Recycling the World Computer: Fault-Tolerant LLM Training on Idle GPU Capacity

Benjamin Cowen, Jason Mancuso*, Shariq Mobin

Agenda

1. What is Modal?
2. A crash course on Modal's core tech
   a. Linear programming for optimal fleet management
   b. Why the fleet will always have idle GPUs
3. How to recycle idle GPUs in a multi-cloud, multi-region fleet
4. Which tasks are most suitable for recycled compute?
   a. Massively parallel + fault-tolerant + reentrant functions with no/low inter-replica communication
5. Pushing the limits of recycled compute toward large-scale LLM training

neurips-talk 84f9cfd4

demo.py

neurips-talk — zsh

demo.py

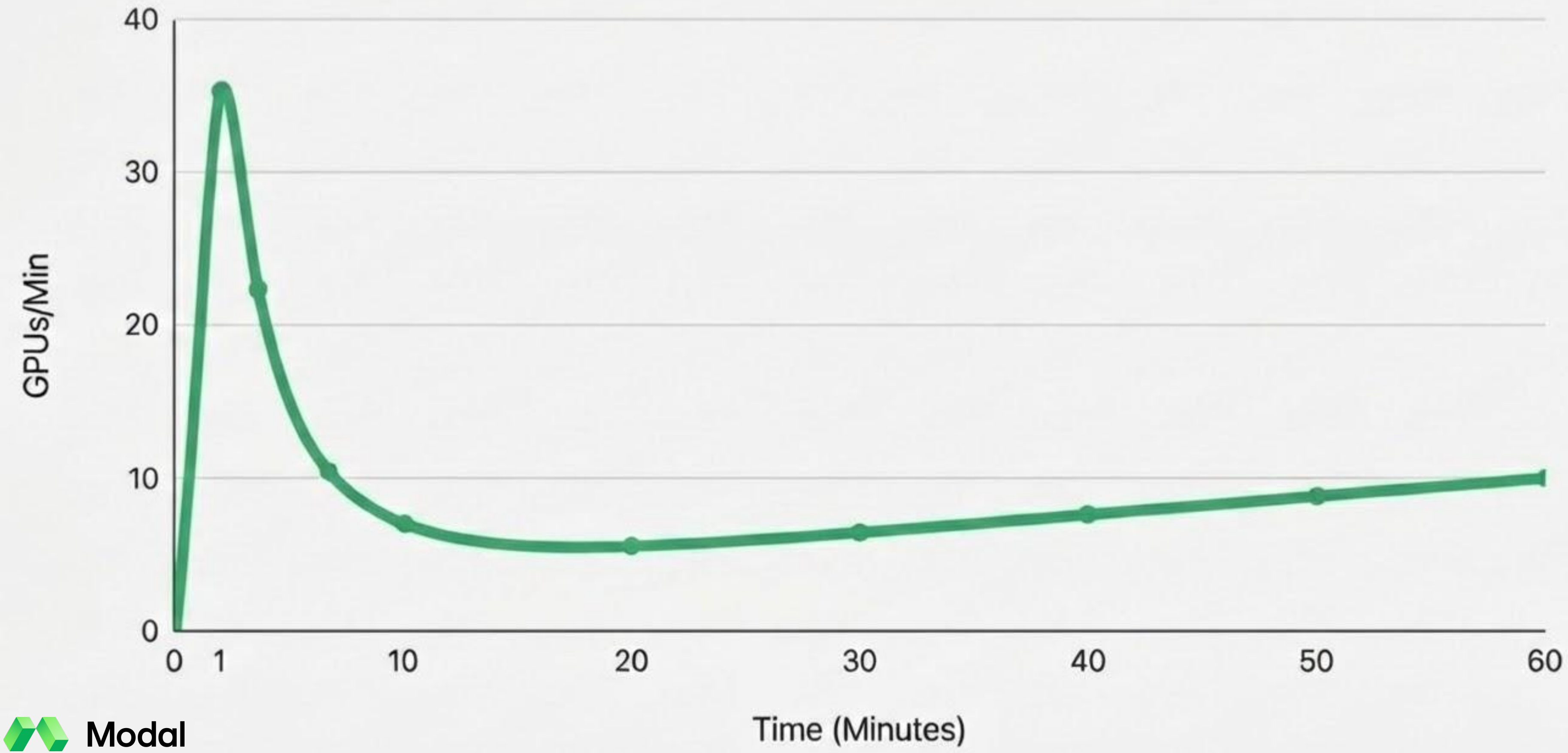
```python
import os
import subprocess
import time

import modal

app = modal.App("h100-scale-test")


@app.function(gpu=["H100", "H200", "B200", "A100", "A100-80GB",
def run_on_gpu(i: int):
    cloud, region, gpu_type = get_container_info()
    print(
        f"Running input {i} in {region} on {cloud} "
        f"with device {gpu_type}..."
    )
    time.sleep(60 * 5)


@app.local_entrypoint()
def main():
    _results = list(run_on_gpu.map(range(10_000)))


def get_container_info():
    cloud, region = os.getenv("MODAL_CLOUD_PROVIDER"), os.getenv('

    # Determine GPU type
    try:
        result = subprocess.run(
```

```
neurips-talk is 📦 v0.1.0 via 🐍 v3.13.7 (neurips-talk) on ☁ (us-east-1)
 on ☁ jason.mancuso@modal.com
❯
```

