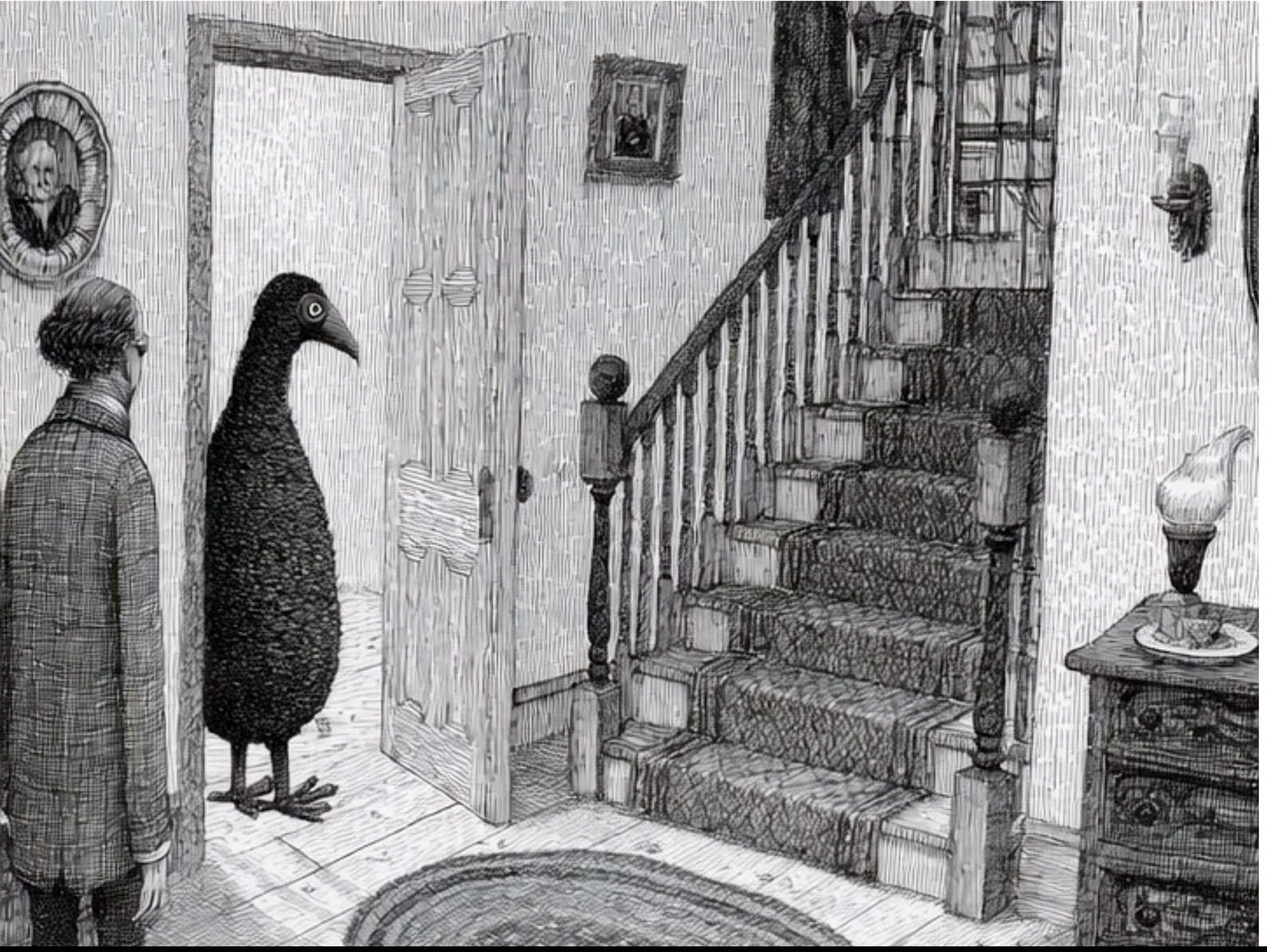


The Doubtful Guest

When the guest appeared on that long-running night,
Nothing had failed, yet nothing felt quite right.

© 2026 Clockwork All rights reserved



The Doubtful Guest

It joined every step and quietly ate,
Some seconds of progress and part of a plate.

© 2026 Clockwork All rights reserved

The Gashlycrumb Tinies



A is for Amy who fell down the stairs



M is for Maud who was swept out to sea

N is for Neville who died of ennui

CLOCKWORK.IO

© 2026 Clockwork All rights reserved

Doubtful Guest: Fail-Slow Failures in AI Training

Lerna Ekmekcioglu
Sr. Solutions Eng, Clockwork.io

© 2026 Clockwork All rights reserved

Training workload

```
# Loop over the dataset multiple times
for epoch in range(epochs):

    # Each step processes one batch of data
    for step in range(num_steps_per_epoch):

        # Forward pass: run inputs through the model & measure error
        predictions = model(inputs)
        loss = loss_fn(predictions, targets)

        # Backward pass: compute gradients and update weights
        optimizer.zero_grad()  # Clear gradients from last step
        loss.backward()        # Compute new gradients
        optimizer.step()       # Apply gradients to update weights
```


© 2026 Clockwork All rights reserved

Training workload

Each epoch

```
# Loop over the dataset multiple times
for epoch in range(epochs):

    # Each step processes one batch of data
    for step in range(num_steps_per_epoch):

        # Forward pass: run inputs through the model & measure error
        predictions = model(inputs)
        loss = loss_fn(predictions, targets)

        # Backward pass: compute gradients and update weights
        optimizer.zero_grad()  # Clear gradients from last step
        loss.backward()        # Compute new gradients
        optimizer.step()       # Apply gradients to update weights
```


© 2026 Clockwork All rights reserved

Training workload

Each epoch

```
# Loop over the dataset multiple times
for epoch in range(epochs):
```

Each training step

```
    # Loop over each batch of data
    for step in range(num_steps_per_epoch):

        # Forward pass: run inputs through the model & measure error
        predictions = model(inputs)
        loss = loss_fn(predictions, targets)

        # Backward pass: compute gradients and update weights
        optimizer.zero_grad()  # Clear gradients from last step
        loss.backward()        # Compute new gradients
        optimizer.step()       # Apply gradients to update weights
```


© 2026 Clockwork All rights reserved

Training workload

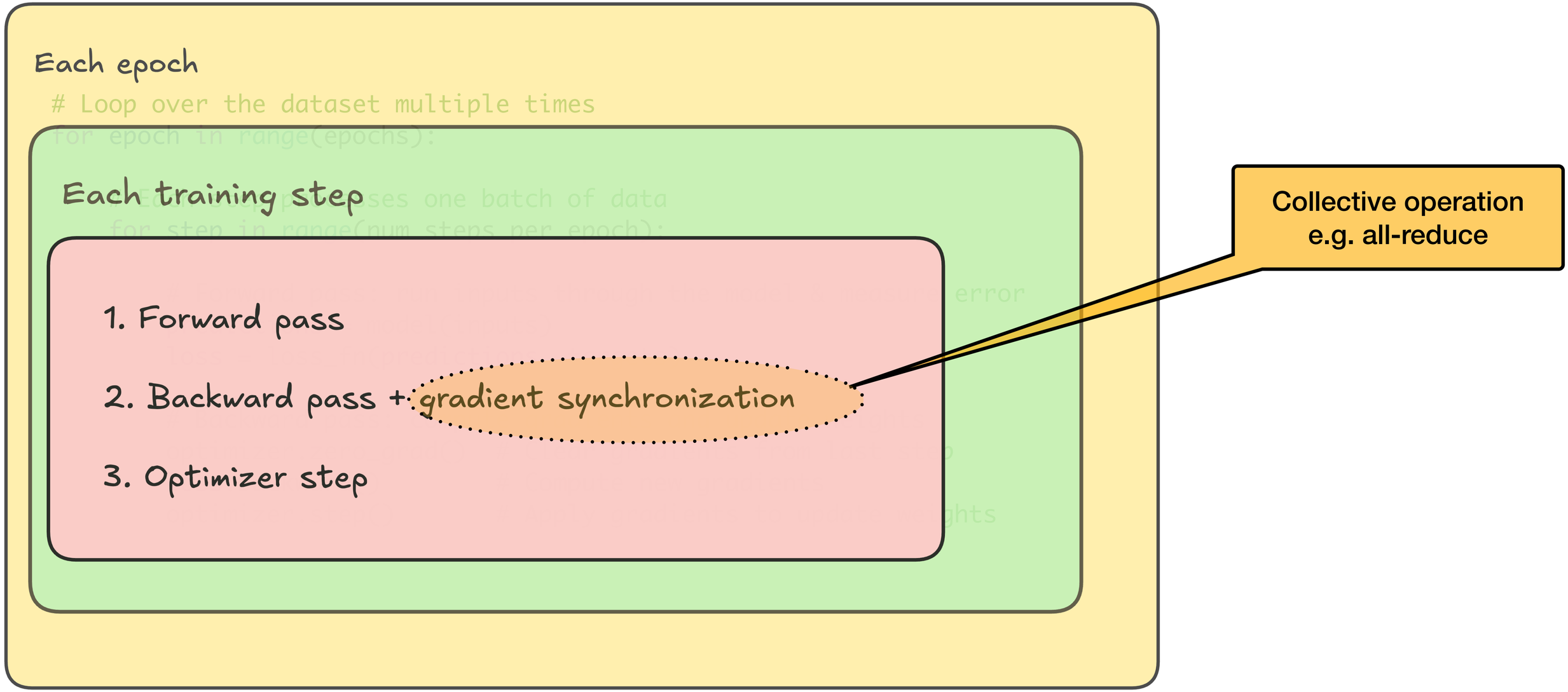
Each epoch

Each training step

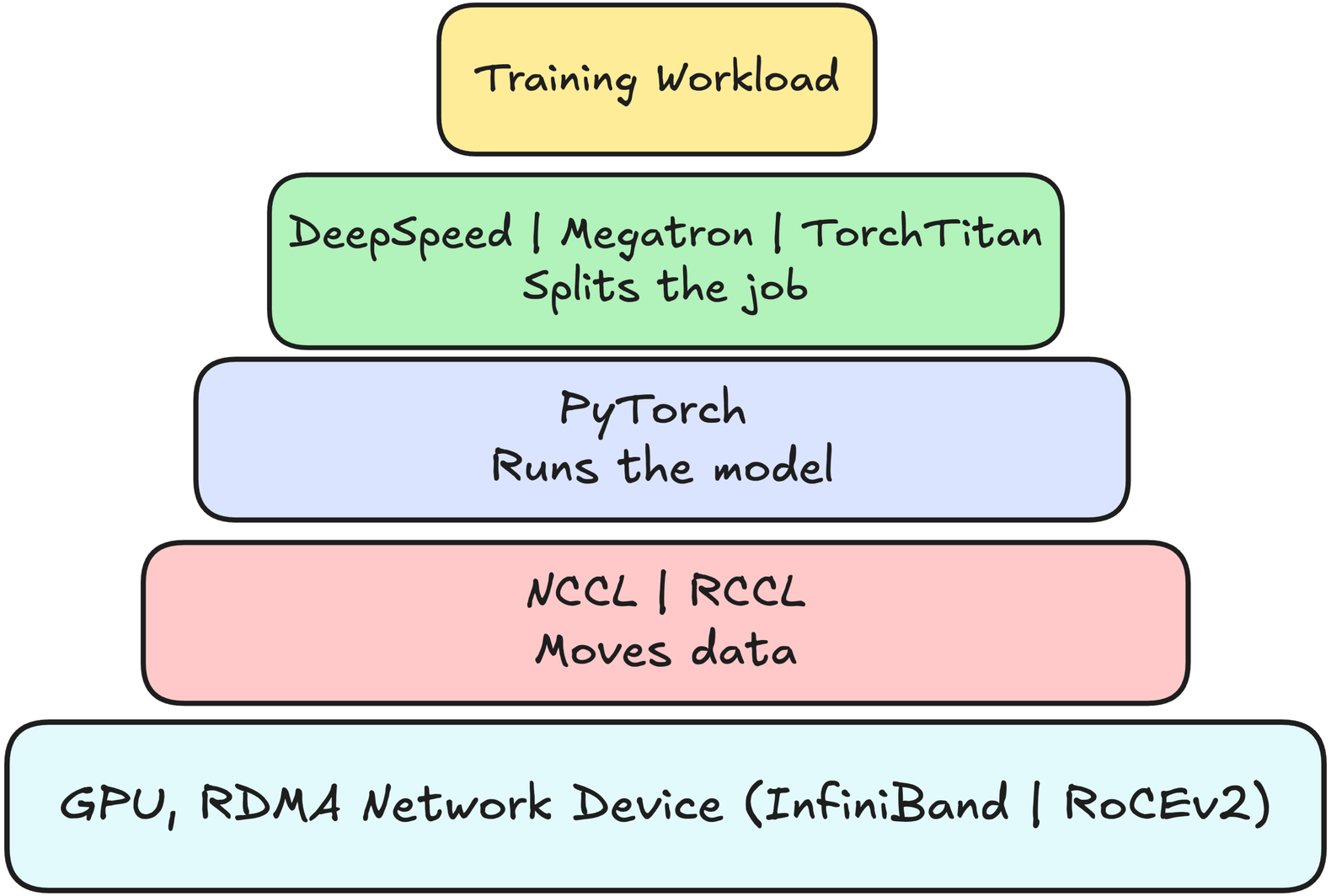
1. Forward pass
2. Backward pass + gradient synchronization
3. Optimizer step

