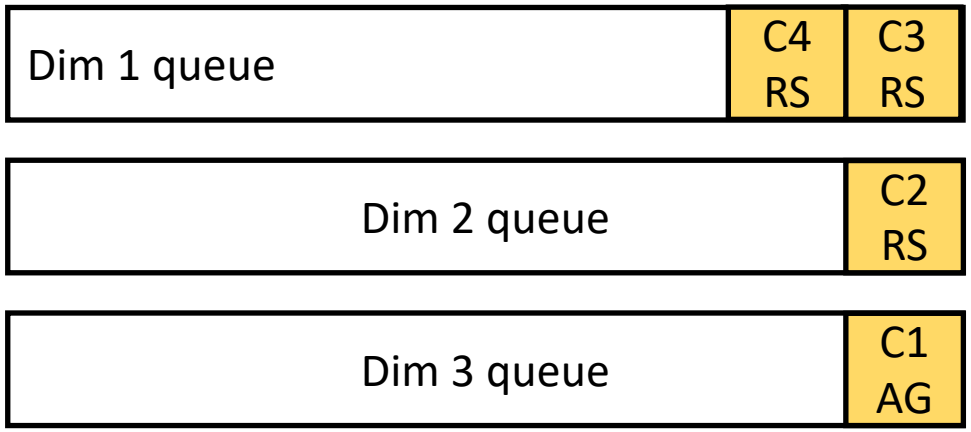


AR: All-Reduce
RS: Reduce-Scatter
AG: All-Gather

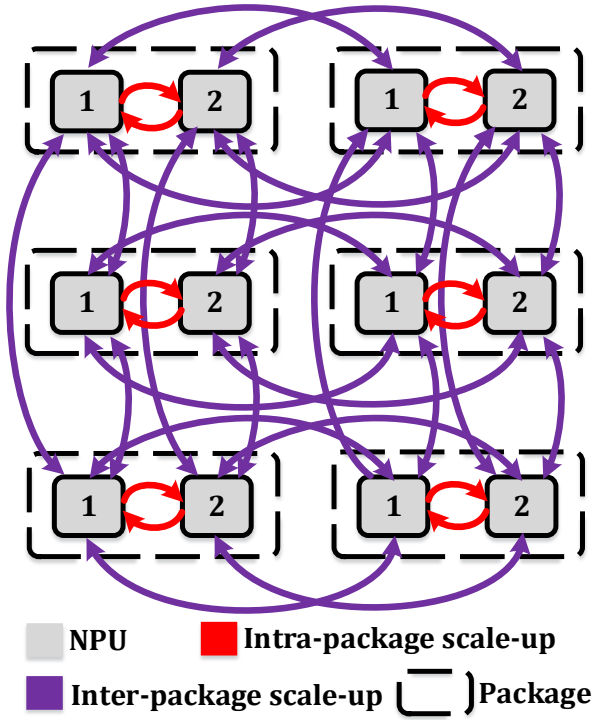


Collective Scheduler

- There are one/multiple queue(s) per each physical network dimension.
- A collective is broken into multiple chunks and inserted into the first queue.
- Queues process chunks in-order.

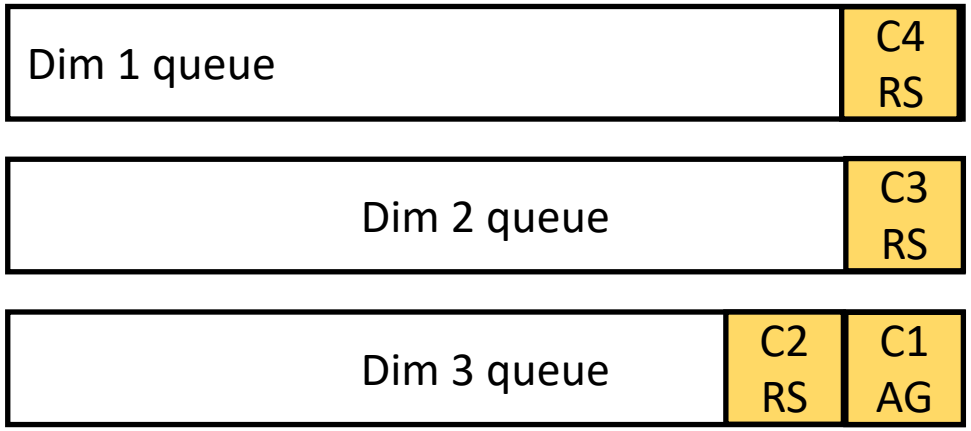


AR: All-Reduce
RS: Reduce-Scatter
AG: All-Gather

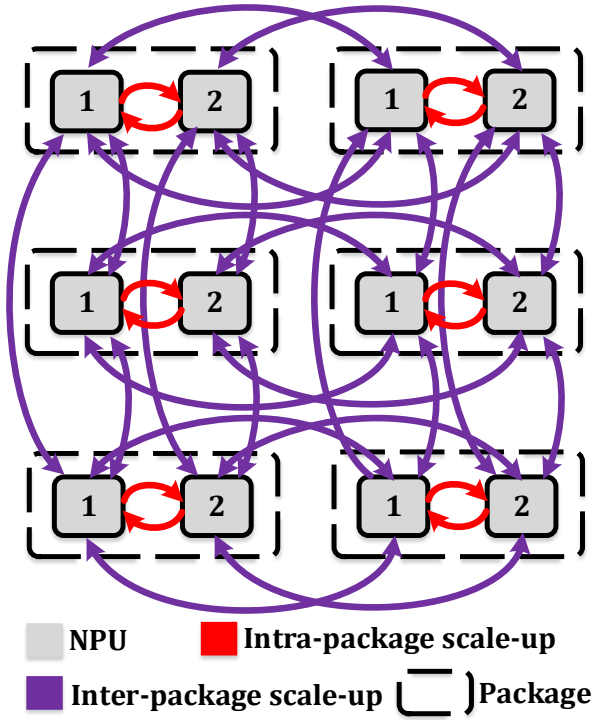


Collective Scheduler

- There are one/multiple queue(s) per each physical network dimension.
- A collective is broken into multiple chunks and inserted into the first queue.
- Queues process chunks in-order.

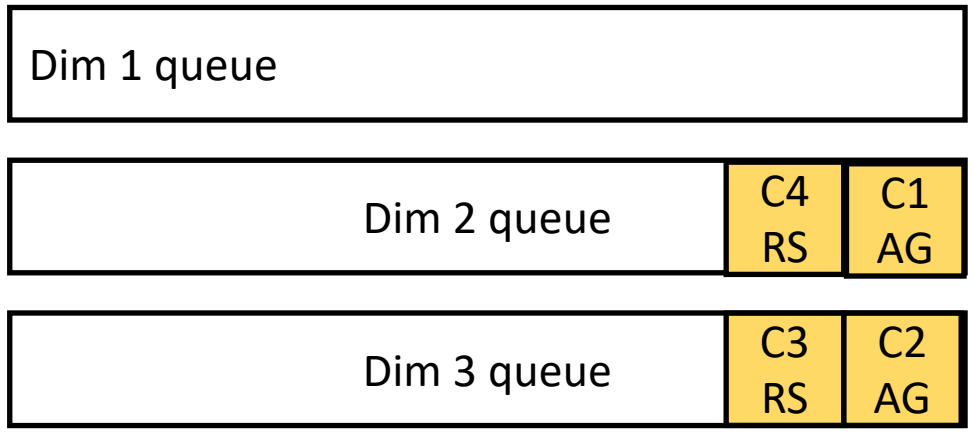


AR: All-Reduce
RS: Reduce-Scatter
AG: All-Gather

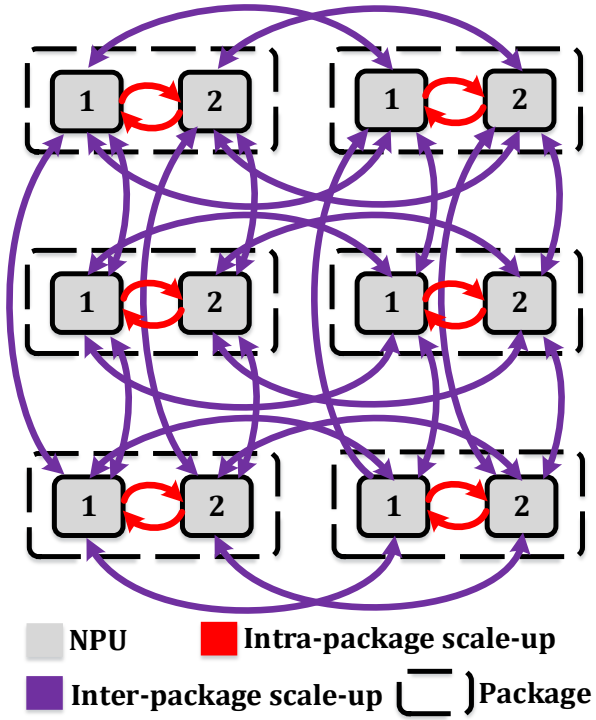


Collective Scheduler

- There are one/multiple queue(s) per each physical network dimension.
- A collective is broken into multiple chunks and inserted into the first queue.
- Queues process chunks in-order.