Click to restart and update Zed

Python 3.11.13 (38758ec89ea2c80; Conda)

GPU Acquisition Rate Over Time (per Function)



Modal

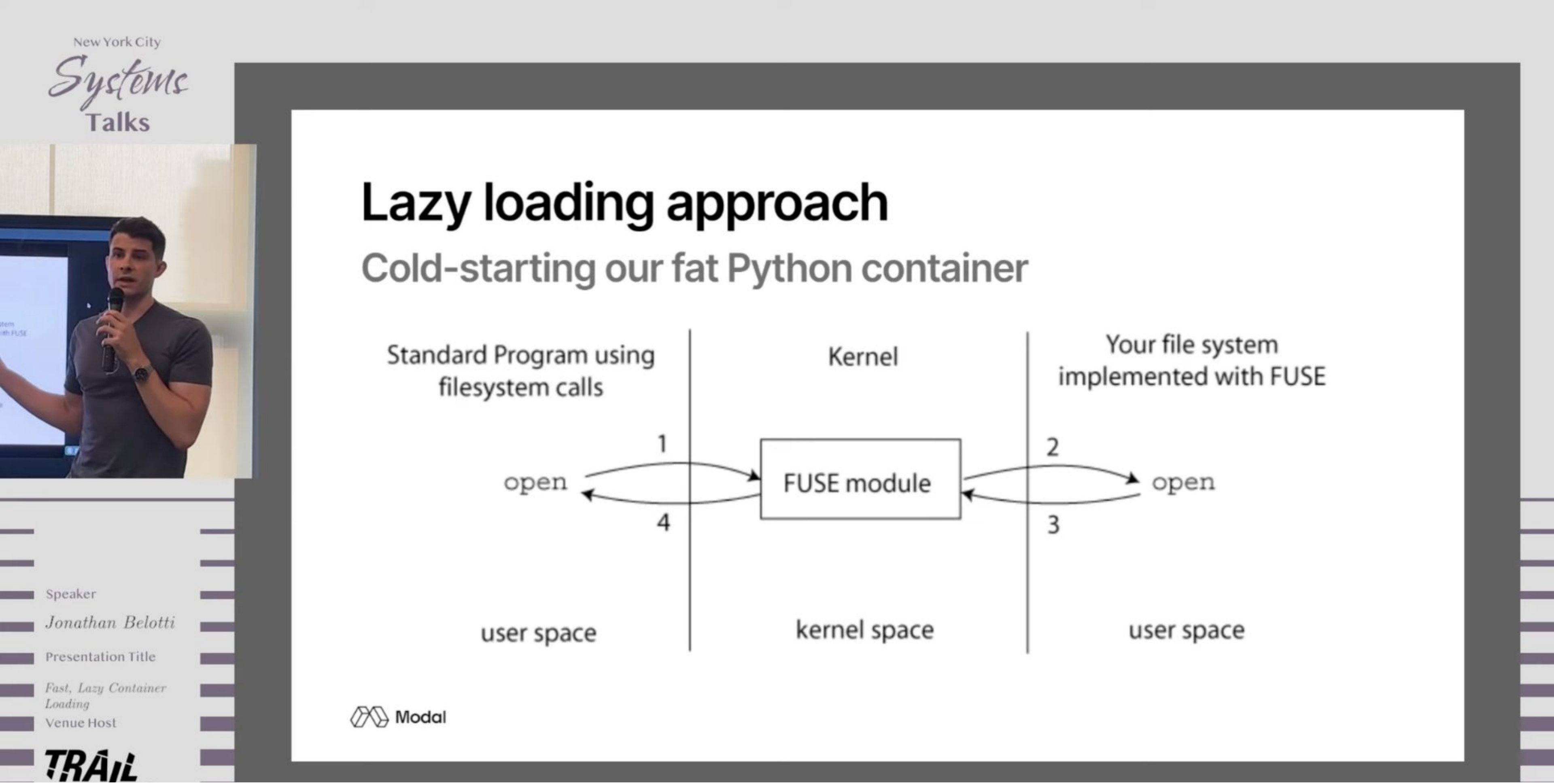
What’s our secret?

