OptiReduce:
Resilient and Tail-Optimal AllReduce for Distributed Deep Learning in the Cloud

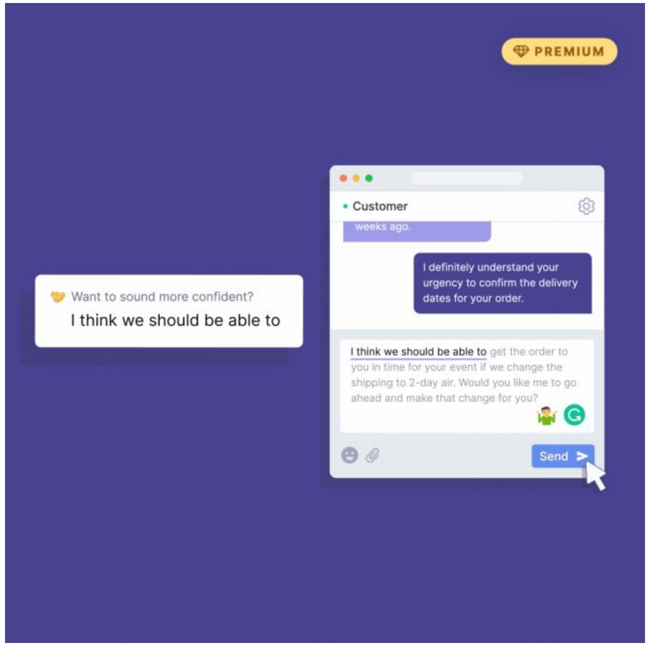
Ertza Warraich, Omer Shabtai, Khalid Manaa, Shay Vargaftik,
Yonatan Piasetzky, Matty Kadosh, Lalith Suresh, Muhammad Shahbaz

PURDUE UNIVERSITY · NVIDIA · vmware by Broadcom · feldera · UNIVERSITY OF MICHIGAN

DNNs are Everywhere

ChatGPT

Midjourney



Want to sound more confident?

I think we should be able to

weeks ago.

I definitely understand your urgency to confirm the delivery dates for your order.

I think we should be able to get the order to you in time for your event if we change the shipping to 2-day air. Would you like me to go ahead and make that change for you?

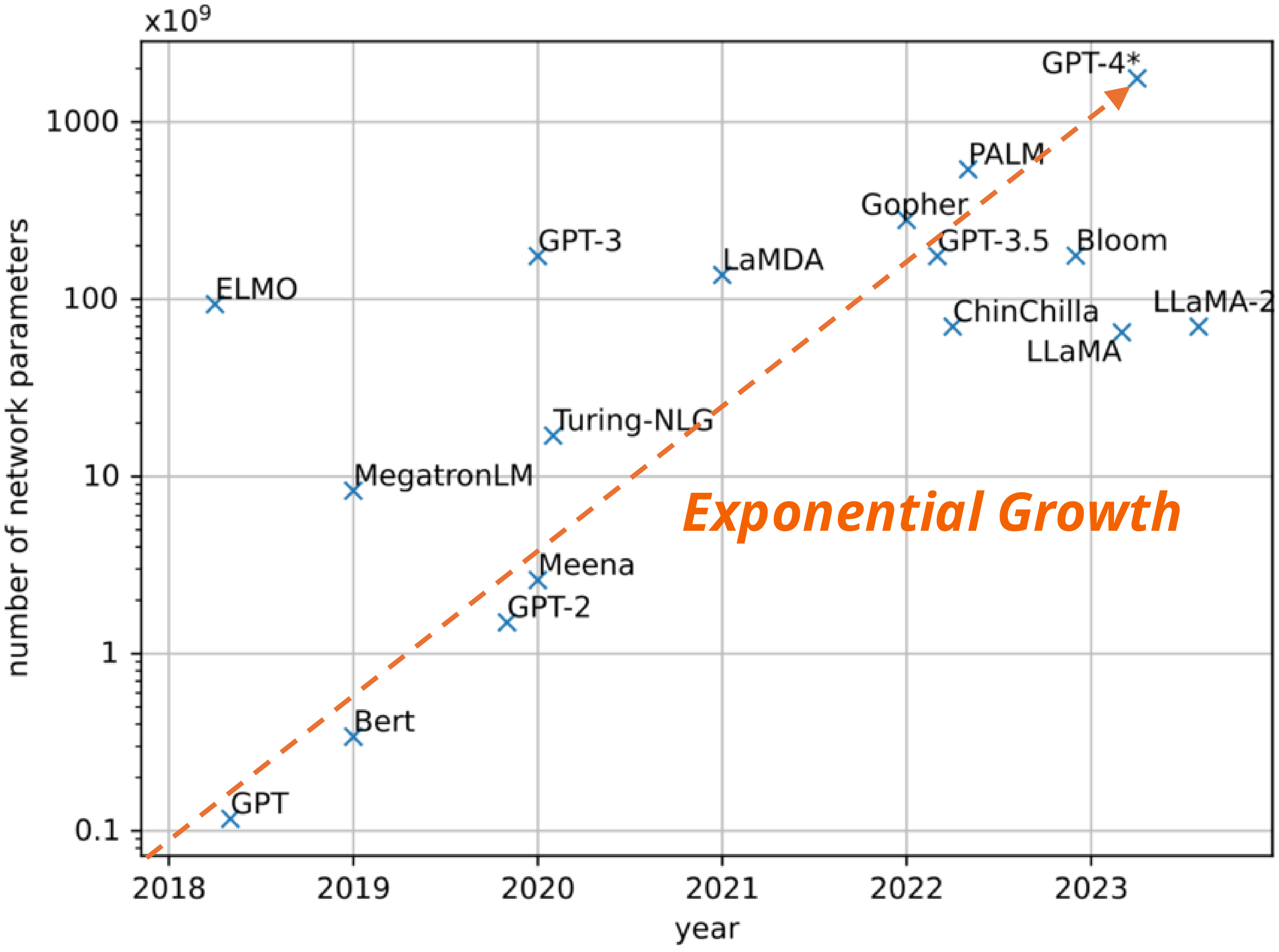
grammarly

Growth of DNNs & Distributed Training

1 Trillion

Log Scale

1 Billion



Exponential Growth

Takes years to train!!!

Distribute training over multiple GPUs: **Distributed Deep Learning (DDL)**

[1] Gerstmayr, Johannes et al. “Multibody Models Generated from Natural Language.” Multibody System Dynamics ‘24

Cloud for Distributed Training

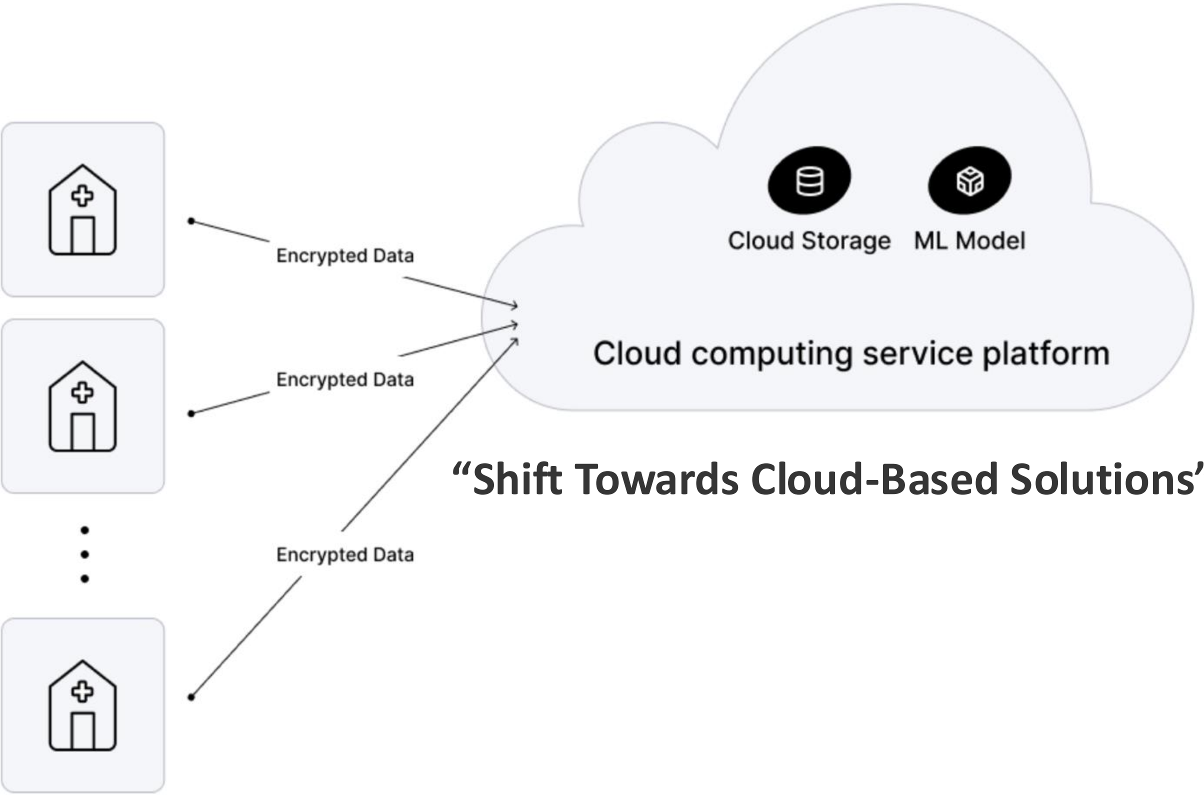
Symphony Solutions

Machine Learning in the Cloud: What Are the Benefits?

mentormate

March 19, 2024

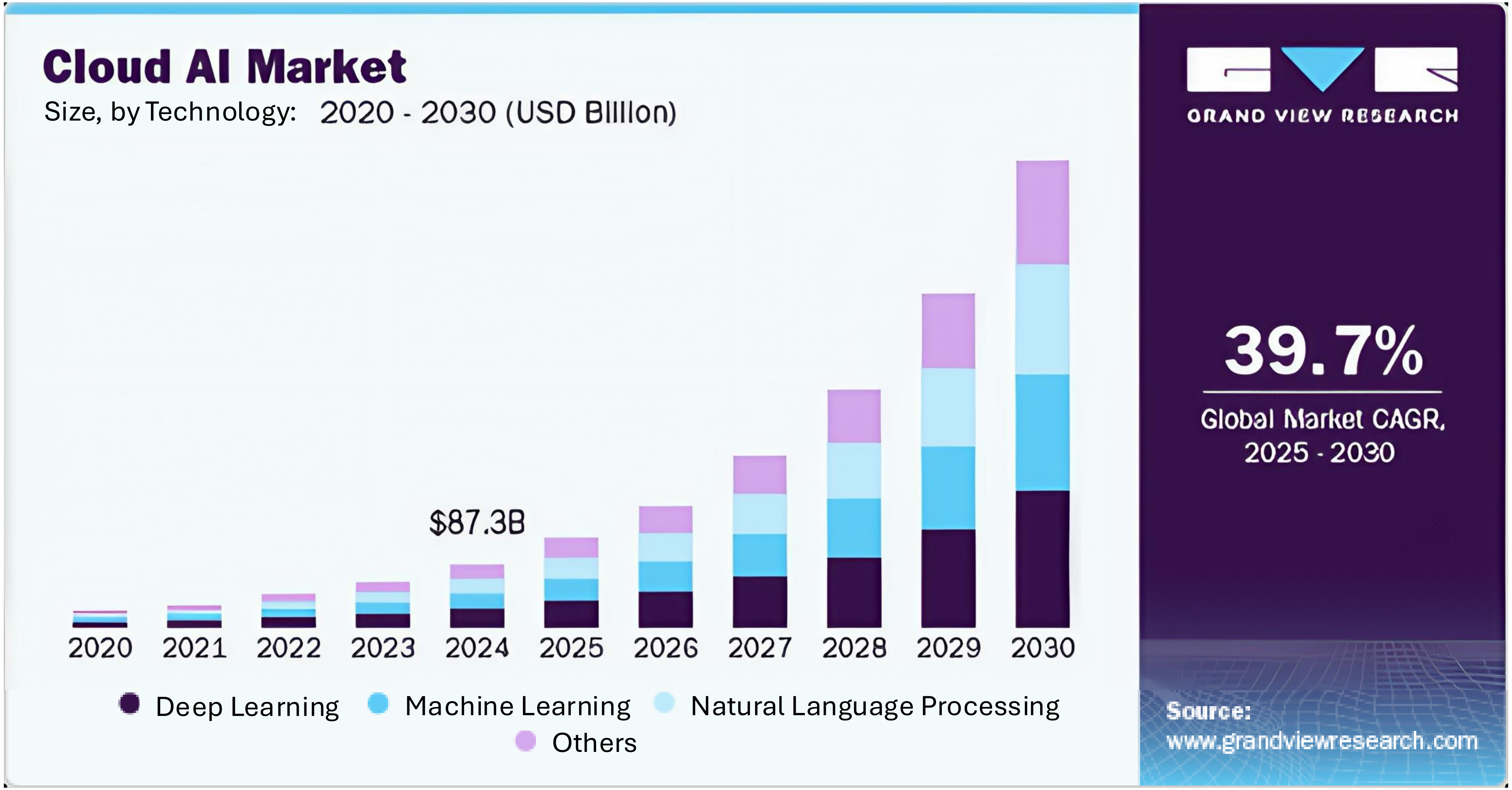
The Power of Machine Learning in the Cloud: Transforming Business Operations



“Shift Towards Cloud-Based Solutions”

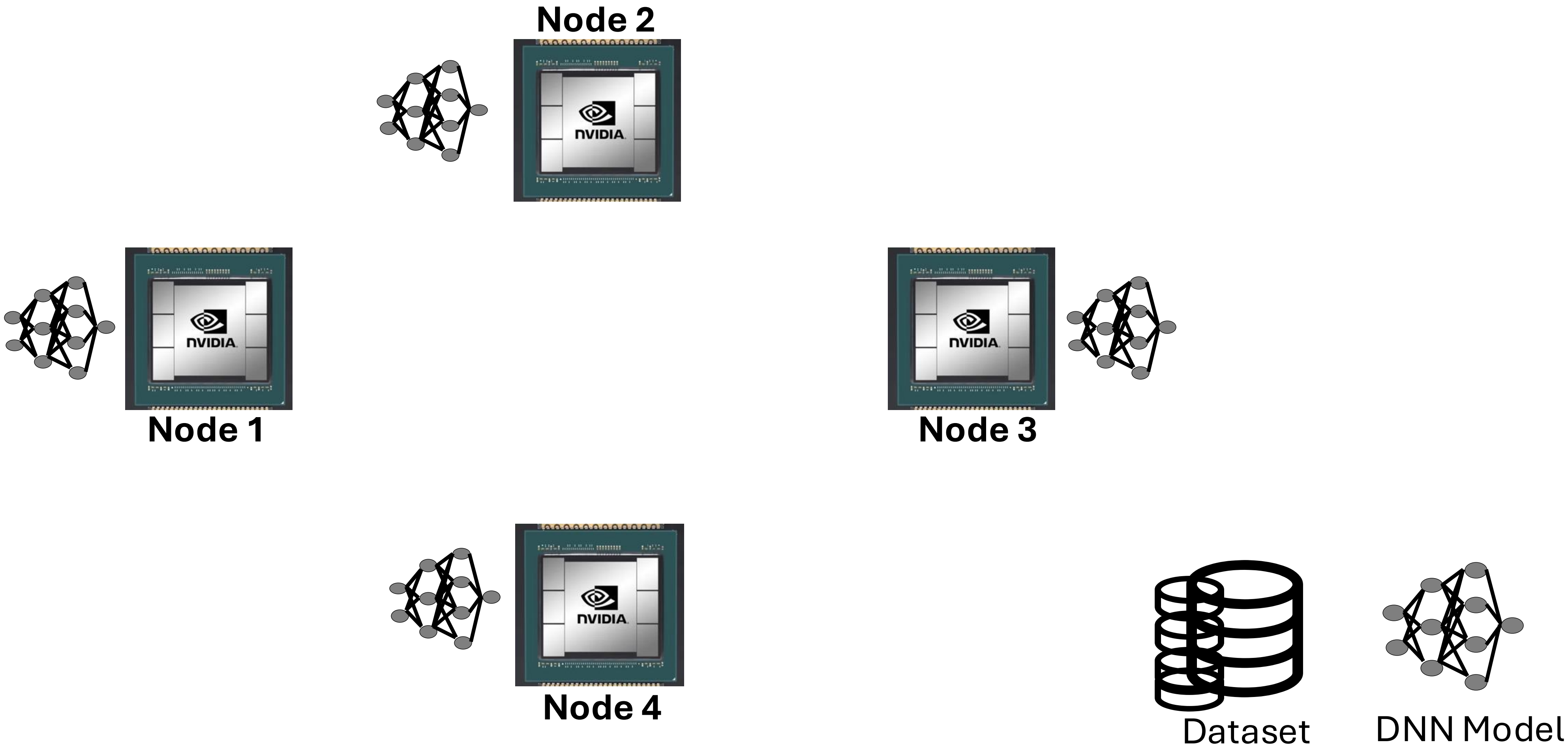
[1] “https://encapture.com/the-rise-of-cloud-machine-learning”
[2] “https://redresscompliance.com/exploring-cloud-based-machine-learning-platforms”
[3] “https://symphony-solutions.com/insights/machine-learning-in-the-cloud-what-are-the-benefits”

Cloud for Distributed Training



Source: www.grandviewresearch.com

Distributed Data Parallel (DDP)



Node 2

Node 1

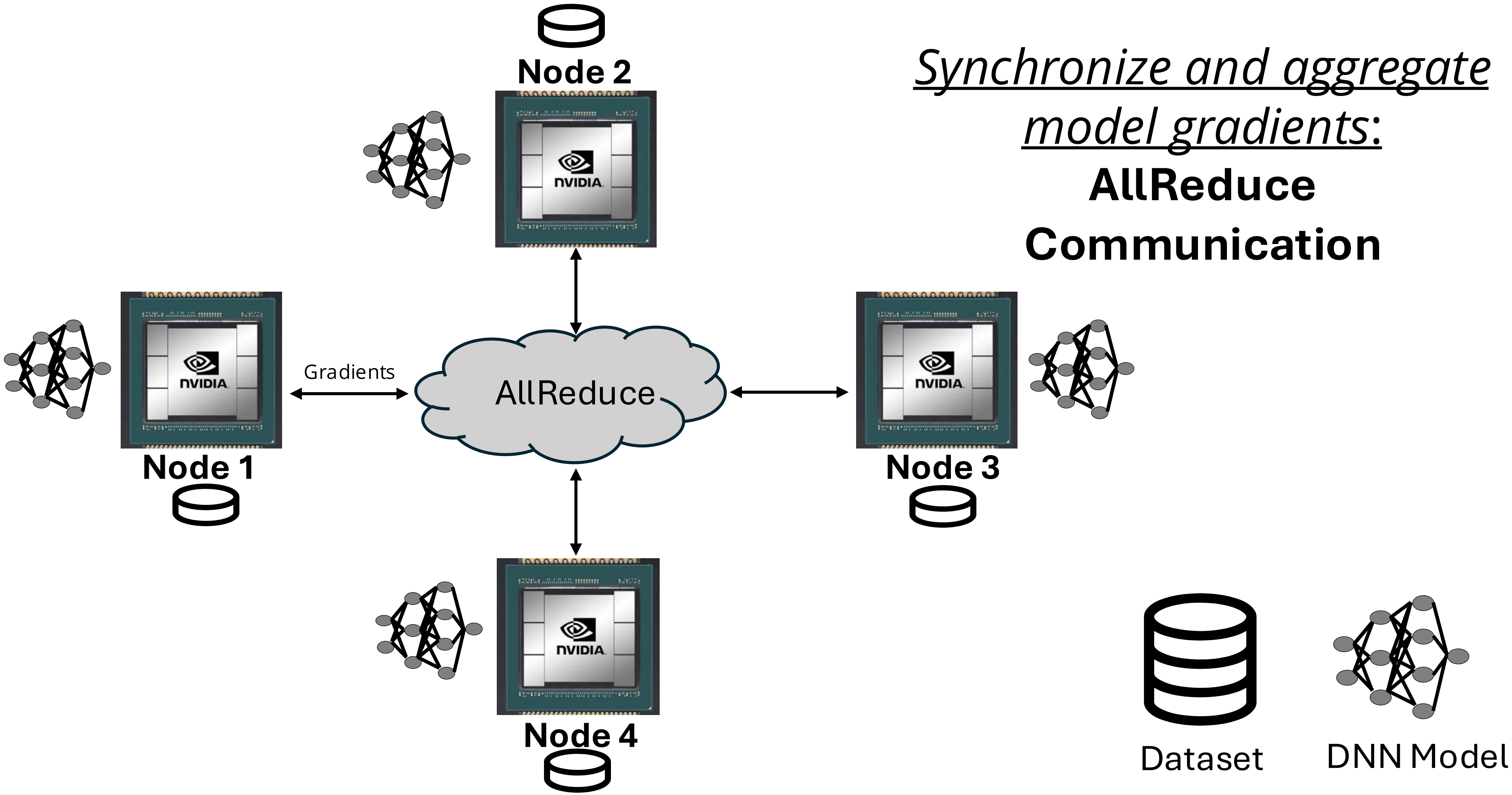
Node 3

Node 4

Dataset

DNN Model

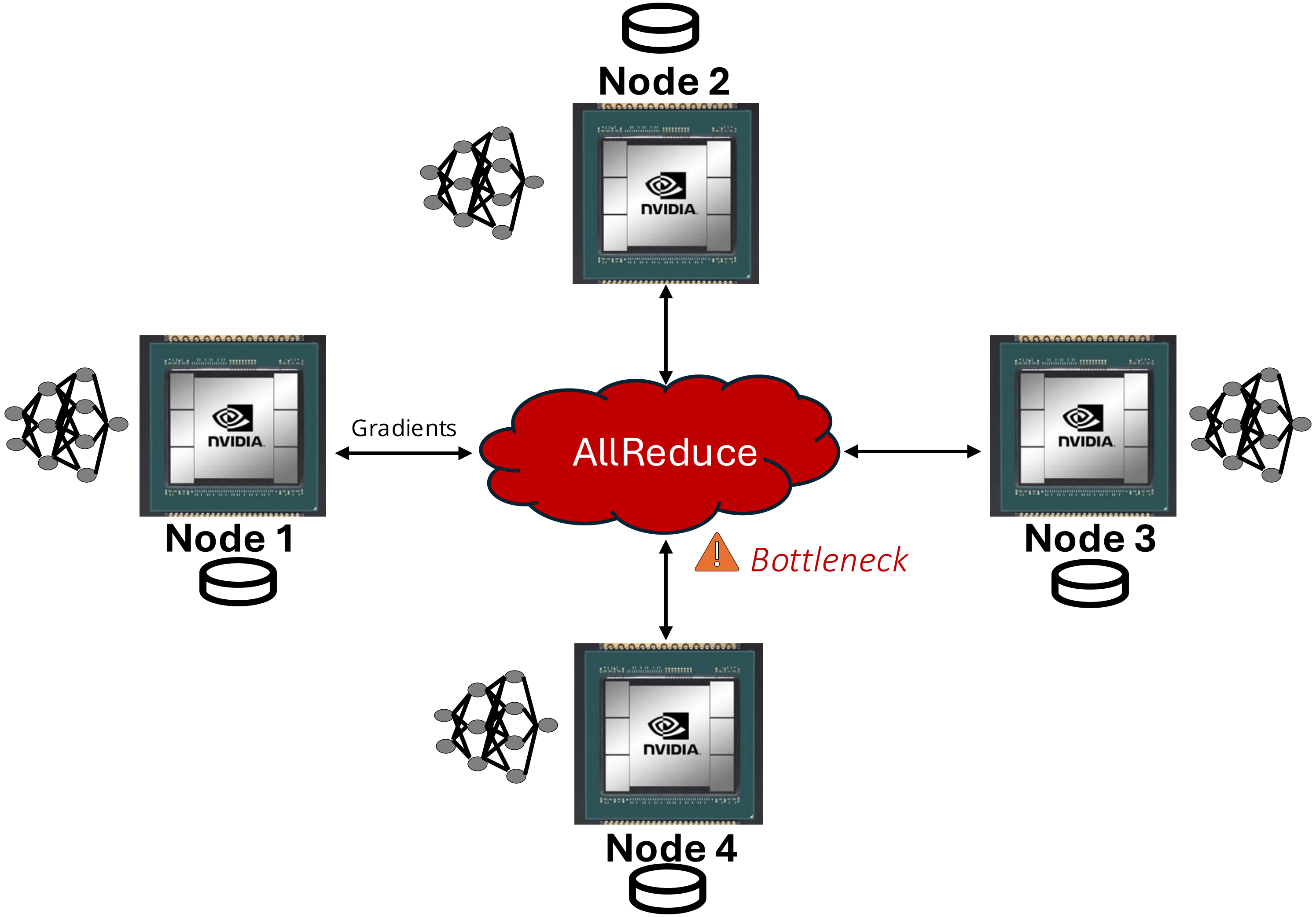
Distributed Data Parallel (DDP)



Synchronize and aggregate model gradients:

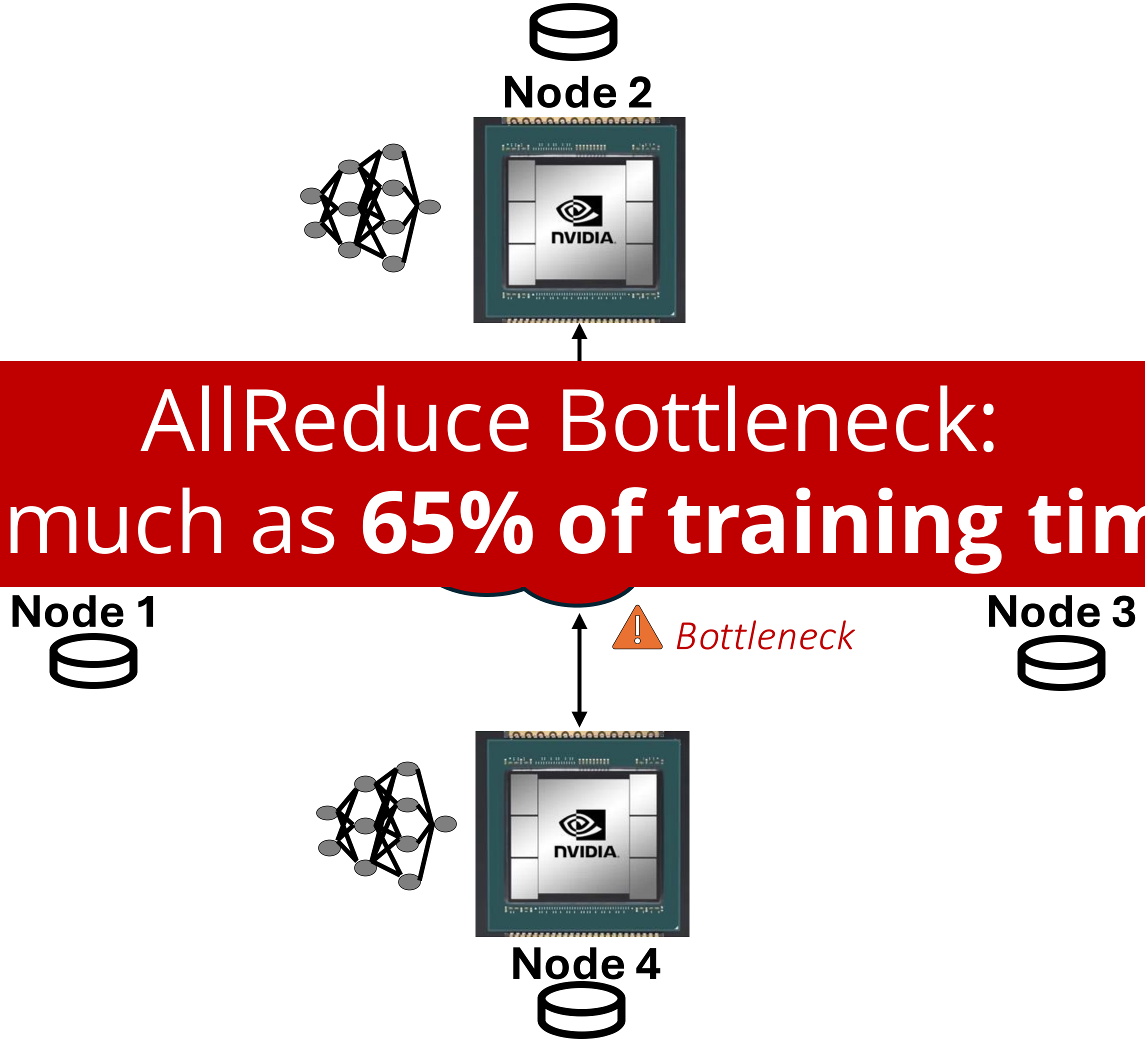
AllReduce Communication

Distributed Data Parallel (DDP)



[1] Sapio, Amedeo, et al. "Scaling distributed machine learning with in-network aggregation." NSDI'21

Distributed Data Parallel (DDP)