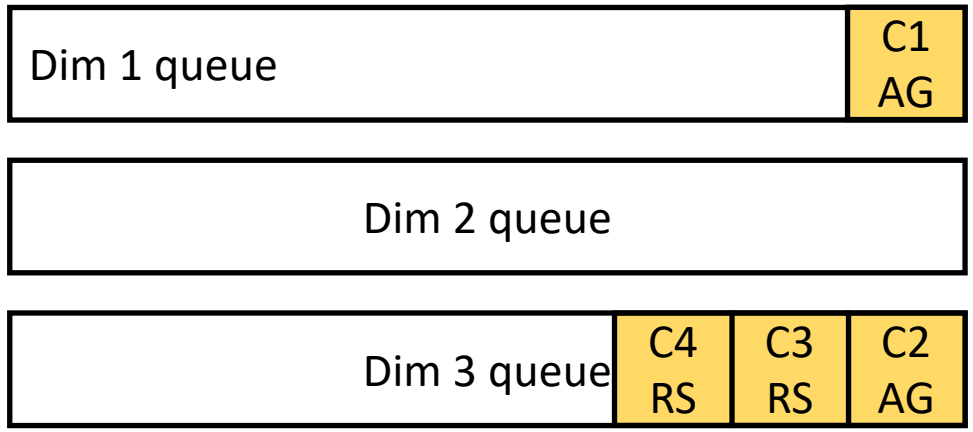


AR: All-Reduce
RS: Reduce-Scatter
AG: All-Gather

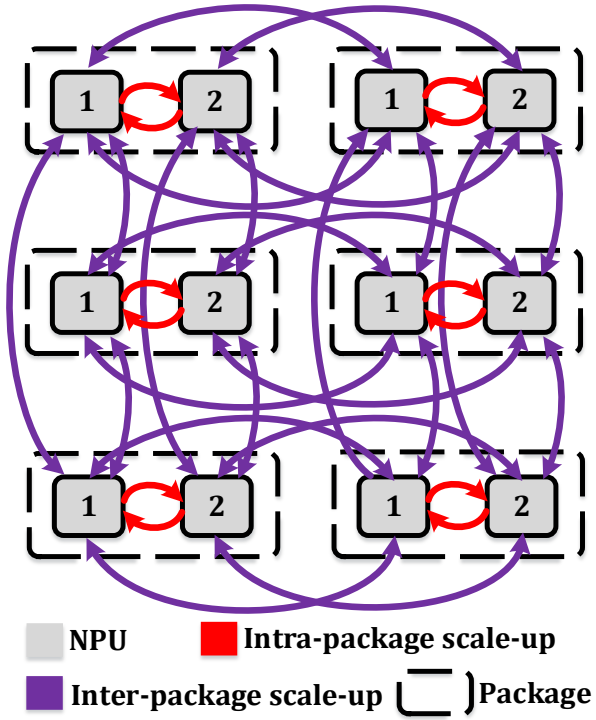


Collective Scheduler

- There are one/multiple queue(s) per each physical network dimension.
- A collective is broken into multiple chunks and inserted into the first queue.
- Queues process chunks in-order.

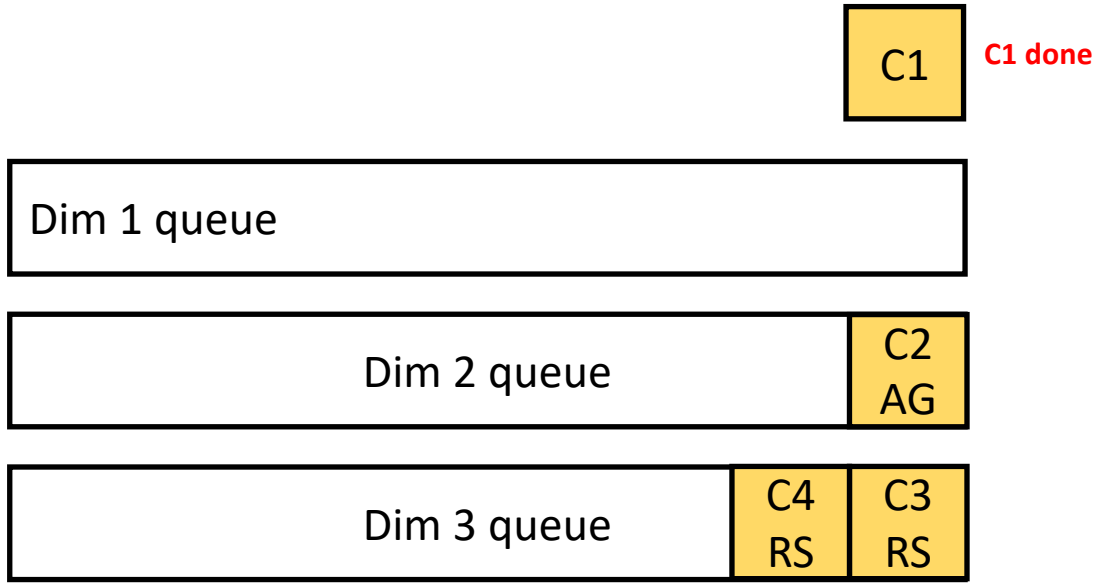


AR: All-Reduce
RS: Reduce-Scatter
AG: All-Gather

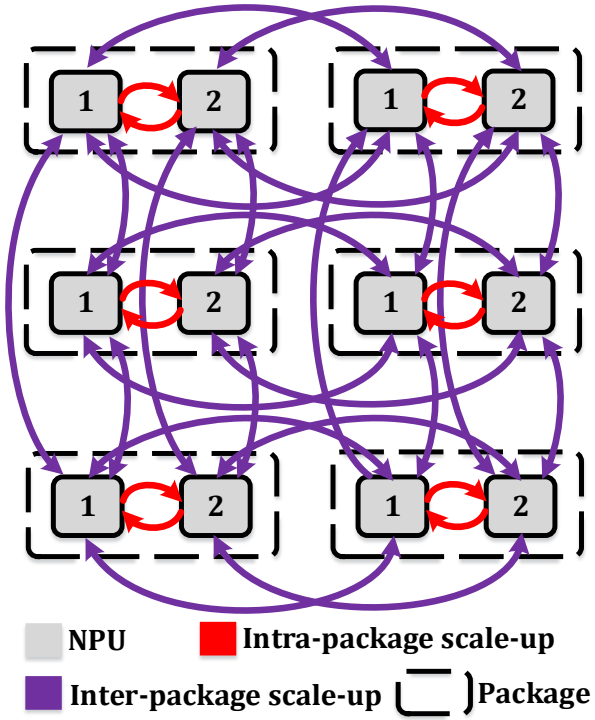


Collective Scheduler

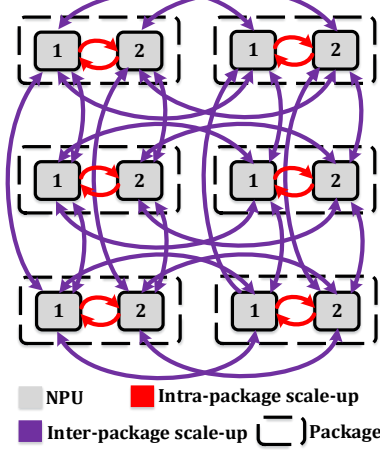
- There are one/multiple queue(s) per each physical network dimension.
- A collective is broken into multiple chunks and inserted into the first queue.
- Queues process chunks in-order.