What’s our secret?

- Near instant cold-starts

What’s our secret?

- Near instant cold-starts
  - Custom container runtime, built to start from lazy-loaded, content-addressed images



Fast, lazy container loading on Modal, Jonathon Belotti (via [YouTube]())

What’s our secret?

- Near instant cold-starts
    - Custom container runtime, built to start from lazy-loaded, content-addressed images

What’s our secret?

- Near instant cold-starts
  - Custom container runtime, built to start from lazy-loaded, content-addressed images
- Large, elastic, global compute pool

What’s our secret?

- Near instant cold-starts
  - Custom container runtime, built to start from lazy-loaded, content-addressed images
- Large, elastic, global compute pool
  - Multi-cloud worker fleet that scales dynamically in response to user workload demand

What's our secret?

- Near instant cold-starts
  - Custom container runtime, built to start from lazy-loaded, content-addressed images
- Large, elastic, global compute pool
  - Multi-cloud worker fleet that scales dynamically in response to user workload demand

Multi-cloud Substrate

ORACLE CLOUD | Google Cloud | aws | Azure | Neoclouds

What’s our secret?

- Near instant cold-starts
  - Custom container runtime, built to start from lazy-loaded, content-addressed images
- Large, elastic, global compute pool
  - Multi-cloud worker fleet that scales dynamically in response to user workload demand

Managing our fleet is not only a logistics problem, but also an arbitrage opportunity.

Multi-cloud Substrate

ORACLE CLOUD | Google Cloud | aws | Azure | Neoclouds

Parameters:

$$A := \text{total GPUs requested by users}$$

$$B := \text{count of GPUs in the buffer}$$

$$I = \{I_1, I_2, \ldots, I_n\} := \text{possible instance types}$$

$$\mathbf{C} = [C_1, \ldots, C_n] := \text{cost of instance types}$$

$$\mathbf{L} = [L_1, \ldots, L_n] := \text{scaling limits for each type}$$

Output:

$$\mathbf{X} = [x_1, \ldots, x_n] := \text{instances of each type to launch}$$

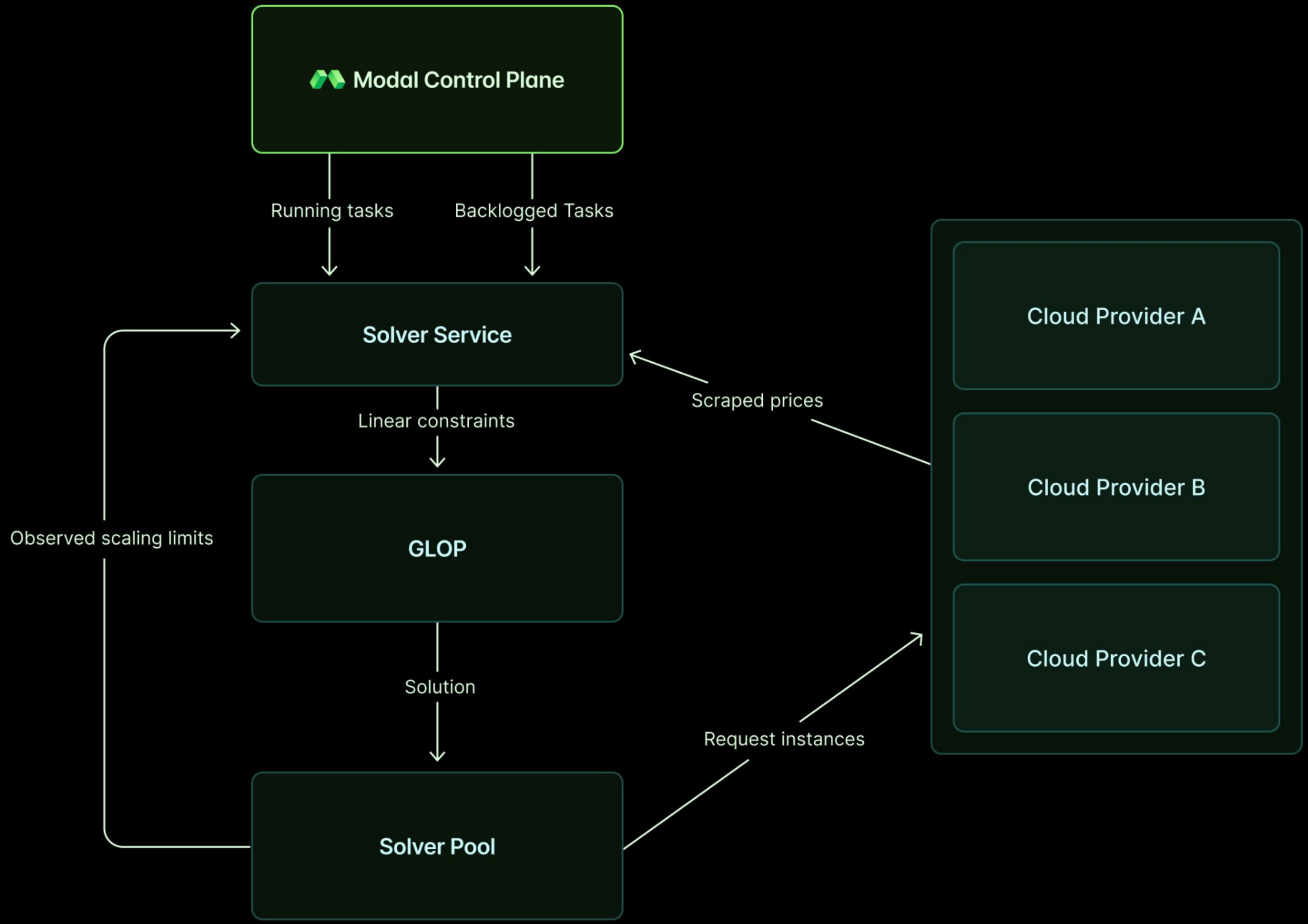
Objective:

$$\text{minimize } \mathbf{C}^{\mathrm{T}}\mathbf{X} = \sum_{i=1}^{n} x_i C_i$$

Constraints:

$$8 \sum_{i=1}^{n} x_i \geq A + B$$

$$x_i \leq L_i \quad (i = 1, \ldots, n)$$


Modal Control Plane
Running tasks
Backlogged Tasks
Solver Service
Linear constraints
GLOP
Solution
Solver Pool
Observed scaling limits
Scraped prices
Request instances
Cloud Provider A
Cloud Provider B
Cloud Provider C

Sources of Idle Compute in the Fleet

Sources of Idle Compute in the Fleet

- Buffer slots on active workers

Sources of Idle Compute in the Fleet

- Buffer slots on active workers
    - These are intentional. We explicitly overprovision so that we can accommodate bursts in our user’s workloads.