© 2026 Clockwork All rights reserved

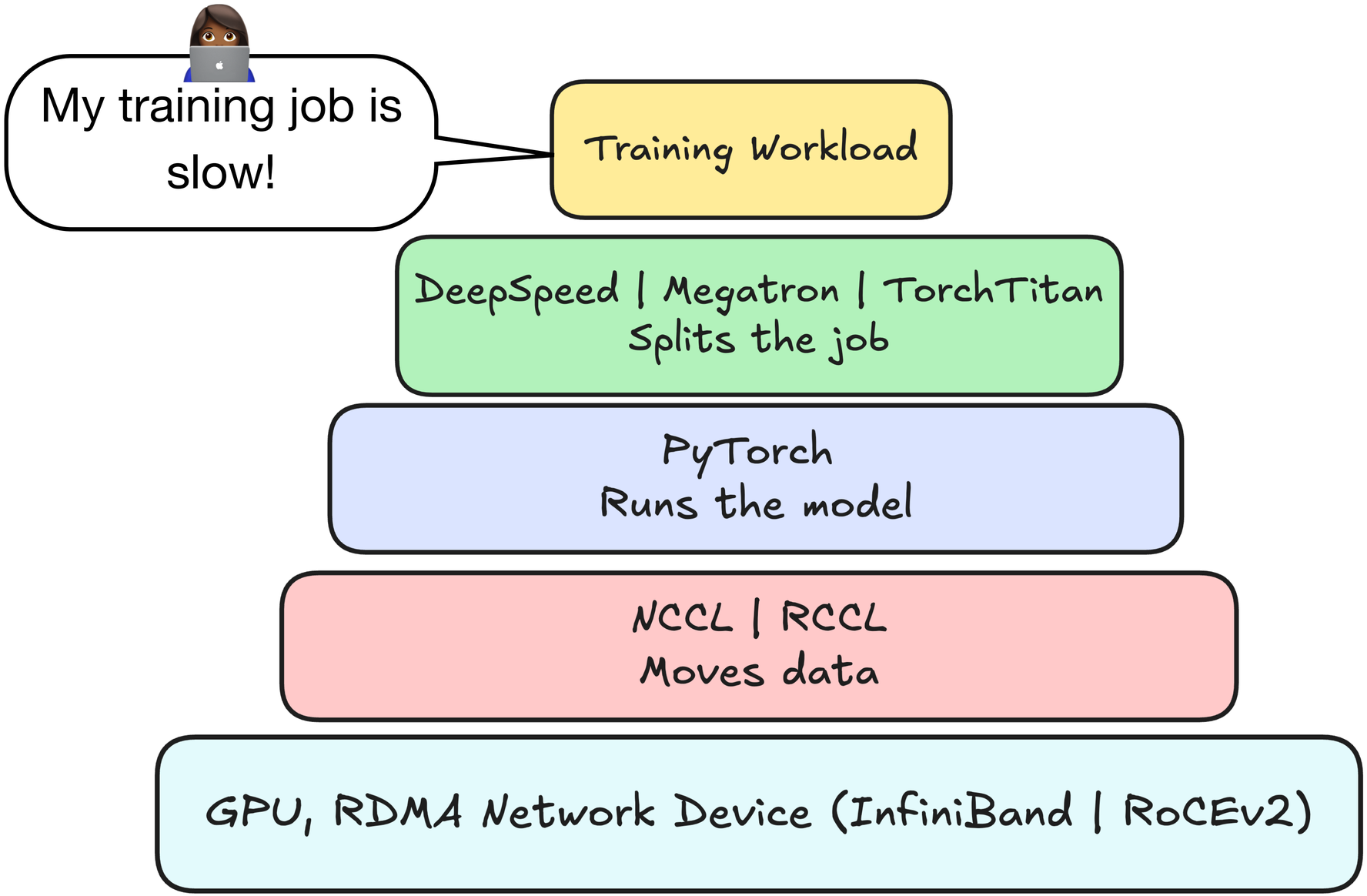
Training workload



© 2026 Clockwork All rights reserved



© 2026 Clockwork All rights reserved



© 2026 Clockwork All rights reserved



My training job is slow!

Training Workload

Is it the app?



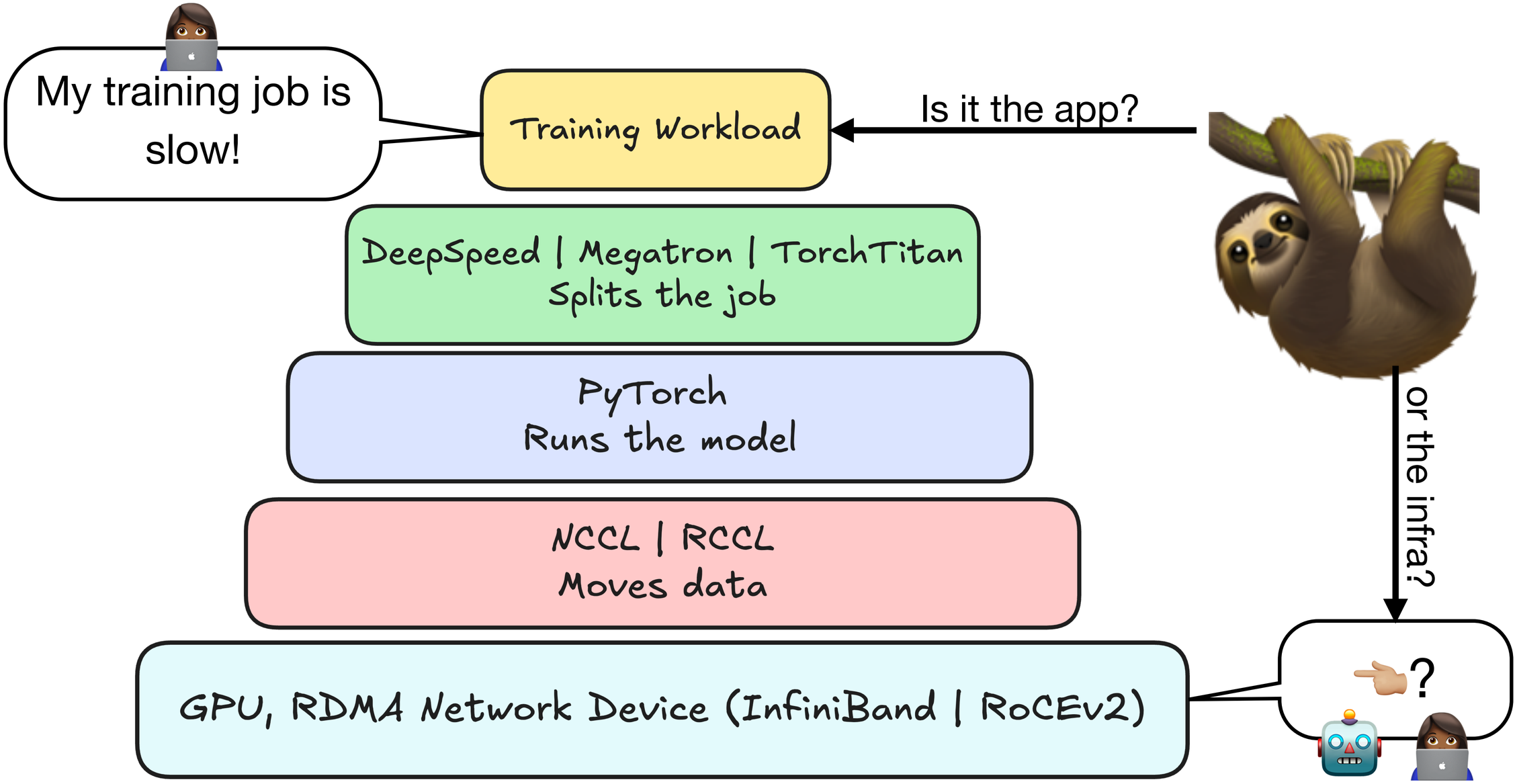
DeepSpeed | Megatron | TorchTitan
Splits the job

PyTorch
Runs the model

NCCL | RCCL
Moves data

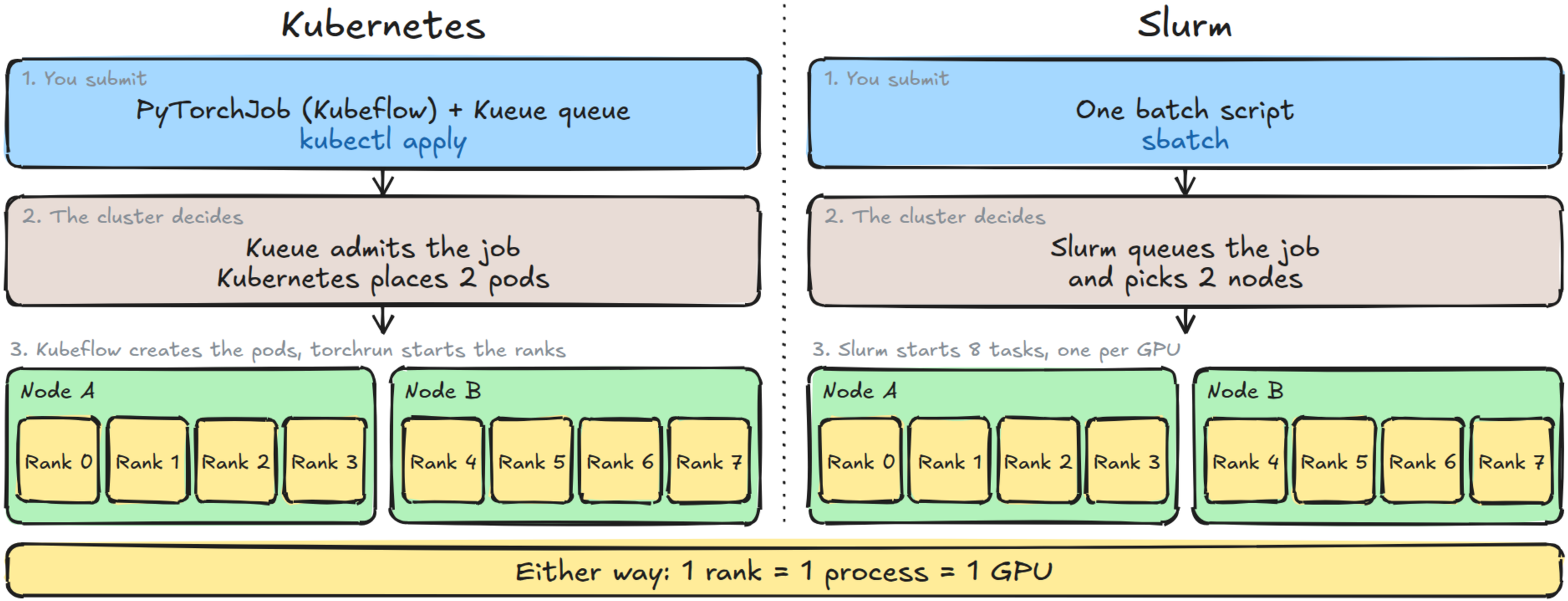
GPU, RDMA Network Device (InfiniBand | RoCEv2)