AR: All-Reduce
RS: Reduce-Scatter
AG: All-Gather



System Input

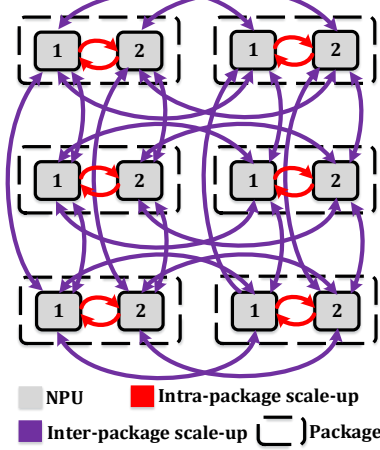


Collective Policy for

sample_torus_sys.txt

```
scheduling-policy: LIFO
endpoint-delay: 1
active-chunks-per-dimension: 1
preferred-dataset-splits: 4
boost-mode: 0
all-reduce-implementation: ring_ring_ring
all-gather-implementation: ring_ring_ring
reduce-scatter-implementation: ring_ring_ring
all-to-all-implementation: ring_ring_ring
collective-optimization: localBWAware
```

System Input

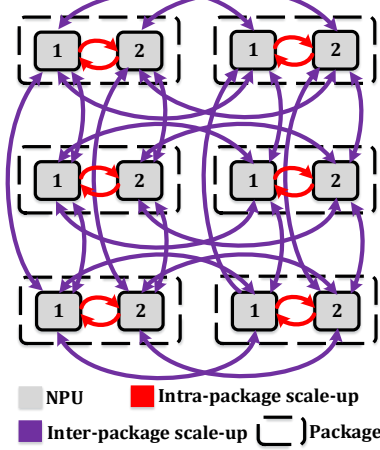


Constant delay before NPU sending a message

sample_torus_sys.txt

```
1  scheduling-policy: LIFO
2  endpoint-delay: 1
3  active-chunks-per-dimension: 1
4  preferred-dataset-splits: 4
5  boost-mode: 0
6  all-reduce-implementation: ring_ring_ring
7  all-gather-implementation: ring_ring_ring
8  reduce-scatter-implementation: ring_ring_ring
9  all-to-all-implementation: ring_ring_ring
10 collective-optimization: localBWAware
11
```

System Input

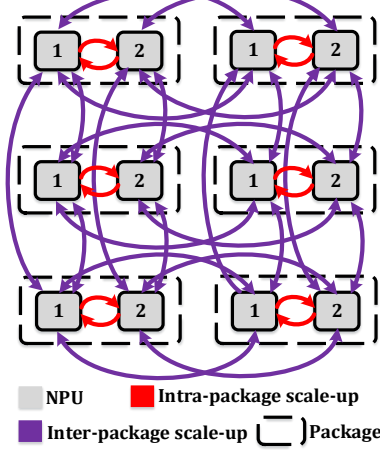


Max running chunks per each physical network dimension

sample_torus_sys.txt

```
1  scheduling-policy: LIFO
2  endpoint-delay: 1
3  active-chunks-per-dimension: 1
4  preferred-dataset-splits: 4
5  boost-mode: 0
6  all-reduce-implementation: ring_ring_ring
7  all-gather-implementation: ring_ring_ring
8  reduce-scatter-implementation: ring_ring_ring
9  all-to-all-implementation: ring_ring_ring
10 collective-optimization: localBWAware
11
```

System Input

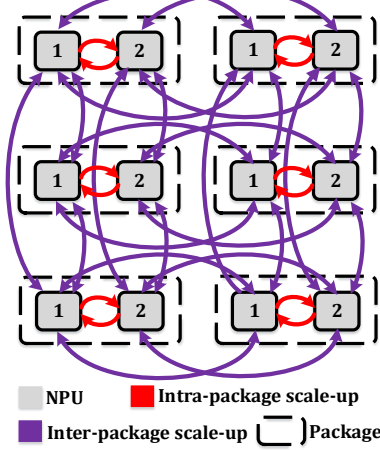


# of chunks to split each collective into

sample_torus_sys.txt

```
1  scheduling-policy: LIFO
2  endpoint-delay: 1
3  active-chunks-per-dimension: 1
4  preferred-dataset-splits: 4
5  boost-mode: 0
6  all-reduce-implementation: ring_ring_ring
7  all-gather-implementation: ring_ring_ring
8  reduce-scatter-implementation: ring_ring_ring
9  all-to-all-implementation: ring_ring_ring
10 collective-optimization: localBWAware
11
```

System Input

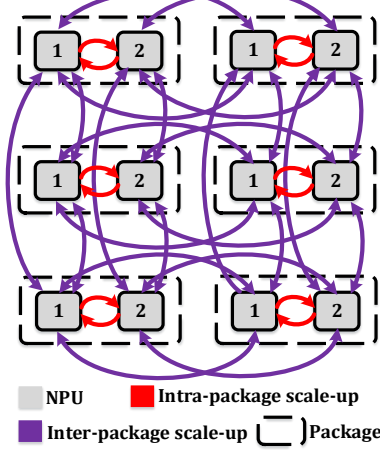


Speed-up the simulation

sample_torus_sys.txt

```
1  scheduling-policy: LIFO
2  endpoint-delay: 1
3  active-chunks-per-dimension: 1
4  preferred-dataset-splits: 4
5  boost-mode: 0
6  all-reduce-implementation: ring_ring_ring
7  all-gather-implementation: ring_ring_ring
8  reduce-scatter-implementation: ring_ring_ring
9  all-to-all-implementation: ring_ring_ring
10 collective-optimization: localBWAware
11
```

System Input



Hierarchical collective algorithm implementation

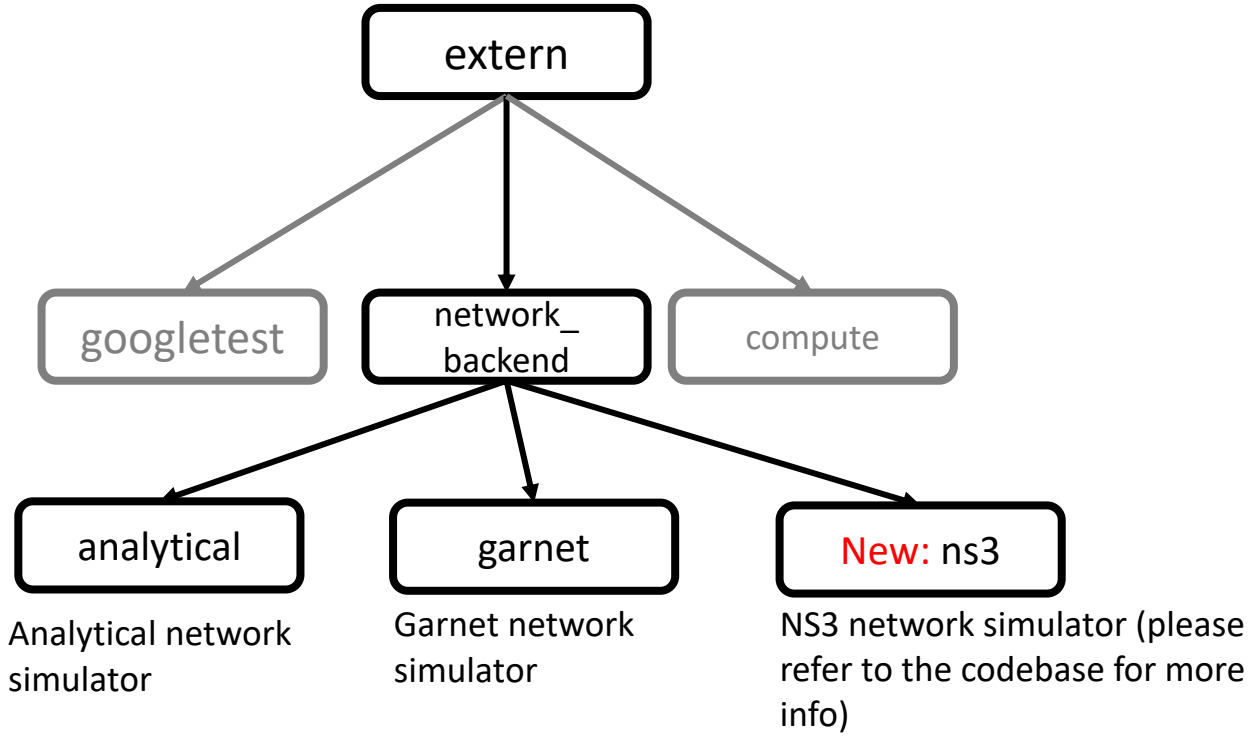
sample_torus_sys.txt

```
1  scheduling-policy: LIFO
2  endpoint-delay: 1
3  active-chunks-per-dimension: 1
4  preferred-dataset-splits: 4
5  boost-mode: 0
6  all-reduce-implementation: ring_ring_ring
7  all-gather-implementation: ring_ring_ring
8  reduce-scatter-implementation: ring_ring_ring
9  all-to-all-implementation: ring_ring_ring
10 collective-optimization: localBWAware
11
```

Network Layer

Network Layer Structure

- Network backends are maintained separately and are imported as **submodule**.



Simulation Control Flow

- The main file is implemented inside network layer.
- Network layer creates corresponding System and NetworkAPI instances.
- Each system layer instance **internally** creates its workload layer instance.

Where do instantiations happen?
Analytical backend: analytical/src/main.cc
Garnet backend: garnet/gem5_astra/src/mem/ruby/network/garnet2.0/NetworkInterface.cc

Network Layer

```
void main(){
        ⋮
Instantiate NetworkAPI[];
Instantiate System[];
        ⋮
for(auto &s:system){
    s->workload.fire();
}
process_all_events();
return;
}
```

Garnet vs. Analytical

- Regular topologies + topology-aware collectives make traffic patterns to be congestionless (in most cases), enabling analytical backend to calculate latencies fast and accurately.