Sources of Idle Compute in the Fleet

- Buffer slots on active workers
  - These are intentional. We explicitly overprovision so that we can accommodate bursts in our user's workloads.
- Idle slots on draining workers

Sources of Idle Compute in the Fleet

- Buffer slots on active workers
  - These are intentional. We explicitly overprovision so that we can accommodate bursts in our user's workloads.
- Idle slots on draining workers
  - When the solver decides to prune a worker, we can't move it over until user tasks have fresh containers elsewhere in the fleet.
  - Unused GPUs on draining workers can't accept new tasks, so they remain idle for the duration.

Recycled Compute

Running background tasks on these “idle” GPUs, whether in the buffer or in the drain.

For background tasks in the buffer, we need to be able to SIGKILL in ~10s or 100s of milliseconds.

How much Recycled Compute is available?

GPU Types (Inclusive)	Min VRAM	Estimated* Idle Compute	% Draining
B200,H200	141GB+	100k GPU-hrs/month	10%+
H100, A100-80 + above	80GB+	300k GPU-hrs/month	10%+
A100-40, L40S + above	40GB+	500k GPU-hrs/month	20%+
A10G, L4 + above	24GB+	1.2M GPU-hrs/month	30%+

How much Recycled Compute is available?

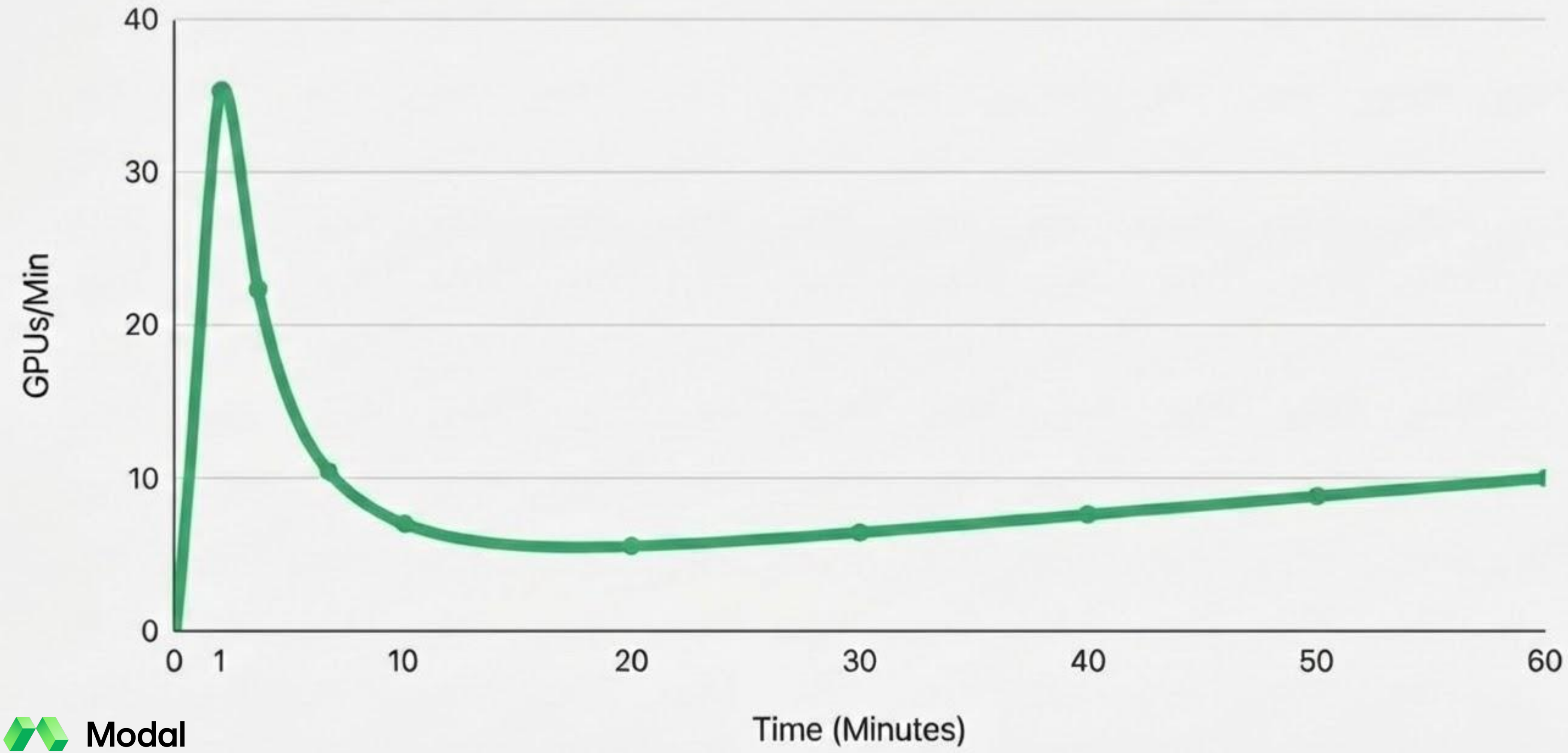
GPU Types (Inclusive)	Min VRAM	Estimated* Idle Compute	% Draining
B200,H200	141GB+	100k GPU-hrs/month	10%+
H100, A100-80 + above	80GB+	300k GPU-hrs/month	10%+
A100-40, L40S + above	40GB+	500k GPU-hrs/month	20%+
A10G, L4 + above	24GB+	1.2M GPU-hrs/month	30%+