© 2026 Clockwork All rights reserved



© 2026 Clockwork All rights reserved

From job to ranks

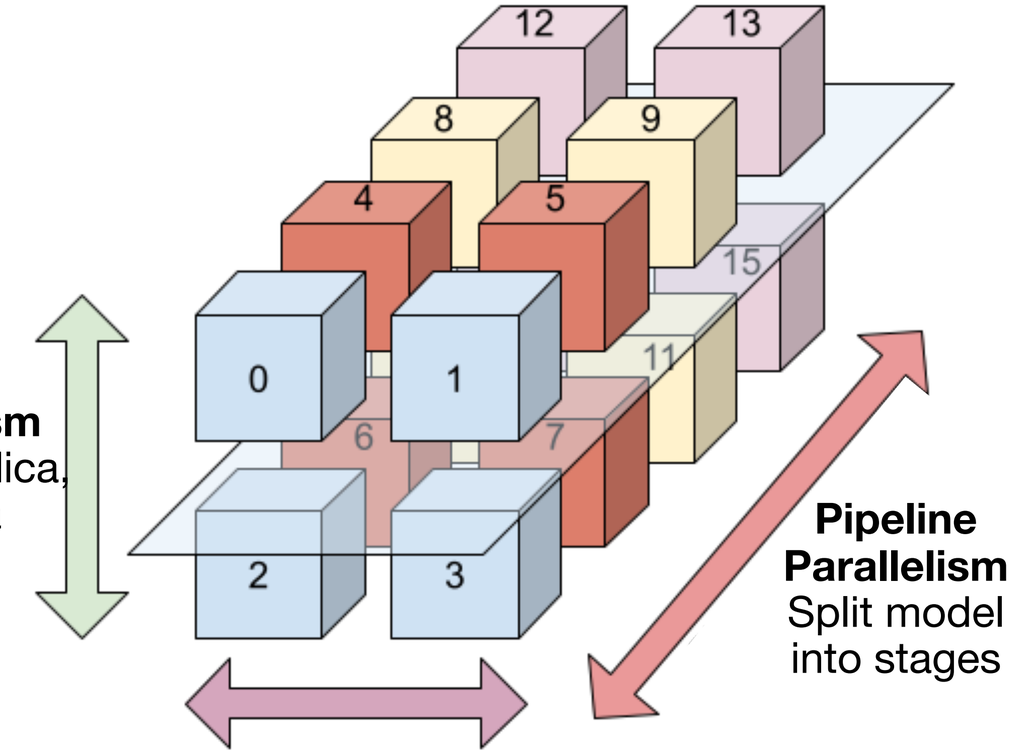


Either way: 1 rank = 1 process = 1 GPU

Slurm ♥ Kubernetes = SUNK · Soperator · Slinky

© 2026 Clockwork All rights reserved

Scaling large jobs



Data Parallelism
Same model replica, different data

Pipeline Parallelism
Split model into stages

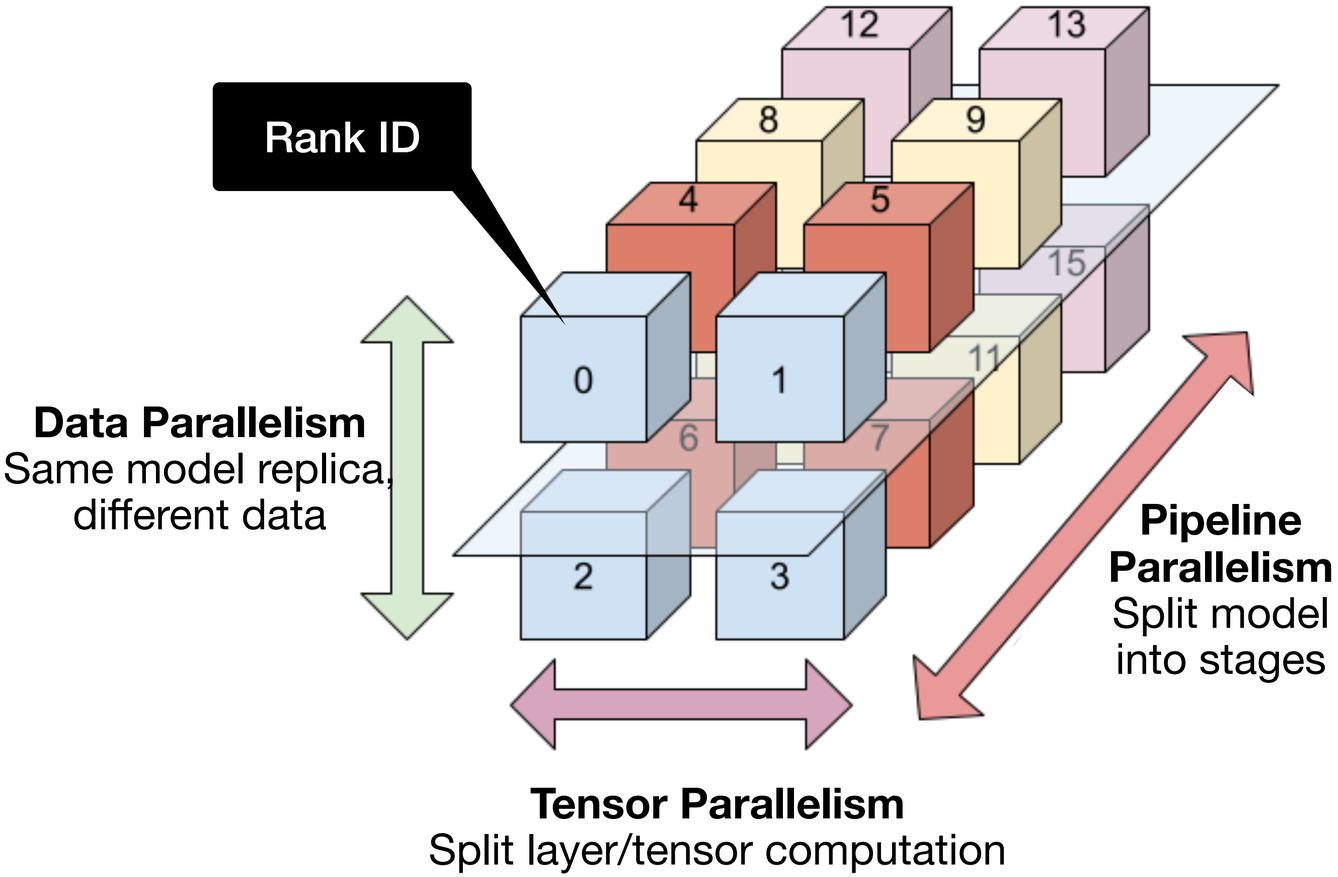
Tensor Parallelism
Split layer/tensor computation

- Large jobs combine parallelism strategies.
- A rank belongs to several communication groups at once.

Solergibert, A Deep Dive into 3D Parallelism with Nanotron
CC BY-NC-ND 4.0

© 2026 Clockwork All rights reserved

Scaling large jobs

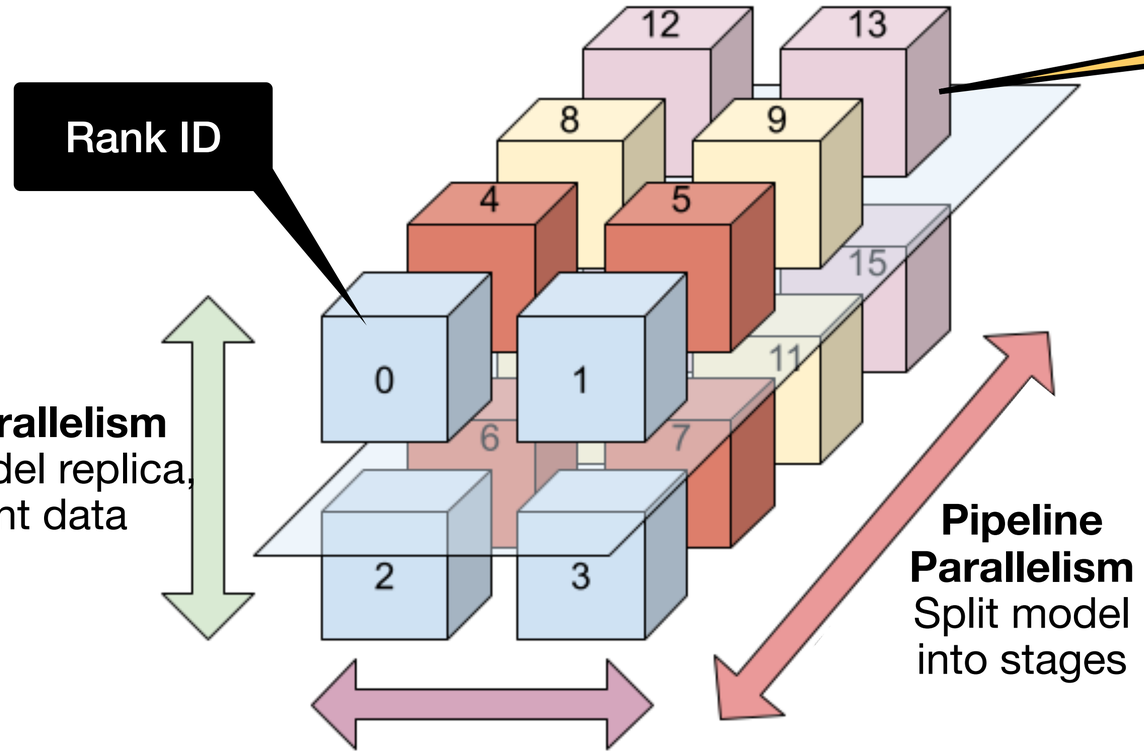


- Large jobs combine parallelism strategies.
- A rank belongs to several communication groups at once.

Solergibert, A Deep Dive into 3D Parallelism with Nanotron
CC BY-NC-ND 4.0

© 2026 Clockwork All rights reserved

Scaling large jobs



- Large jobs combine parallelism strategies.
- A rank belongs to several communication groups at once.

Solergibert, A Deep Dive into 3D Parallelism with Nanotron
CC BY-NC-ND 4.0

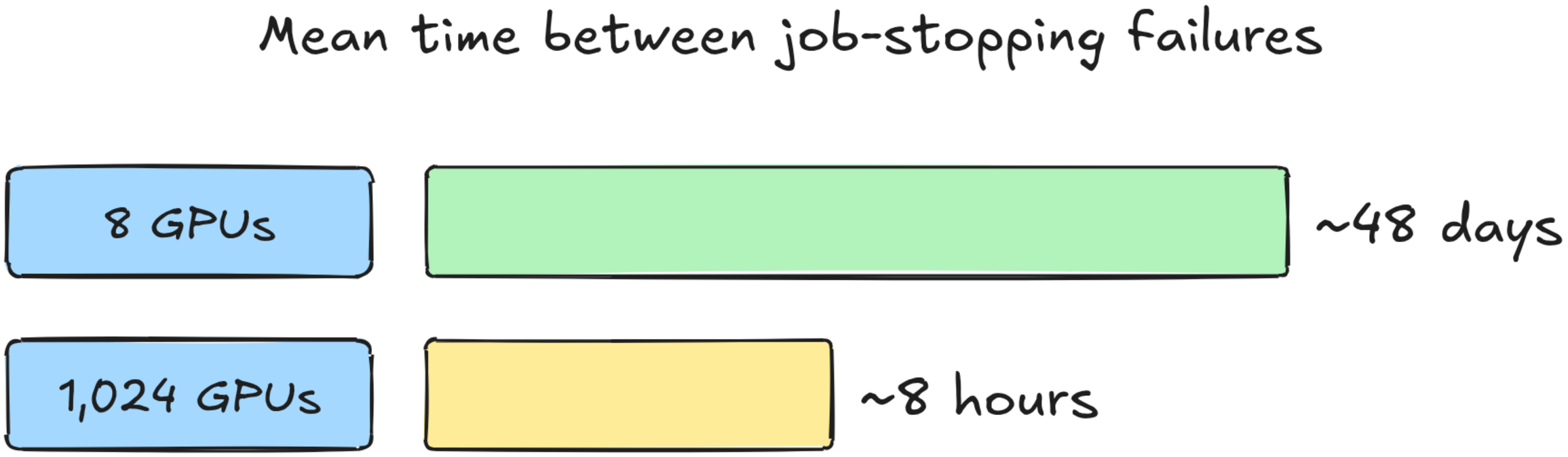
© 2026 Clockwork All rights reserved

The Dark Arts of Distributed Communication



A séance!

Loud failures at scale



Kokolis et al. (Meta), Revisiting Reliability in Large-Scale Machine Learning Research Clusters (arXiv 2410.21680)

© 2026 Clockwork All rights reserved

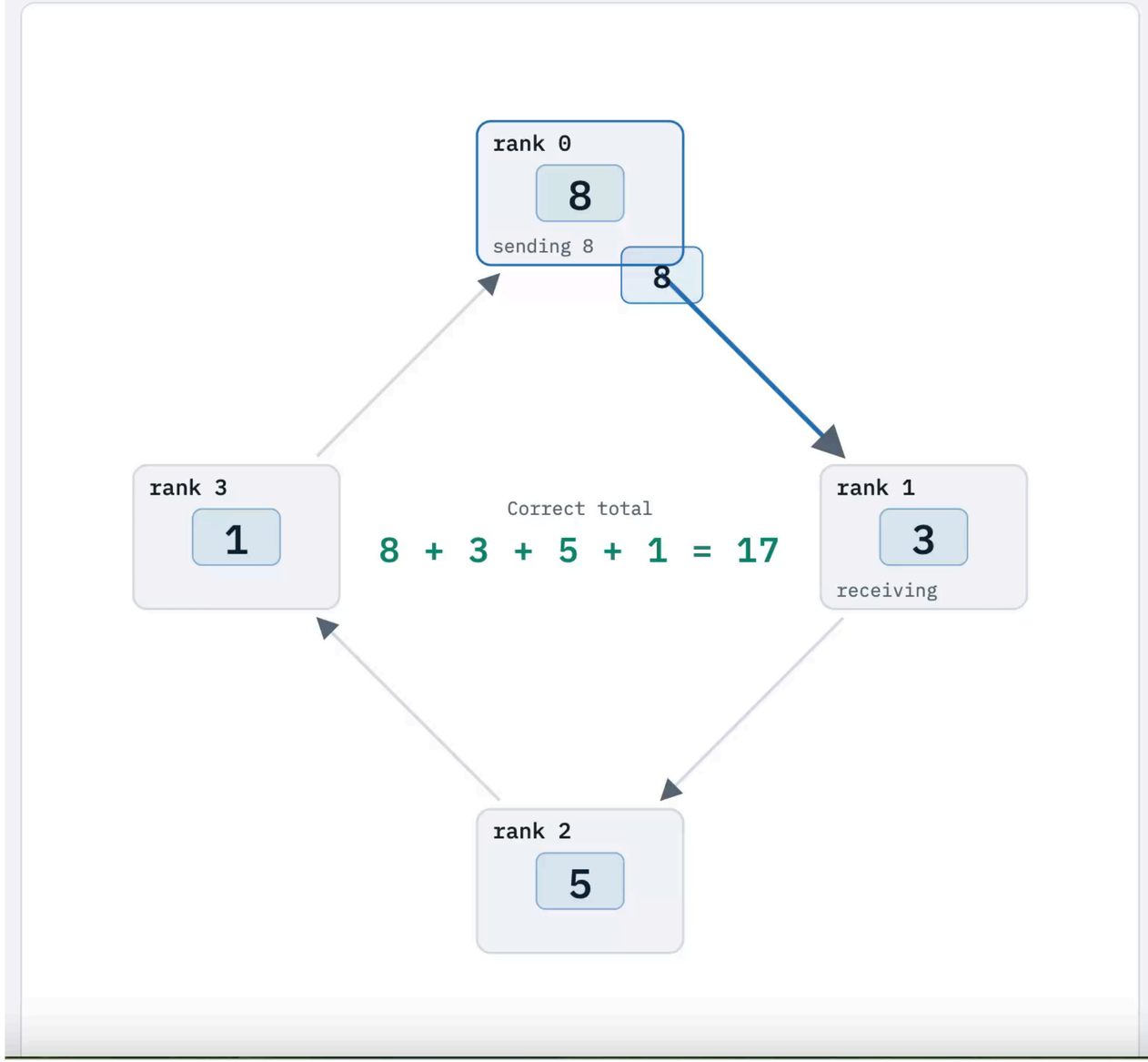


A is for All-Reduce, where eight took their place,
And each bore the same sum, with a most solemn face.