Analytical	Garnet
Supports hierarchical topologies	Supports switch-based and torus-based topologies
Each level in hierarchy can be switch, ring, FC	Supports credit-based flow control
Uses simple link latency, BW analytical model to get latency.	Performs packetization, flow control, congestion modeling, etc.
Fast	Slow for large systems & big models
Accurate when comm patterns are congestionless.	Accurate in all scenarios.

Network Input

Garnet network input

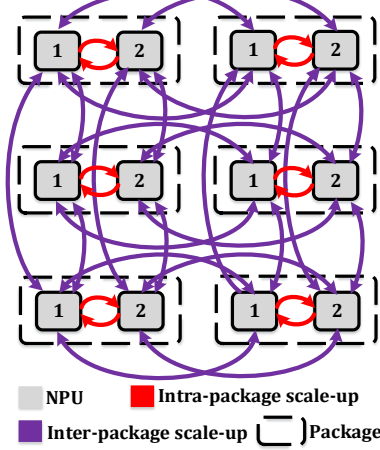
sample_torus.txt

```
num-npus: 12
num-packages: 6
package-rows: 3
topology: Torus3D
local-rings: 2
vertical-rings: 1
horizontal-rings: 1
flit-width: 2048
local-packet-size: 4096
package-packet-size: 4096
tile-link-width: 256
package-link-width: 256
vcs-per-vnet: 50
routing-algorithm: Ring_XY
router-latency: 1
local-link-latency: 90
package-link-latency: 200
buffers-per-vc: 5000
local-link-efficiency: 1.0
package-link-efficiency: 1.0
```

Analytical network input

sample_Torus3D.json

```
{
  "topology-name": "Hierarchical",
  "topologies-per-dim": ["Ring", "Ring", "Ring"],
  "dimension-type": ["N", "N", "N"],
  "dimensions-count": 3,
  "units-count": [2, 2, 3],
  "links-count": [2, 2, 2],
  "link-latency": [10, 100, 100],
  "link-bandwidth": [32, 16, 16],
  "nic-latency": [0, 0, 0],
  "router-latency": [0, 0, 0],
  "hbm-latency": [500, 500, 500],
  "hbm-bandwidth": [370, 370, 370],
  "hbm-scale": [0, 0, 0]
}
```



ASTRA-SIM Run Script

A Sample Run Script

run_DLRM_analytical.sh

```
#! /bin/bash -v

# Absolue path to this script
SCRIPT_DIR=$(dirname "$(realpath $0)")

# Absolute paths to useful directories
BINARY="${SCRIPT_DIR:?}"/../build/astra_analytical/build/AnalyticalAstra/bin/AnalyticalAstra
NETWORK="${SCRIPT_DIR:?}"/../inputs/network/analytical/sample_Torus3D.json
SYSTEM="${SCRIPT_DIR:?}"/../inputs/system/sample_torus_sys.txt
WORKLOAD="${SCRIPT_DIR:?}"/../inputs/workload/DLRM_HybridParallel.txt
STATS="${SCRIPT_DIR:?}"/results/run_DLRM_analytical

rm -rf "${STATS}"
mkdir "${STATS}"

"${BINARY}" \
--network-configuration="${NETWORK}" \
--system-configuration="${SYSTEM}" \
--workload-configuration="${WORKLOAD}" \
--path="${STATS}/" \
--run-name="sample_DLRM" \
--num-passes=2 \
--total-stat-rows=1 \
--stat-row=0

```

A Sample Run Script

run_DLRM_analytical.sh

```
#! /bin/bash -v

# Absolue path to this script
SCRIPT_DIR=$(dirname "$(realpath $0)")

# Absolute paths to useful directories
BINARY="${SCRIPT_DIR:?}"/../build/astra_analytical/build/AnalyticalAstra/bin/AnalyticalAstra
NETWORK="${SCRIPT_DIR:?}"/../inputs/network/analytical/sample_Torus3D.json
SYSTEM="${SCRIPT_DIR:?}"/../inputs/system/sample_torus_sys.txt
WORKLOAD="${SCRIPT_DIR:?}"/../inputs/workload/DLRM_HybridParallel.txt
STATS="${SCRIPT_DIR:?}"/results/run_DLRM_analytical

rm -rf "${STATS}"
mkdir "${STATS}"

"${BINARY}" \
--network-configuration="${NETWORK}" \
--system-configuration="${SYSTEM}" \
--workload-configuration="${WORKLOAD}" \
--path="${STATS}/" \
--run-name="sample_DLRM" \
--num-passes=2 \
--total-stat-rows=1 \
--stat-row=0
```

Runtime parameters

ASTRA-SIM Reports

Overall Results

- Endtoend.csv.

Layer name

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			fwd compute	wg compute	ig compute	fwd exposed	wg exposed c	ig exposed cc	fwd total con	wg total com	ig total comn	workload fini	total comp	total exposed	comm
2	conv1	sample_Resnet	26.006	64.582	0	0	17.364	0	0	17.366	0	4875.201	2164.9	2710.301	
3	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	4.796	0	4875.201			
4	layer_64_1_c	sample_Resnet	6.912	14.976	6.912	0	0	0	0	27.08	0	4875.201			
5	layer_64_1_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	47.906	0	4875.201			
6	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	13.648	0	4875.201			
7	layer_64_2_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	10.166	0	4875.201			
8	layer_64_2_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	20.102	0	4875.201			
9	layer_64_2_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	22.048	0	4875.201			
10	layer_64_3_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	30.082	0	4875.201			
11	layer_64_3_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	11.334	0	4875.201			
12	layer_64_3_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	7.526	0	4875.201			
13	layer_128_1_	sample_Resnet	5.184	12.288	5.184	0	0	0	0	36.03	0	4875.201			
14	layer_128_1_	sample_Resnet	7.296	19.968	7.04	0	0	0	0	9.08	0	4875.201			
15	layer_128_1_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	28.13	0	4875.201			
16	layer_128_1_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	13.272	0	4875.201			
17	layer_128_2_	sample_Resnet	5.184	8.192	4.672	0	0	0	0	30.868	0	4875.201			
18	layer_128_2_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	33.952	0	4875.201			
19	layer_128_2_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	101.056	0	4875.201			

Overall Results

- Endtoend.csv.

Run name

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			fwd compute	wg compute	ig compute	fwd exposed	wg exposed c	ig exposed cc	fwd total com	wg total com	ig total comn	workload fini	total comp	total exposed	comm
2	conv1	sample_Resnet	26.006	64.582	0	0	17.364	0	0	17.366	0	4875.201	2164.9	2710.301	
3	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	4.796	0	4875.201			
4	layer_64_1_c	sample_Resnet	6.912	14.976	6.912	0	0	0	0	27.08	0	4875.201			
5	layer_64_1_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	47.906	0	4875.201			
6	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	13.648	0	4875.201			
7	layer_64_2_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	10.166	0	4875.201			
8	layer_64_2_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	20.102	0	4875.201			
9	layer_64_2_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	22.048	0	4875.201			
10	layer_64_3_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	30.082	0	4875.201			
11	layer_64_3_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	11.334	0	4875.201			
12	layer_64_3_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	7.526	0	4875.201			
13	layer_128_1_	sample_Resnet	5.184	12.288	5.184	0	0	0	0	36.03	0	4875.201			
14	layer_128_1_	sample_Resnet	7.296	19.968	7.04	0	0	0	0	9.08	0	4875.201			
15	layer_128_1_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	28.13	0	4875.201			
16	layer_128_1_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	13.272	0	4875.201			
17	layer_128_2_	sample_Resnet	5.184	8.192	4.672	0	0	0	0	30.868	0	4875.201			
18	layer_128_2_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	33.952	0	4875.201			
19	layer_128_2_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	101.056	0	4875.201			

Overall Results

- Endtoend.csv.

Compute times (us)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			fwd compute	wg compute	ig compute	fwd exposed	wg exposed c	ig exposed cc	fwd total con	wg total com	ig total comn	workload fini	total comp	total exposed	comm
2	conv1	sample_Resnet	26.006	64.582	0	0	17.364	0	0	17.366	0	4875.201	2164.9	2710.301	
3	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	4.796	0	4875.201			
4	layer_64_1_c	sample_Resnet	6.912	14.976	6.912	0	0	0	0	27.08	0	4875.201			
5	layer_64_1_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	47.906	0	4875.201			
6	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	13.648	0	4875.201			
7	layer_64_2_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	10.166	0	4875.201			
8	layer_64_2_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	20.102	0	4875.201			
9	layer_64_2_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	22.048	0	4875.201			
10	layer_64_3_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	30.082	0	4875.201			
11	layer_64_3_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	11.334	0	4875.201			
12	layer_64_3_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	7.526	0	4875.201			
13	layer_128_1_	sample_Resnet	5.184	12.288	5.184	0	0	0	0	36.03	0	4875.201			
14	layer_128_1_	sample_Resnet	7.296	19.968	7.04	0	0	0	0	9.08	0	4875.201			
15	layer_128_1_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	28.13	0	4875.201			
16	layer_128_1_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	13.272	0	4875.201			
17	layer_128_2_	sample_Resnet	5.184	8.192	4.672	0	0	0	0	30.868	0	4875.201			
18	layer_128_2_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	33.952	0	4875.201			
19	layer_128_2_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	101.056	0	4875.201			

Overall Results

- Endtoend.csv.