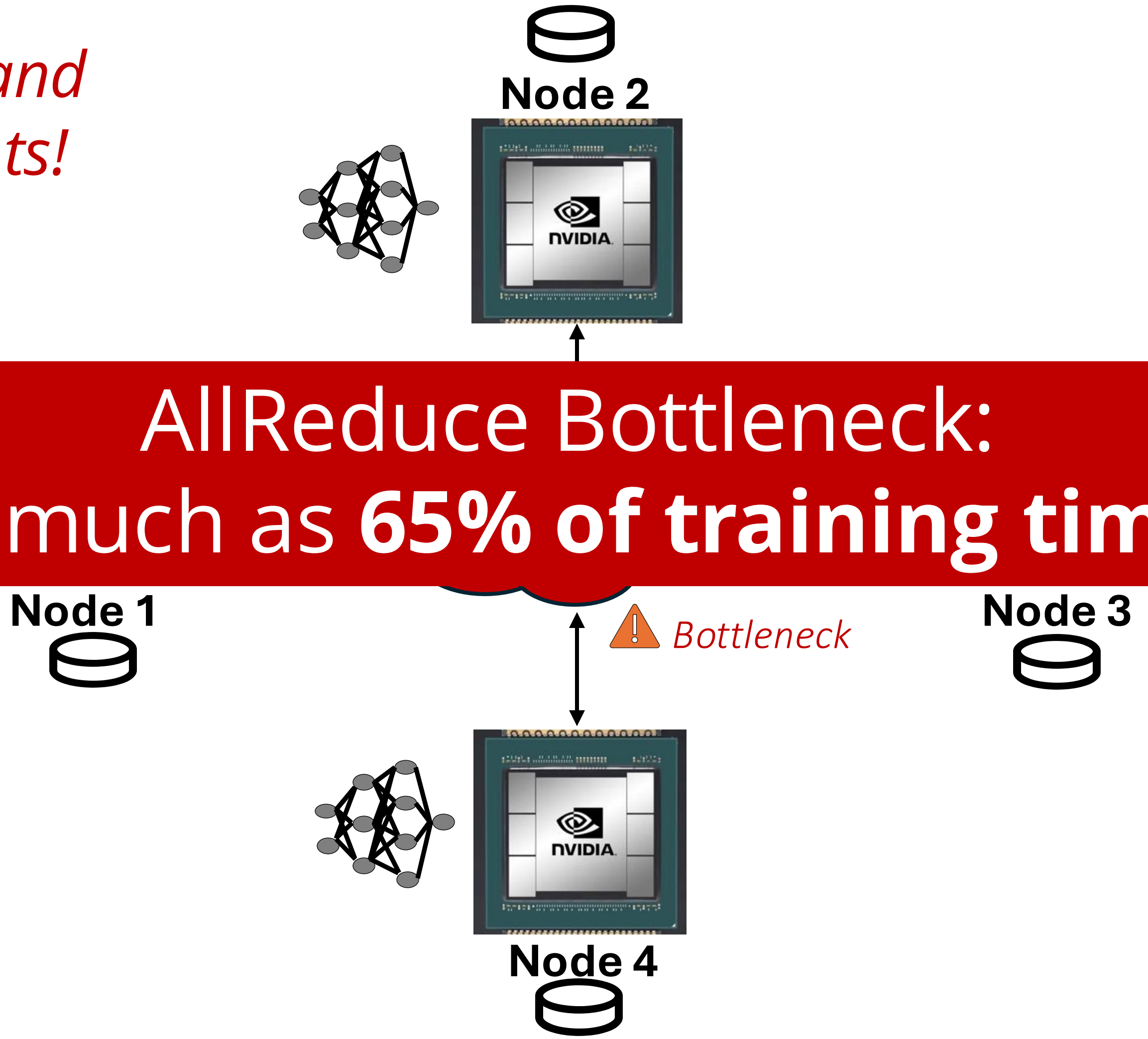


AllReduce Bottleneck:
As much as **65% of training time**[1]

[1] Sapio, Amedeo, et al. "Scaling distributed machine learning with in-network aggregation.” NSDI’21

Distributed Data Parallel (DDP)

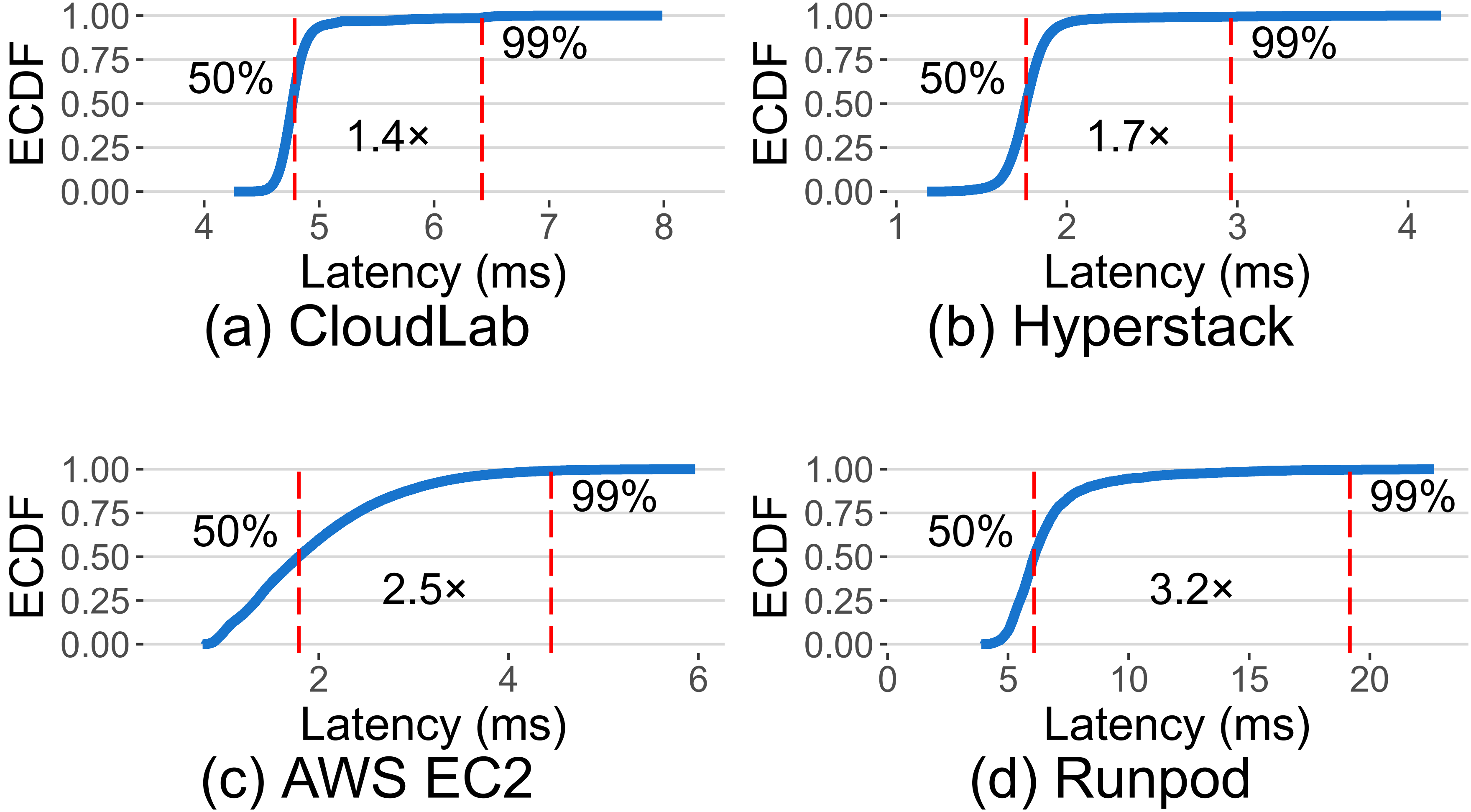
Worsened in cloud and shared environments!



AllReduce Bottleneck:
As much as **65% of training time**[1]

[1] Sapio, Amedeo, et al. "Scaling distributed machine learning with in-network aggregation." NSDI'21

Tail in the Cloud



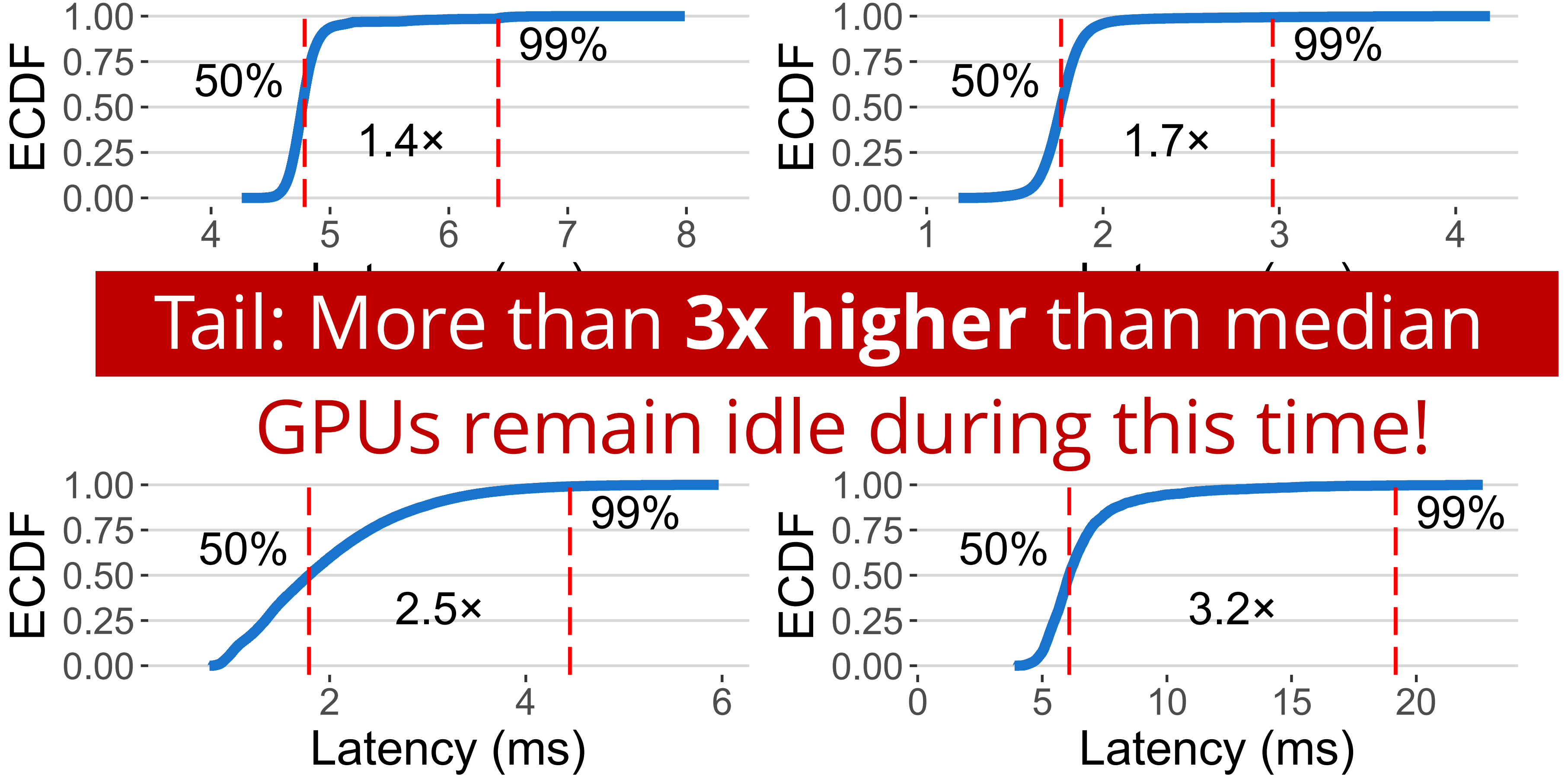
(a) CloudLab

(b) Hyperstack

(c) AWS EC2

(d) Runpod

Tail in the Cloud



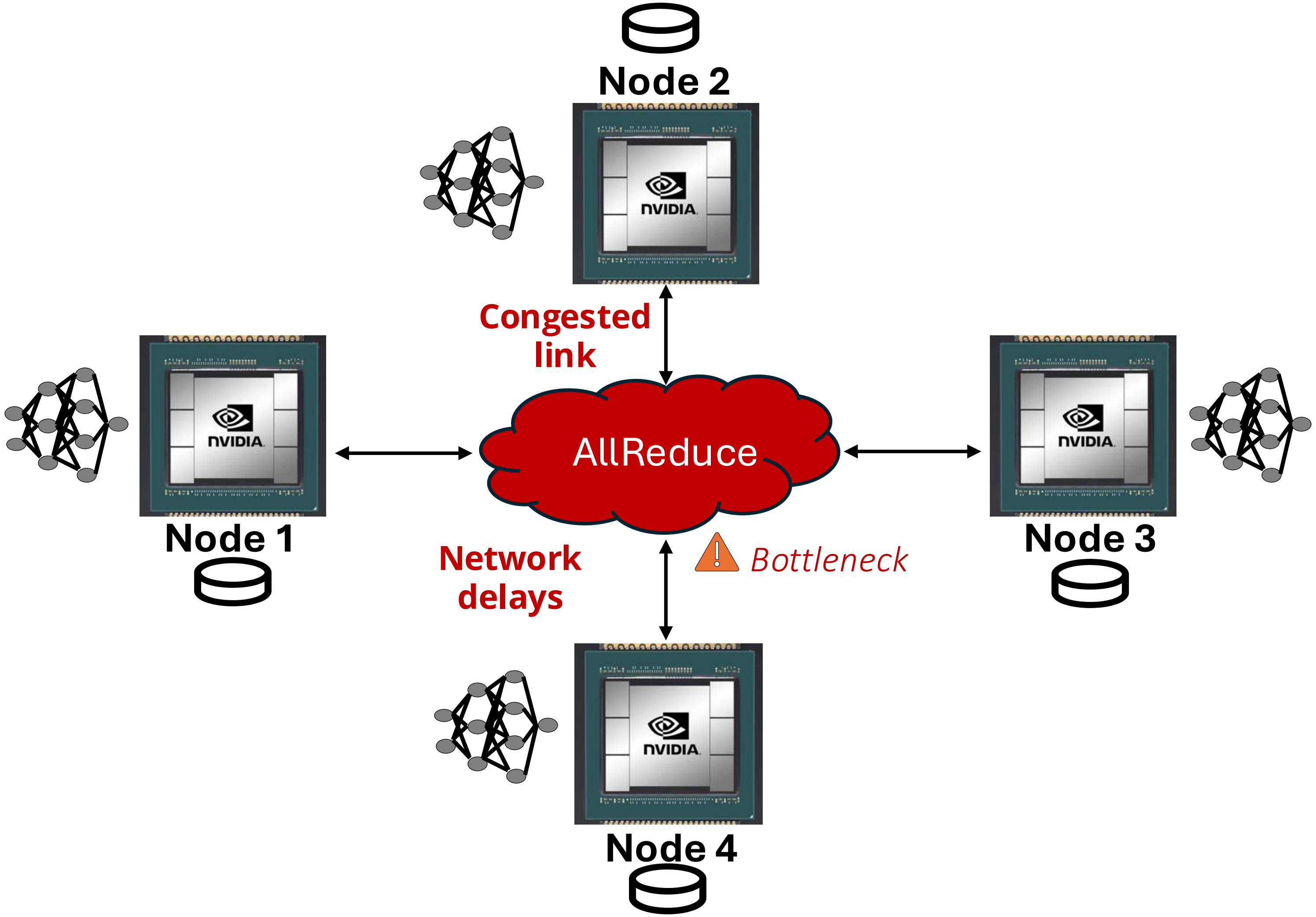
Tail: More than **3x higher** than median

GPUs remain idle during this time!

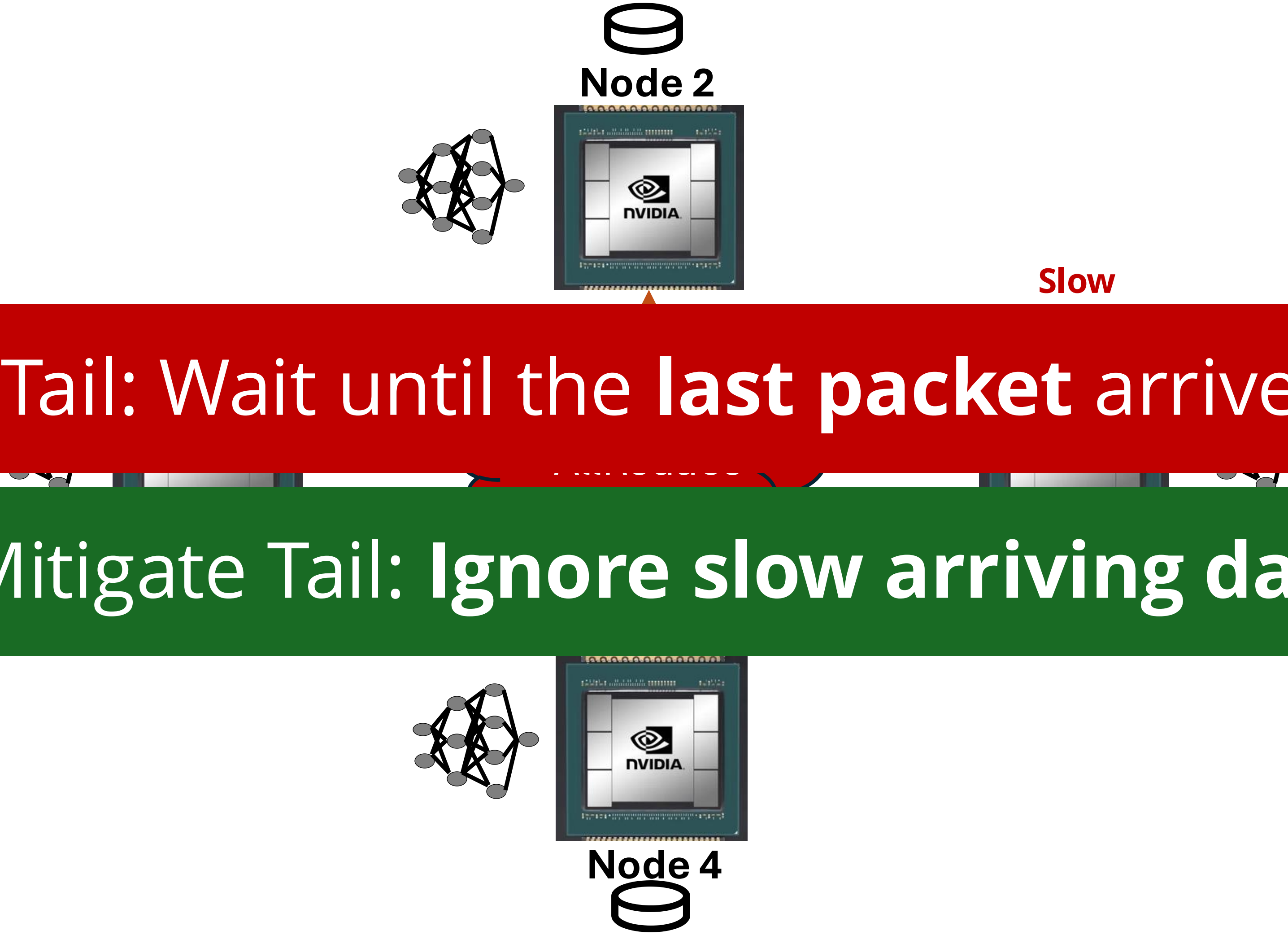
(c) AWS EC2

(d) Runpod

How to Mitigate this Tail?



How to Mitigate this Tail?



Tail: Wait until the **last packet** arrives

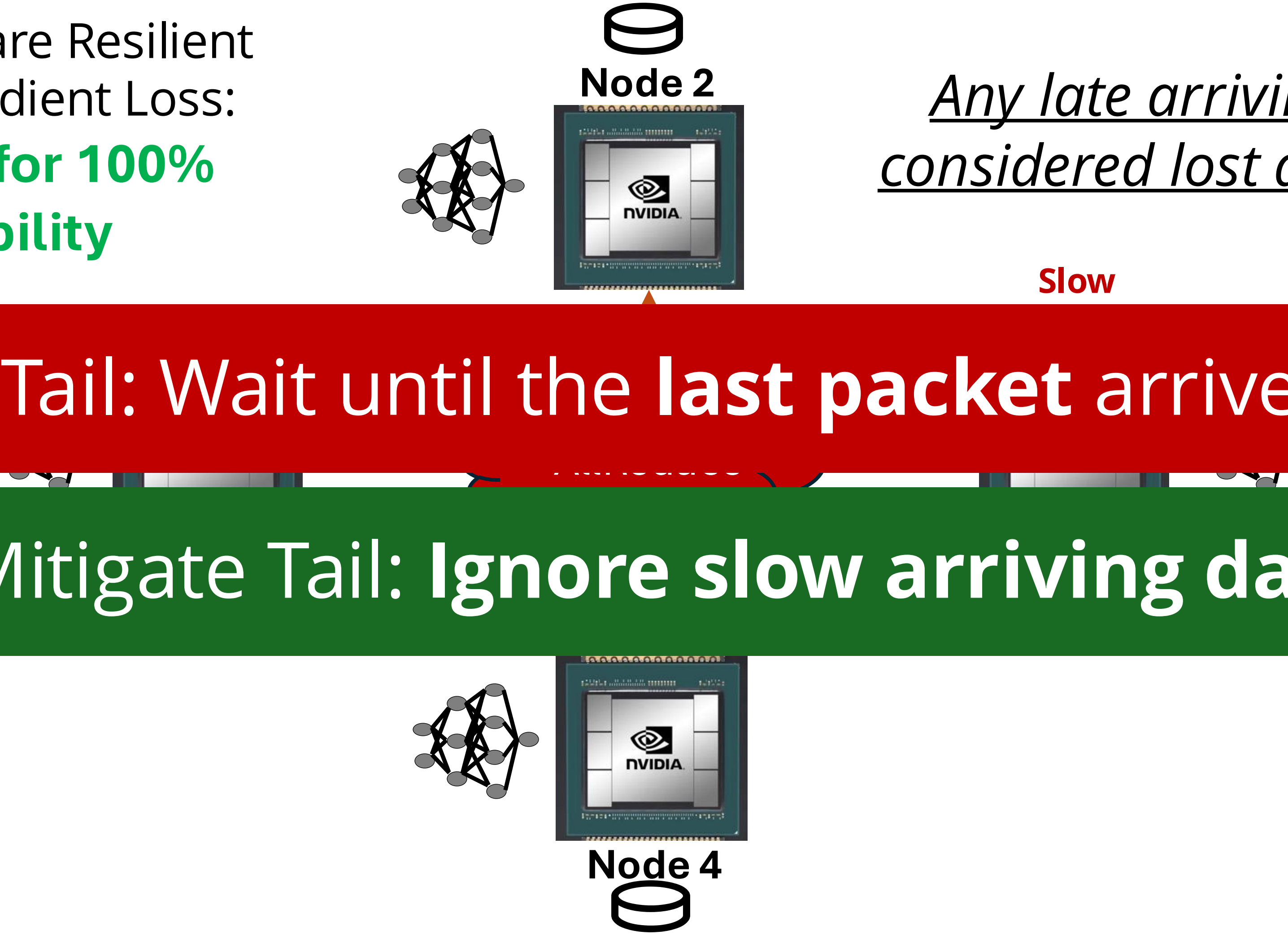
Mitigate Tail: **Ignore slow arriving data**

How to Mitigate this Tail?

ML Models are Resilient Against Gradient Loss:

No need for 100% reliability

Any late arriving data is considered lost and ignored

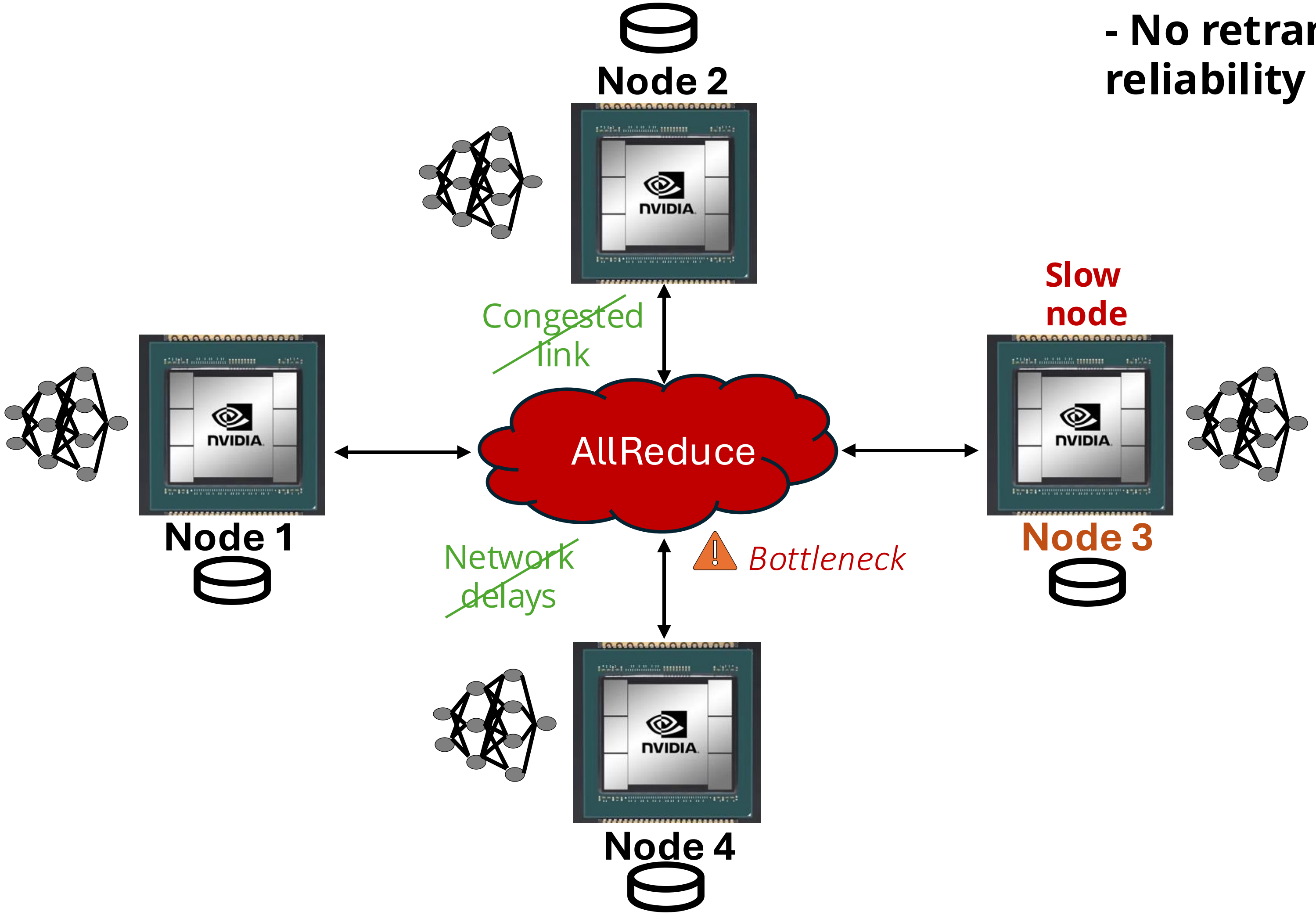


Tail: Wait until the **last packet** arrives

Mitigate Tail: **Ignore slow arriving data**

Unreliable Communication

- No retransmissions or reliability guarantees



Node 2

~~Congested link~~

Slow node

Node 1

AllReduce

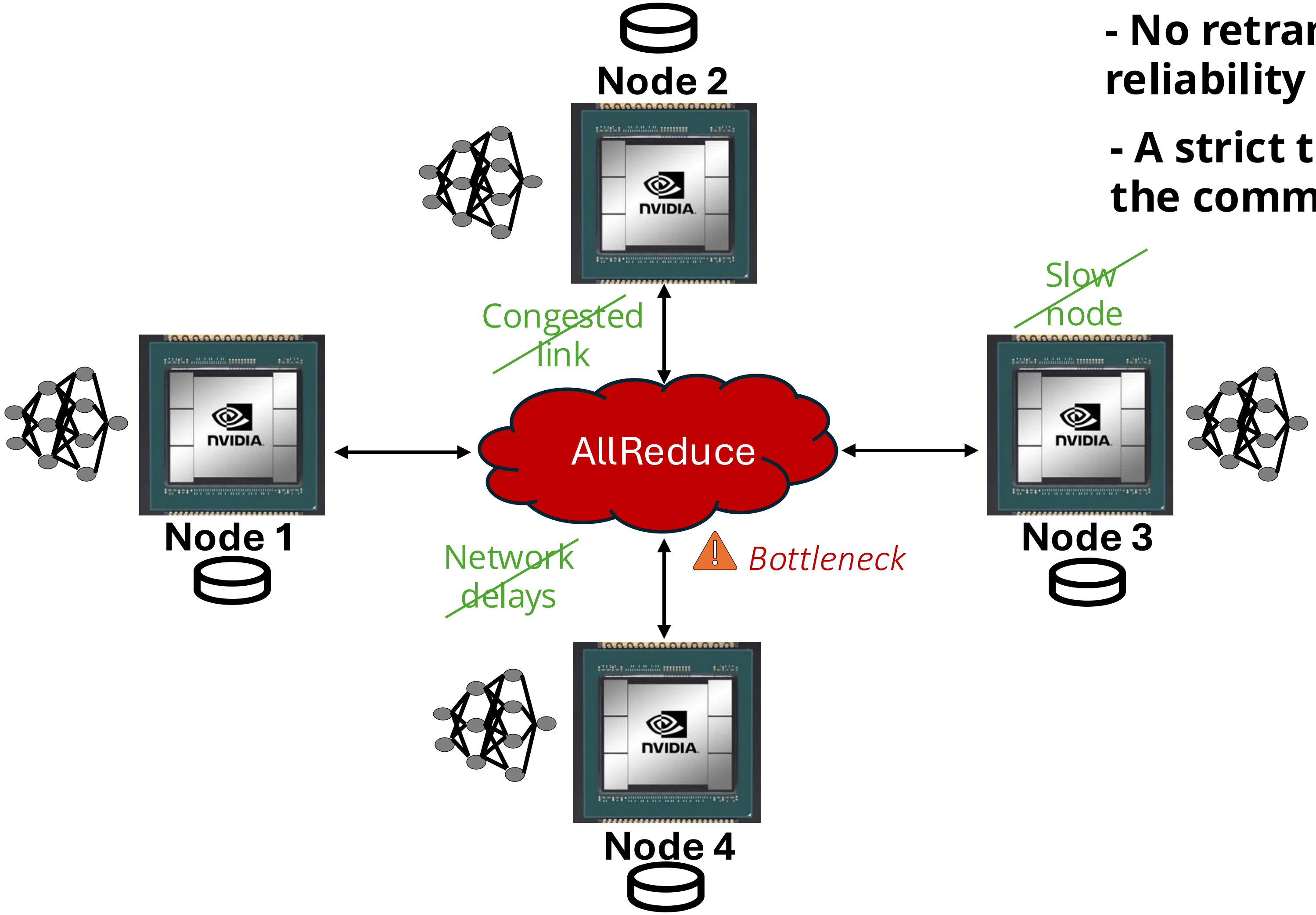
Node 3

~~Network delays~~

⚠ *Bottleneck*

Node 4

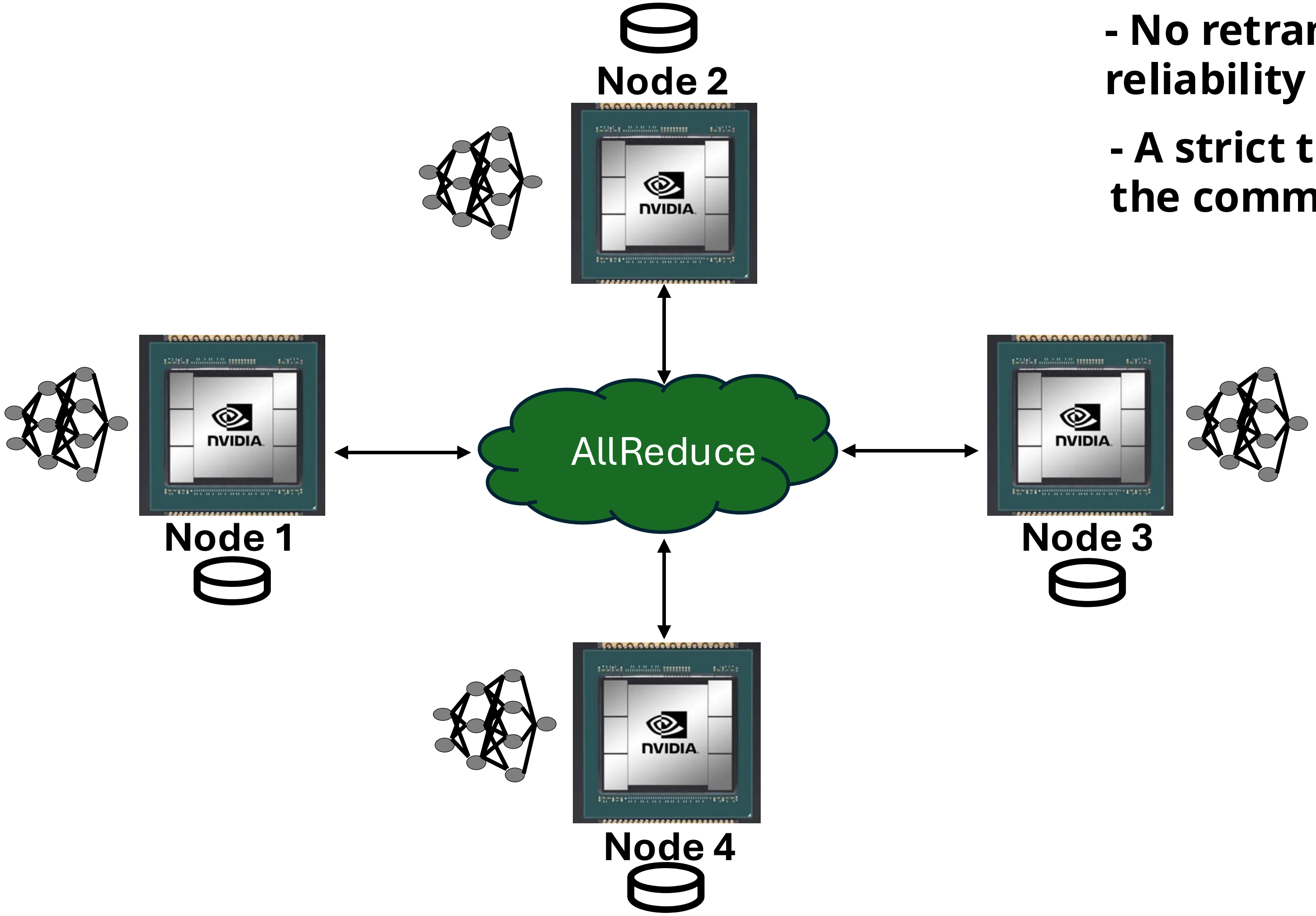
Unreliable Communication



- No retransmissions or reliability guarantees

- A strict time-bound on the communication

Unreliable Communication



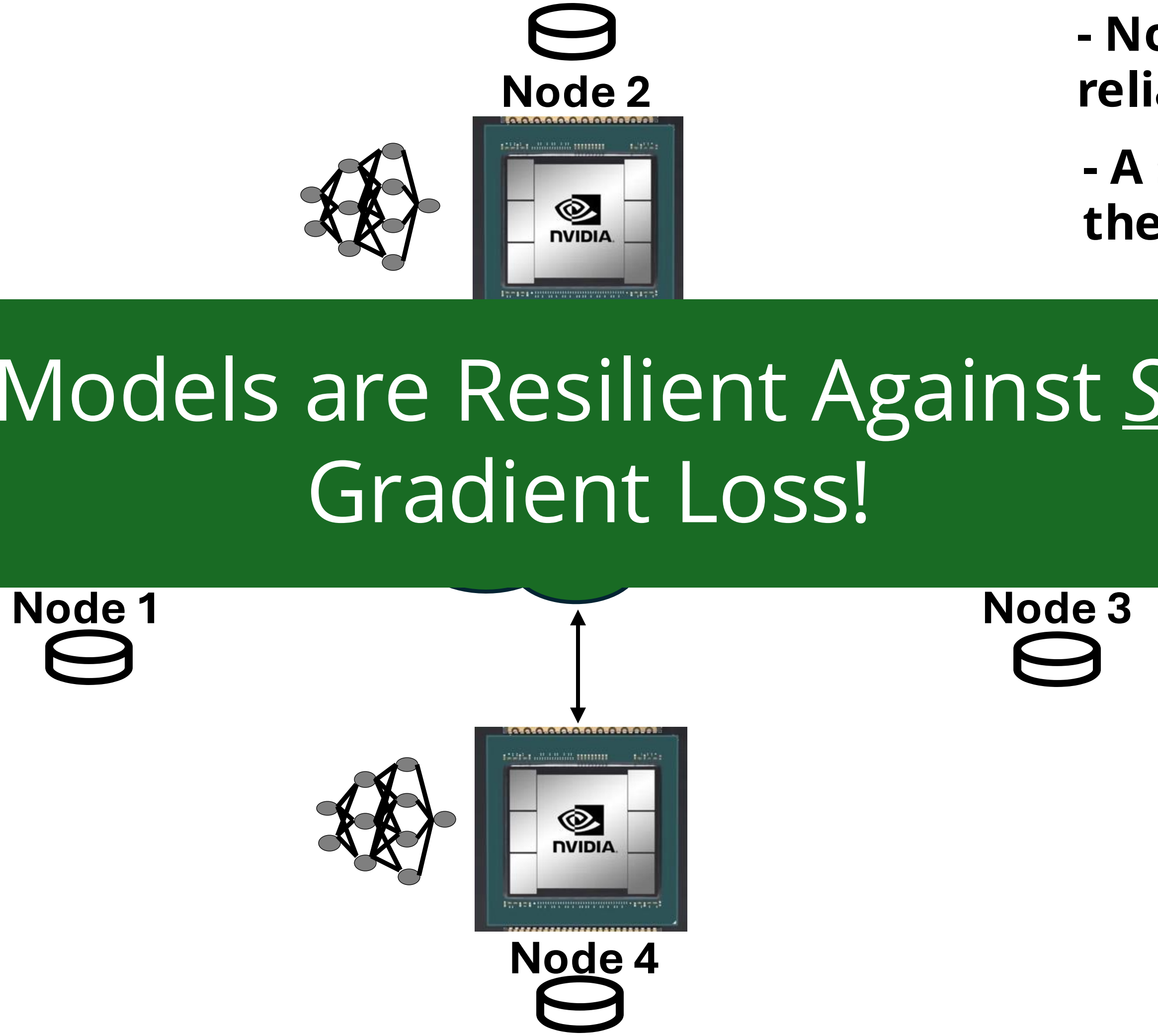
- No retransmissions or reliability guarantees

- A strict time-bound on the communication

Unreliable Communication

- No retransmissions or reliability guarantees

- A strict time-bound on the communication



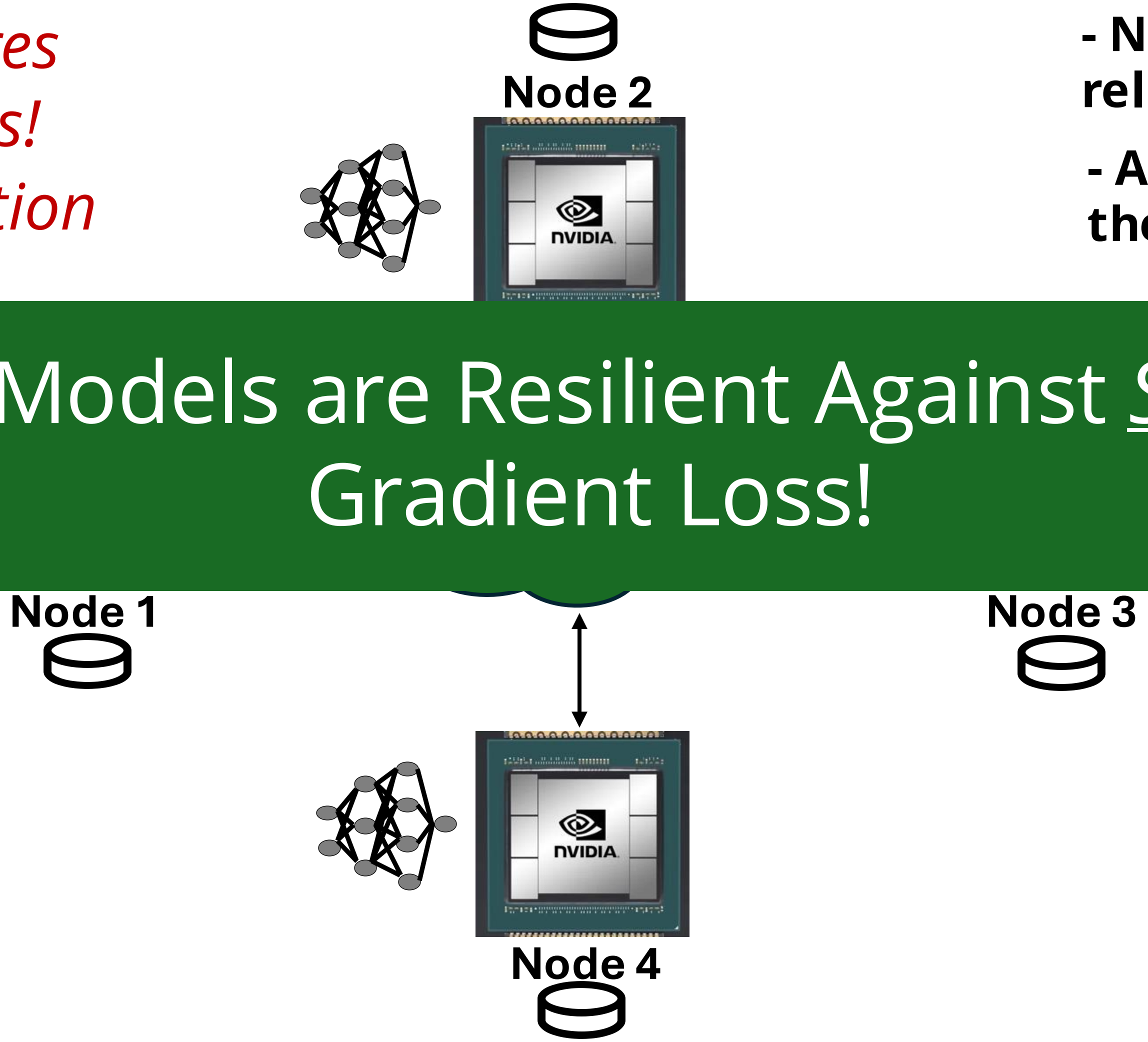
ML Models are Resilient Against *Some* Gradient Loss!

Unreliable Communication

Existing architectures exacerbate the loss!
→ Accuracy degradation

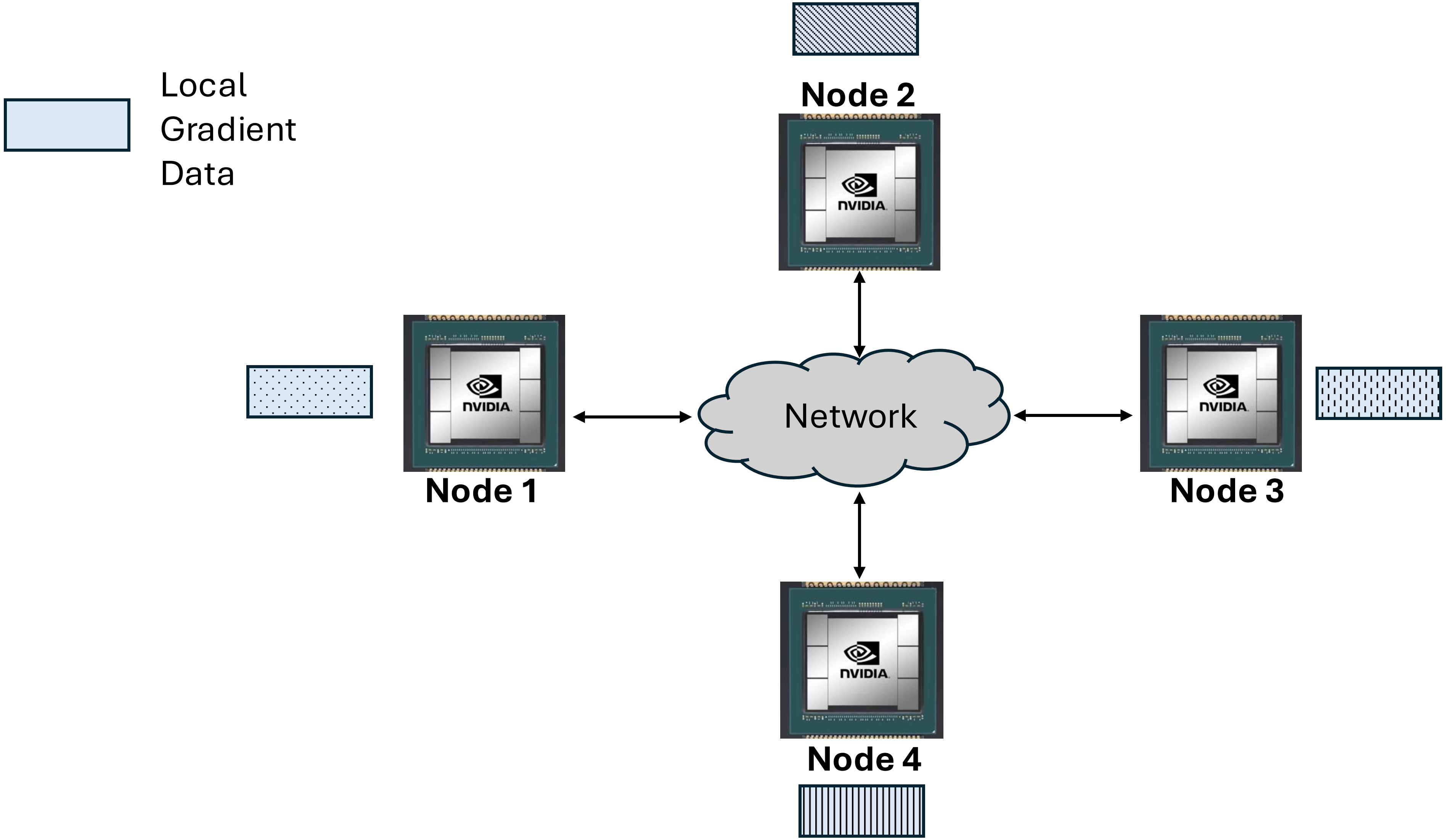
- No retransmissions or reliability guarantees

- A strict time-bound on the communication



ML Models are Resilient Against *<u>Some</u>* Gradient Loss!