© 2026 Clockwork All rights reserved

Ring All-Reduce

How GPUs add up their gradients after every training step, and why one slow rank holds up all the others.



SCENARIO

All ranks healthy | Rank 2 is slow

Pause | Restart | Auto demo

PROGRESS

Add up, 3 steps — Share the total, 3 steps

Reduction operation	sum
Step	1 / 6
This all-reduce	0.0 s
Last all-reduce	5.7 s
All-reduces per minute	10.5

SECONDS PER ALL-REDUCE, LATEST 8

5.7 5.7 5.7 5.7 5.7 5.8 5.7 5.7

A rank's own number

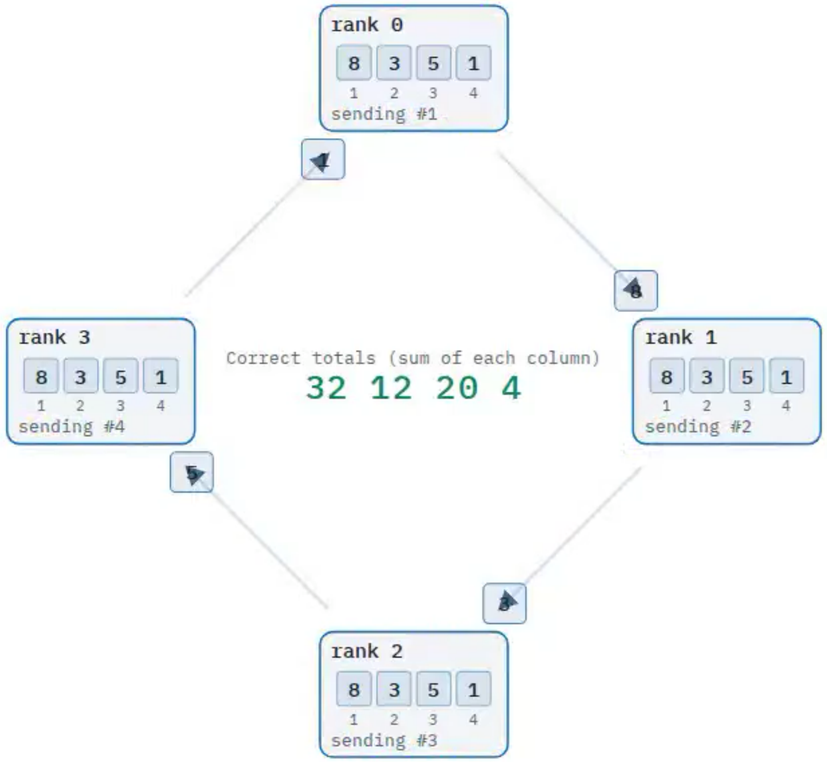
A partial sum

The final total, 17

© 2026 Clockwork All rights reserved

Chunked Ring All-Reduce

How GPUs add up their gradients after every training step, and why one slow rank holds up all the others.



Reduce-scatter: each rank passes one chunk to its neighbor, which adds it to its own copy.

SCENARIO

All ranks healthy | Rank 2 is slow

Pause | Restart | Auto demo

PROGRESS

Reduce-scatter, 3 steps — All-gather, 3 steps

Reduction operation	sum
Step	1 / 6
This all-reduce	0.6 s
Last all-reduce	–
All-reduces per minute	–

SECONDS PER ALL-REDUCE, LATEST 8

- One chunk, a few ranks summed
- One chunk, most ranks summed
- Fully summed across all ranks

Simplified to 4 ranks. Every rank holds the same four numbers, 8, 3, 5 and 1, as the four chunks of its gradient, so each column sums to 4 times its number: 32, 12, 20 and 4. Each cell shows the running sum of its column, and the small numbers under the cells are the chunk positions. In every step all four ranks send a different chunk at the same time, which keeps every link busy. That is how NCCL works, and every step still waits for the slowest sender. The recorded demos use 16 ranks on two hosts, with rank 11 playing the part of rank 2.

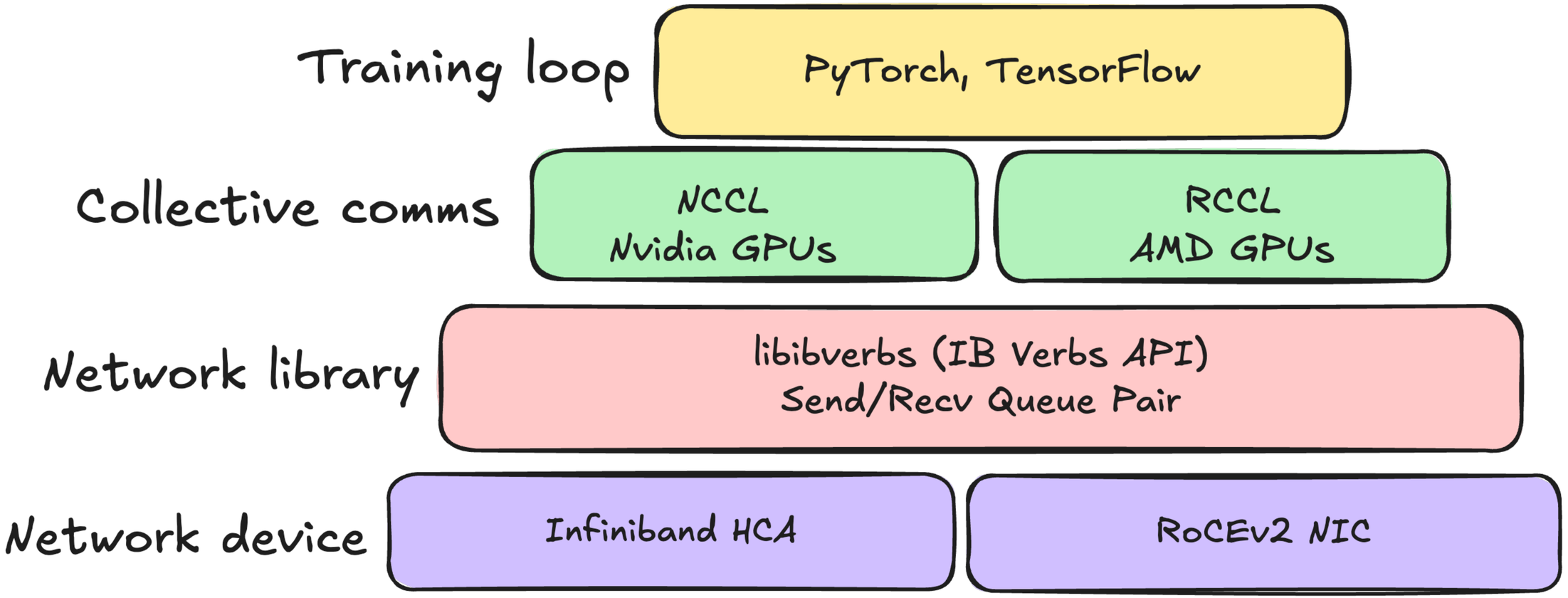
© 2026 Clockwork All rights reserved



N is for NCCL, where messages flow,
Till one rank falls silent, and none of them go.

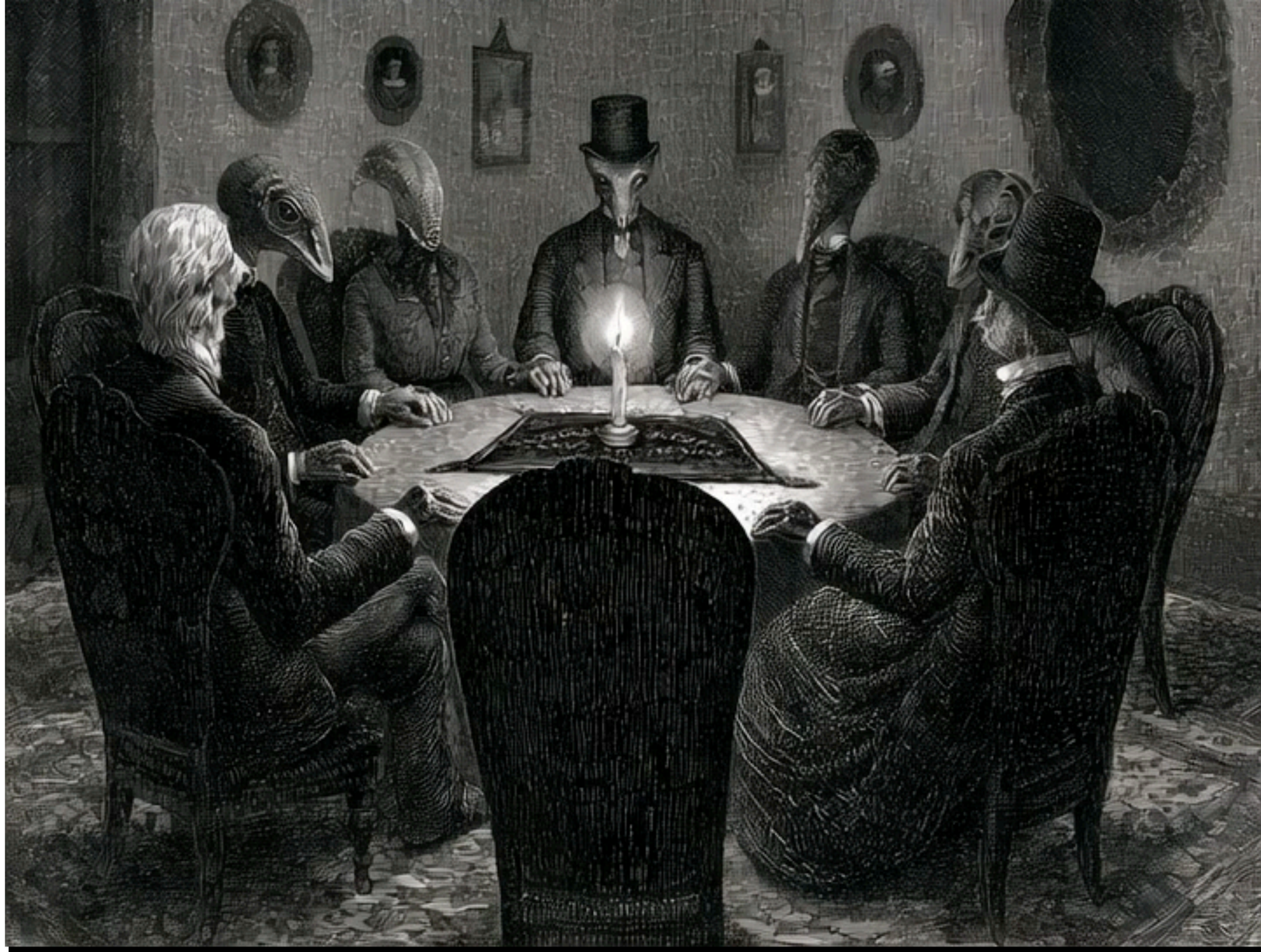
© 2026 Clockwork All rights reserved

N is for NCCL



© 2026 Clockwork All rights reserved

G is for Gang Scheduling



G is for Gang, who must sit as a whole,
Seven of eight can not summon a soul.