Training Costs	Pre-Training	Context Extension	Post-Training	Total
in H800 GPU Hours	2664K	119K	5K	2788K
in USD	$5.328M	$0.238M	$0.01M	$5.576M

Table 1 | Training costs of DeepSeek-V3, assuming the rental price of H800 is $2 per GPU hour.

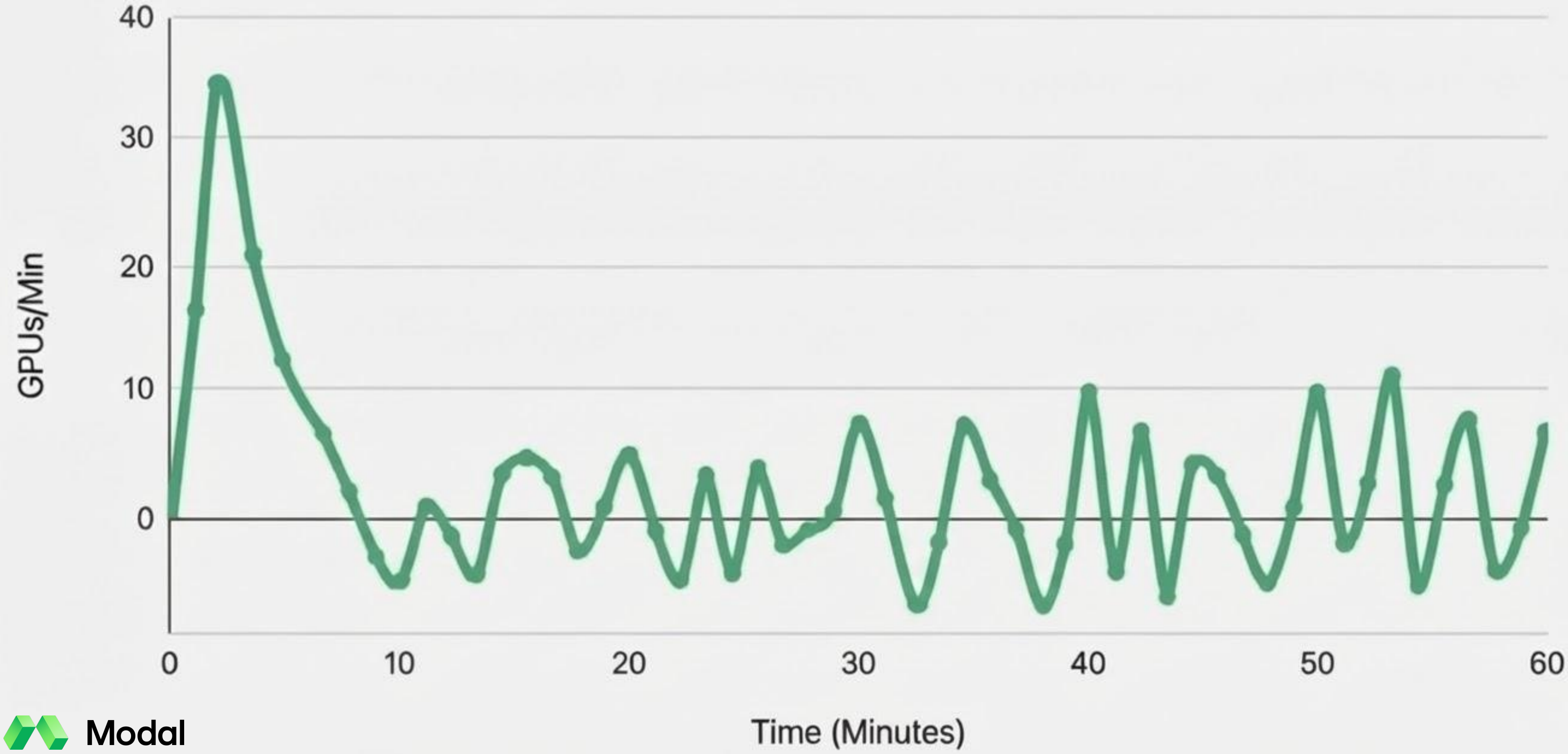
Deepseek-AI 2025 (arxiv: 2412.19437)