Raw communication times (us)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			fwd compute	wg compute	ig compute	fwd exposed	wg exposed c	ig exposed cc	fwd total com	wg total com	ig total comn	workload fini	total comp	total exposed	comm
2	conv1	sample_Resnet	26.006	64.582	0	0	17.364	0	0	17.366	0	4875.201	2164.9	2710.301	
3	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	4.796	0	4875.201			
4	layer_64_1_c	sample_Resnet	6.912	14.976	6.912	0	0	0	0	27.08	0	4875.201			
5	layer_64_1_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	47.906	0	4875.201			
6	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	13.648	0	4875.201			
7	layer_64_2_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	10.166	0	4875.201			
8	layer_64_2_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	20.102	0	4875.201			
9	layer_64_2_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	22.048	0	4875.201			
10	layer_64_3_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	30.082	0	4875.201			
11	layer_64_3_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	11.334	0	4875.201			
12	layer_64_3_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	7.526	0	4875.201			
13	layer_128_1_	sample_Resnet	5.184	12.288	5.184	0	0	0	0	36.03	0	4875.201			
14	layer_128_1_	sample_Resnet	7.296	19.968	7.04	0	0	0	0	9.08	0	4875.201			
15	layer_128_1_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	28.13	0	4875.201			
16	layer_128_1_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	13.272	0	4875.201			
17	layer_128_2_	sample_Resnet	5.184	8.192	4.672	0	0	0	0	30.868	0	4875.201			
18	layer_128_2_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	33.952	0	4875.201			
19	layer_128_2_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	101.056	0	4875.201			

Overall Results

- Endtoend.csv.

Exposed communication times (us)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			fwd compute	wg compute	ig compute	fwd exposed	wg exposed c	ig exposed cc	fwd total com	wg total com	ig total comn	workload fini	total comp	total exposed comm	
2	conv1	sample_Resnet	26.006	64.582	0	0	17.364	0	0	17.366	0	4875.201	2164.9	2710.301	
3	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	4.796	0	4875.201			
4	layer_64_1_c	sample_Resnet	6.912	14.976	6.912	0	0	0	0	27.08	0	4875.201			
5	layer_64_1_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	47.906	0	4875.201			
6	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	13.648	0	4875.201			
7	layer_64_2_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	10.166	0	4875.201			
8	layer_64_2_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	20.102	0	4875.201			
9	layer_64_2_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	22.048	0	4875.201			
10	layer_64_3_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	30.082	0	4875.201			
11	layer_64_3_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	11.334	0	4875.201			
12	layer_64_3_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	7.526	0	4875.201			
13	layer_128_1_	sample_Resnet	5.184	12.288	5.184	0	0	0	0	36.03	0	4875.201			
14	layer_128_1_	sample_Resnet	7.296	19.968	7.04	0	0	0	0	9.08	0	4875.201			
15	layer_128_1_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	28.13	0	4875.201			
16	layer_128_1_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	13.272	0	4875.201			
17	layer_128_2_	sample_Resnet	5.184	8.192	4.672	0	0	0	0	30.868	0	4875.201			
18	layer_128_2_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	33.952	0	4875.201			
19	layer_128_2_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	101.056	0	4875.201			

Overall Results

- Endtoend.csv.

Total compute & exposed communication times across all layers (us)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			fwd compute	wg compute	ig compute	fwd exposed	wg exposed c	ig exposed cc	fwd total con	wg total com	ig total comn	workload fini	total comp	total exposed comm	
2	conv1	sample_Resnet	26.006	64.582	0	0	17.364	0	0	17.366	0	4875.201	2164.9	2710.301	
3	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	4.796	0	4875.201			
4	layer_64_1_c	sample_Resnet	6.912	14.976	6.912	0	0	0	0	27.08	0	4875.201			
5	layer_64_1_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	47.906	0	4875.201			
6	layer_64_1_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	13.648	0	4875.201			
7	layer_64_2_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	10.166	0	4875.201			
8	layer_64_2_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	20.102	0	4875.201			
9	layer_64_2_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	22.048	0	4875.201			
10	layer_64_3_c	sample_Resnet	7.296	19.968	6.912	0	0	0	0	30.082	0	4875.201			
11	layer_64_3_c	sample_Resnet	21.888	28.288	20.736	0	0	0	0	11.334	0	4875.201			
12	layer_64_3_c	sample_Resnet	6.912	14.976	7.296	0	0	0	0	7.526	0	4875.201			
13	layer_128_1_	sample_Resnet	5.184	12.288	5.184	0	0	0	0	36.03	0	4875.201			
14	layer_128_1_	sample_Resnet	7.296	19.968	7.04	0	0	0	0	9.08	0	4875.201			
15	layer_128_1_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	28.13	0	4875.201			
16	layer_128_1_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	13.272	0	4875.201			
17	layer_128_2_	sample_Resnet	5.184	8.192	4.672	0	0	0	0	30.868	0	4875.201			
18	layer_128_2_	sample_Resnet	12.96	13.312	11.68	0	0	0	0	33.952	0	4875.201			
19	layer_128_2_	sample_Resnet	4.672	10.24	5.184	0	0	0	0	101.056	0	4875.201			

Overall Results

- Detailed.csv.

Average chunk queueing delay per each collective phase (us)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1			queuing d	queuing delay phase 1	queuing delay phase 2	queuing delay phase 3	queuing delay phase 4	queuing delay phase 5	network delay phase 1	network delay phase 2	network delay phase 3	network delay phase 4	network delay phase 5
2	conv1	sample_Resnet	0	1.8795	0.5455	0	0.137	2.3575	0.035	0.018	0.005	0.018	0.035
3	layer_64_1_	sample_Resnet	0	0.453	0.03925	0	0.02725	0.2425	0.06	0.03	0.008	0.03	0.06
4	layer_64_1_	sample_Resnet	0	2.9225	2.3655	0	0.065	7.1915	0.015	0.008	0.002	0.008	0.015
5	layer_64_1_	sample_Resnet	0	6.8005	2.5125	1.2645	3.2625	6.7425	0.135	0.068	0.017	0.068	0.135
6	layer_64_1_	sample_Resnet	0	2.013	0.412	0	0.227	2.536	0.06	0.03	0.008	0.03	0.06
7	layer_64_2_	sample_Resnet	0	2.889	0.03925	0	0.02725	0.2425	0.06	0.03	0.008	0.03	0.06
8	layer_64_2_	sample_Resnet	0	5.9785	0.08275	0	0.07075	0.5485	0.135	0.068	0.017	0.068	0.135
9	layer_64_2_	sample_Resnet	0	1.901	2.947	0	0.227	4.313	0.06	0.03	0.008	0.03	0.06
10	layer_64_3_	sample_Resnet	0	5.669	1.306	0	0.227	5.954	0.06	0.03	0.008	0.03	0.06
11	layer_64_3_	sample_Resnet	0	1.5945	0.08275	0	0.07075	0.5485	0.135	0.068	0.017	0.068	0.135
12	layer_64_3_	sample_Resnet	0	0.383	0.409	0	0.227	1.108	0.06	0.03	0.008	0.03	0.06
13	layer_128_1	sample_Resnet	0	6.6015	0.4635	0	0.262	3.0585	0.477	0.239	0.06	0.239	0.477
14	layer_128_1	sample_Resnet	0	1.176	0.07225	0	0.06325	0.4915	0.12	0.06	0.015	0.06	0.12
15	layer_128_1	sample_Resnet	0	2.9015	0.31	0	0.2935	2.2045	0.537	0.269	0.068	0.269	0.537
16	layer_128_1	sample_Resnet	0	1.4375	0.14125	0	0.12925	0.9775	0.239	0.12	0.03	0.12	0.239
17	layer_128_2	sample_Resnet	0	1.3395	2.3275	0	0.905	4.1295	0.239	0.12	0.03	0.12	0.239
18	layer_128_2	sample_Resnet	0	2.9285	0.37	0	0.487	3.393	0.537	0.269	0.068	0.269	0.537
19	layer_128_2	sample_Resnet	0	6.5115	0.8545	0.8115	8.02825	20.68175	0.239	0.12	0.03	0.12	0.239
20	layer_128_3	sample_Resnet	0	4.6365	0.42475	0.01275	0.549	16.7855	0.239	0.12	0.03	0.12	0.239