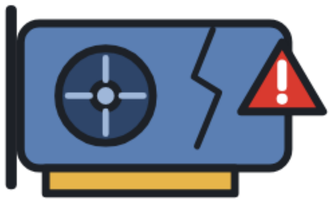
Fail-stop and fail-slow

FAIL-STOP loud

quiet FAIL-SLOW



Worker crash



GPU fell off the bus



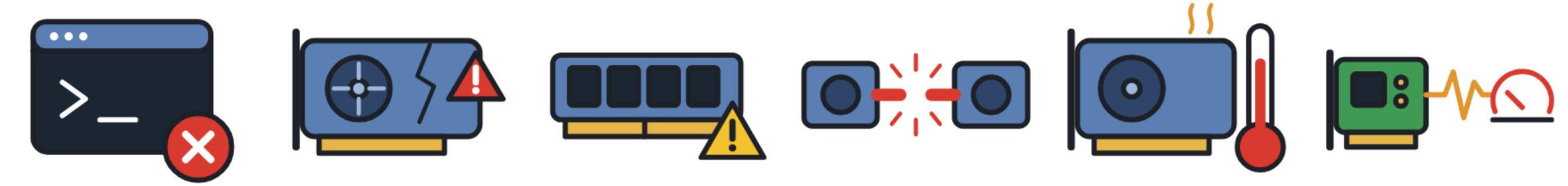
Uncorrectable memory error

© 2026 Clockwork All rights reserved

Fail-stop and fail-slow

FAIL-STOP loud

quiet FAIL-SLOW



Worker crash

GPU fell off the bus

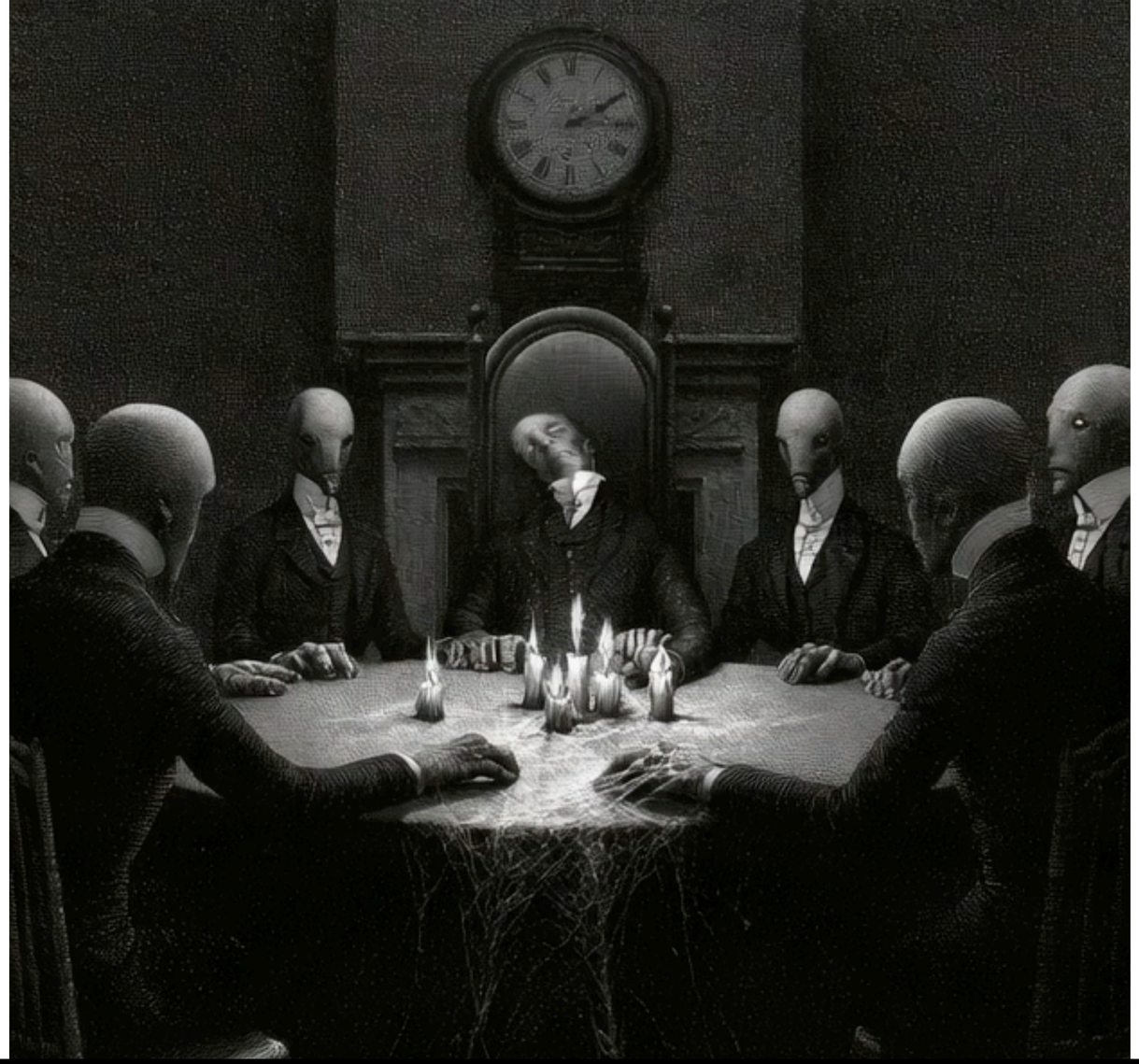
Uncorrectable memory error

NVLink errors

Throttled GPU

Network degradation

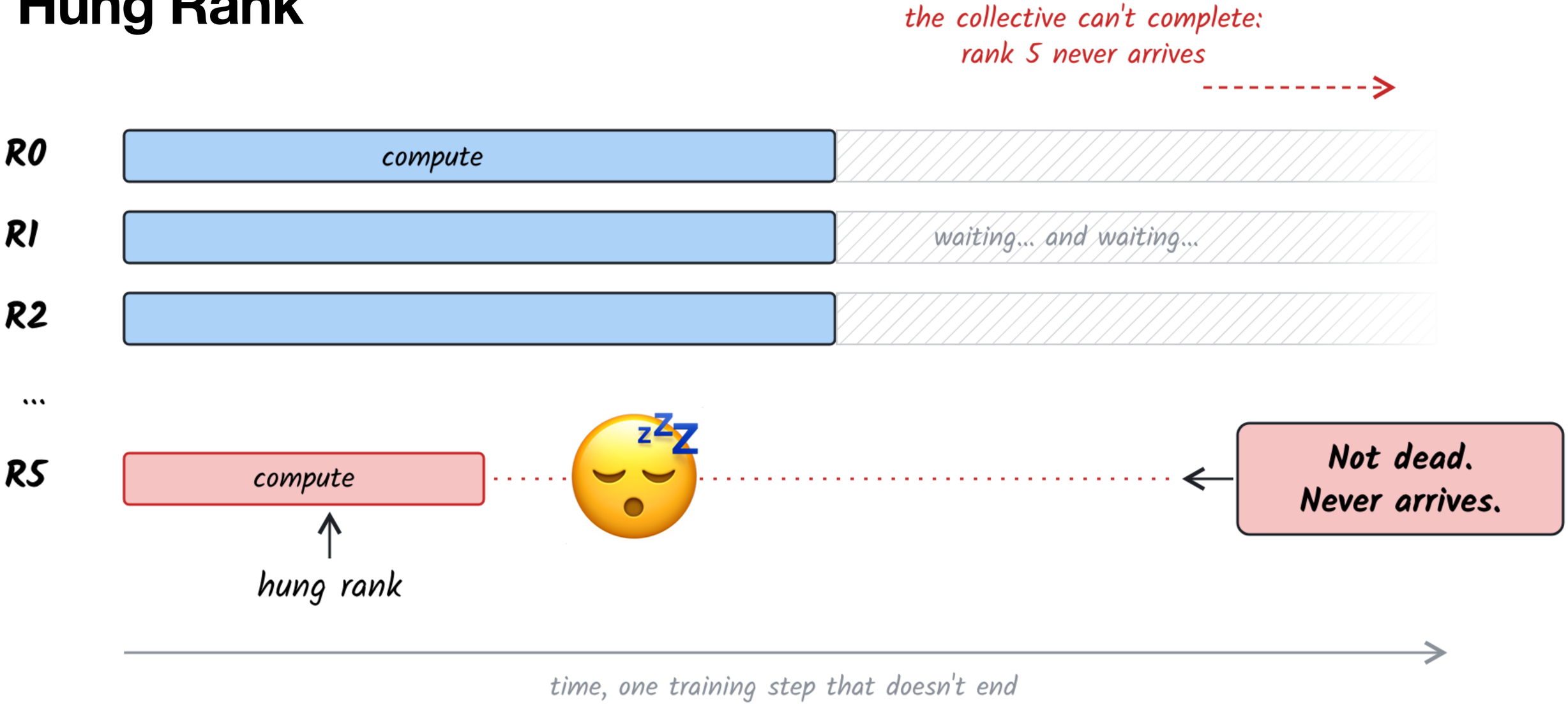
© 2026 Clockwork All rights reserved



H is for Hung (rank), who sat still in his chair;
the others all waited; he never got there.

© 2026 Clockwork All rights reserved

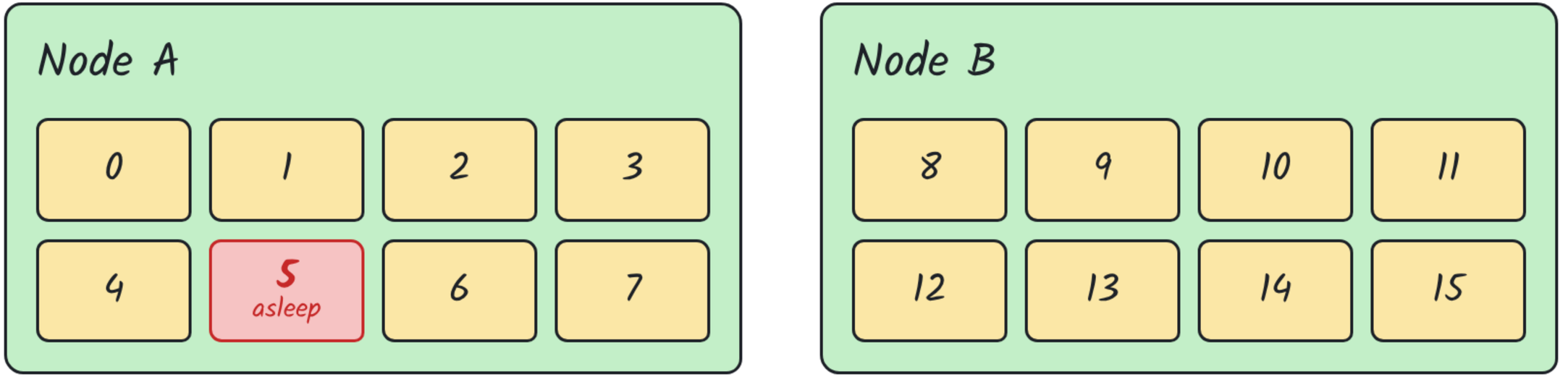
Hung Rank



A hung rank never reaches the synchronization point.

© 2026 Clockwork All rights reserved

Demo 1: a hung rank



At 4 min, **rank 5 sleeps for 300 s**. The other 15 wait.

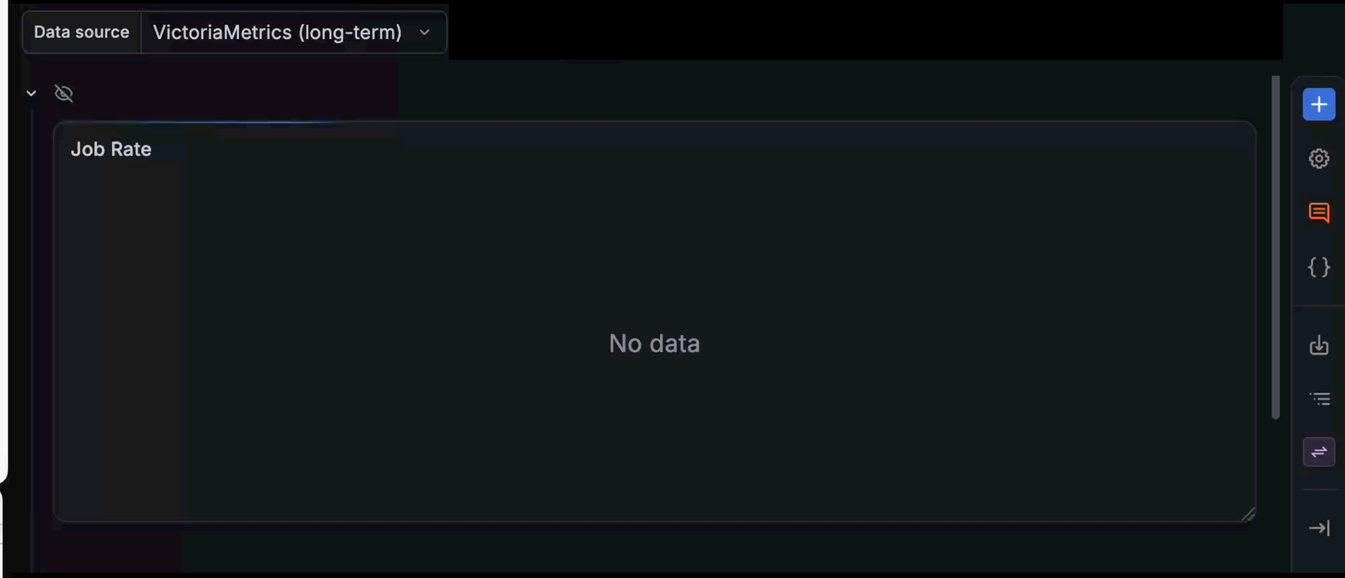
Watch: step time · NCCL ops/sec · GPU utilization

© 2026 Clockwork All rights reserved

😴 Hung Rank

```
$ sed -n 86,94p wm-job.tmpl
            opt.step()
            it += 1
            el = time.time() - t0
            if r == HR and not hung and el > HA:
                hung = True
                print(f"[rank {r}] HANG START for {HF}s", flush=True)
                time.sleep(HF)
                print(f"[rank {r}] HANG END", flush=True)
            if it % 50 == 0:
$
```

H is for Hung (rank), who sat still in his chair;
the others all waited; he never got there.



lerna — ubuntu@ecmp-controller: ~ — ssh ecmp-controller — 68×14

```
$ kubectl apply -f wm-hang-0.yaml -f wm-hang-1.yaml
kubectl -n wm-demo get pods -w
pod/wm-hang-0 created
pod/wm-hang-1 created
NAME         READY   STATUS     RESTARTS   AGE
wm-hang-0    0/1     Init:0/1   0          0s
wm-hang-1    0/1     Init:0/1   0          0s
wm-hang-0    0/1     PodInitializing   0          2s
wm-hang-1    0/1     PodInitializing   0          2s
wm-hang-0    1/1     Running           0          3s
wm-hang-1    1/1     Running           0          3s
```