UDP for Unreliable Transport



Local Gradient Data

Node 2

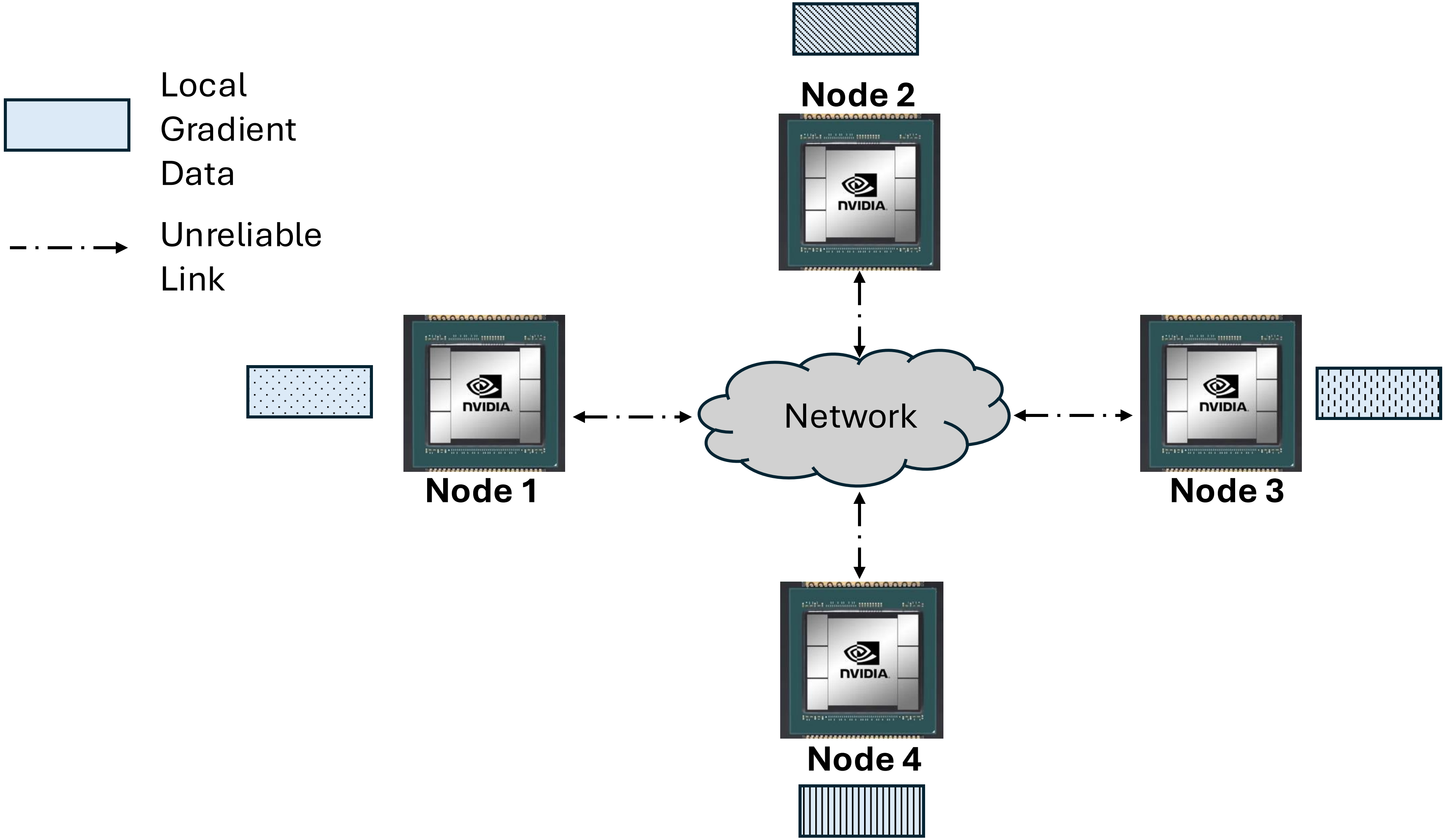
Node 1

Network

Node 3

Node 4

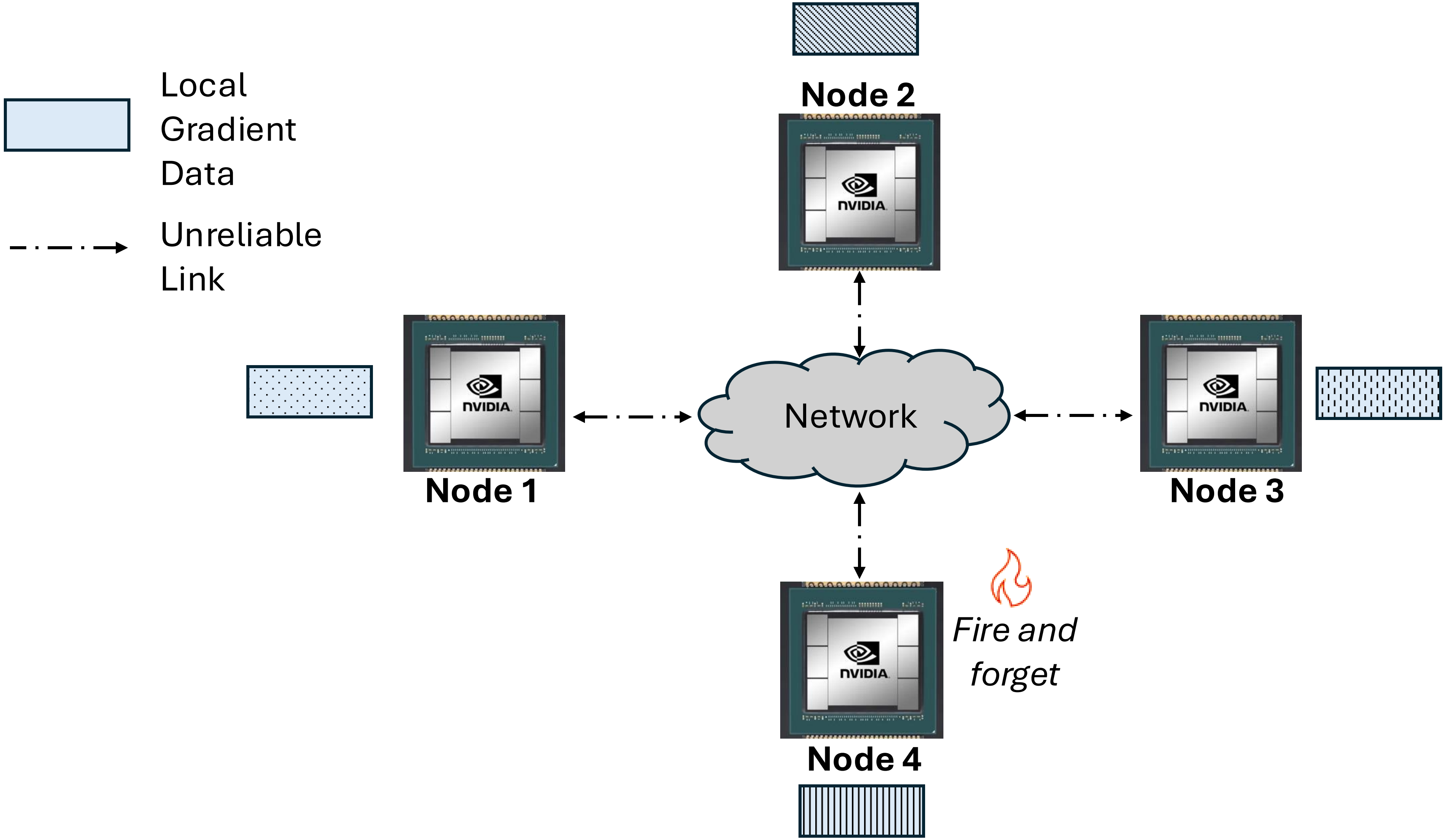
UDP for Unreliable Transport



Local Gradient Data

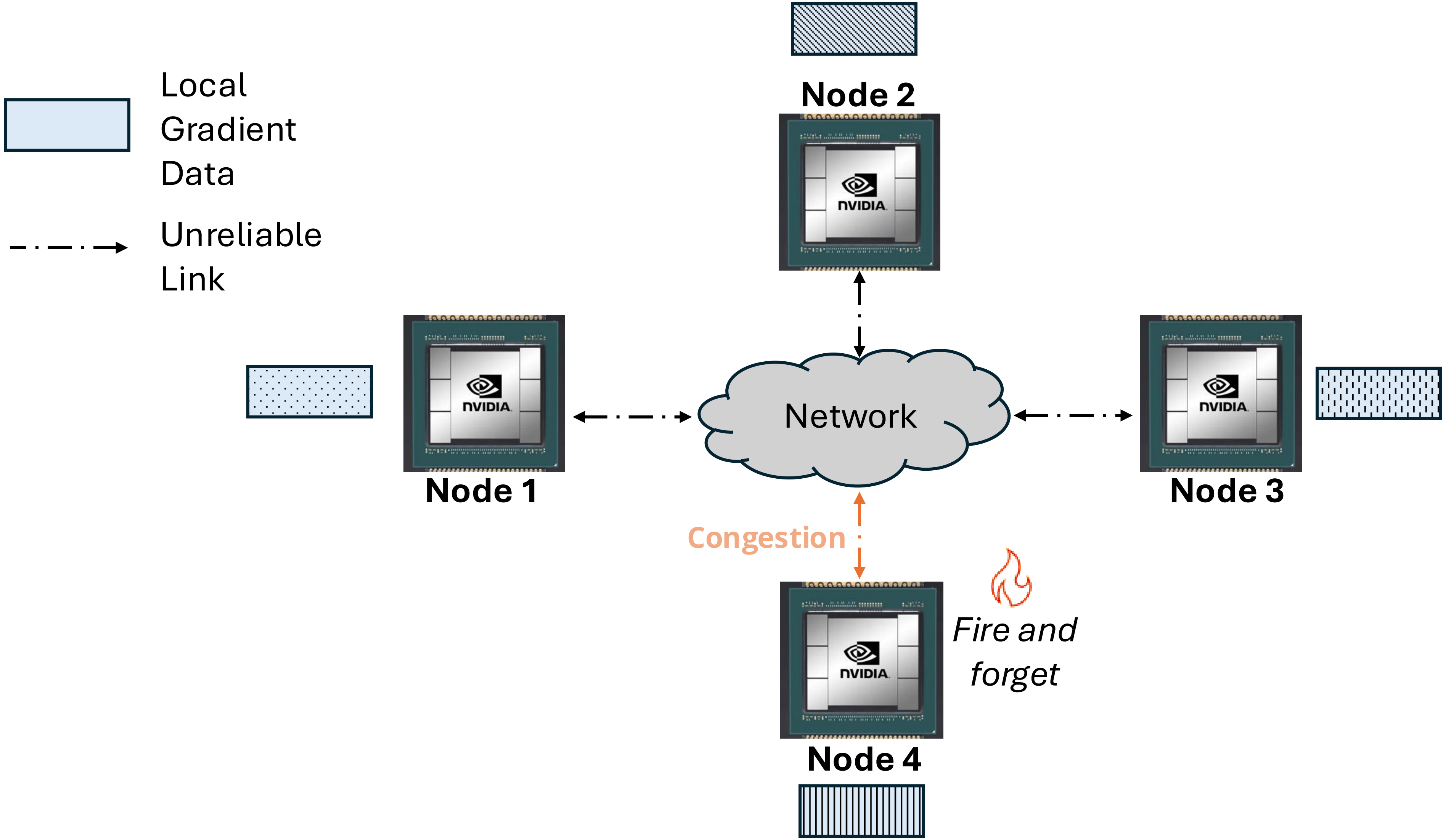
Unreliable Link

UDP for Unreliable Transport

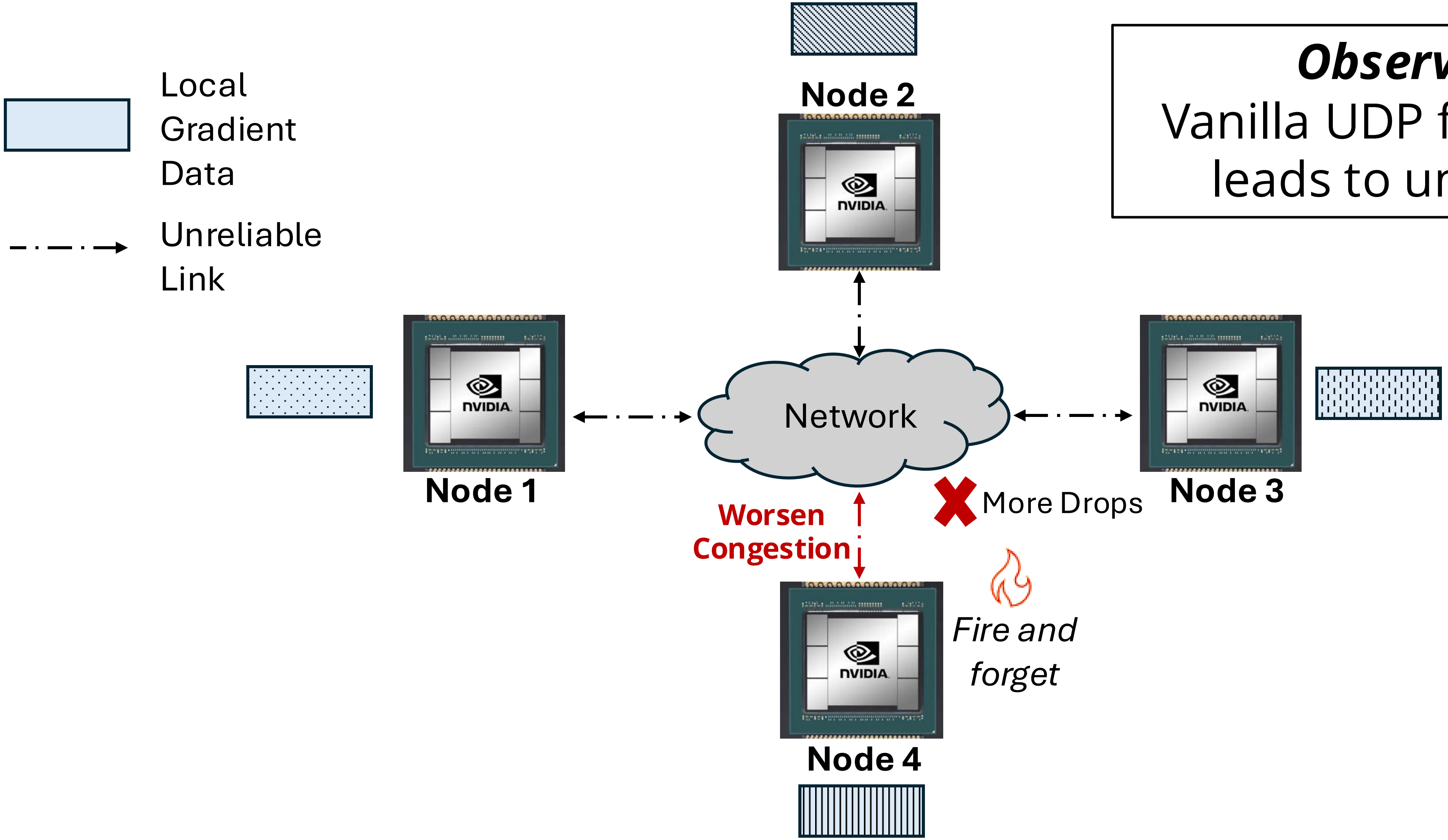


Fire and forget

UDP for Unreliable Transport

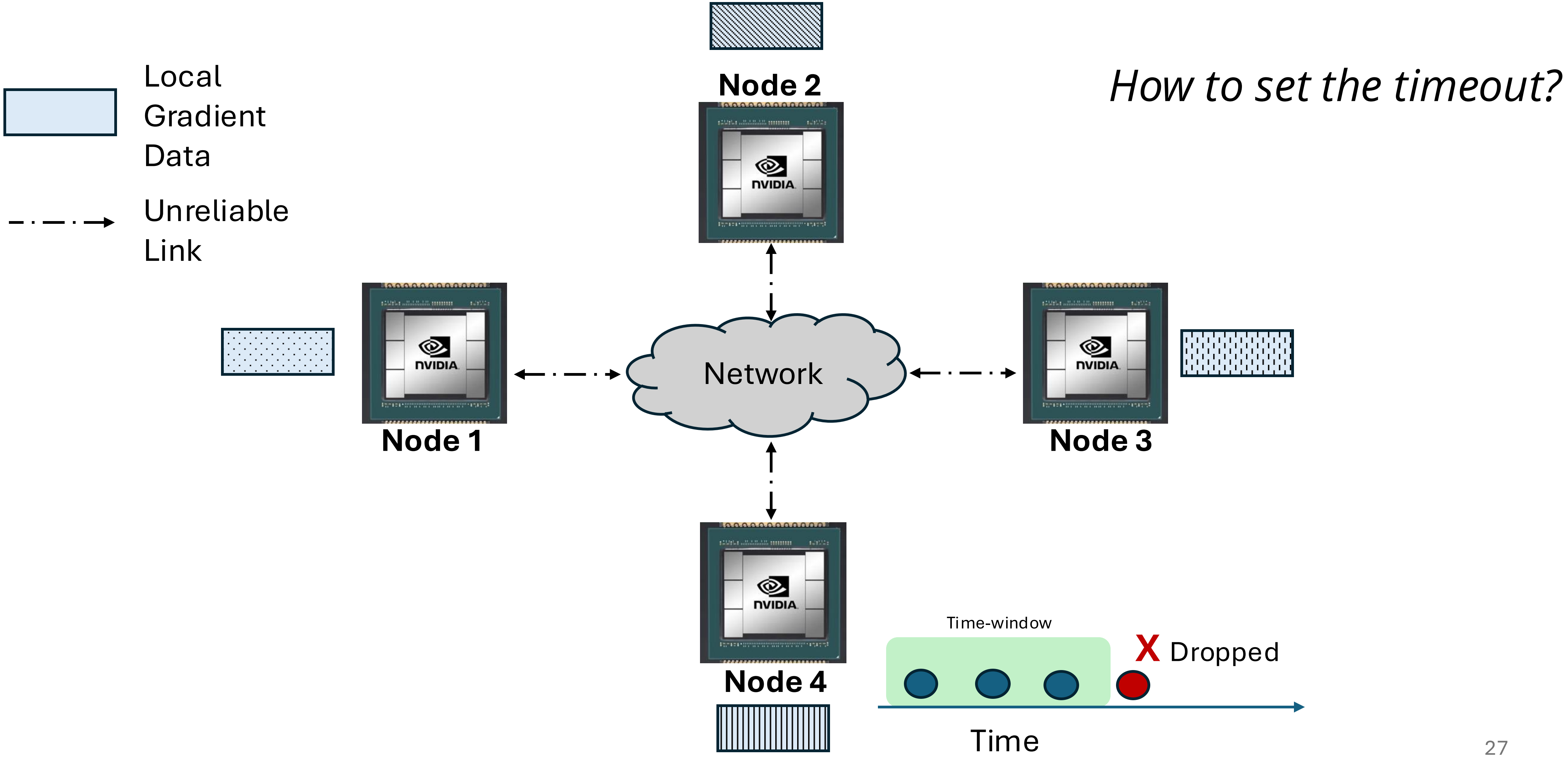


UDP for Unreliable Transport



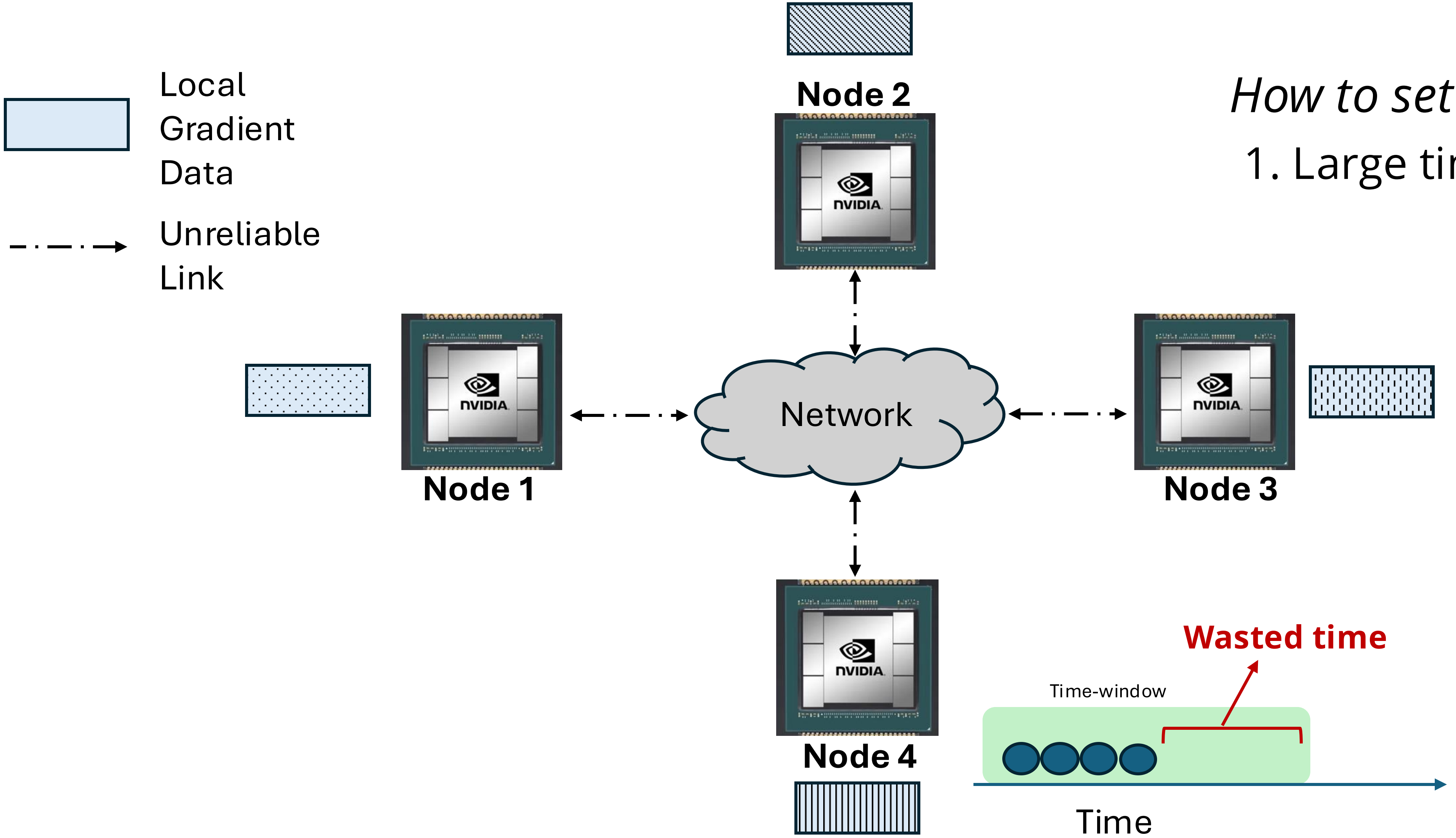
Observation 1:
Vanilla UDP fires away and leads to undue losses!

Timeouts to Mitigate Slow Workers



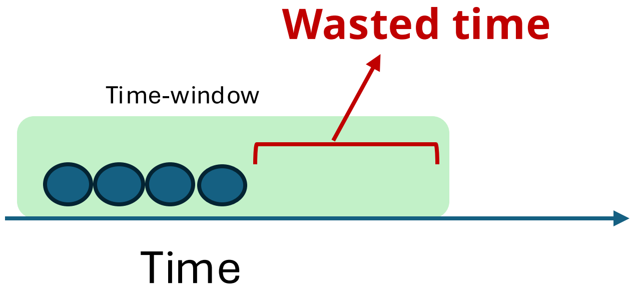
How to set the timeout?

Timeouts to Mitigate Slow Workers

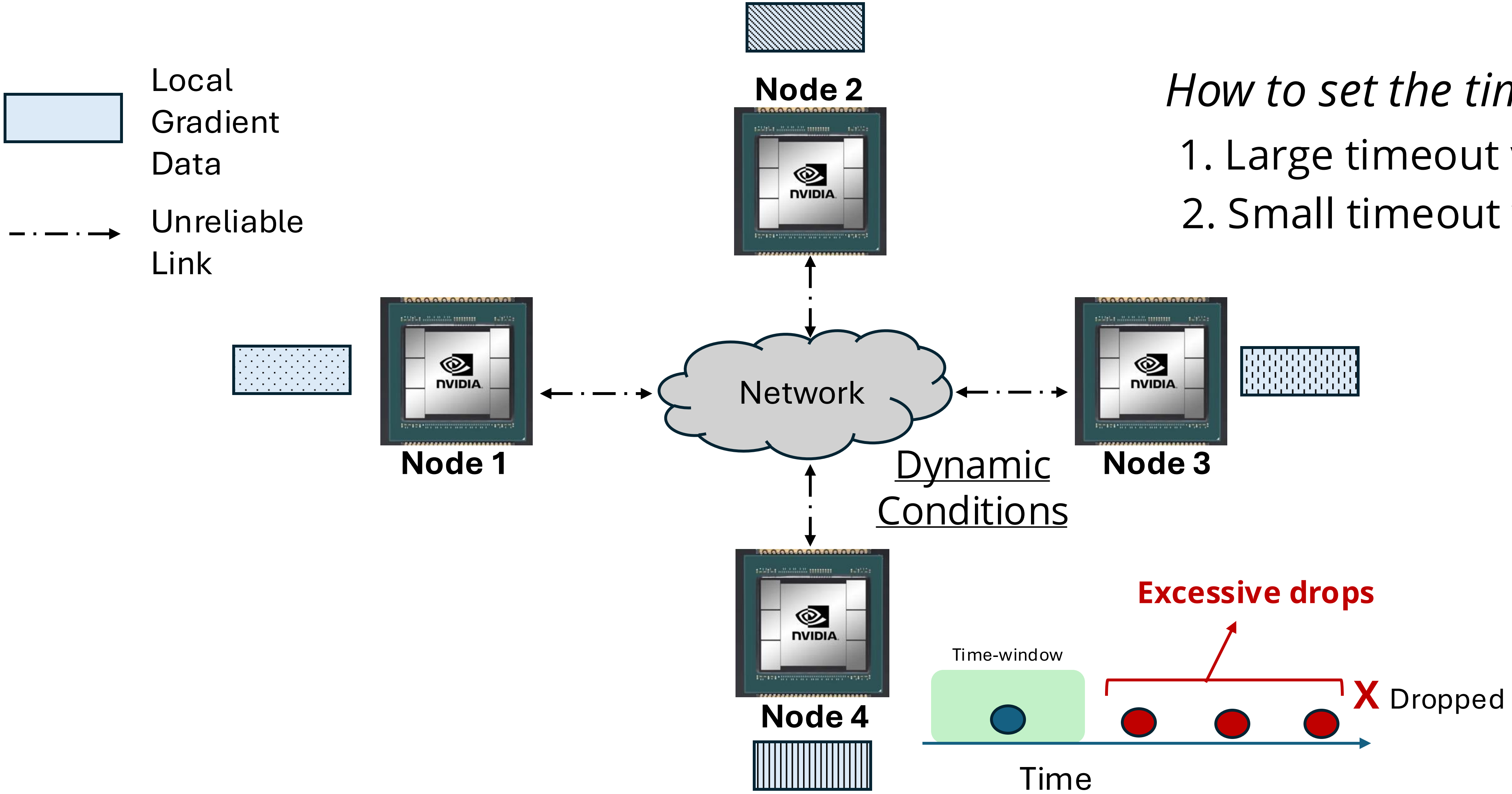


How to set the timeout?

1. Large timeout value



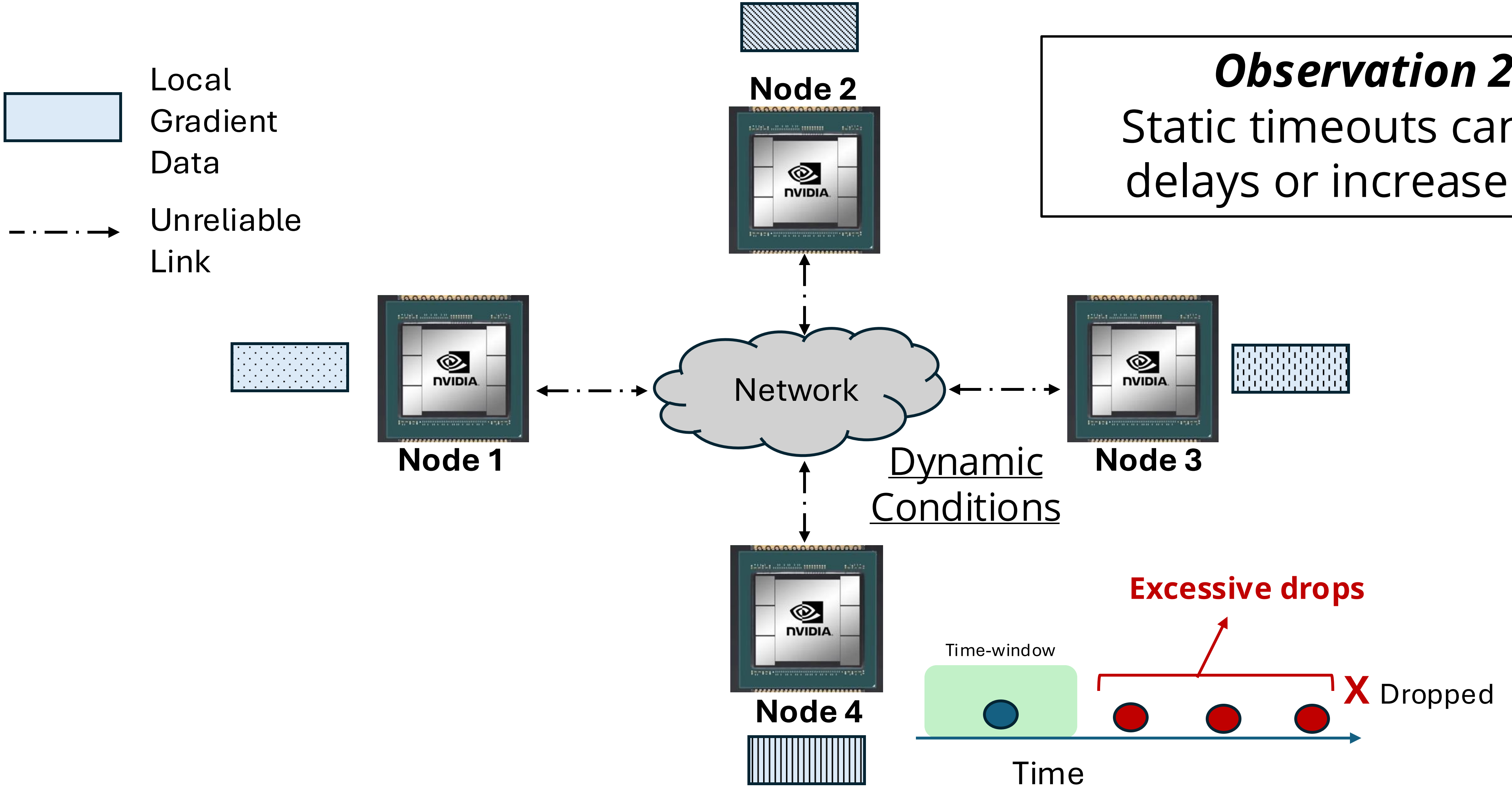
Timeouts to Mitigate Slow Workers



How to set the timeout?

1. Large timeout value
2. Small timeout value

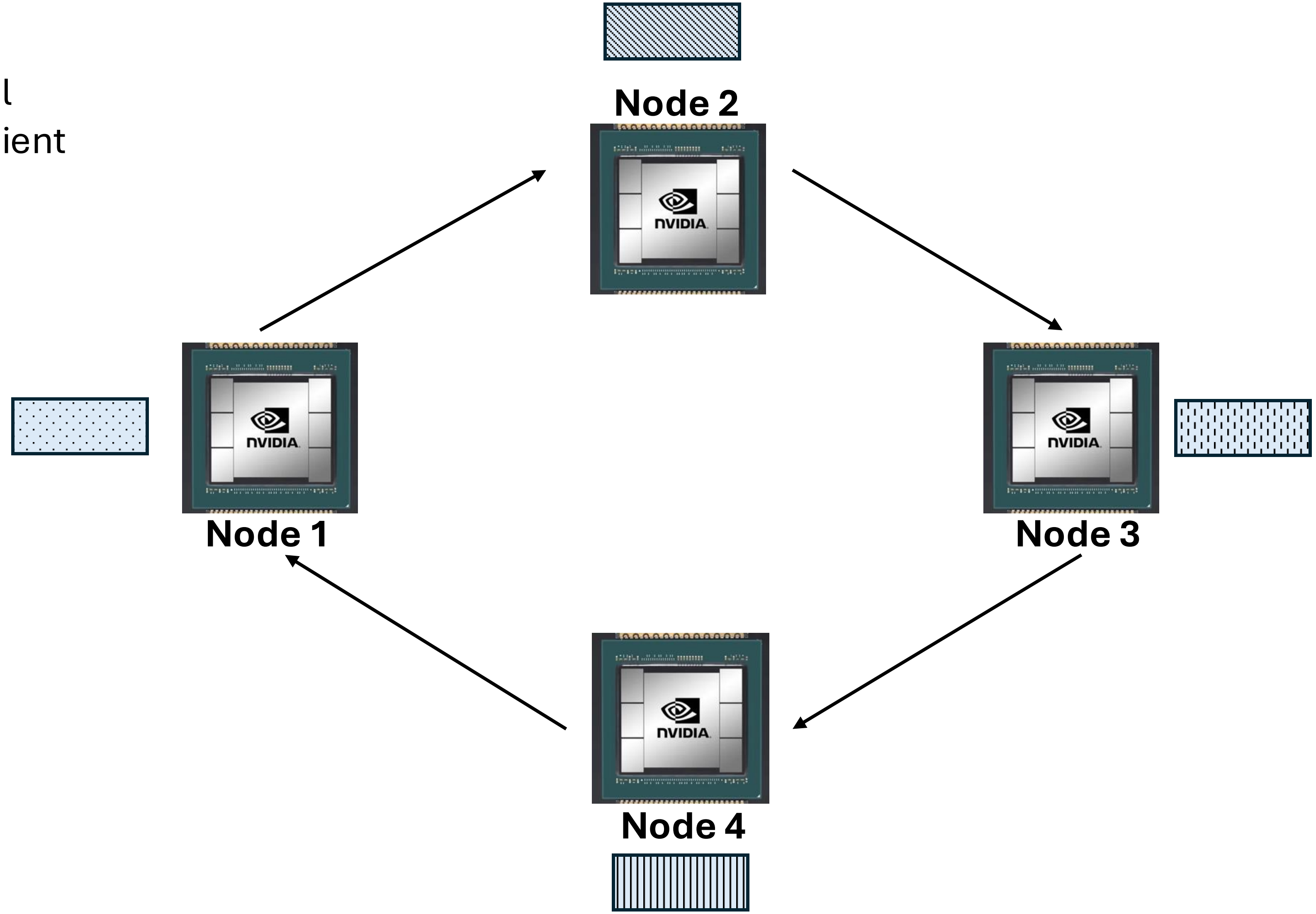
Timeouts to Mitigate Slow Workers



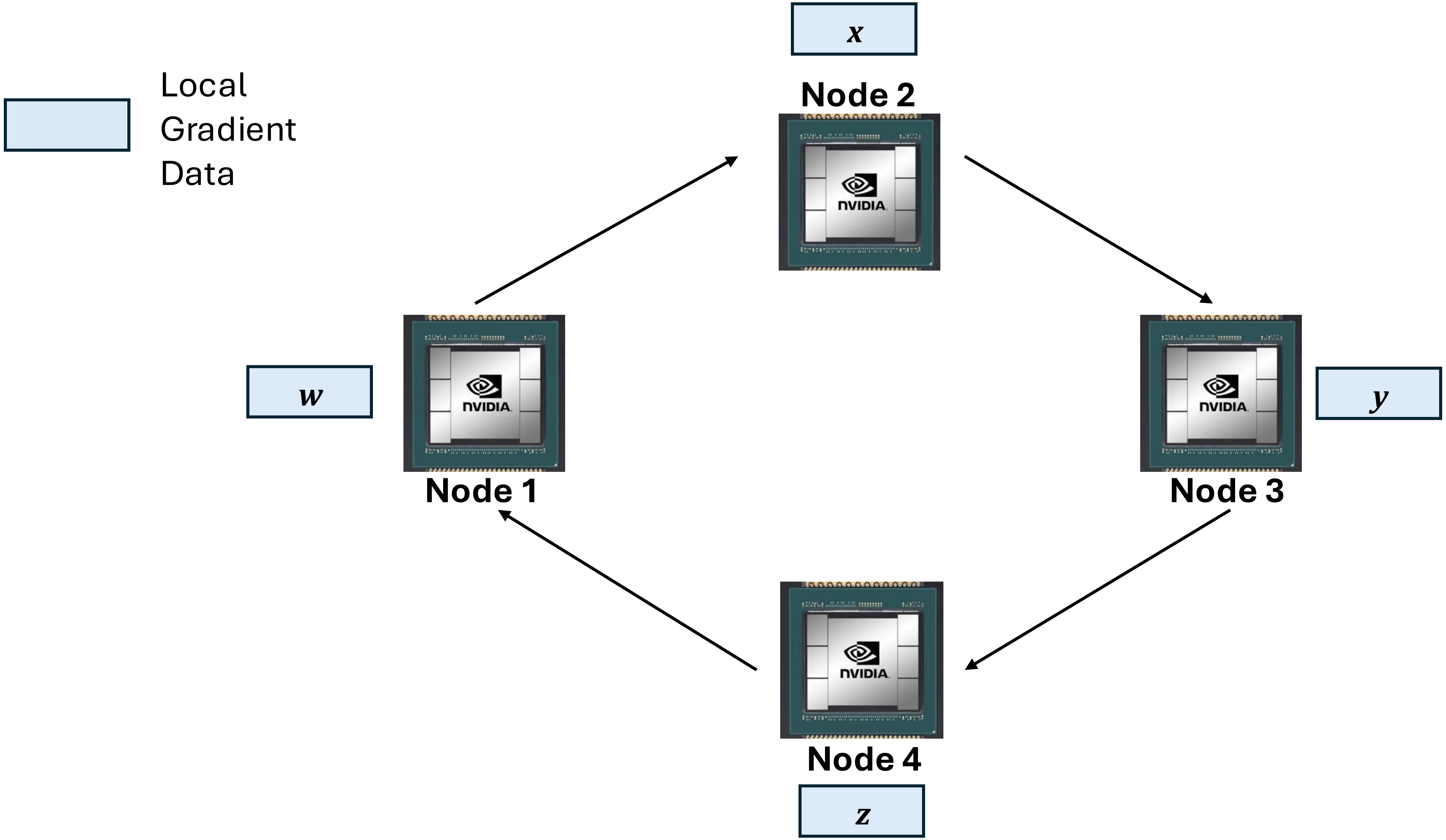
Observation 2:
Static timeouts can add delays or increase loss!