GPU Acquisition Rate Over Time (per Function)



Modal

GPU Acquisition Rate Over Time (per **Recycled** Function)



Modal

The Shape of Recycled Compute

- Massively parallel
- Skewed hardware heterogeneity
- High availability in aggregate, but low for any one device
    - High preemption likelihood
- High variance interconnect
    - Inter-container network could be low-latency, high-bandwidth if within single AZ
        - But can’t rely on Infiniband/NVLink/RDMA :(
    - Could be low-latency if within single geographic region
    - Could also be in completely different regions, communicating over unreliable WAN

The Shape of *Ideal Tasks* for Recycled Compute

- Massively parallel
- Hardware-agnostic
- Reentrant
  - Easy to checkpoint + resume on fresh container
- Parallel-wise fault-tolerant
  - Can still make progress if many individual replicas are preempted
- Little to no inter-replica communication

Ideal Tasks for Recycled Compute

	Hardware agnostic?	Reentrant?	Fault-tolerant parallelism?	Low comm?
Classical protein folding	😍	😍	😍	😍
Molecular physics simulation	😍	😍	😍	😍
Offline batch inference	😕	😍	👍	😍
Synthetic data (agentic rollouts)	😕	😍	👍	😍
Distributed LLM training	😠	👍	👹	😭

Low-Communication Distributed Training, aka Federated Learning

DiLoCo: Distributed Low-Communication Training of Language Models

Arthur Douillard[1], Qixuan Feng[1], Andrei A. Rusu[1], Rachita Chhaparia[1], Yani Donchev[1], Adhiguna Kuncoro[1], Marc'Aurelio Ranzato[1], Arthur Szlam[1] and Jiajun Shen[1]

[1]Google DeepMind

arxiv: 2311.08105

Low-Communication Distributed Training, aka Federated Learning

Google DeepMind

Streaming DiLoCo with overlapping communication:

Towards a Distributed Free Lunch

Arthur Douillard[*,1], Yanislav Donchev[*,1], Keith Rush[2], Satyen Kale[†,2], Zachary Charles[2], Zachary Garrett[2], Gabriel Teston[3], Dave Lacey[1], Ross McIlroy[1], Jiajun Shen[1], Alexandre Ramé[1], Arthur Szlam[1], Marc'Aurelio Ranzato[1] and Paul Barham[1]