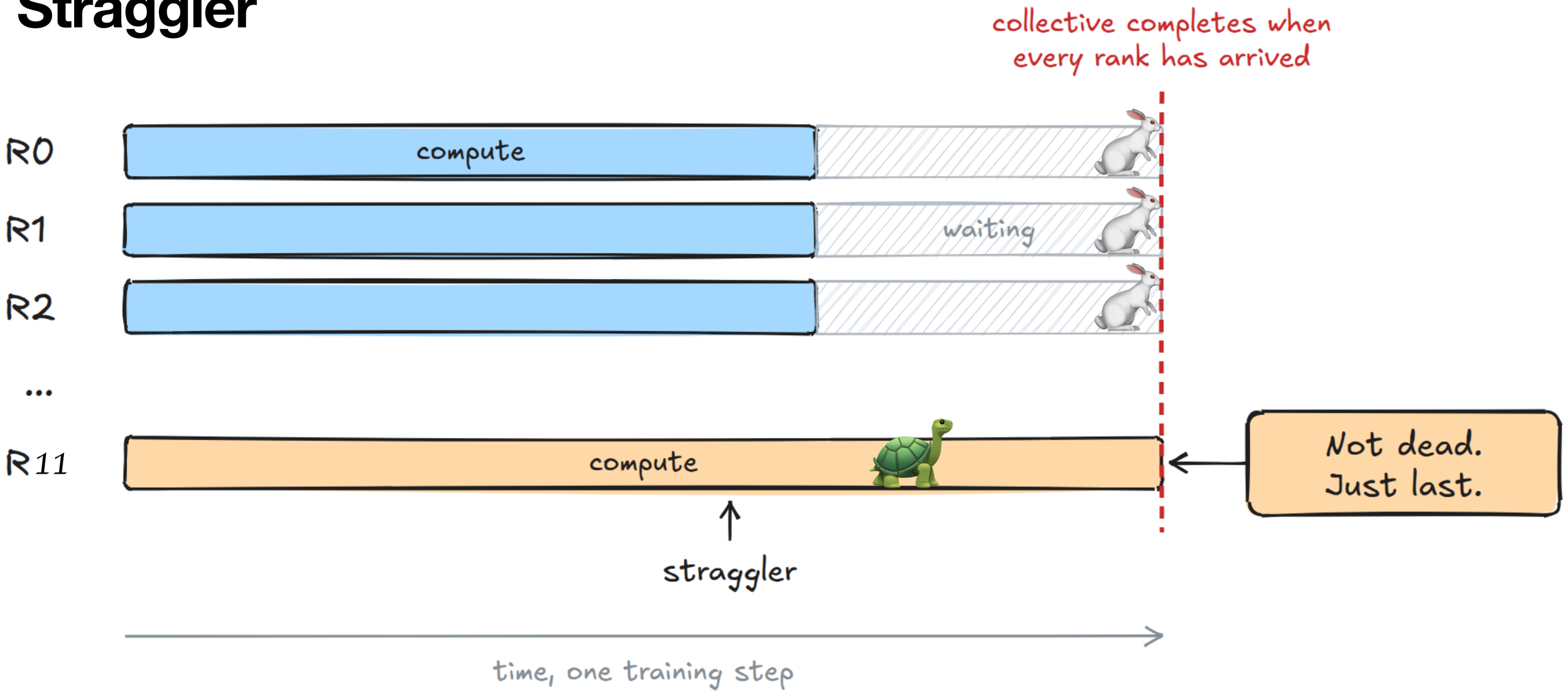

© 2026 Clockwork All rights reserved



S is for Straggler, who lagged by a beat,
and fifteen more waited, each step, in their seat.

© 2026 Clockwork All rights reserved

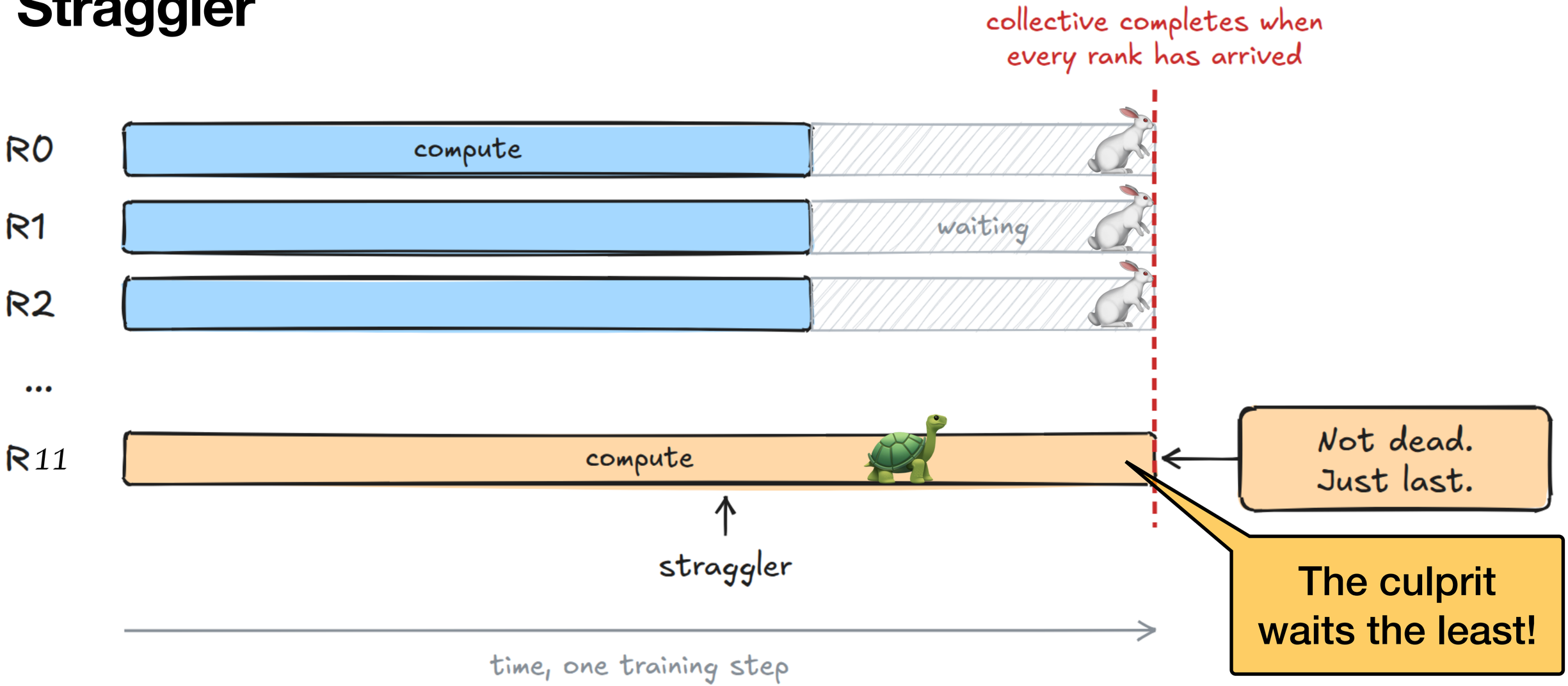
Straggler



The job runs at the speed of its slowest rank.

© 2026 Clockwork All rights reserved

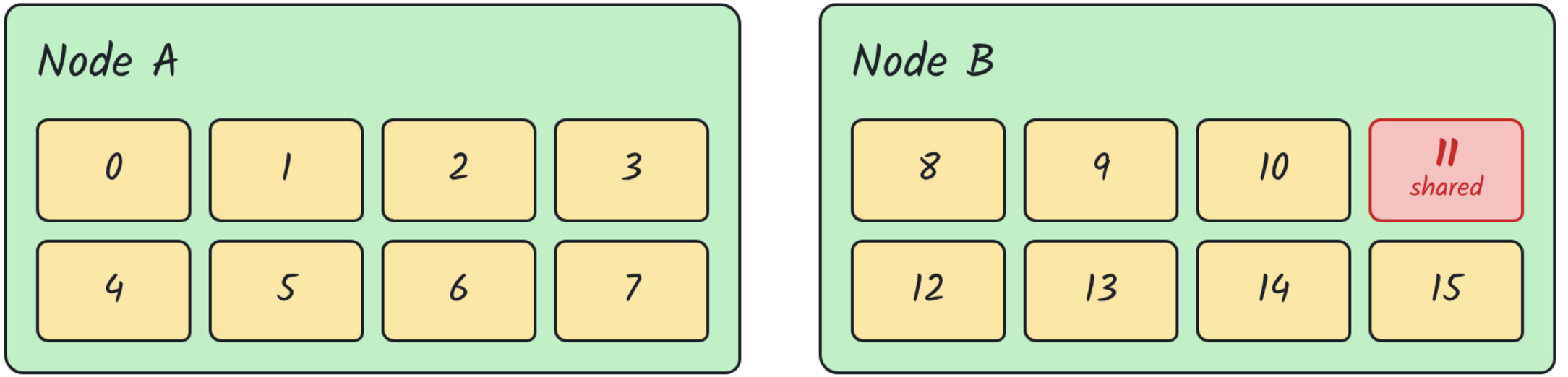
Straggler



The job runs at the speed of its slowest rank.

© 2026 Clockwork All rights reserved

Demo 2: a straggler



A second process **shares rank 11's GPU**. Every step slows to match.

Watch: step time · NCCL ops/sec

© 2026 Clockwork All rights reserved

🐢 Straggler

S is for Straggler, who lagged by a beat,
and fifteen more waited, each step, in their seat.

```
$
```



lerna — ubuntu@ecmp-controller: ~ — ssh ecmp-controller — 74×15

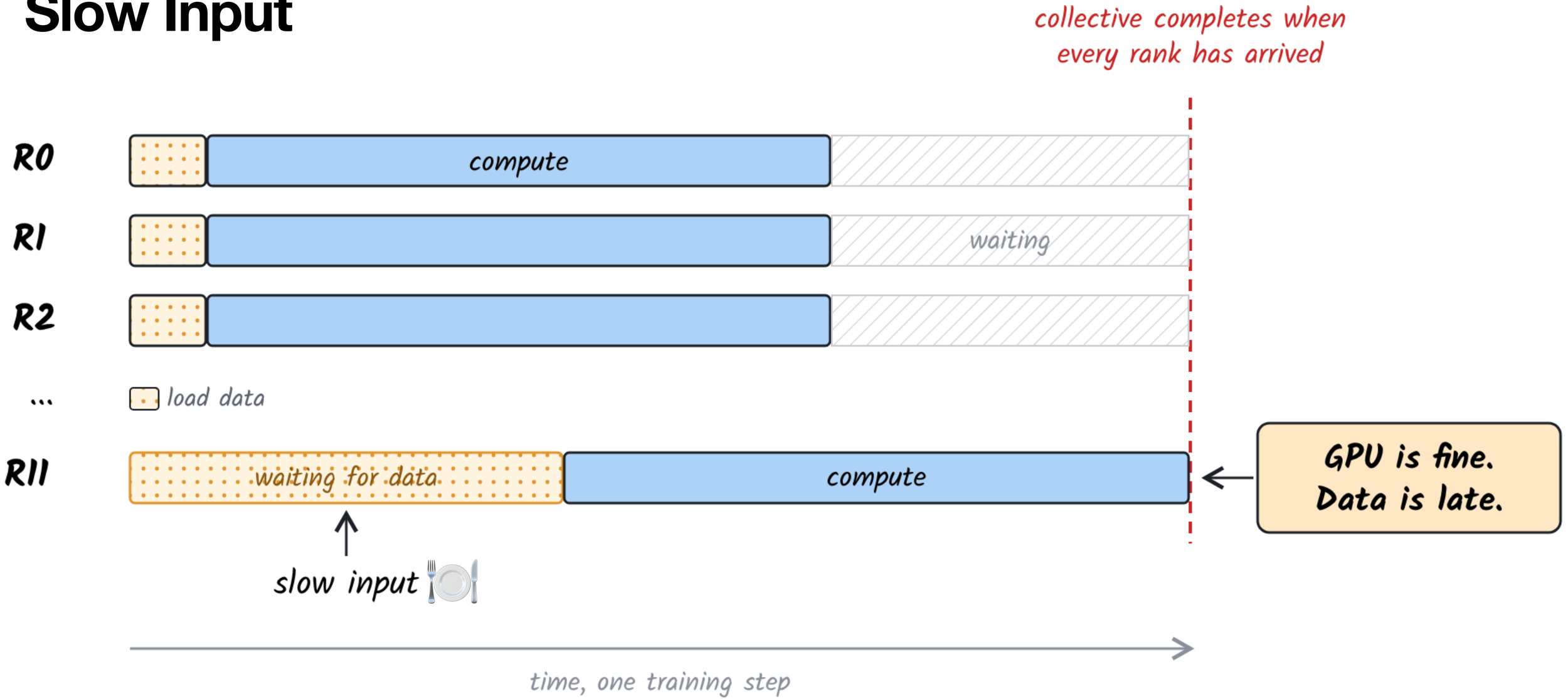
```
$ cat ~/lerna/wm/demo/burn.py
import sys, time, torch
g, dur = int(sys.argv[1]), int(sys.argv[2])
torch.cuda.set_device(g)
a = torch.randn(8192, 8192, device="cuda"); b = torch.randn(8192, 8192, de
vice="cuda")
end = time.time() + dur
while time.time() < end:
    for _ in range(20):
        c = a @ b
    torch.cuda.synchronize()
$
```

lerna — ubuntu@ecmp-controller: ~ — ssh ecmp-controller — 64×12

```
$
```


© 2026 Clockwork All rights reserved

Slow Input



The straggler isn't the GPU. It's what feeds it.