Overall Results

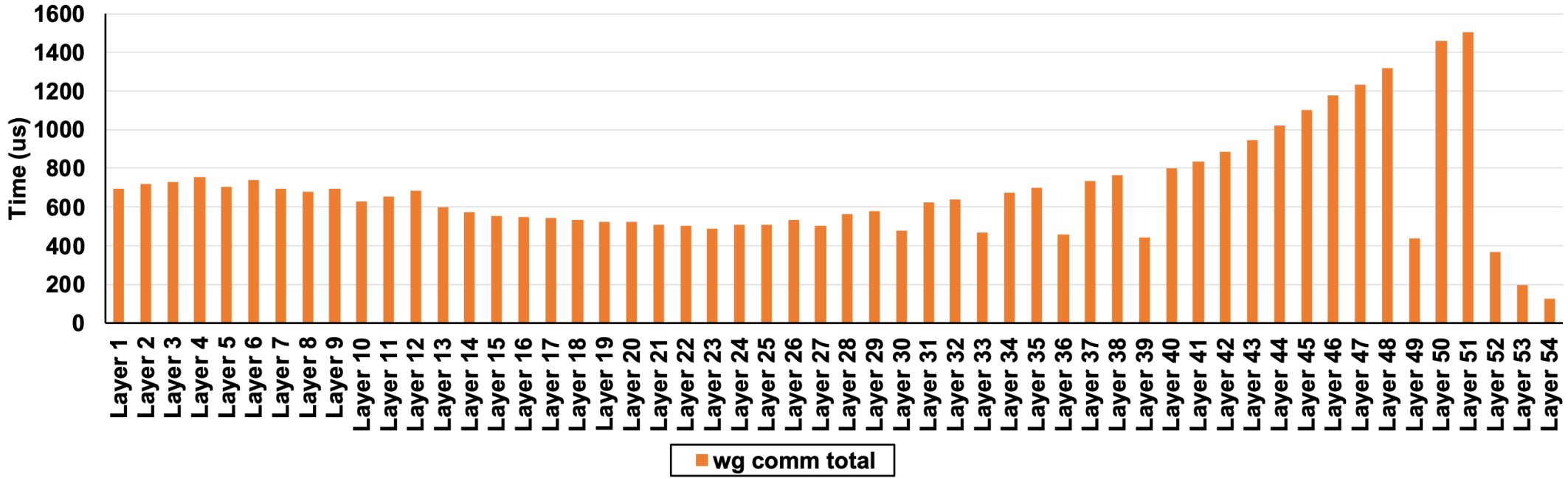
- Detailed.csv.

Average message latency per each collective phase (us)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1			queuing d	queuing delay phase 1	queuing delay phase 2	queuing delay phase 3	queuing delay phase 4	queuing delay phase 5	network delay phase 1	network delay phase 2	network delay phase 3	network delay phase 4	network delay phase 5
2	conv1	sample_Resnet	0	1.8795	0.5455	0	0.137	2.3575	0.035	0.018	0.005	0.018	0.035
3	layer_64_1_	sample_Resnet	0	0.453	0.03925	0	0.02725	0.2425	0.06	0.03	0.008	0.03	0.06
4	layer_64_1_	sample_Resnet	0	2.9225	2.3655	0	0.065	7.1915	0.015	0.008	0.002	0.008	0.015
5	layer_64_1_	sample_Resnet	0	6.8005	2.5125	1.2645	3.2625	6.7425	0.135	0.068	0.017	0.068	0.135
6	layer_64_1_	sample_Resnet	0	2.013	0.412	0	0.227	2.536	0.06	0.03	0.008	0.03	0.06
7	layer_64_2_	sample_Resnet	0	2.889	0.03925	0	0.02725	0.2425	0.06	0.03	0.008	0.03	0.06
8	layer_64_2_	sample_Resnet	0	5.9785	0.08275	0	0.07075	0.5485	0.135	0.068	0.017	0.068	0.135
9	layer_64_2_	sample_Resnet	0	1.901	2.947	0	0.227	4.313	0.06	0.03	0.008	0.03	0.06
10	layer_64_3_	sample_Resnet	0	5.669	1.306	0	0.227	5.954	0.06	0.03	0.008	0.03	0.06
11	layer_64_3_	sample_Resnet	0	1.5945	0.08275	0	0.07075	0.5485	0.135	0.068	0.017	0.068	0.135
12	layer_64_3_	sample_Resnet	0	0.383	0.409	0	0.227	1.108	0.06	0.03	0.008	0.03	0.06
13	layer_128_1	sample_Resnet	0	6.6015	0.4635	0	0.262	3.0585	0.477	0.239	0.06	0.239	0.477
14	layer_128_1	sample_Resnet	0	1.176	0.07225	0	0.06325	0.4915	0.12	0.06	0.015	0.06	0.12
15	layer_128_1	sample_Resnet	0	2.9015	0.31	0	0.2935	2.2045	0.537	0.269	0.068	0.269	0.537
16	layer_128_1	sample_Resnet	0	1.4375	0.14125	0	0.12925	0.9775	0.239	0.12	0.03	0.12	0.239
17	layer_128_2	sample_Resnet	0	1.3395	2.3275	0	0.905	4.1295	0.239	0.12	0.03	0.12	0.239
18	layer_128_2	sample_Resnet	0	2.9285	0.37	0	0.487	3.393	0.537	0.269	0.068	0.269	0.537
19	layer_128_2	sample_Resnet	0	6.5115	0.8545	0.8115	8.02825	20.68175	0.239	0.12	0.03	0.12	0.239
20	layer_128_3	sample_Resnet	0	4.6365	0.42475	0.01275	0.549	16.7855	0.239	0.12	0.03	0.12	0.239

ResNet-50 Layer-Wise Raw Comm Latency

- A Torus 3D with total of 32 (2X4X4) nodes is used.
- Data parallel approach is used.
- Raw latency depends on the comm size plus the priority of each layer comm (queuing delay).

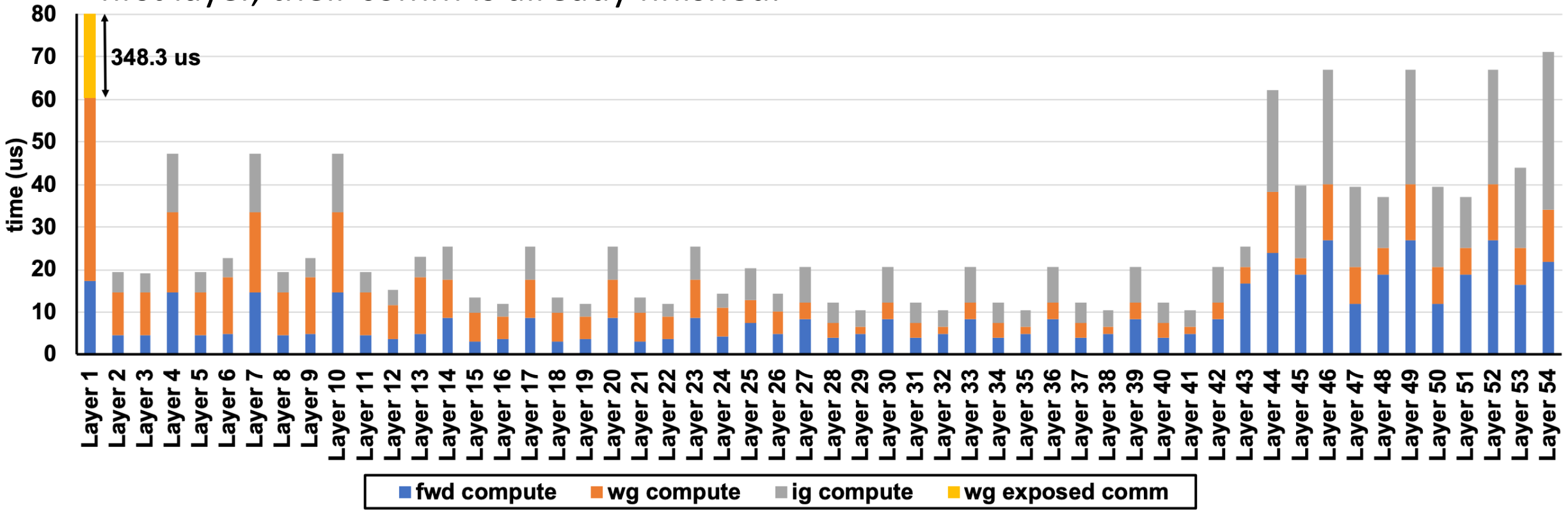


ResNet-50 Layer-Wise Compute vs. Exposed Comm Latency

- Exposed comm latency is observed for the first layer.