Existing Architectures: Ring AR

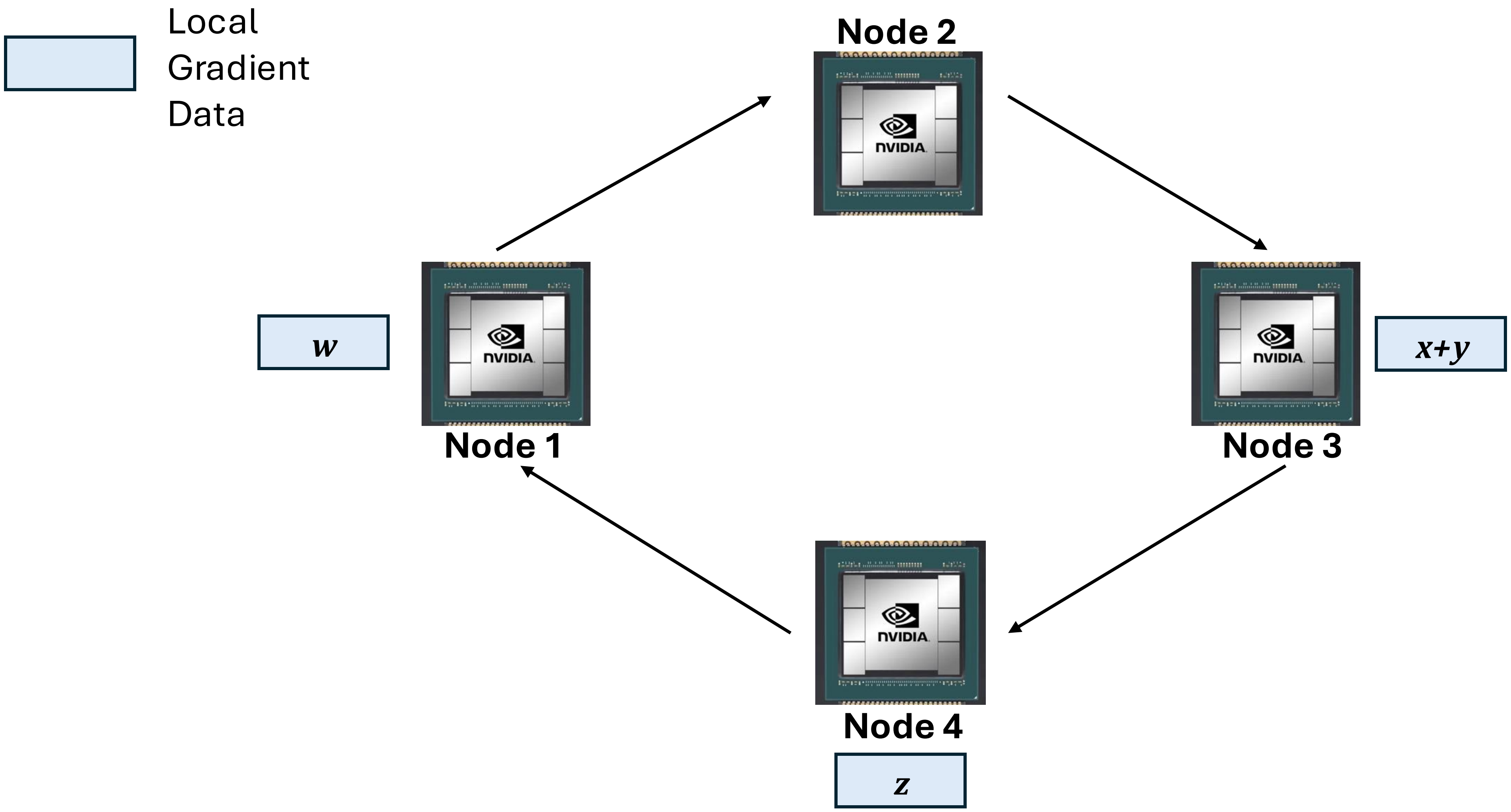
▭ Local Gradient Data



Existing Architectures: Ring AR



Existing Architectures: Ring AR



Local Gradient Data

Node 2

Node 1

w

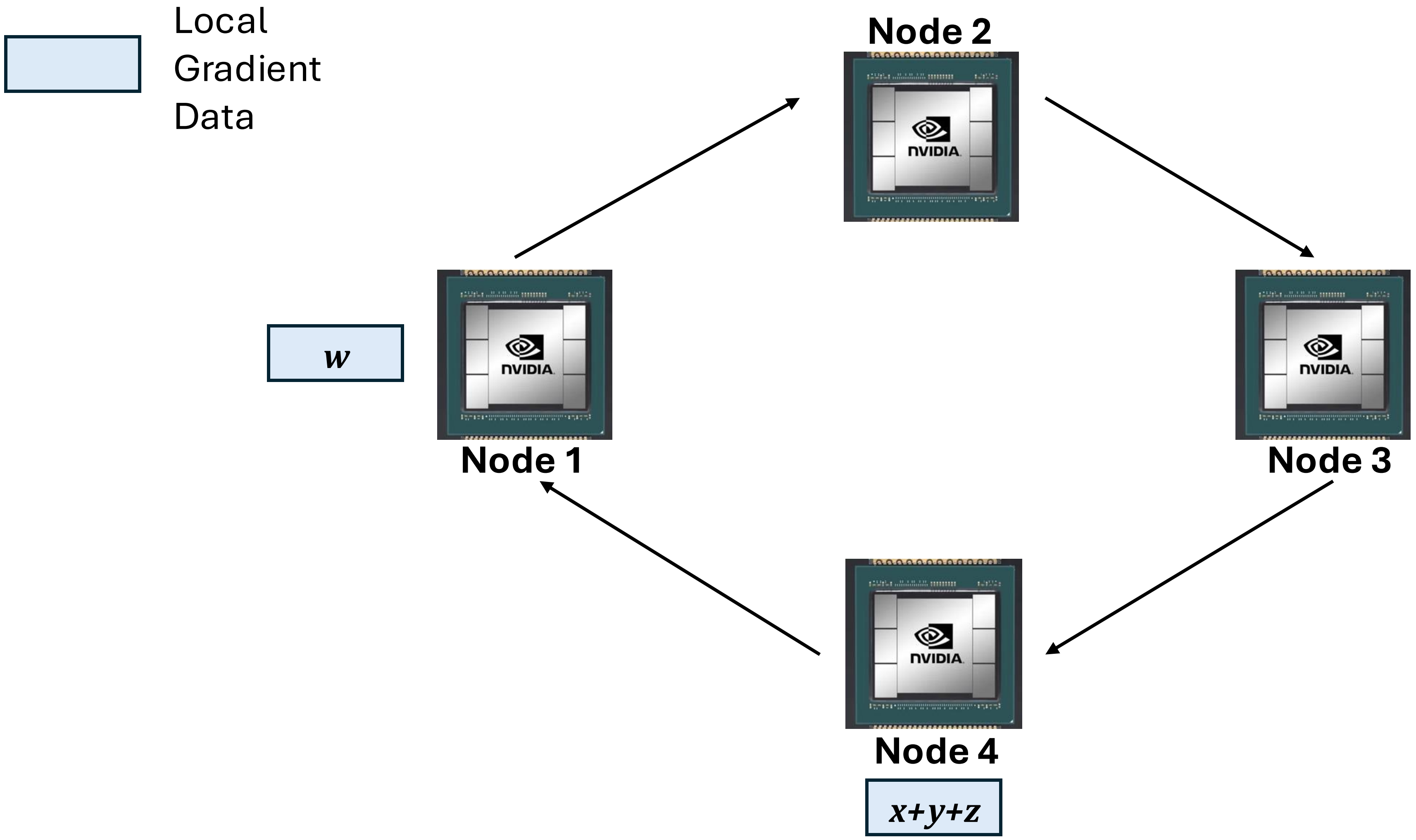
Node 3

$x+y$

Node 4

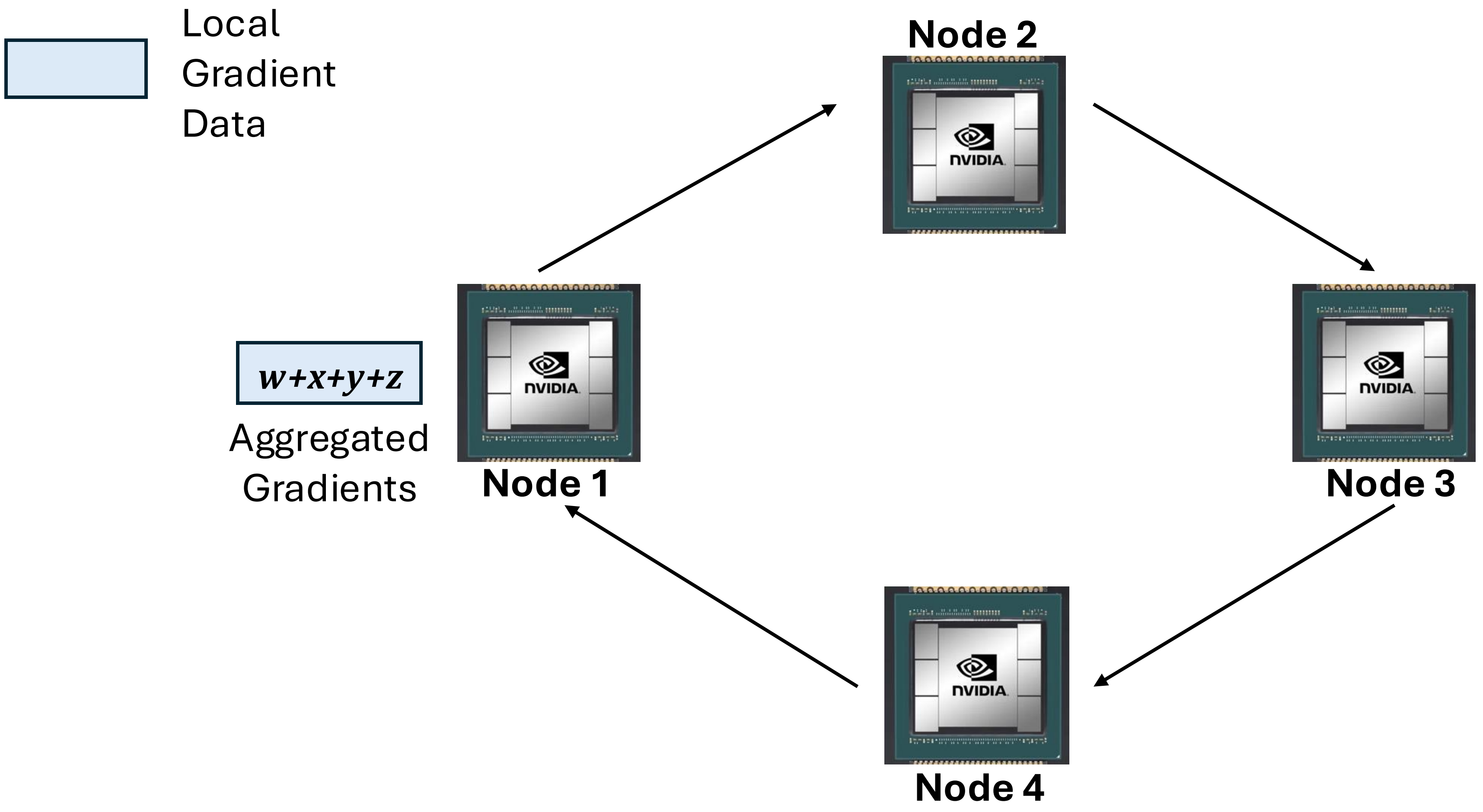
z

Existing Architectures: Ring AR

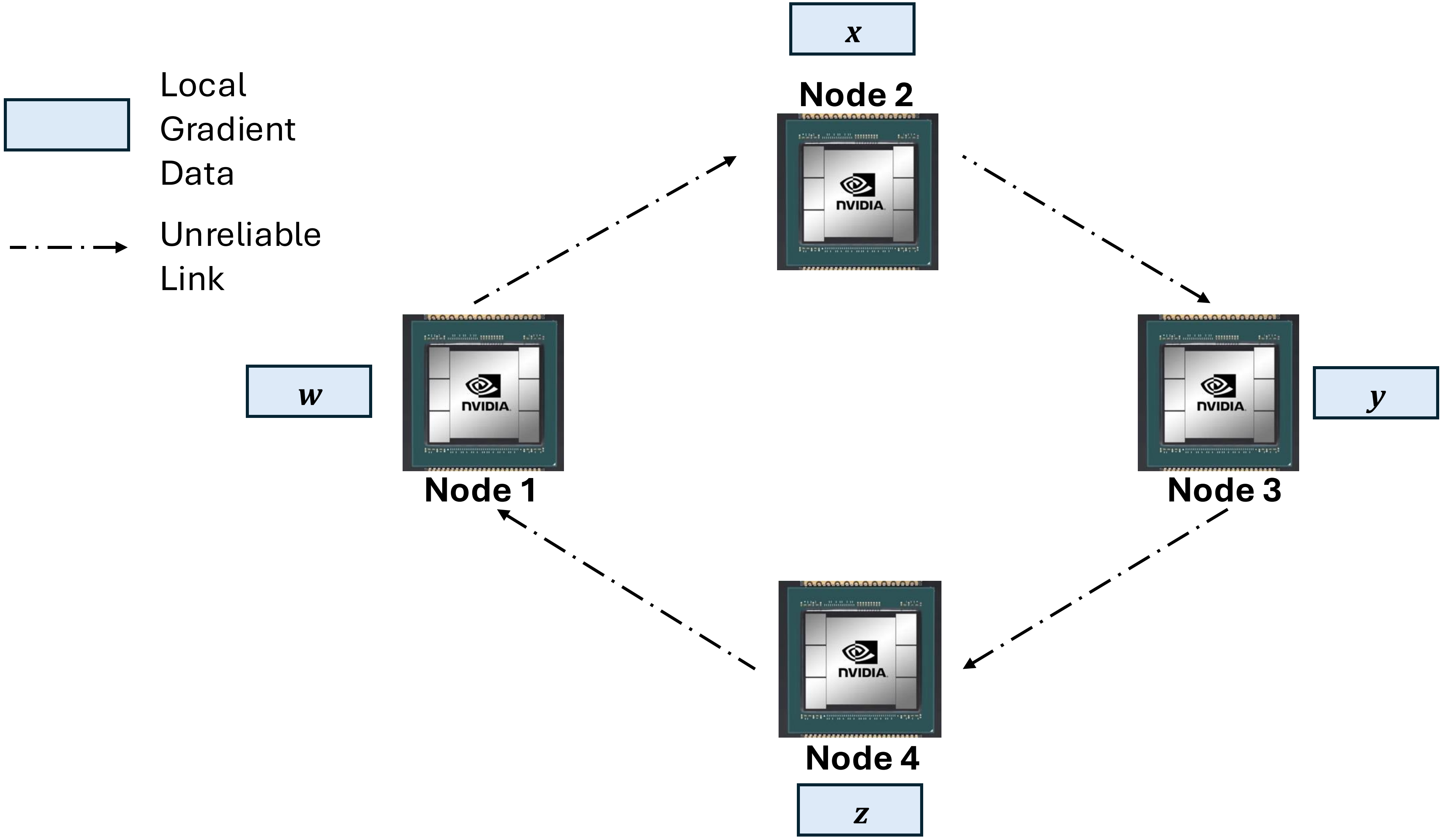


▭ Local Gradient Data

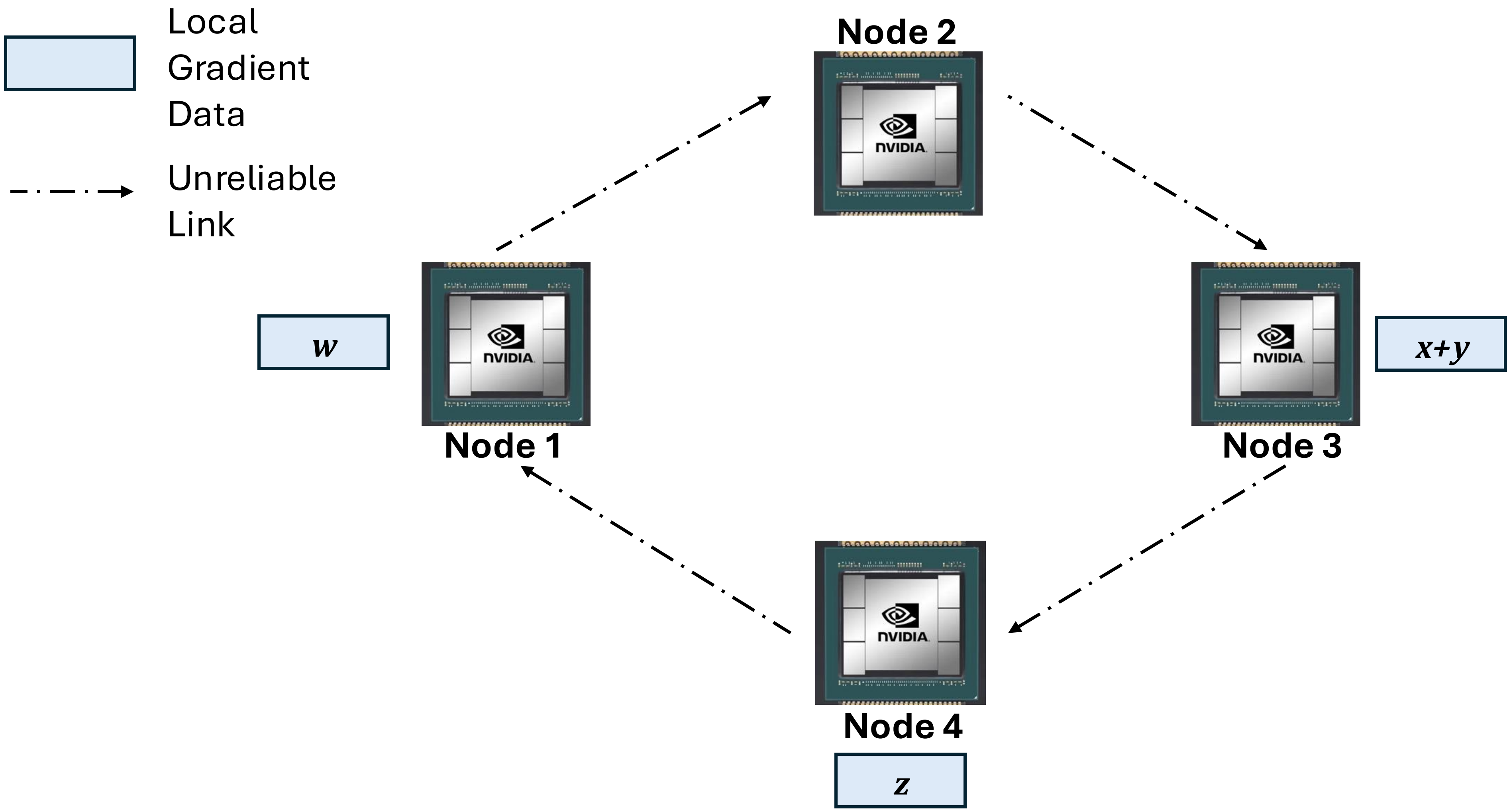
Existing Architectures: Ring AR



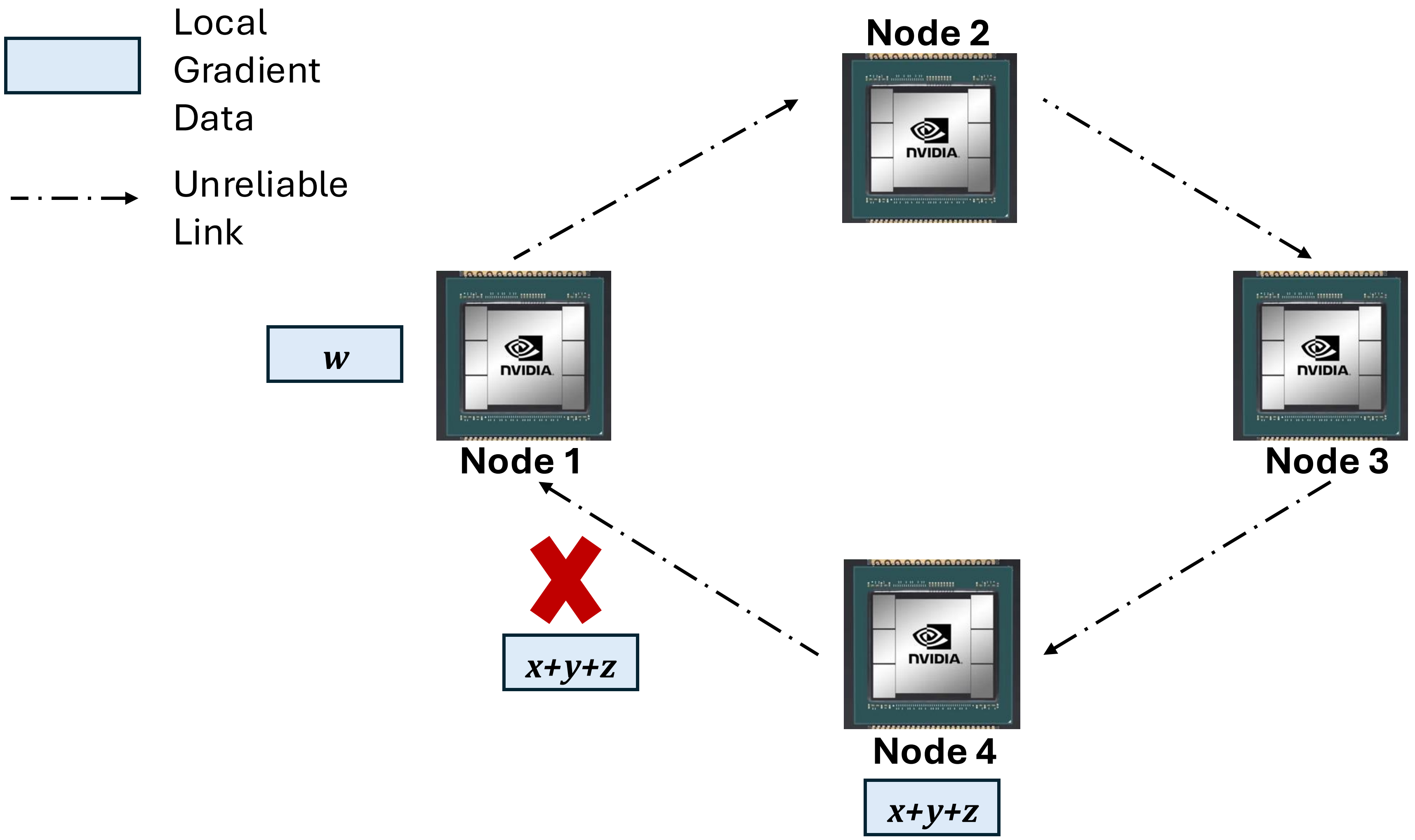
Existing Architectures: Ring AR



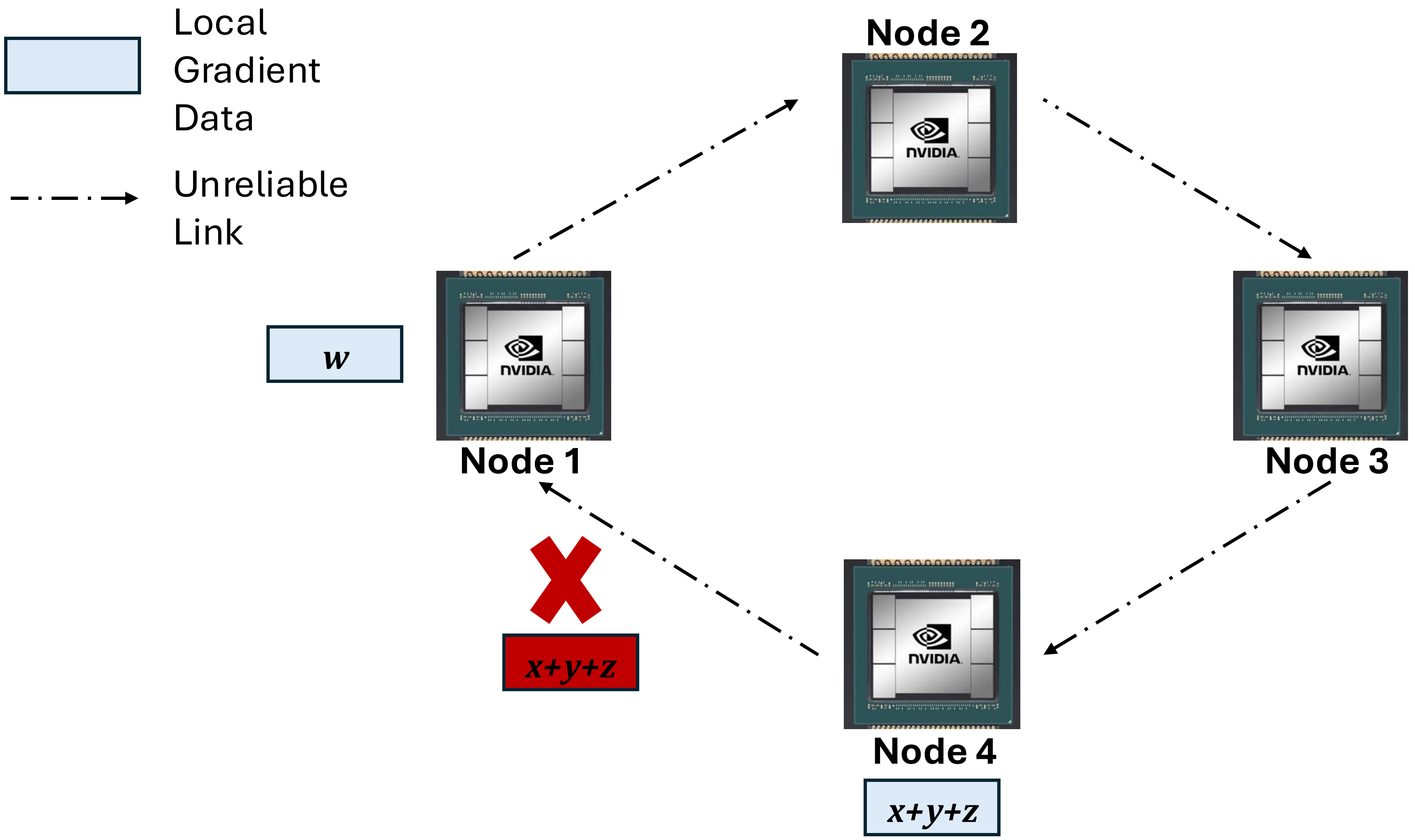
Existing Architectures: Ring AR



Existing Architectures: Ring AR



Existing Architectures: Ring AR



Local Gradient Data

Unreliable Link

Node 2

Node 1

Node 3

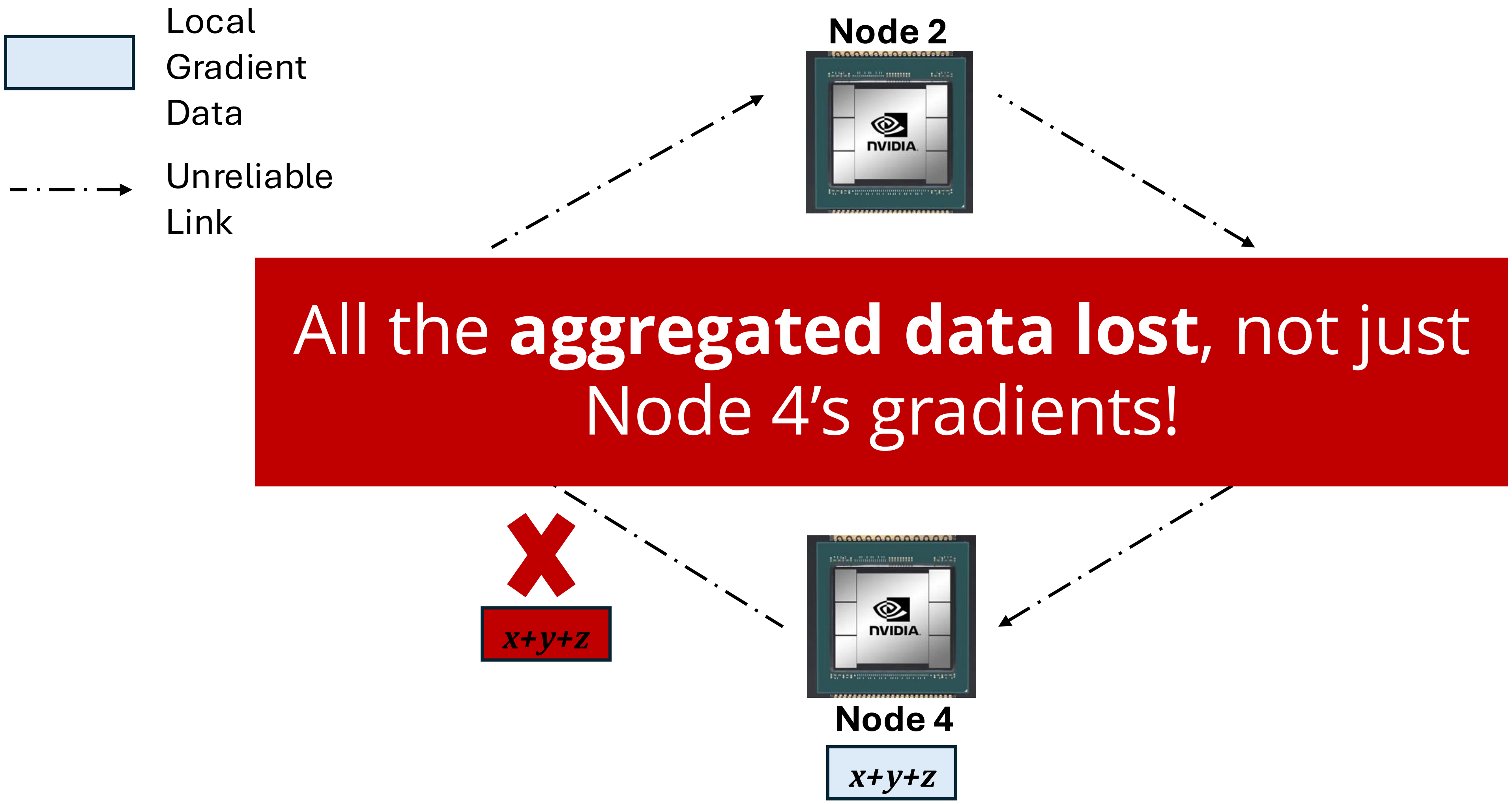
Node 4

w

x+y+z

x+y+z

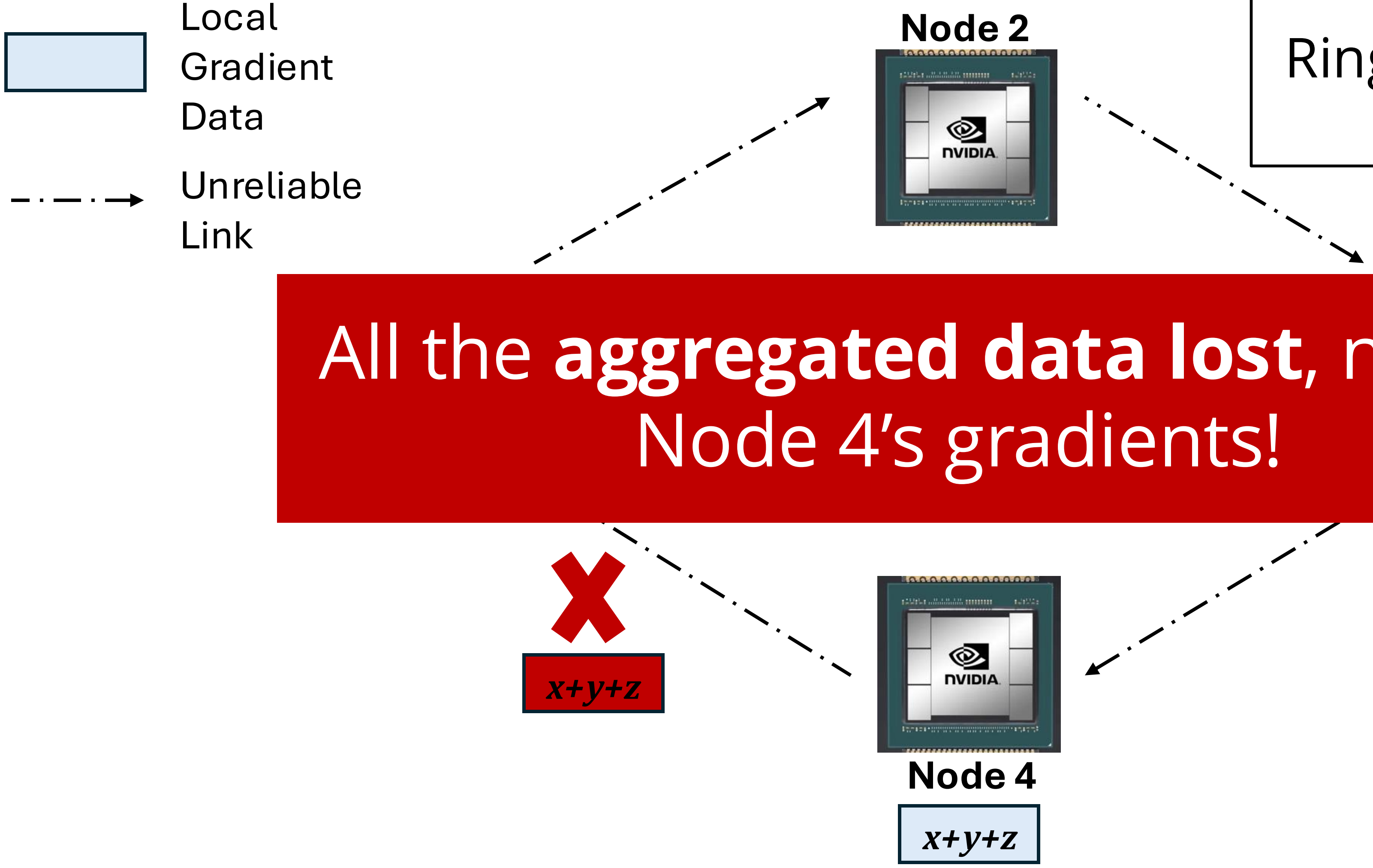
Existing Architectures: Ring AR



All the **aggregated data lost**, not just Node 4’s gradients!