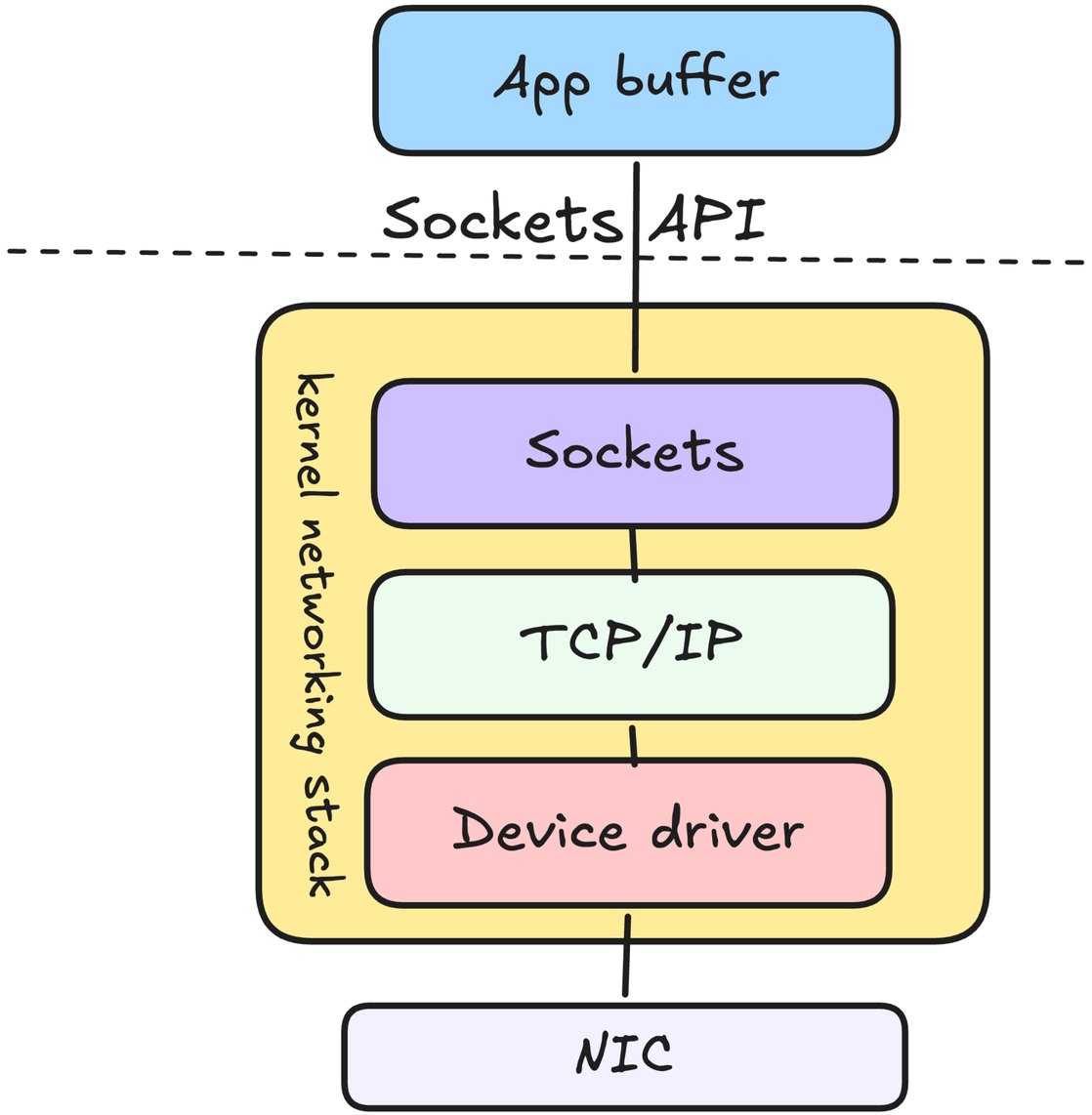
© 2026 Clockwork All rights reserved



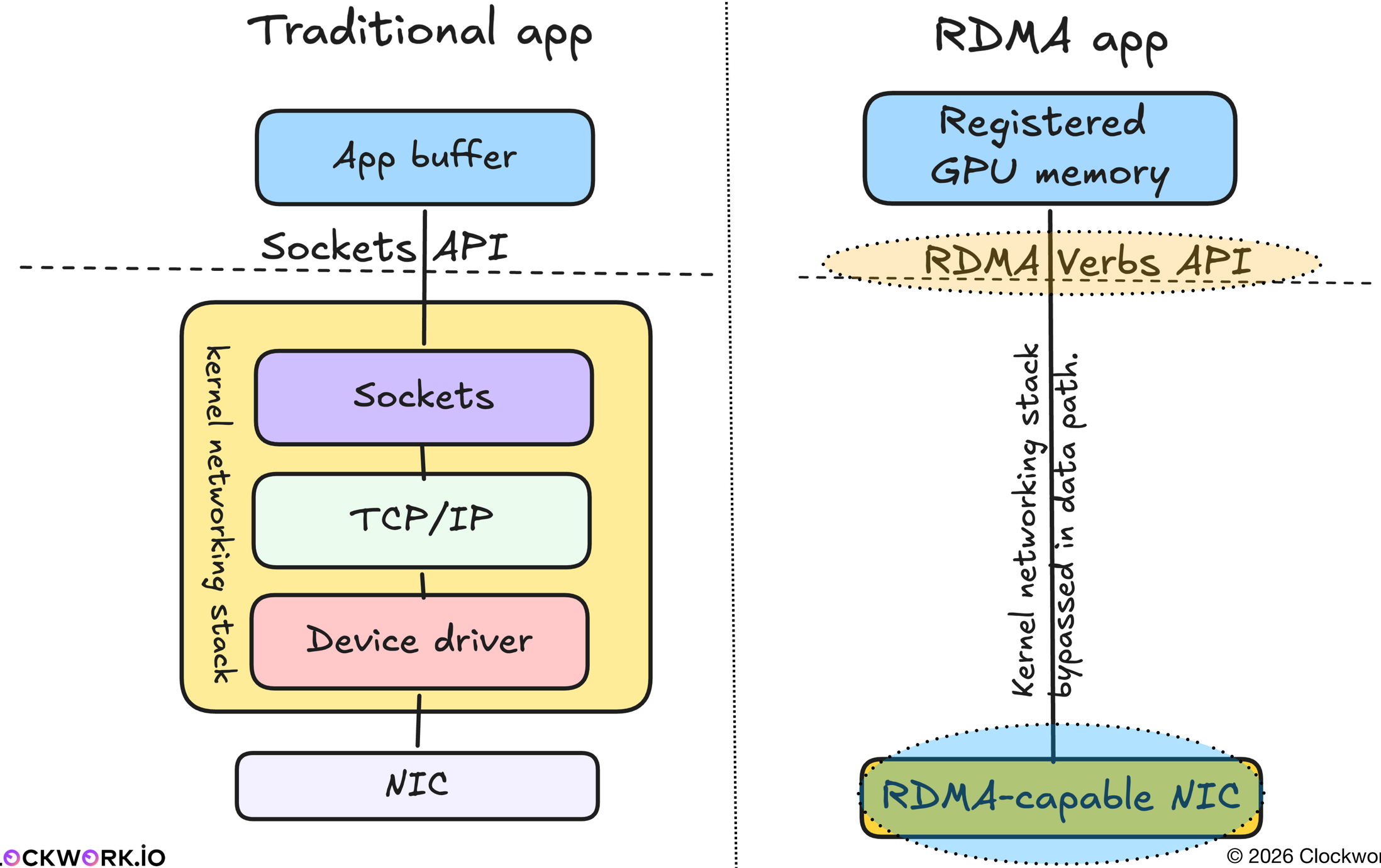
R is for RDMA, the swift private lane;
lose it, and traffic crawls back through the main.

© 2026 Clockwork All rights reserved

Traditional app



© 2026 Clockwork All rights reserved



© 2026 Clockwork All rights reserved



T is for TCP, the long way around;
the whole circle slowed, and it made not a sound.

© 2026 Clockwork All rights reserved

Demo: TCP fallback

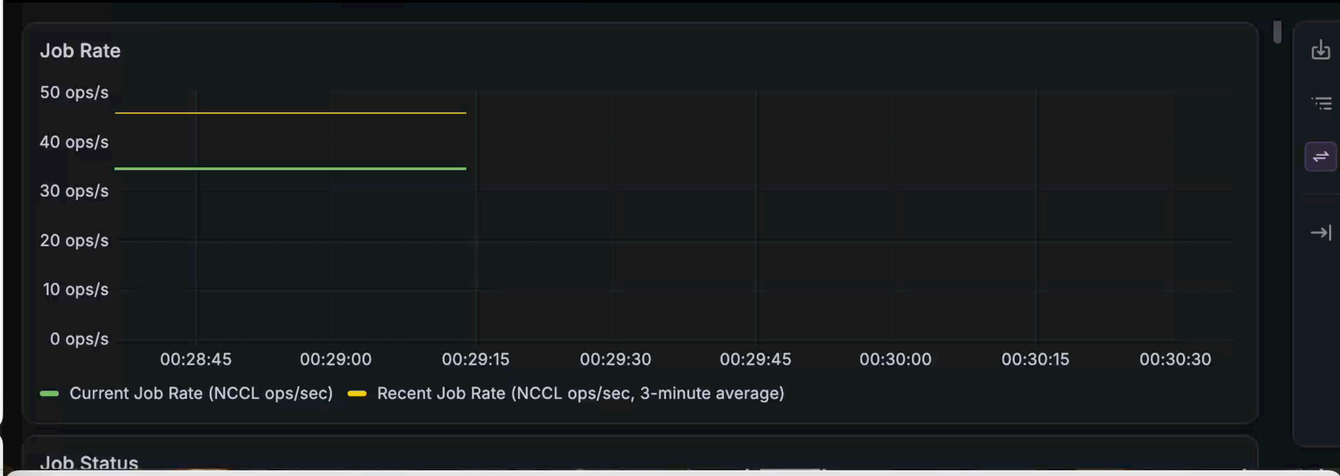
```
$
```

Run over IB for 90s
Then FallBack to TCP

lerna — ubuntu@ecmp-controller: ~ — ssh ecmp-controller — 74×15

```
$ grep -n -E "torchrun|NCCL_IB_DISABLE|sleep 5" wm-ibtcp-1.yaml
113:          RUN_SECONDS=90 torchrun --nnodes=2 --nproc_per_node=8 --node
_rank=1 --master_addr=172.16.7.70 --master_port=29500 /tmp/train.py
114:          sleep 5
115:          NCCL_IB_DISABLE=1 RUN_SECONDS=120 torchrun --nnodes=2 --npro
c_per_node=8 --node_rank=1 --master_addr=172.16.7.70 --master_port=29501 /
tmp/train.py
$
```

T is for TCP, the long way around;
the whole circle slowed, and it made not a sound.



lerna — ubuntu@ecmp-controller: ~ — ssh ecmp-controller — 64×12

```
Every 2.0s: nvidia...  oci-a100-host-3: Wed Oct  7 04:30:35 2026

0, 1410 MHz, 89.93 W, 0 %
1, 1410 MHz, 87.52 W, 0 %
2, 1410 MHz, 85.60 W, 0 %
3, 210 MHz, 60.50 W, 0 %
4, 1410 MHz, 93.75 W, 0 %
5, 1410 MHz, 83.29 W, 0 %
6, 1410 MHz, 89.07 W, 0 %
7, 1410 MHz, 89.17 W, 0 %
```


© 2026 Clockwork All rights reserved

How to Find the Guest?



Don’t burn GPU-hours while solving the mystery.

Adapted from Google Cloud

[Taming the stragglers: Maximize AI training performance with automated straggler detection]

© 2026 Clockwork All rights reserved

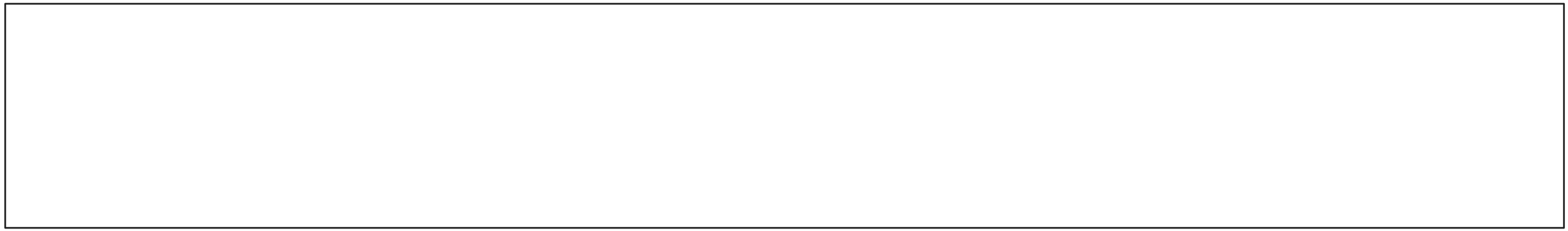


C is for Checkpoint, who wrote down the past,
In case the next séance should prove to be the last.