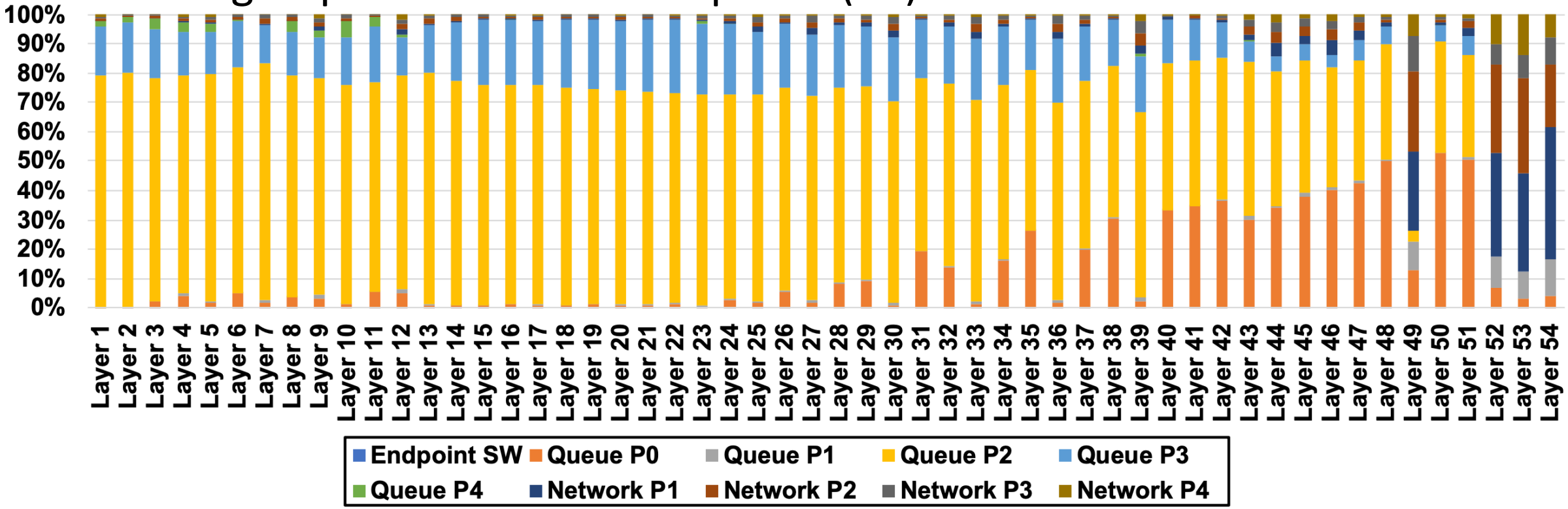
because by the time we reach other layers except that.

first layer, their comm is already finished.



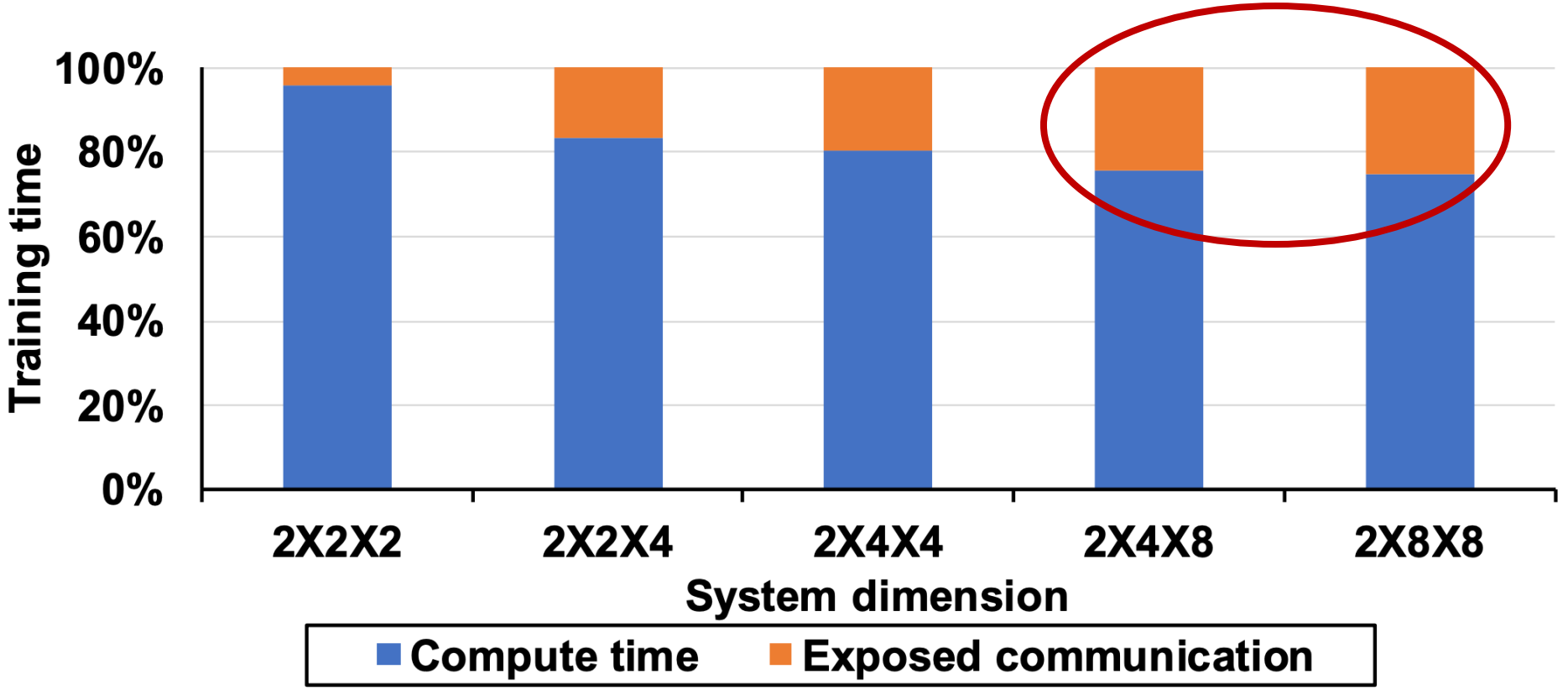
ResNet-50 Layer-Wise detailed latency

- Queue P2 is becoming the dominant factor due to very high speed of P1 (within package) that results most of the chunks get queued for the next phase (P2).



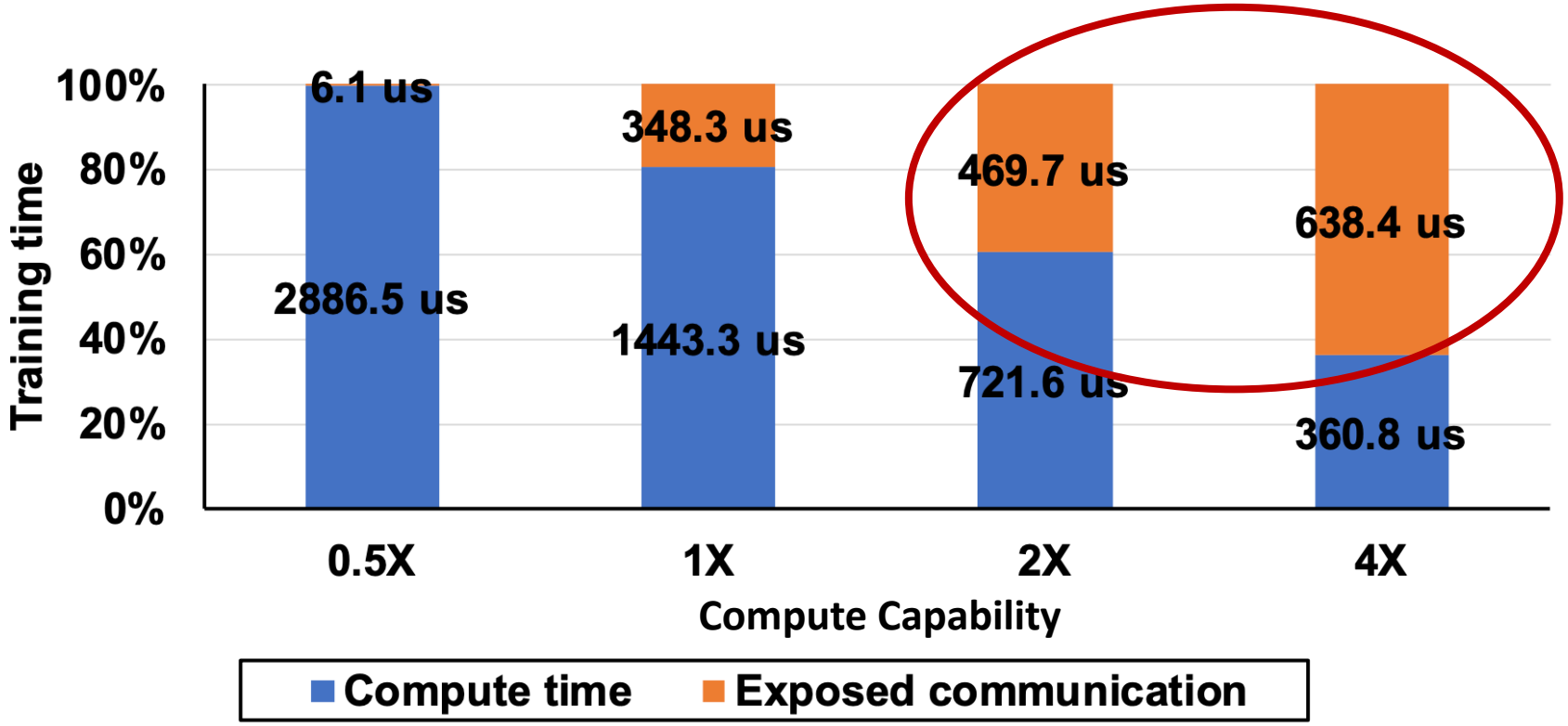
Effect of # of nodes on the Ratio of Total Compute vs Total Exposed Comm for ResNet-50

- A Torus 3D with total of 8, 16, 32, 64, 128 nodes are used.