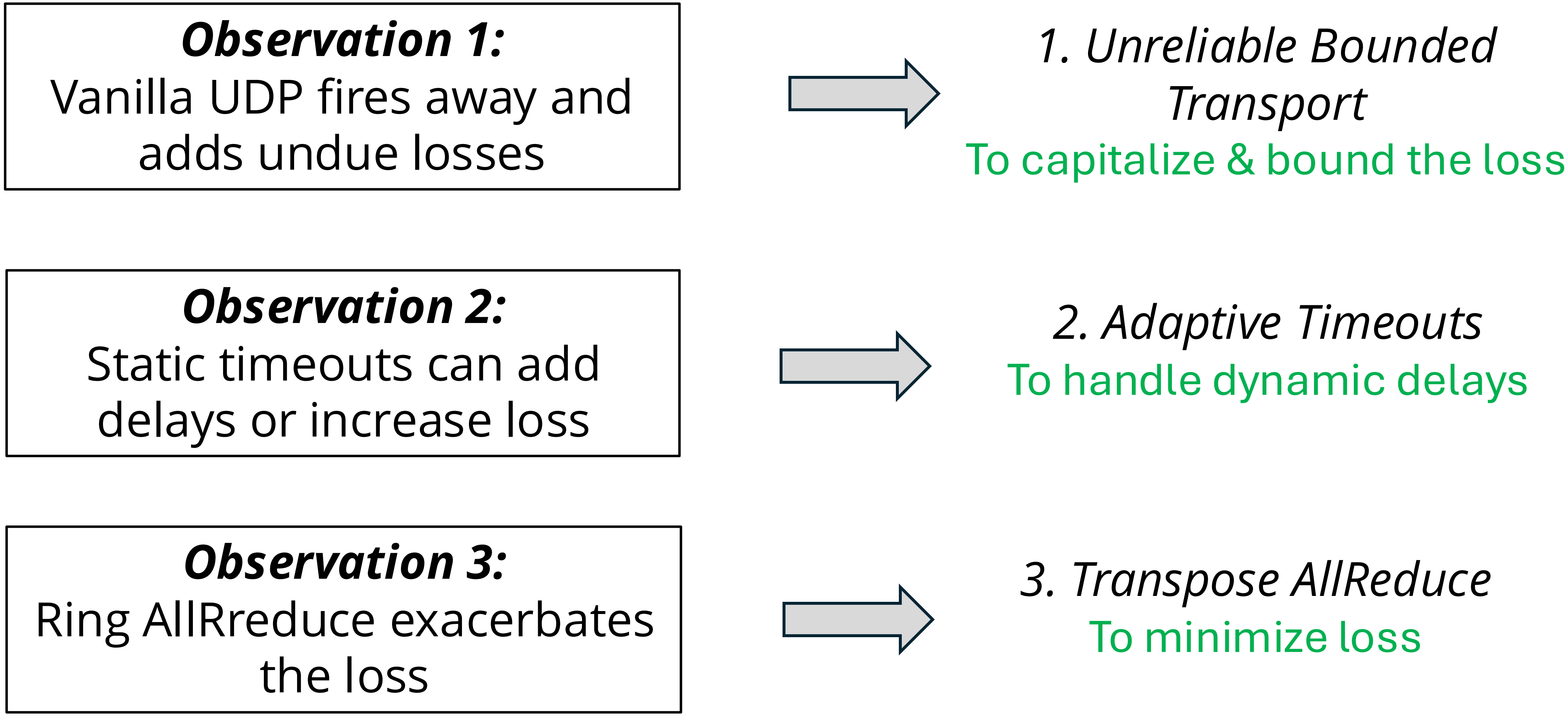
Existing Architectures: Ring AR

Observation 3:
Ring AllRreduce exacerbates the loss!



All the **aggregated data lost**, not just Node 4’s gradients!

OptiReduce Design



Observation 1:
Vanilla UDP fires away and adds undue losses

1. Unreliable Bounded Transport
To capitalize & bound the loss

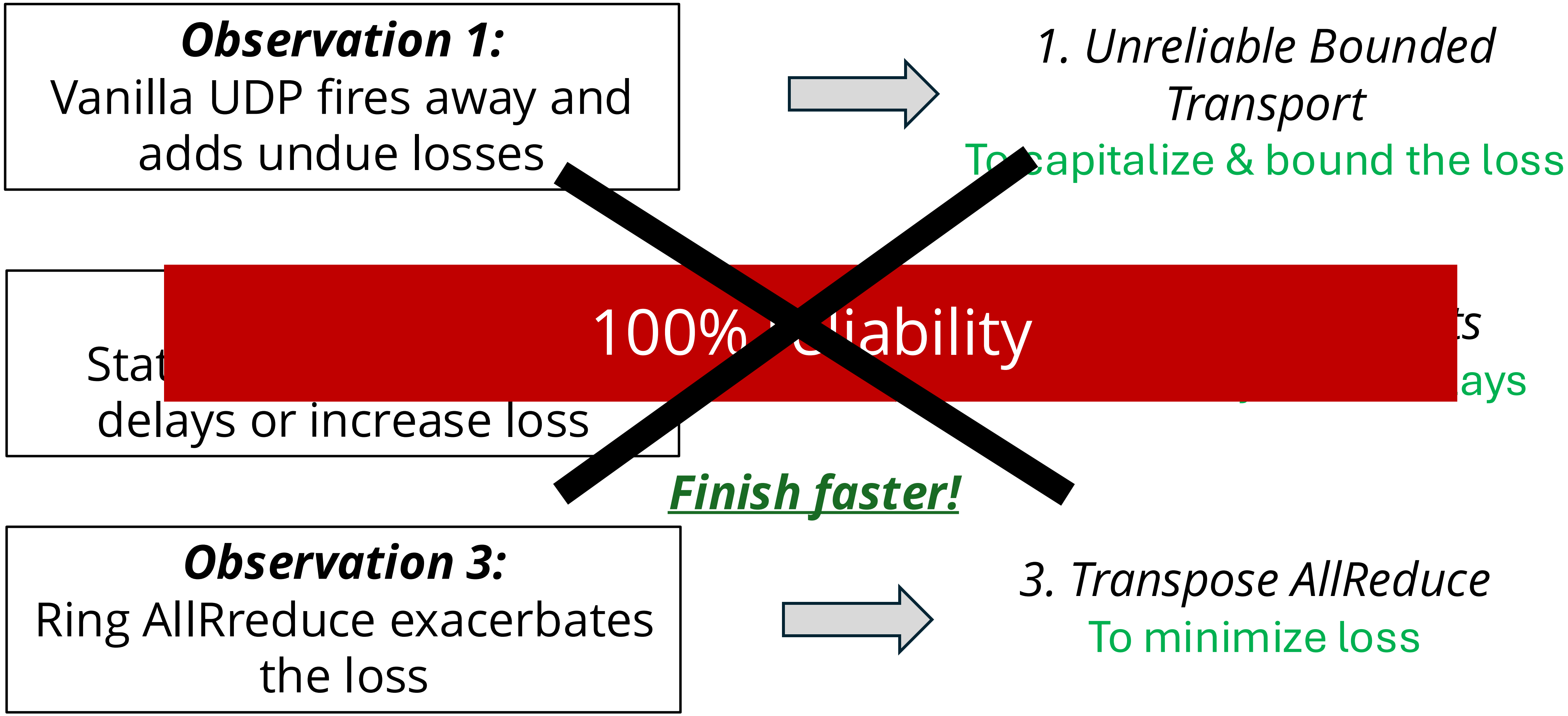
Observation 2:
Static timeouts can add delays or increase loss

2. Adaptive Timeouts
To handle dynamic delays

Observation 3:
Ring AllRreduce exacerbates the loss

3. Transpose AllReduce
To minimize loss

OptiReduce Design



Observation 1:
Vanilla UDP fires away and adds undue losses

1. Unreliable Bounded Transport
To capitalize & bound the loss

Stat
delays or increase loss

ts
ays

100% Reliability

Finish faster!

Observation 3:
Ring AllRreduce exacerbates the loss

3. Transpose AllReduce
To minimize loss

OptiReduce Design

Observation 1:
Vanilla UDP fires away and adds undue losses

1. Unreliable Bounded Transport
To capitalize & bound the loss

Observation 2:
Static timeouts can add delays or increase loss

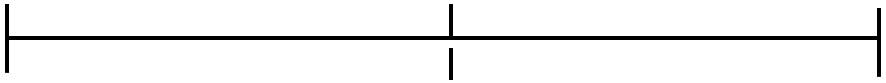
2. Adaptive Timeouts
To handle dynamic delays

Observation 1:
Ring AllRreduce exacerbates the loss

1. Transpose AllReduce
To minimize loss

Unreliable Bounded Transport

Reliability adds to the tail ***Firing away exacerbates loss***



<u>Features</u>	TCP	UDP
- Retransmissions	✓	*X*
- In-order delivery	✓	*X*
- Congestion Control	✓	*X*
- Flow Control	✓	*X*

Unreliable Bounded Transport

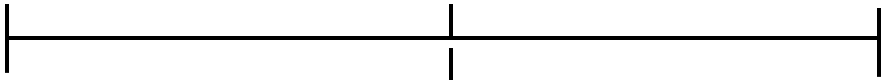
Reliability adds to the tail ***Firing away exacerbates loss***

Features	TCP	UBT	UDP
- Retransmissions	✓	X	X
- In-order delivery	✓	X	X
- Congestion Control	✓		X
- Flow Control	✓		X

🎯Direct placement of packets, enabled by *Offset* field!

Unreliable Bounded Transport

Reliability adds to the tail *Firing away exacerbates loss*



Features	TCP	UBT	UDP
- Retransmissions	✓	X	X
- In-order delivery	✓	X	X
- Congestion Control	✓	✓	X
- Flow Control	✓	✓	X

Direct placement of packets, enabled by *Offset* field!

Timely-inspired rate-control to throttle the sending!

Unreliable Bounded Transport

Effect of this approach:

Capitalizes and bounds the loss

Reliability adds to the tail

<u>Features</u>	TCP	UBT	UDP
- Retransmissions	✓	X	X
- In-order delivery	✓	X	X
- Congestion Control	✓	✓	X
- Flow Control	✓	✓	X

Direct placement of packets, enabled by *Offset* field!

Timely-inspired rate-control to throttle the sending!

Unreliable Bounded Transport

Effect of this approach:
Capitalizes and bounds the loss

Reliability adds to the tail

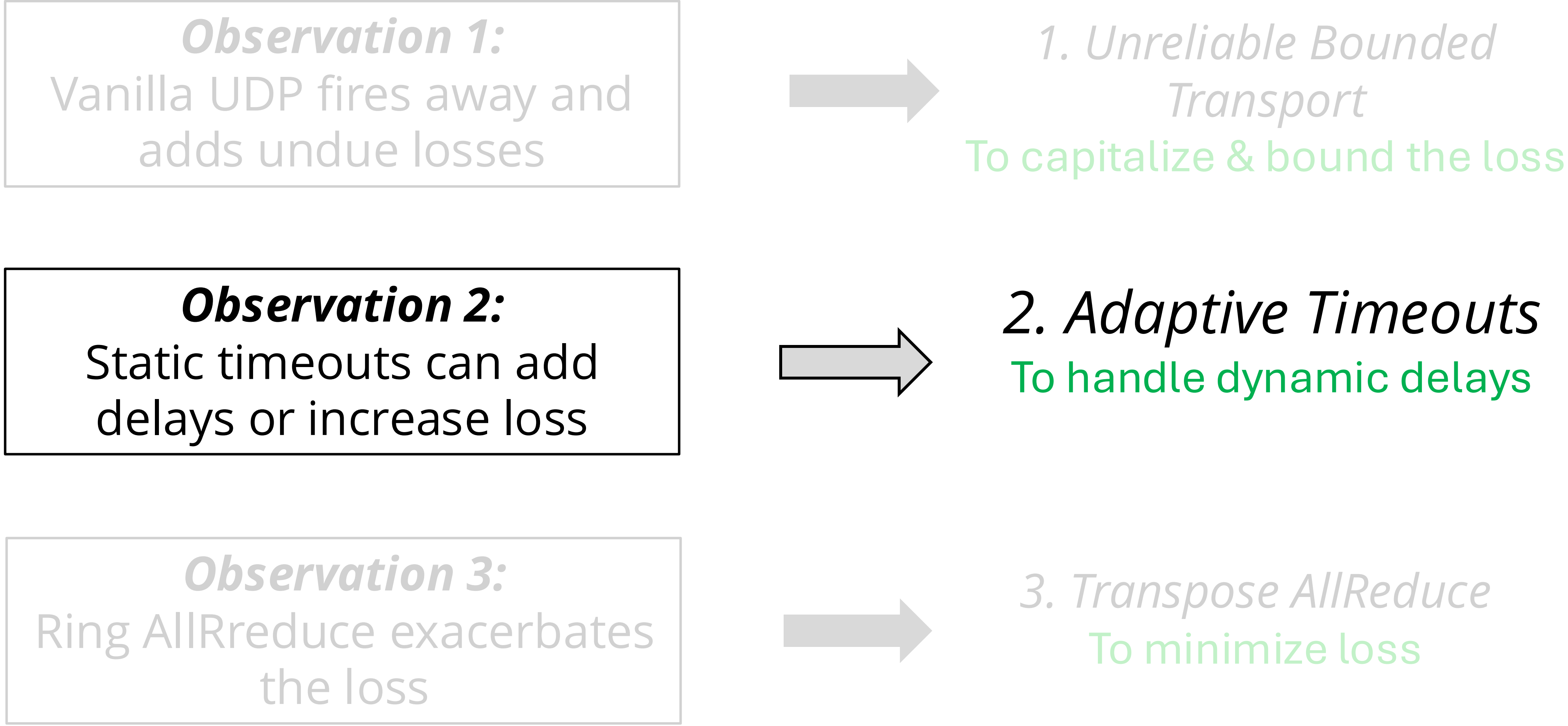
Features	TCP	UBT	UDP
- Retransmissions	✓	X	X
- In-order delivery	✓	X	X
- Congestion Control	✓	✓	X
- Flow Control	✓	✓	X

🎯 Direct placement of packets, enabled by *Offset* field!

Timely-inspired rate-control to throttle the sending!

Receiver-driven multicast, signaled using *Incast* field!

OptiReduce Design



Observation 1:
Vanilla UDP fires away and adds undue losses

1. Unreliable Bounded Transport
To capitalize & bound the loss

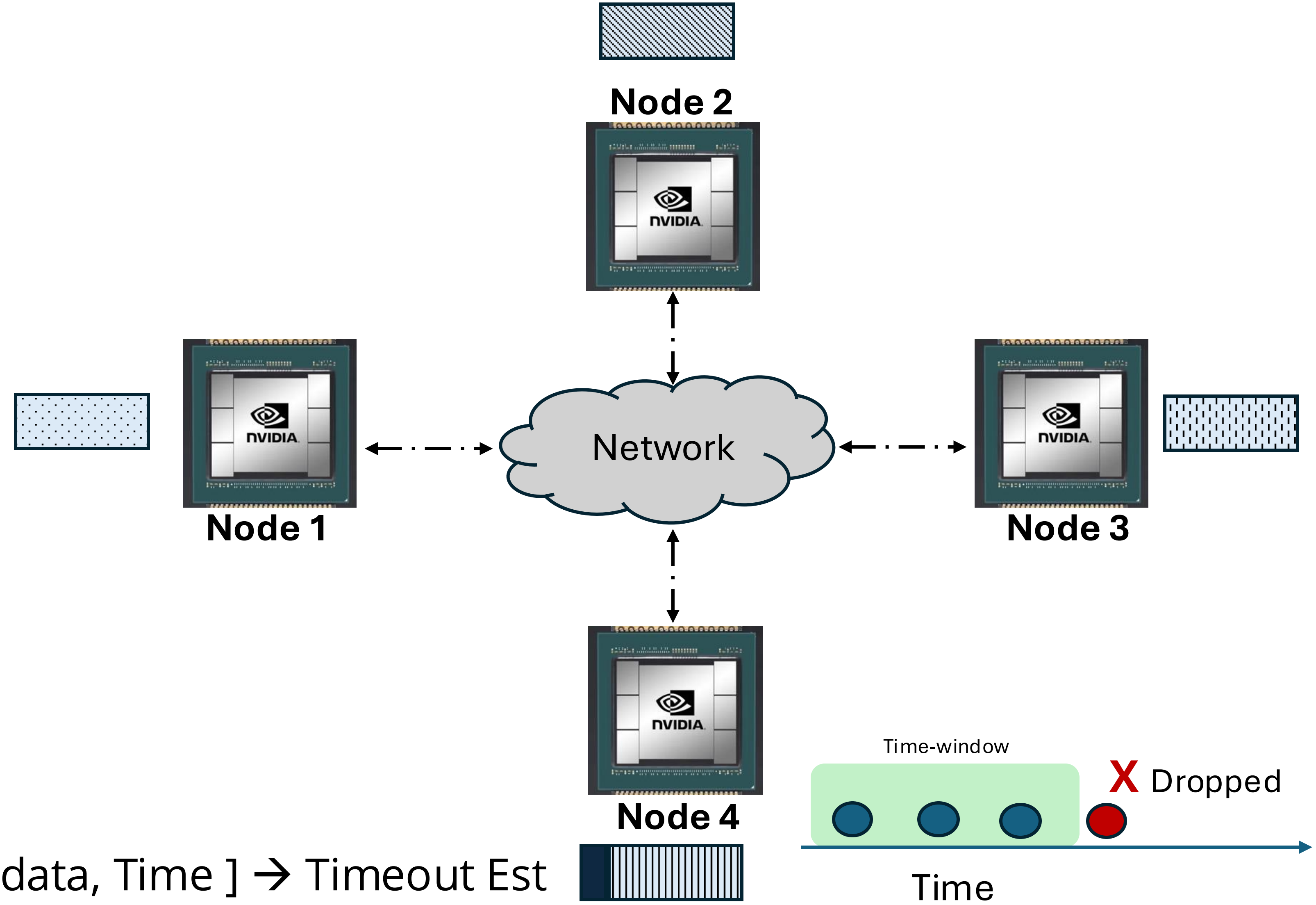
Observation 2:
Static timeouts can add delays or increase loss

2. Adaptive Timeouts
To handle dynamic delays

Observation 3:
Ring AllRreduce exacerbates the loss

3. Transpose AllReduce
To minimize loss

Adaptive Timeouts



[Recvd_data, Time] → Timeout Est

Adaptive Timeouts

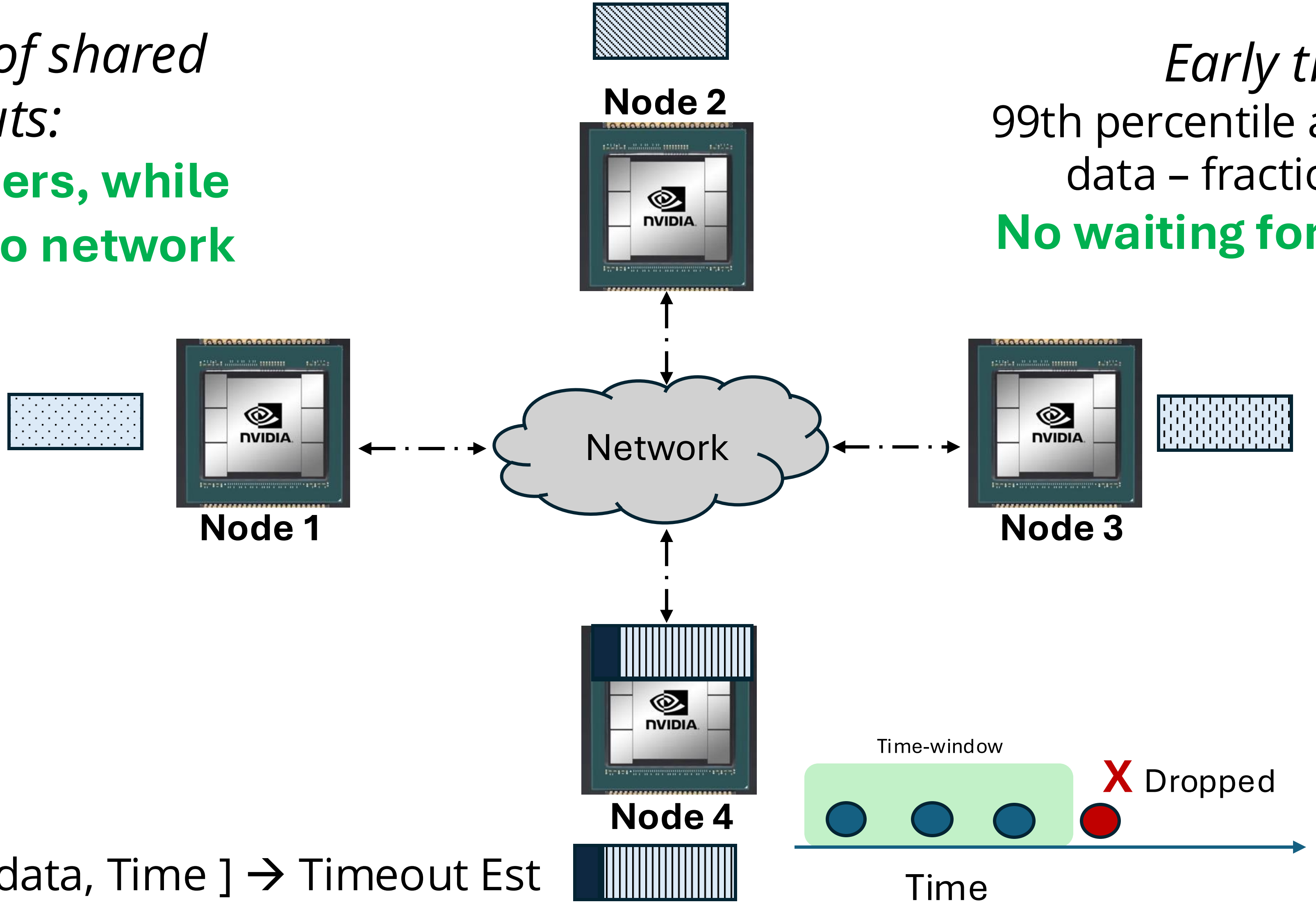
Use median of shared timeouts:

Ignore stragglers, while still reacting to network

Early timeout:

99th percentile arrived & no new data – fraction of timeout

No waiting for dropped data



[Recvd_data, Time] → Timeout Est

OptiReduce Design

Observation 1:
Vanilla UDP fires away and adds undue losses

→ *1. Unreliable Bounded Transport*
To capitalize & bound the loss

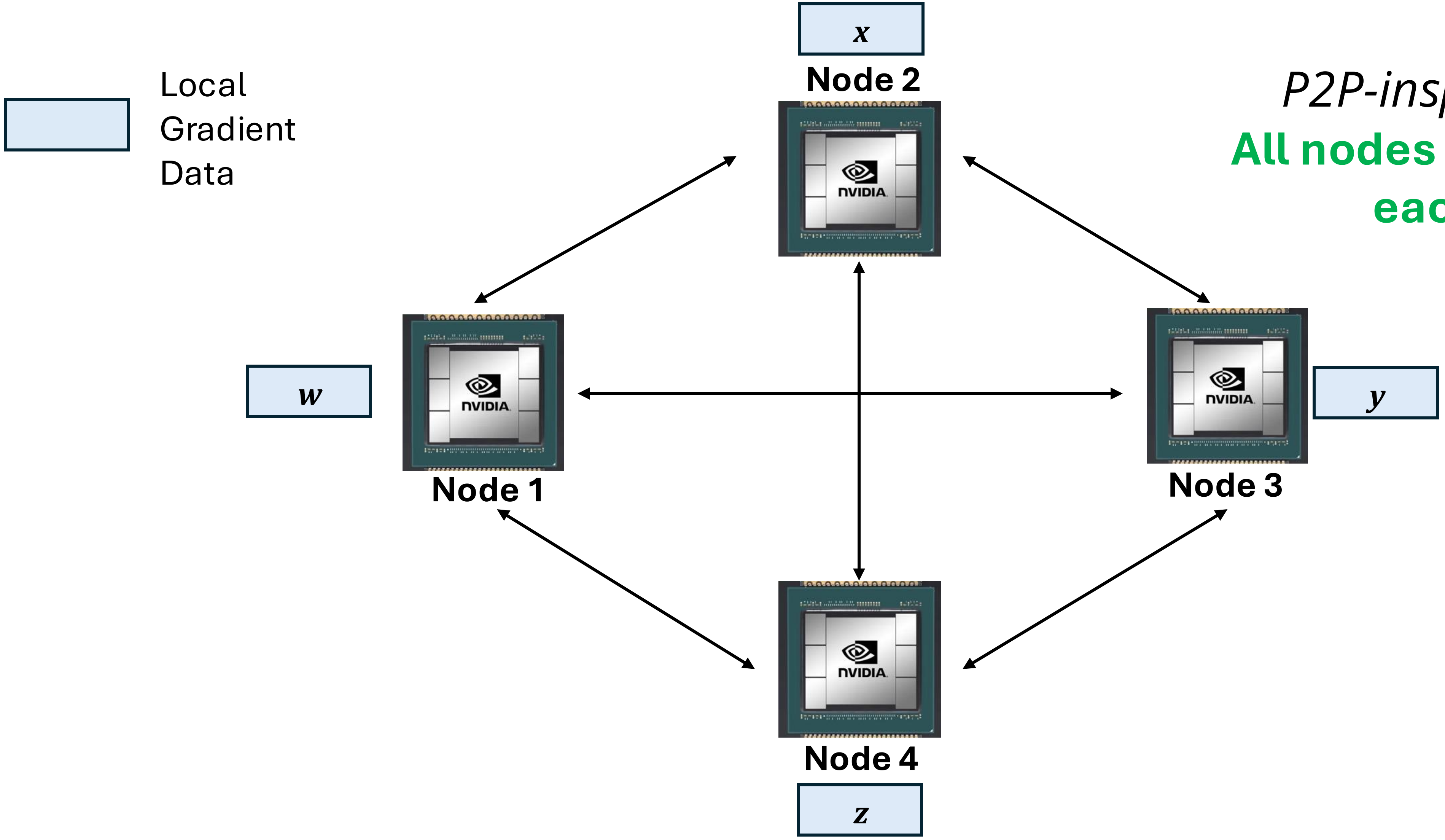
Observation 2:
Static timeouts can add delays or increase loss

→ *2. Adaptive Timeouts*
To handle dynamic delays

Observation 3:
Ring AllRreduce exacerbates the loss

→ *3. Transpose AllReduce*
To minimize loss

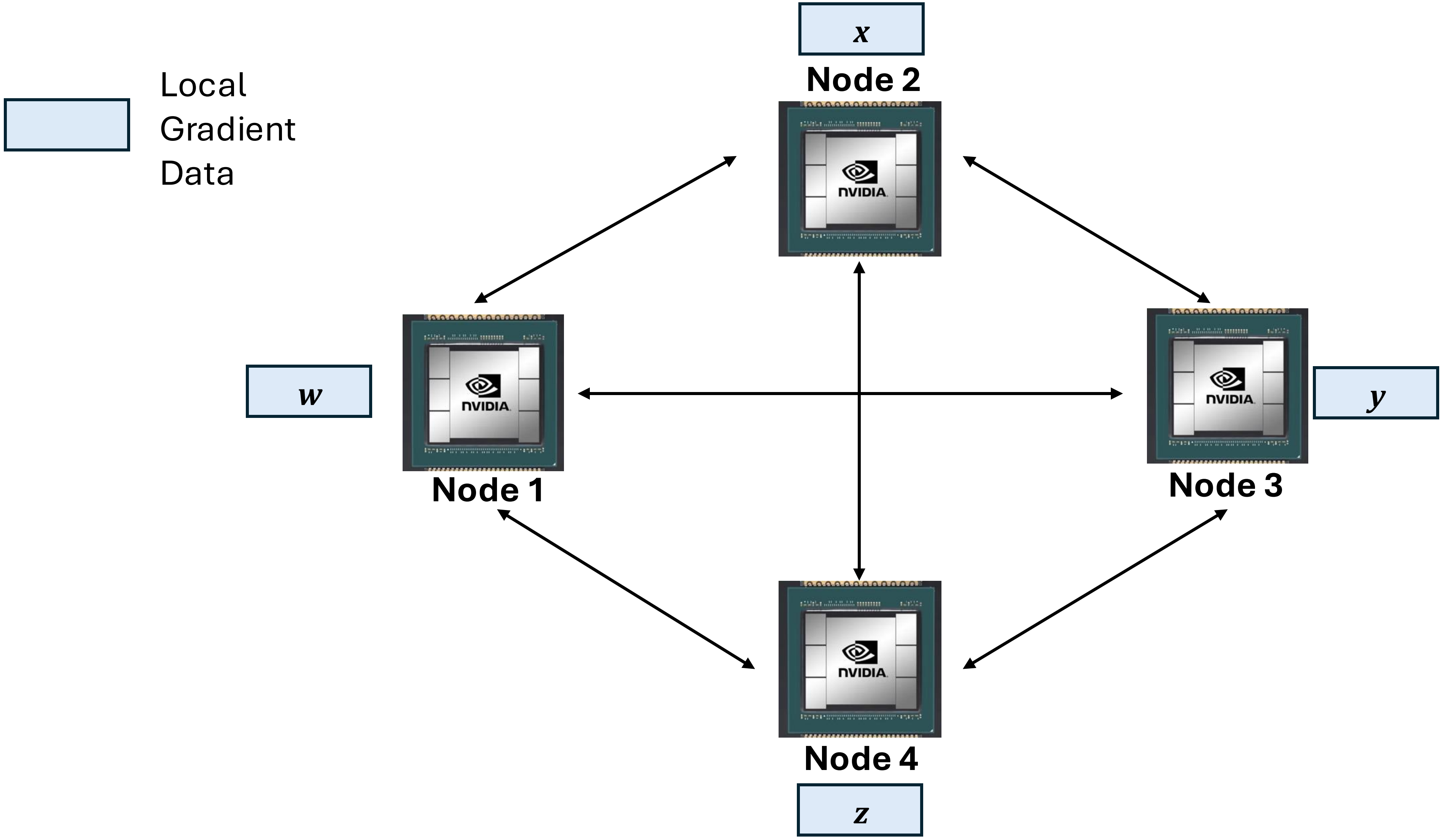
Transpose AllReduce (TAR)