© 2026 Clockwork All rights reserved

Approaches to recovery

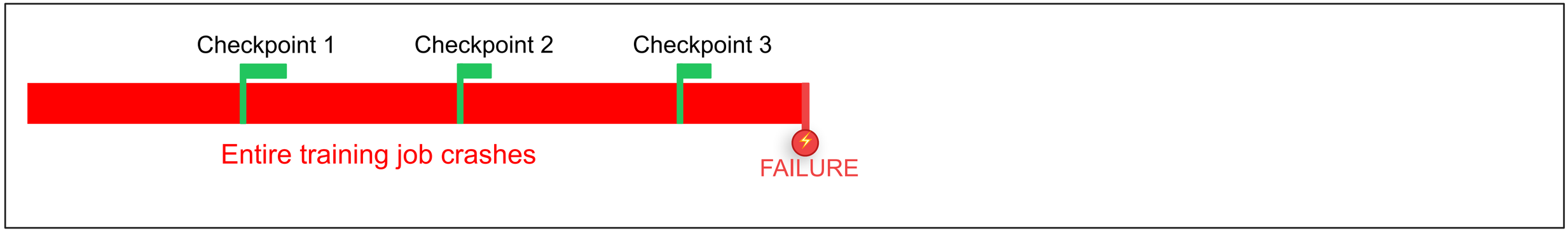
Checkpoint Restart – the “industry standard”



© 2026 Clockwork All rights reserved

Approaches to recovery

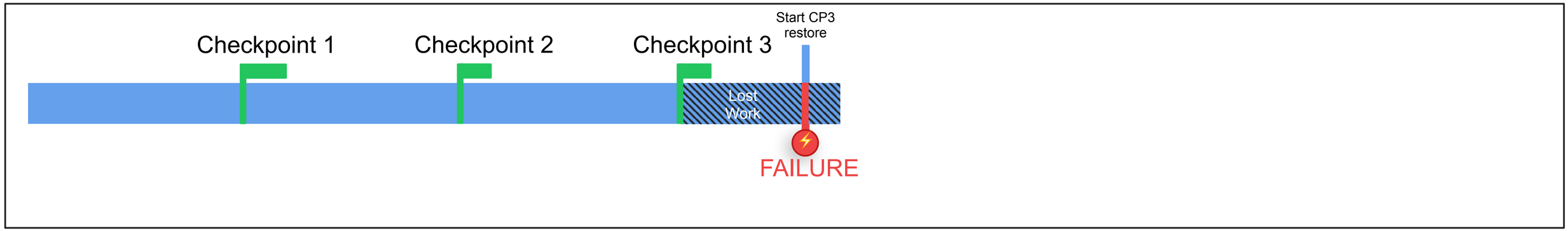
Checkpoint Restart – the “industry standard”



© 2026 Clockwork All rights reserved

Approaches to recovery

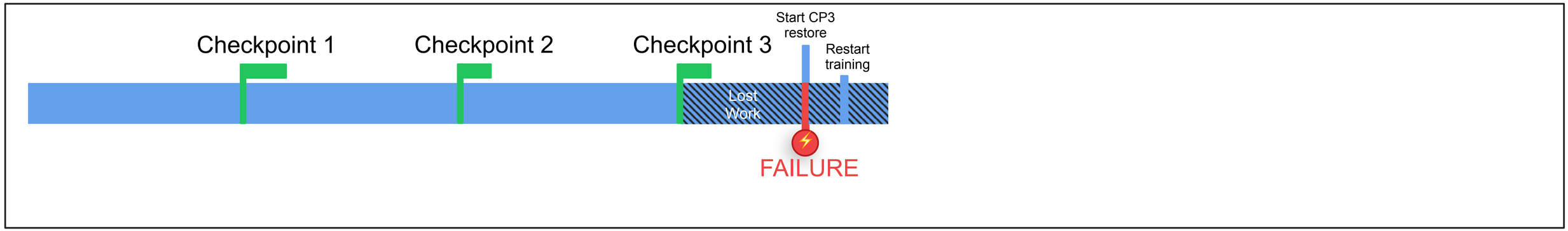
Checkpoint Restart – the “industry standard”



© 2026 Clockwork All rights reserved

Approaches to recovery

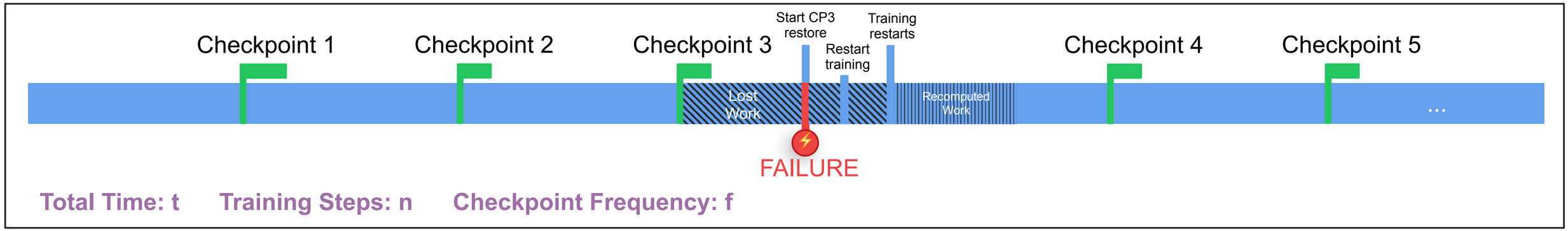
Checkpoint Restart – the “industry standard”



© 2026 Clockwork All rights reserved

Approaches to recovery

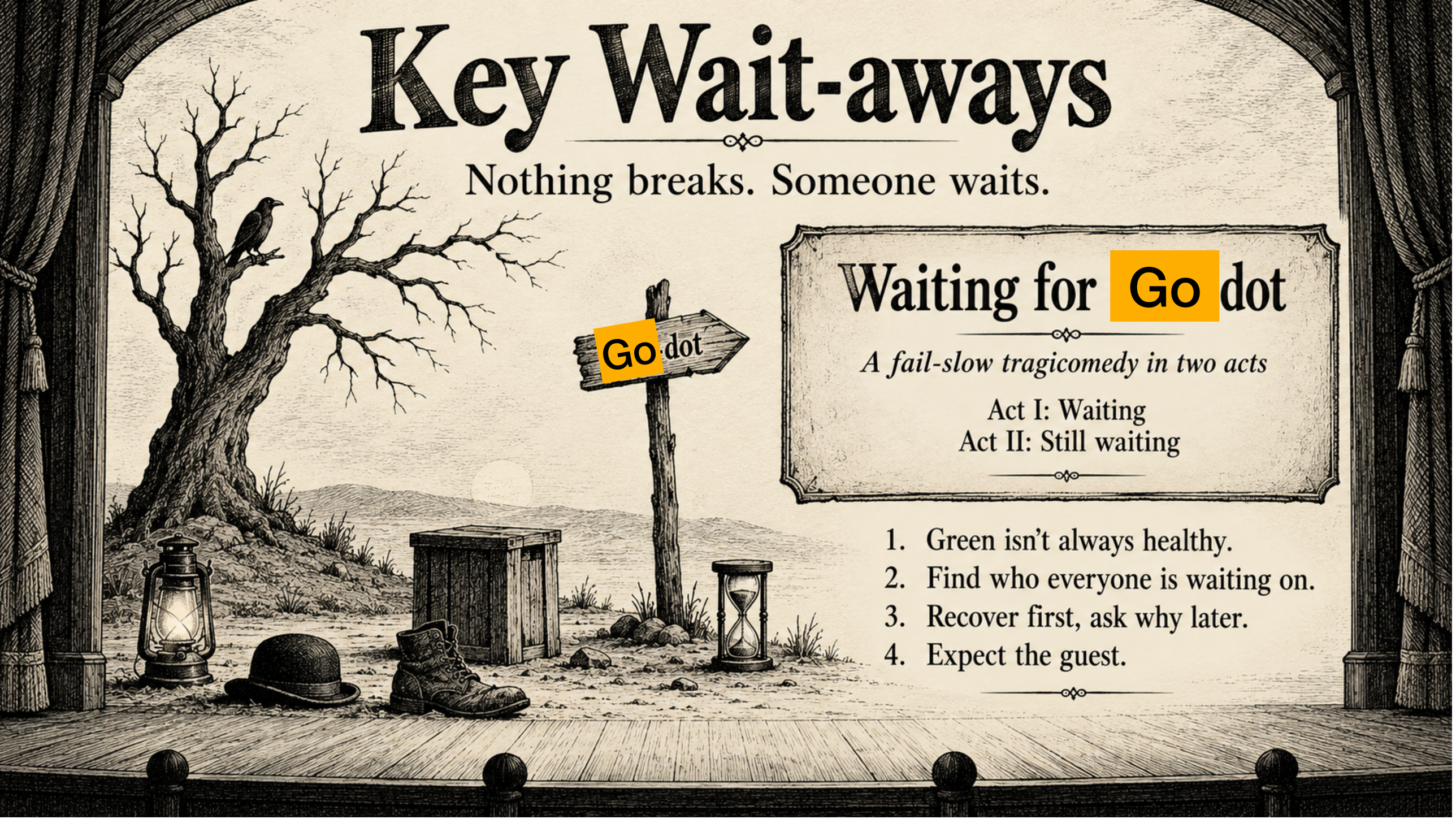
Checkpoint Restart – the “industry standard”



© 2026 Clockwork All rights reserved

Key Wait-aways

Nothing breaks. Someone waits.



Waiting for Go dot

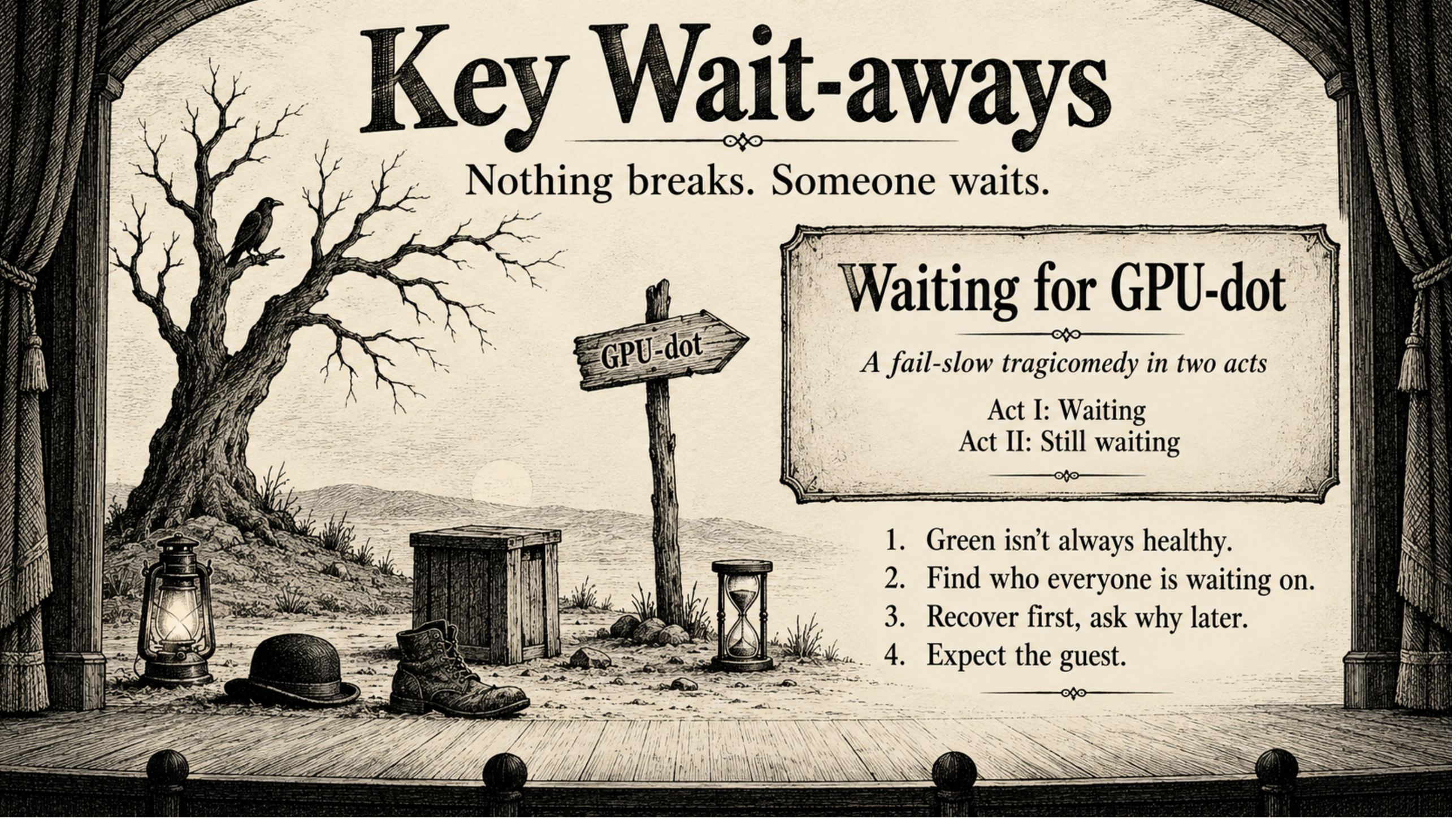
A fail-slow tragicomedy in two acts

Act I: Waiting
Act II: Still waiting

1. Green isn't always healthy.
2. Find who everyone is waiting on.
3. Recover first, ask why later.
4. Expect the guest.

Key Wait-aways

Nothing breaks. Someone waits.



Waiting for GPU-dot

A fail-slow tragicomedy in two acts

Act I: Waiting
Act II: Still waiting

1. Green isn't always healthy.
2. Find who everyone is waiting on.
3. Recover first, ask why later.
4. Expect the guest.


Thank you for coming and waiting so well;
the guest has departed. Or has it? Who can tell?
linkedin.com/in/lerna

Credits

- Inspired by Edward Gorey's *The Doubtful Guest* (1957) and *The Gashlycrumb Tinies* (1963); lines from *The Gashlycrumb Tinies* quoted.
- 3D parallelism figure: Solergibert, *A Deep Dive into 3D Parallelism with Nanotron*, CC BY-NC-ND 4.0.
- Images: Midjourney ("Key Wait-aways": ChatGPT).
- Diagrams and ring all-reduce animations: Claude.

References

- Kokolis et al. (Meta), *Revisiting Reliability in Large-Scale ML Research Clusters*, HPCA 2025.
- Google Cloud, *Taming the stragglers: Maximize AI training performance with automated straggler detection*, (straggler impact; four-step process).
- AWS, *Checkpointless training on Amazon SageMaker HyperPod*, 2025 (restart-time figures).

© 2026 Clockwork All rights reserved