Effect of Enhanced Compute Time per Node on the Ratio of Total Compute vs Total Exposed Comm for ResNet-50

- A Torus 3D with total of 32 nodes (2X4X4) is used.



Workload Generator

Workload Generator

- Should receive the GEMM operations (M, N, K dimensions) and the parallelization strategy as input.
- It uses SCALE-SIM simulator to find the compute times.
- Please see astra-sim/scripts/workload_generator/README.md.

Sample script to call workload generator

```
# For data-parallel
$ python3 gen_astrasim_workload_input.py \
  --datatype_size=2 \
  --mnk=mnk_inputs/example.csv \
  --num_npus=16 \
  --num_packages=2 \
  --output_file=../../inputs/workload/example_DATA.txt \
  --parallel=DATA \
  --run_name=example \
  --scalesim_config=../../extern/compute/SCALE-Sim/configs/google.cfg \
  --scalesim_path=../../extern/compute/SCALE-Sim
```

Sample MNK input file

	A	B	C	D	E
1	Layer	m	n	k	
2	MLP_Bottom	1024	128	512	
3	MLP_Bottom	1024	512	512	
4	MLP_Bottom	1024	512	512	
5	MLP_Bottom	1024	512	16	
6	MLP_Top_0	1024	1024	512	
7	MLP_Top_1	1024	512	512	
8	MLP_Top_2	1024	512	512	
9					

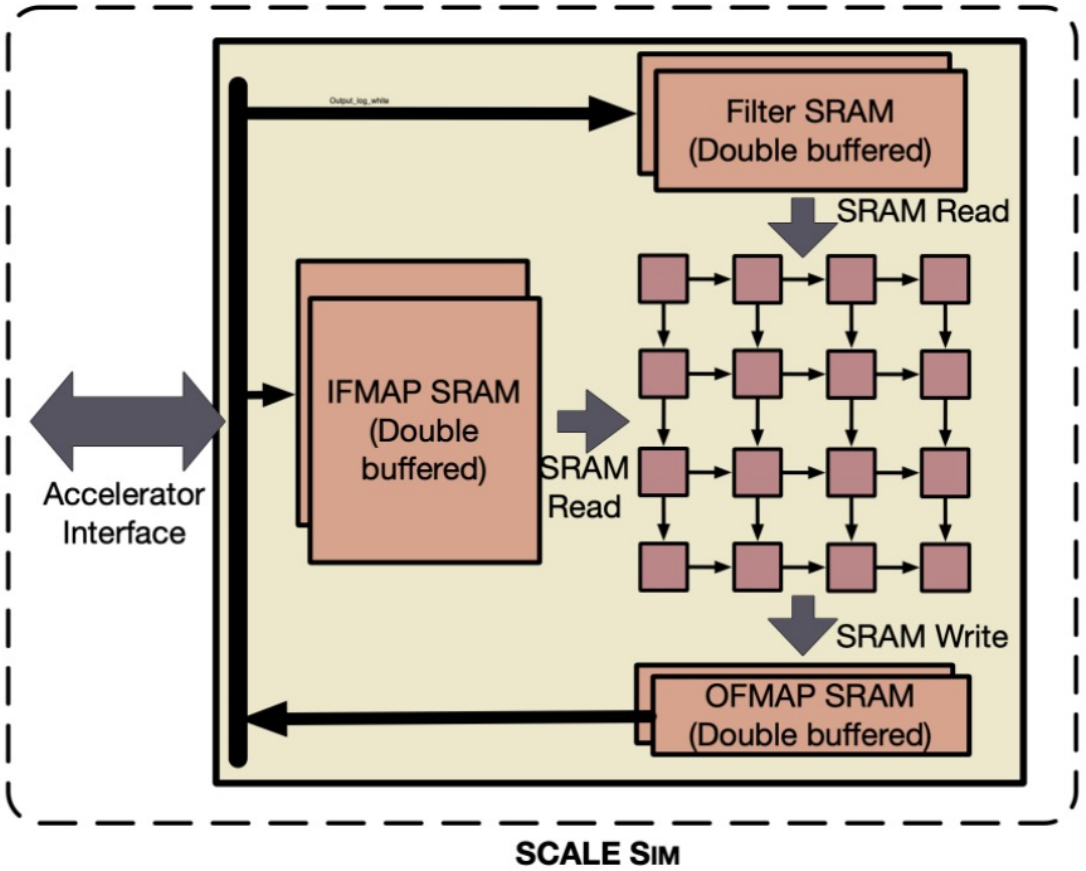
SCALE-SIM

- https://github.com/scalesim-project/scale-sim-v2

Parameter	Value
Array Height	32
Array Width	32
IFMAP SRAM Size	1024
Filter SRAM Size	1024
OFRAM SRAM Size	128
Dataflow	WS

Config file with architecture specs

Topology.csv



Cycle and IF BW

Traces
SRAM Read/Write
DRAM Read/Write

Thank you!

Agenda

Time (PDT)	Topic	Presenter
1:00 – 2:00	**Introduction to Distributed DL Training**	Tushar Krishna
2:00 – 2:20	**Challenges on Distributed Training Systems**	Srinivas Sridharan
2:20 – 3:30	**Introduction to ASTRA-sim simulator**	Saeed Rashidi
3:30 – 4:00	**Coffee Break**	
4:00 – 4:50	**Hands-on Exercises on Using ASTRA-sim**	William Won and Taekyung Heo
4:50 – 5:00	**Closing Remarks and Future Developments**	Taekyung Heo

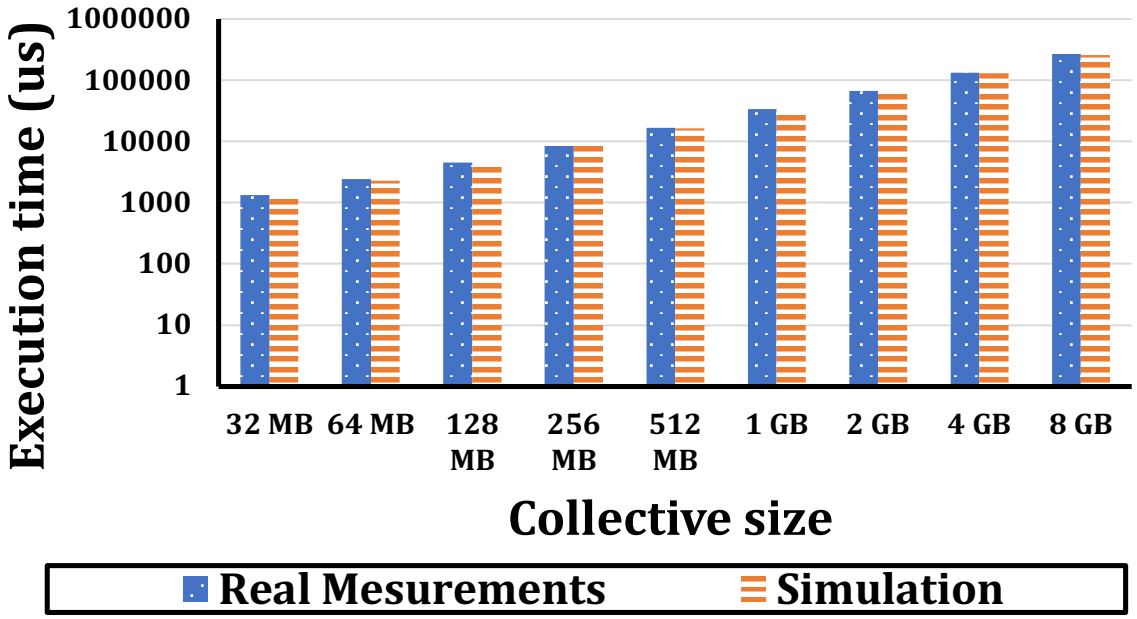
Tutorial Website

includes agenda, slides, ASTRA-sim installation instructions (via source + docker image)
https://astra-sim.github.io/tutorials/mlsys-2022

Attention: Tutorial is being recorded

ASTRA-SIM Validation

- 8 servers, each having 8 V100 GPUs.
- GPUs within a server are connected through NVSwitch (first gen).
- Each GPU has a dedicated 100 Gbps NIC, connecting it to the TOR switch.
- Below is the single All-Reduce performance comparison of real system measurements vs. ASTRA-SIM with analytical backend.



On average only ~4% difference is observed.