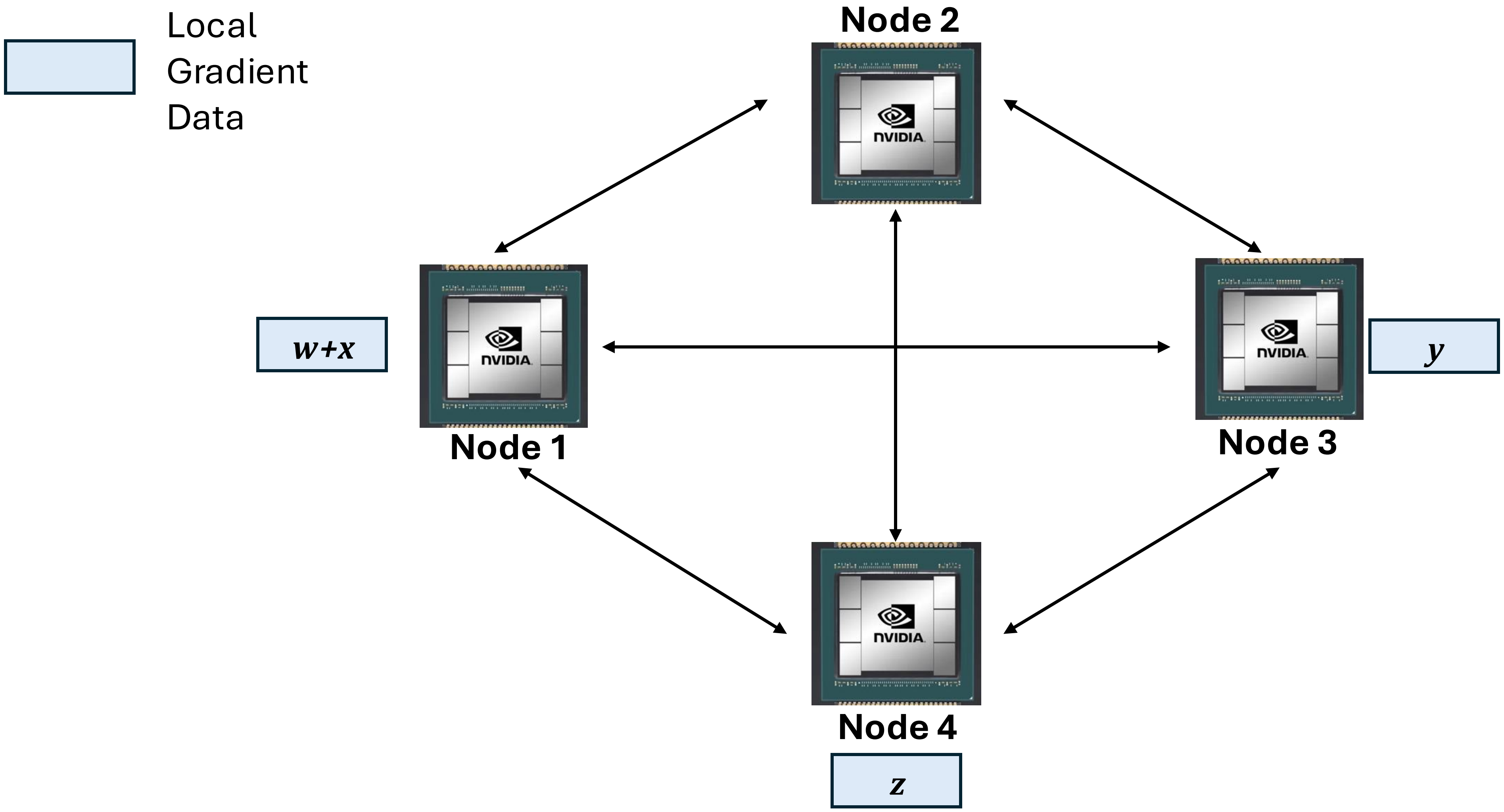
P2P-inspired design:

All nodes directly talk to each other!

Transpose AllReduce (TAR)

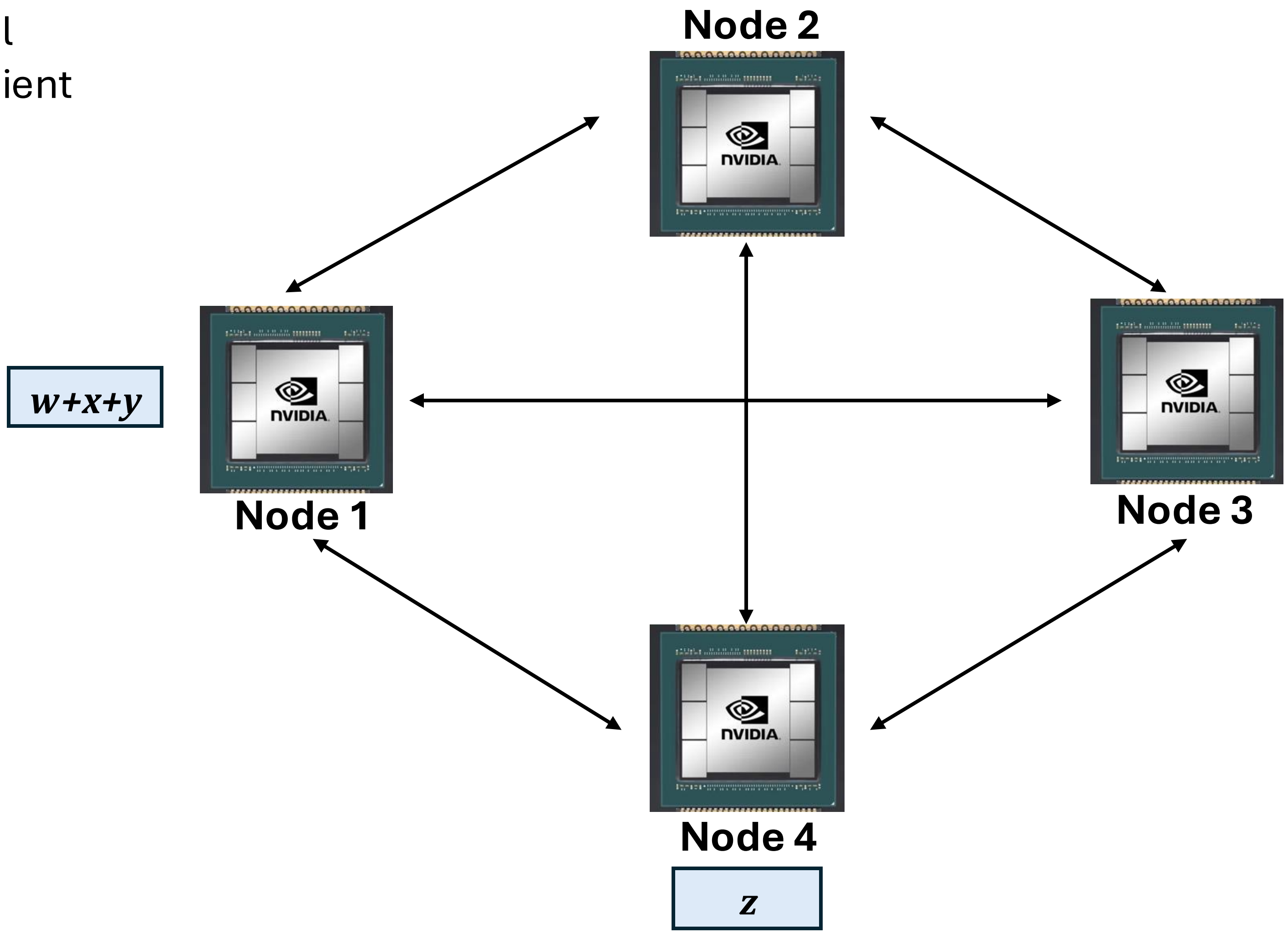


Transpose AllReduce (TAR)

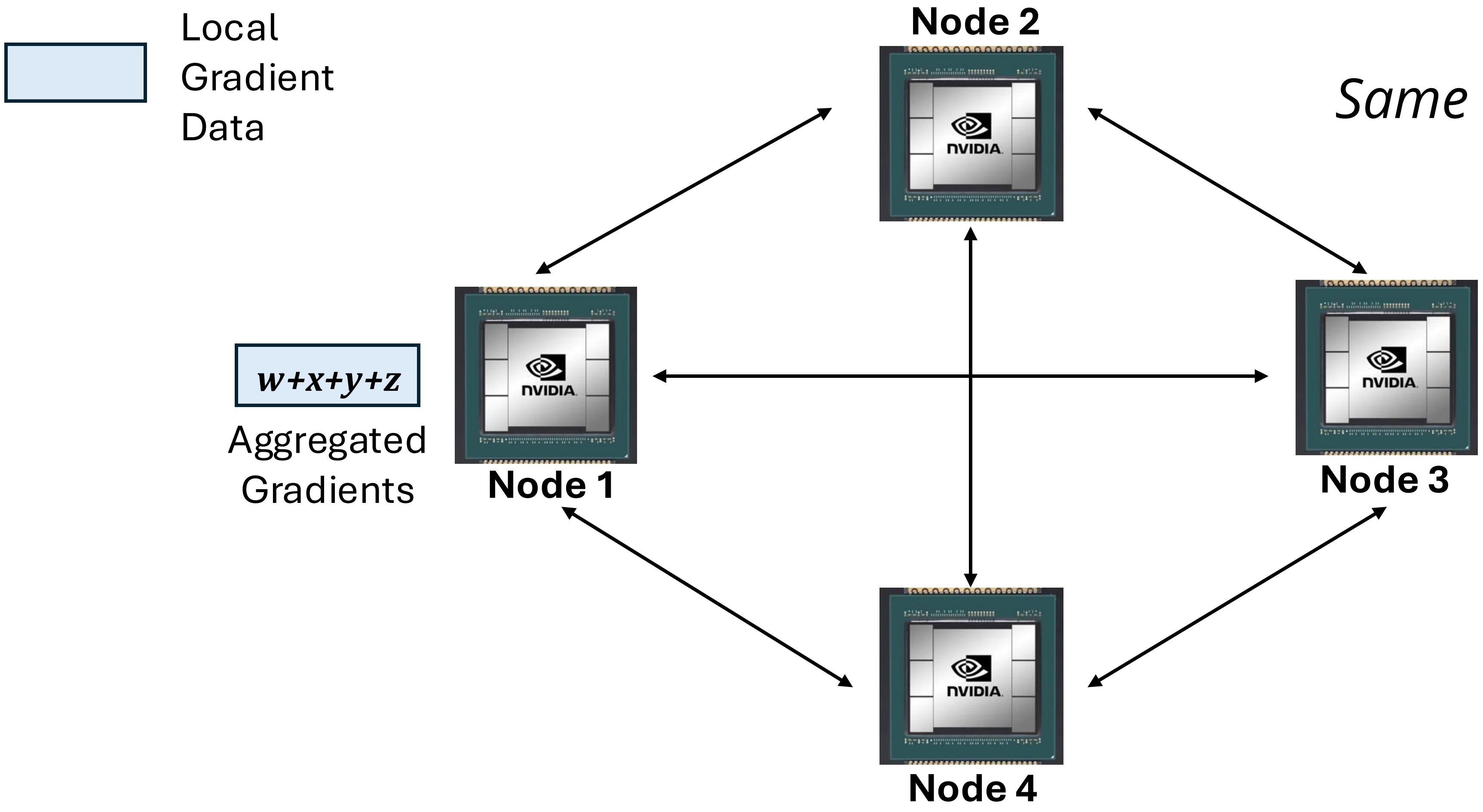


Transpose AllReduce (TAR)

▭ Local Gradient Data

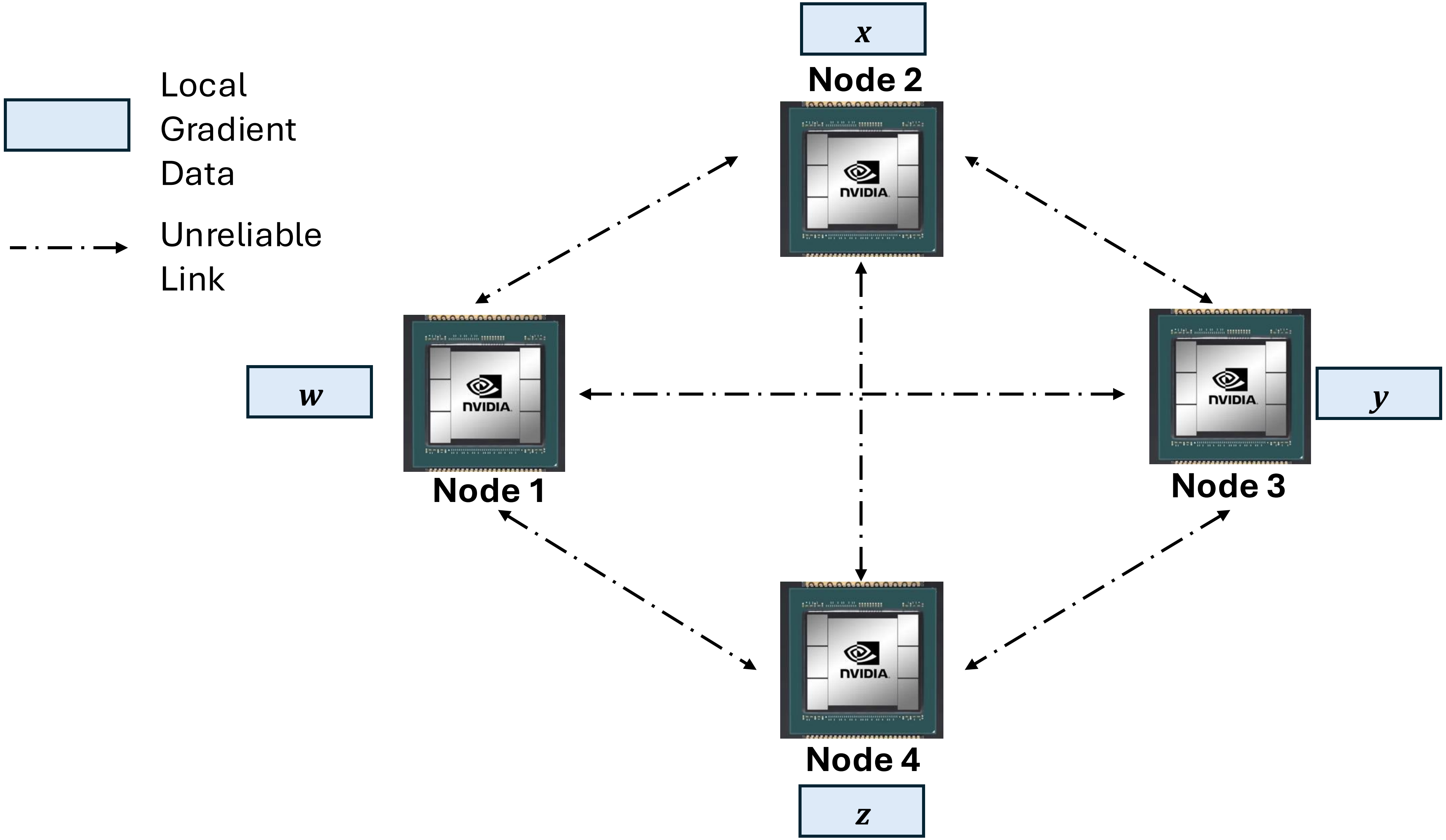


Transpose AllReduce (TAR)

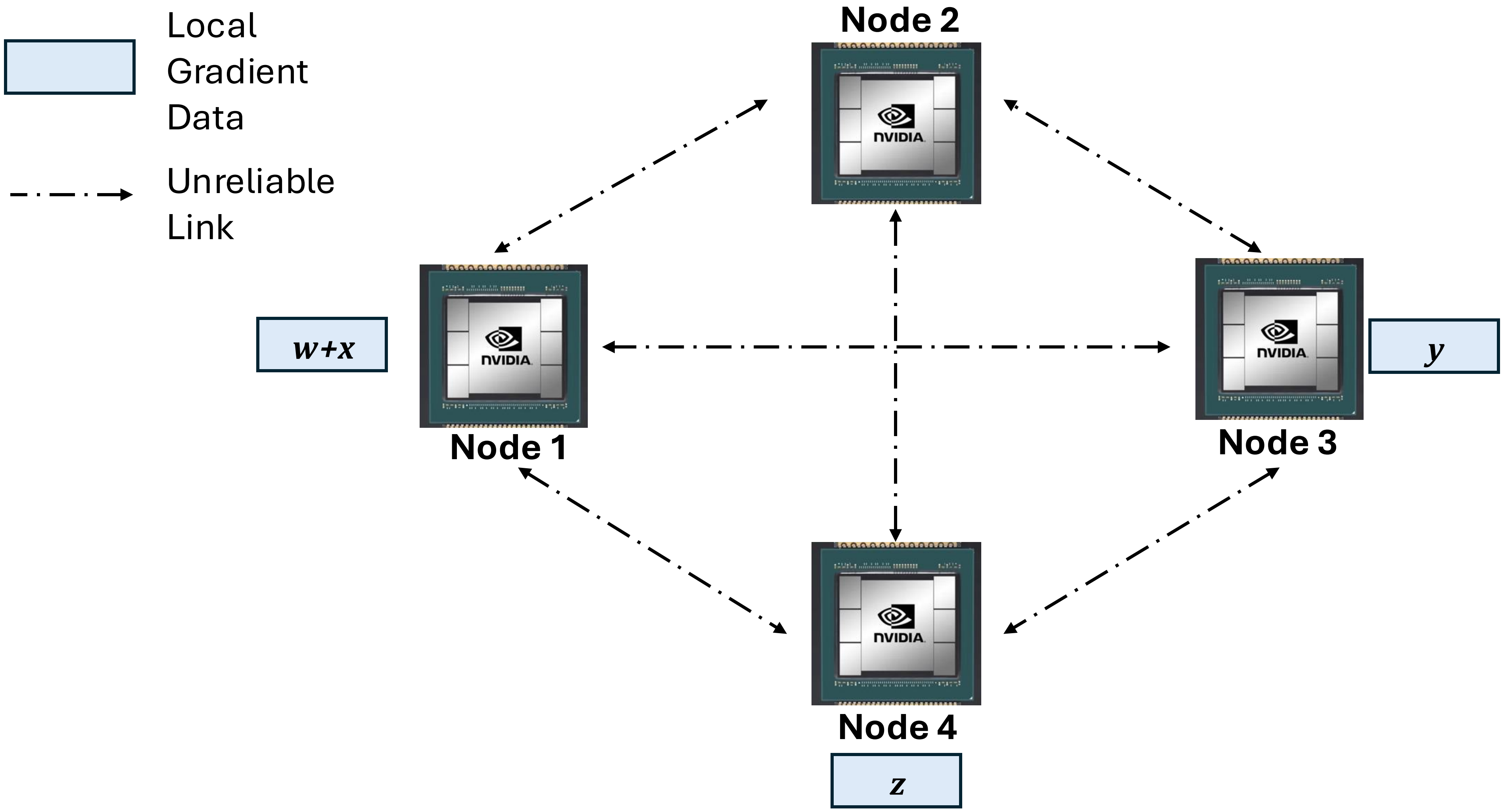


Same number of rounds as Ring AllReduce

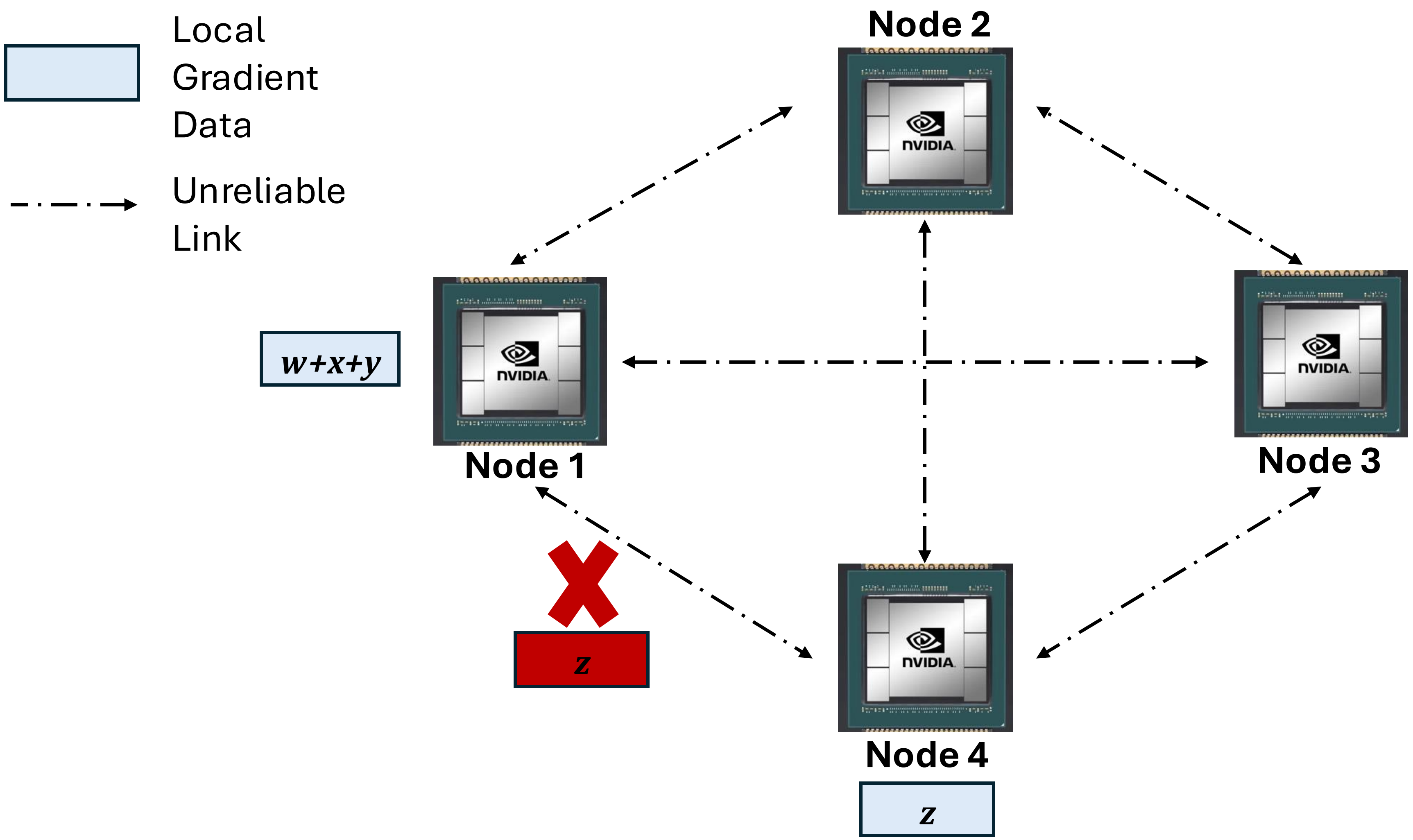
Transpose AllReduce (TAR)



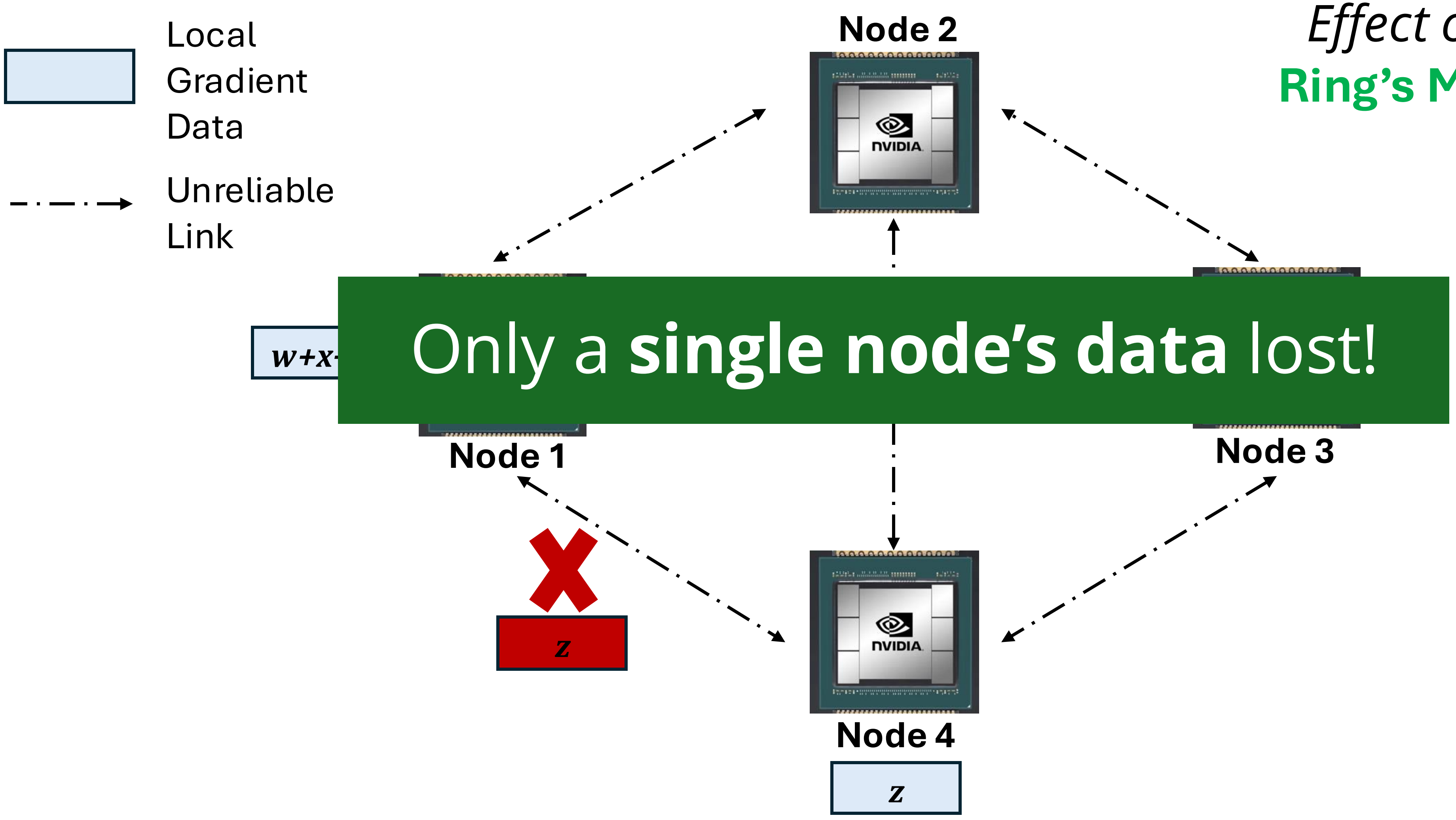
Transpose AllReduce (TAR)



Transpose AllReduce (TAR)



Transpose AllReduce (TAR)



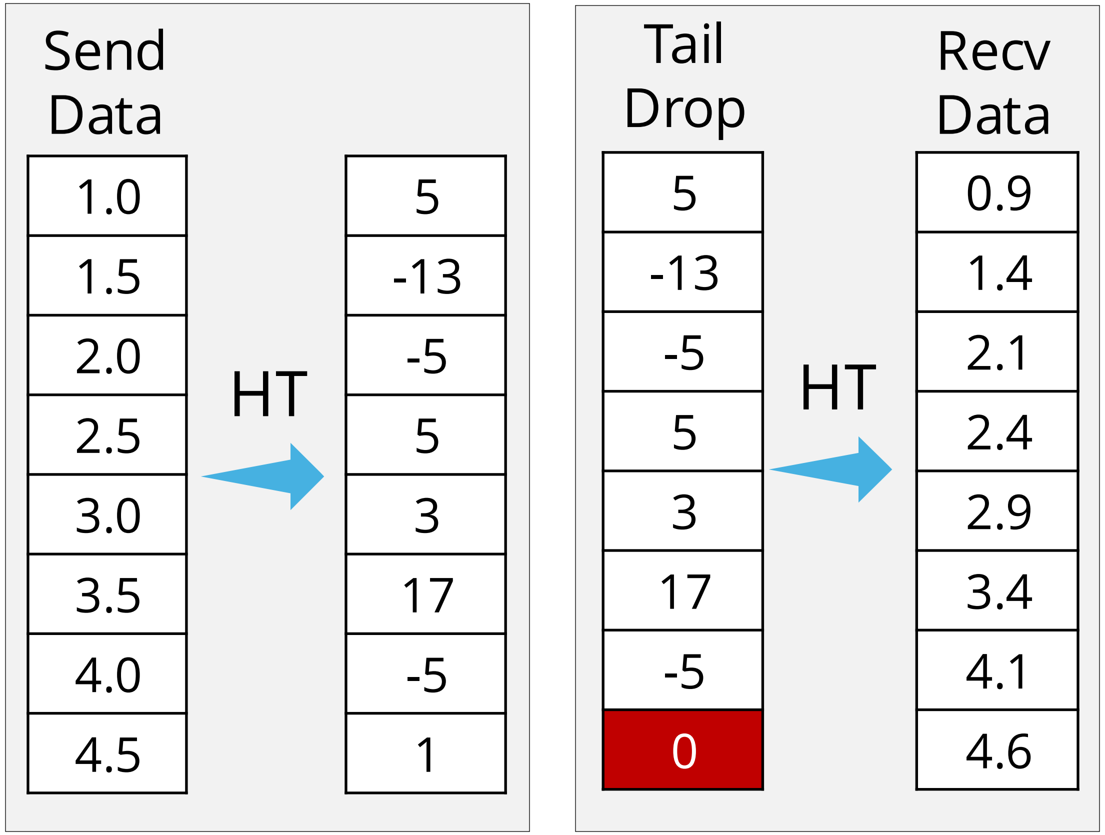
Only a **single node’s data** lost!

Effect of this approach:

Ring’s MSE 7x larger than TAR’s!

Optimizations for Sustaining Accuracy

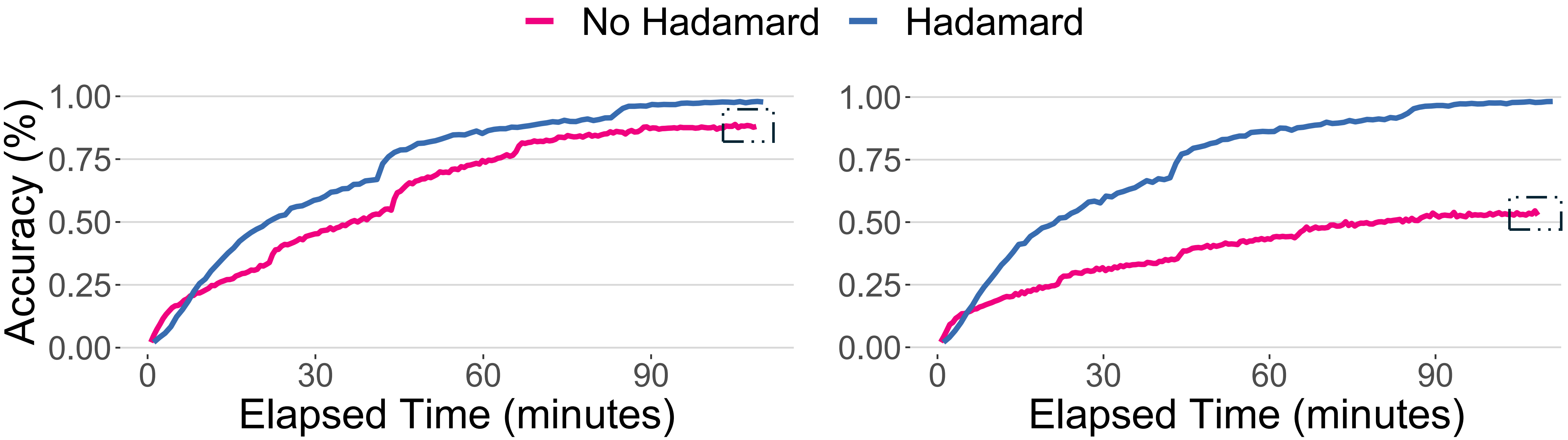
- Safeguards against excessive loss
- Randomized Hadamard Transform (RHT)
- Encode data before sending in the network
- Approximate lost gradients, essentially recovering data



Send Data		Tail Drop	Recv Data
1.0	5	5	0.9
1.5	-13	-13	1.4
2.0	-5	-5	2.1
2.5	5	5	2.4
3.0	3	3	2.9
3.5	17	17	3.4
4.0	-5	-5	4.1
4.5	1	0	4.6

“Hadamard transform applied to a vector of length 8”

Effect of RHT on Accuracy

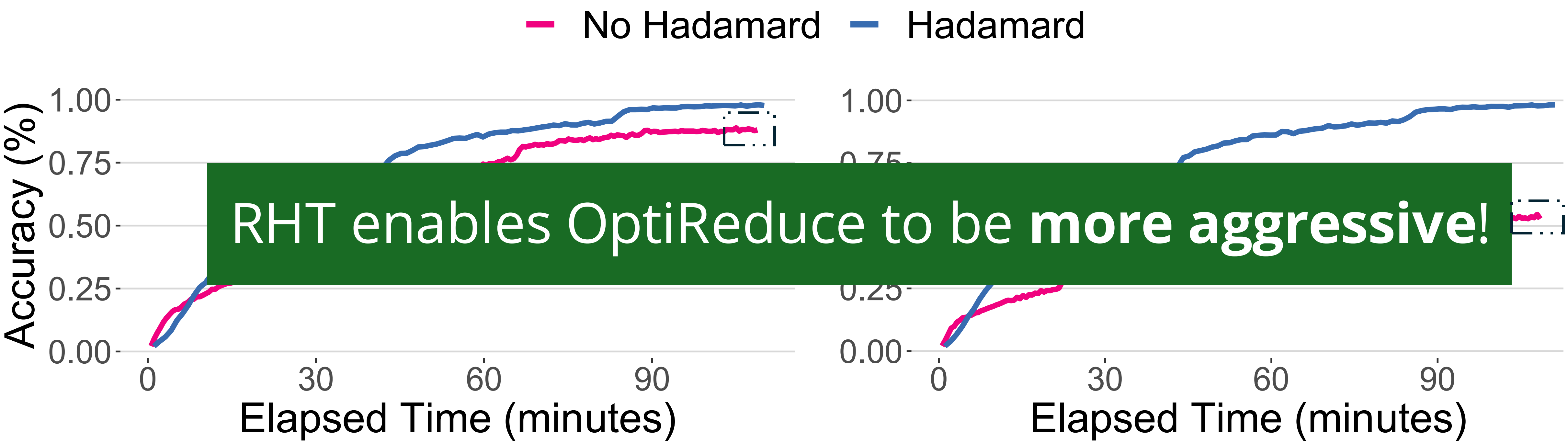


(a) 5% Gradient Drop

(b) 10% Gradient Drop

Model: VGG-19

Effect of RHT on Accuracy



RHT enables OptiReduce to be **more aggressive**!

(a) 5% Gradient Drop

(b) 10% Gradient Drop

Model: VGG-19

Evaluating OptiReduce

🟩 Gloo Ring 🟪 Gloo BCube 🟧 NCCL Ring 🟥 NCCL Tree 🟫 TAR+TCP 🟦 OptiReduce

Baselines

- NCCL
- Gloo

Environments

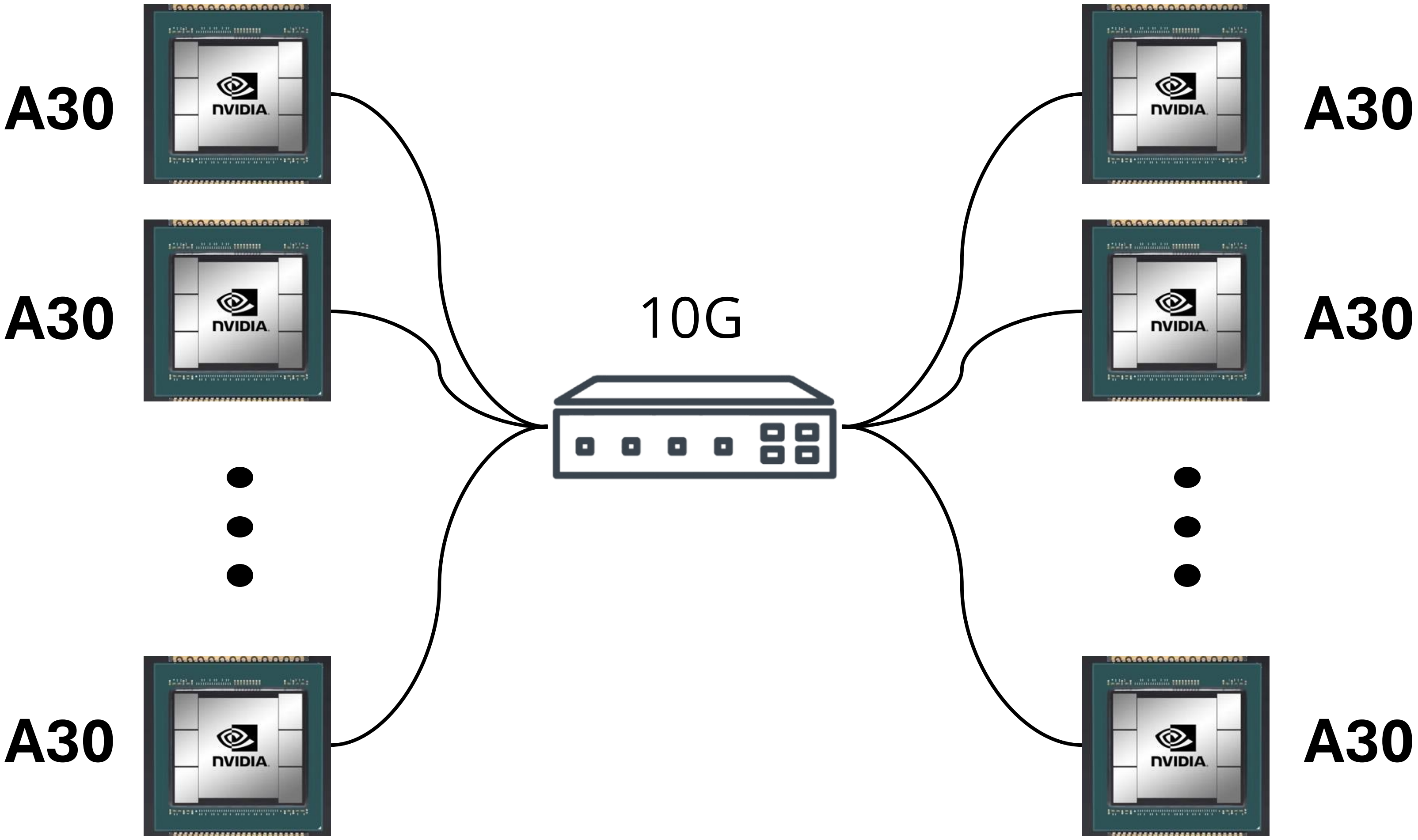
- Cloudlab
- Local Setup

Metrics

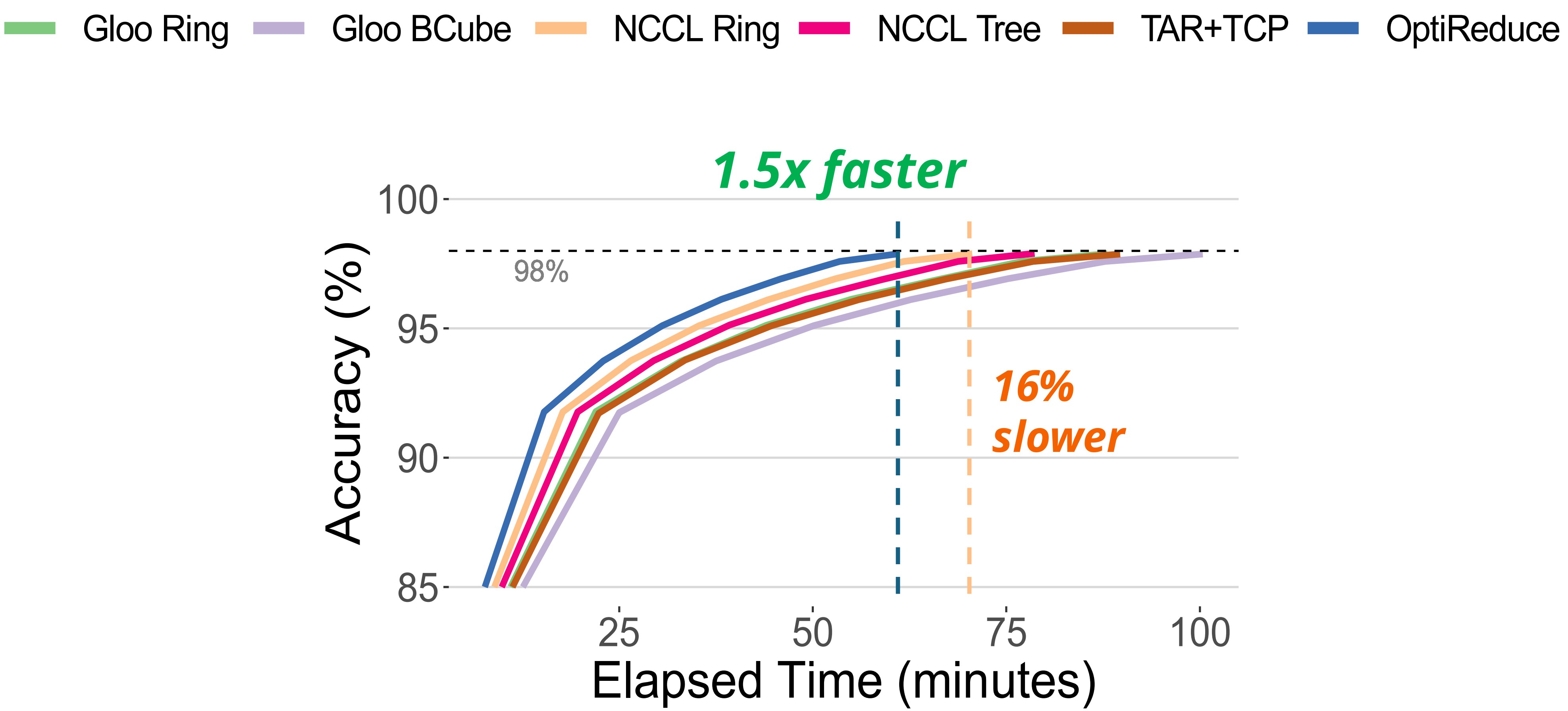
- TTA
- Throughput

Cloudlab: A Public Cloud Environment

🟩 Gloo Ring 🟪 Gloo BCube 🟧 NCCL Ring 🟥 NCCL Tree 🟫 TAR+TCP 🟦 OptiReduce



Cloudlab: A Public Cloud Environment

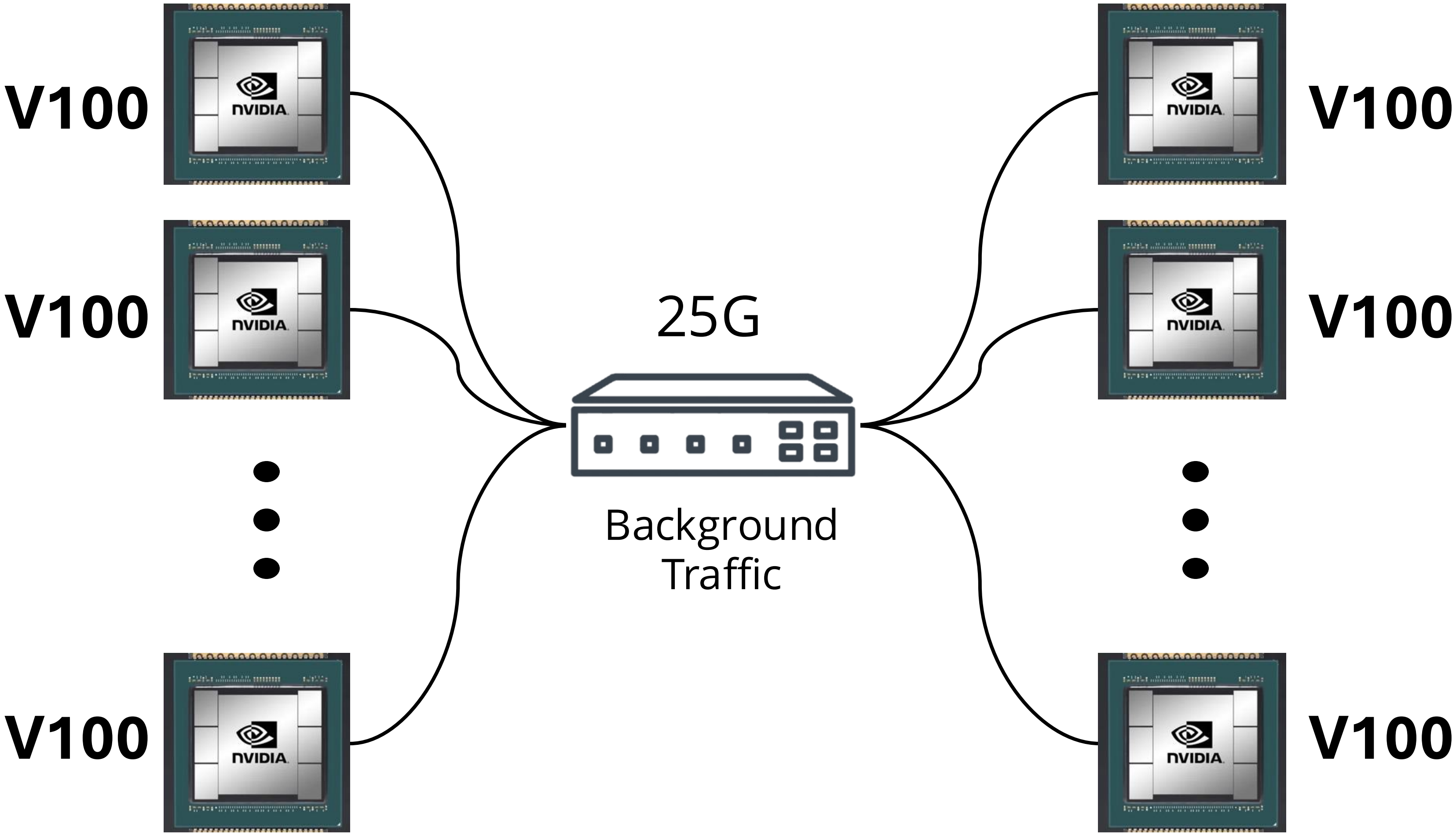


1.5x faster

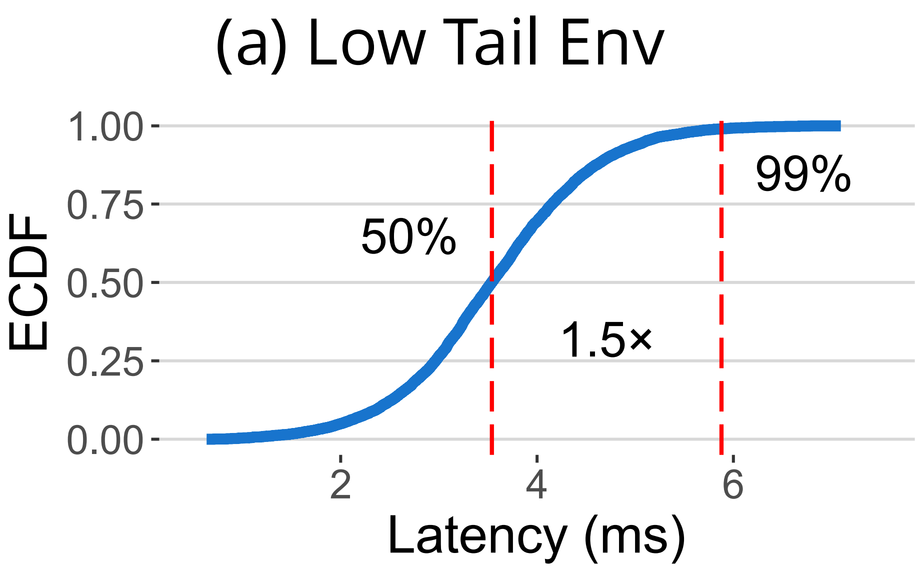
16% slower

TTA GPT-2

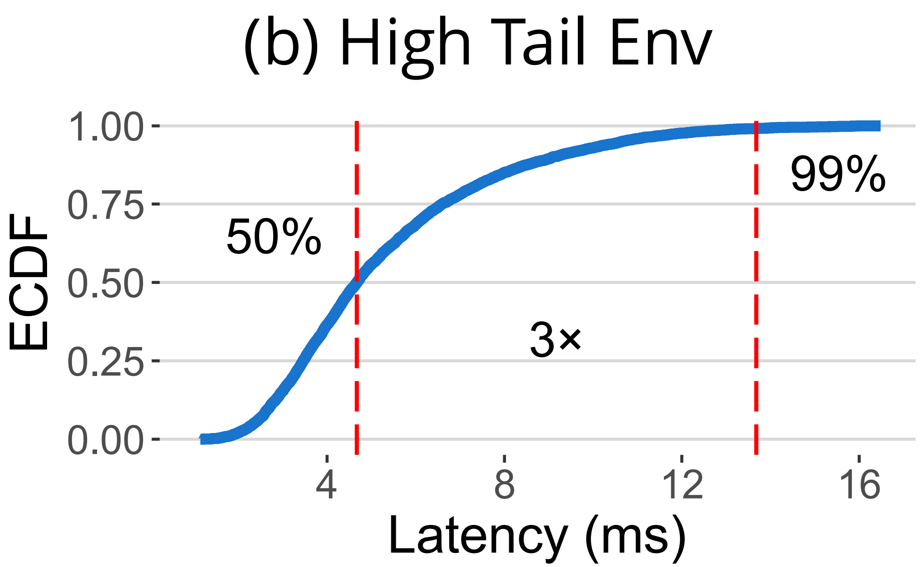
Our Local Setup



Our Local Setup

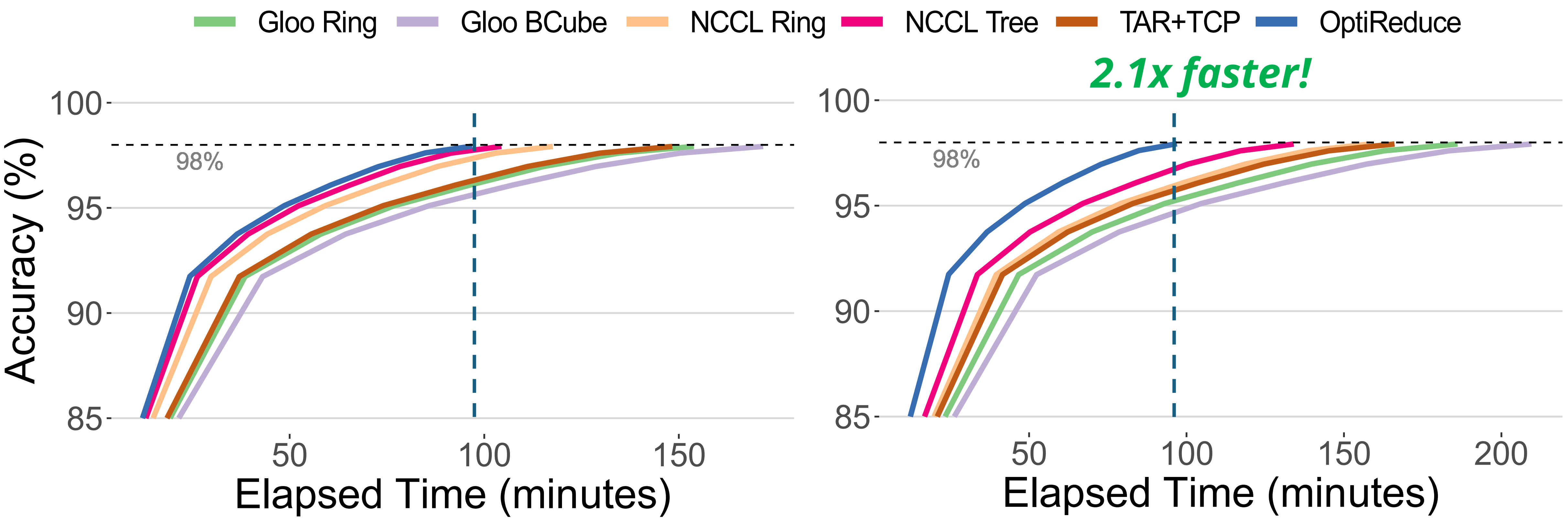


(a) $P_{99/50}$ = 1.5



(b) $P_{99/50}$ = 3

Time-to-Accuracy (TTA)

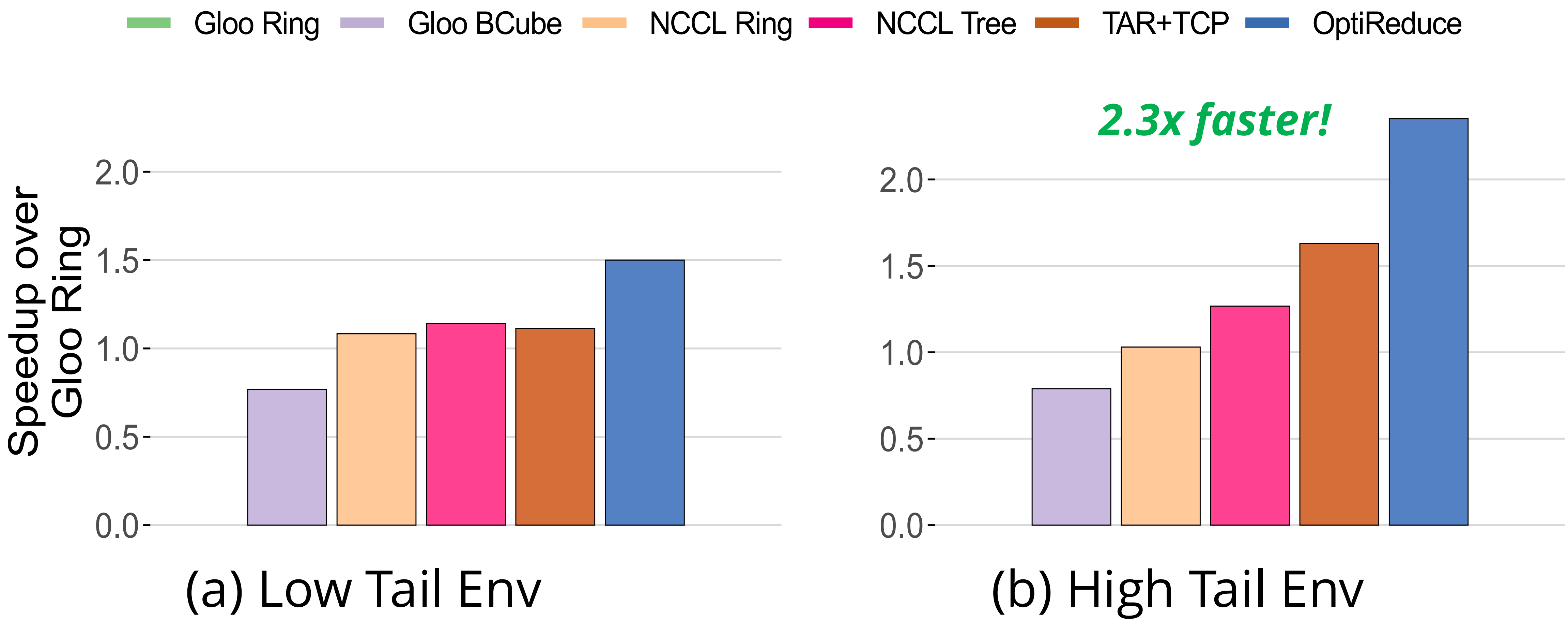


(a) Low Tail Env

(b) High Tail Env

TTA: GPT-2

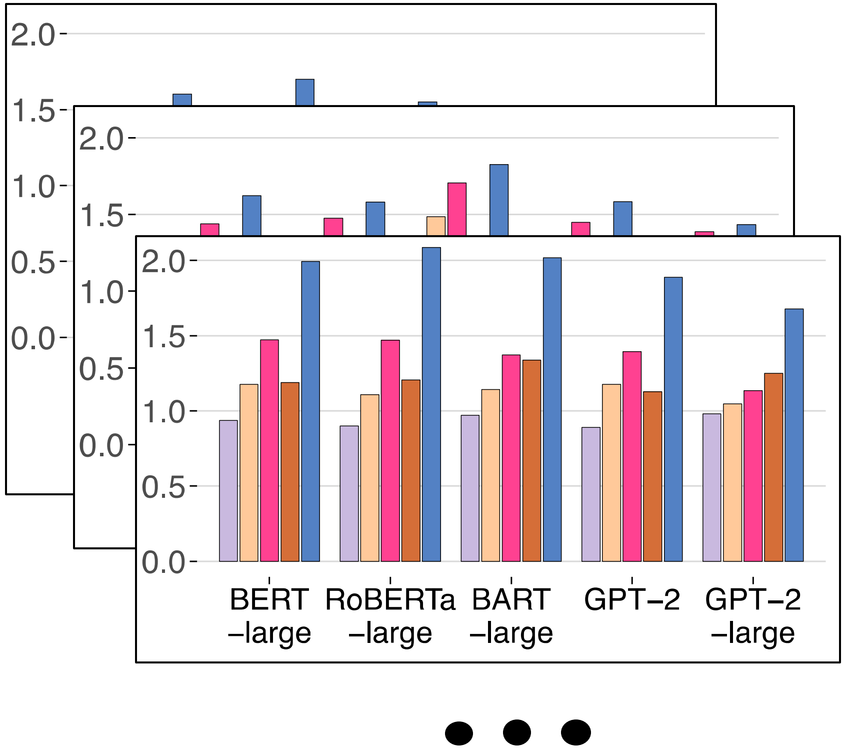
Training Throughput (Speedup)



Training Throughput: Llama-3.2

More Evaluations in the Paper

- OptiReduce **scalability** results
  - **24 nodes** evaluation in local cluster
  - **144 nodes** in simulations
- Comparison with **other schemes**
  - **Parameter Server** approach
  - **Quantization** and **Sparsification** schemes
  - **In-network aggregation** (INA) approaches
- More Models and Datasets
  - More **LLMs** and **vision models**



Conclusion

- **AllReduce bottleneck** in cloud training
- OptiReduce → **Time-bounded AR** for **Cloud**
- More than **2× Speedup** in **high-tail**
- Try **OptiReduce** – follow the link!

optireduce.github.io

Thank You!

OptiReduce - Optimizing Large-Scale ML Training

optireduce.github.io