[1]Google DeepMind, [2]Google Research, [3]Google, [*]Equal core contributions, [†]Currently at Apple.

arxiv: 2501.18512

Low-Communication Distributed Training, aka Federated Learning

Communication-Efficient Language Model Training Scales Reliably and Robustly: **Scaling Laws for DiLoCo**

Zachary Charles[1*] Gabriel Teston[2] Lucio Dery[3] Keith Rush[1]
Nova Fallen[1] Zachary Garrett[1] Arthur Szlam[3] Arthur Douillard[3]

[1]Google Research [2]Google Search [3]Google DeepMind

arxiv: 2503.09799

Low-Communication Distributed Training, aka Federated Learning

Communication-Efficient Language Model Training Scales Reliably

Harder: DiLoCo's hyperparameters are robust and predictable across model scales.

Better: DiLoCo further improves over data-parallel training as model size increases.

Faster: DiLoCo uses orders of magnitude less bandwidth than data-parallel training.

Stronger: DiLoCo tolerates a significantly larger batch size than data-parallel training.

arxiv: 2503.09799

But what about fault-tolerance?!

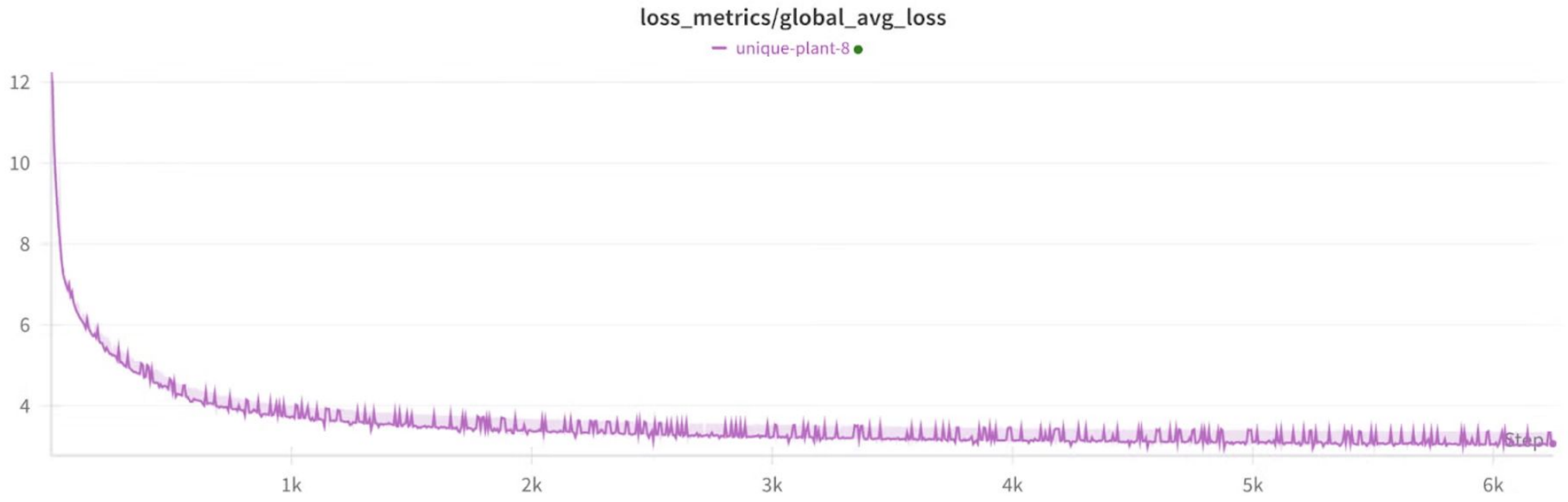
Blog

Fault Tolerant Llama: training with 2000 synthetic failures every ~15 seconds and no checkpoints on Crusoe L40S

By Tristan Rice, Howard Huang | June 20, 2025

Via pytorch.org

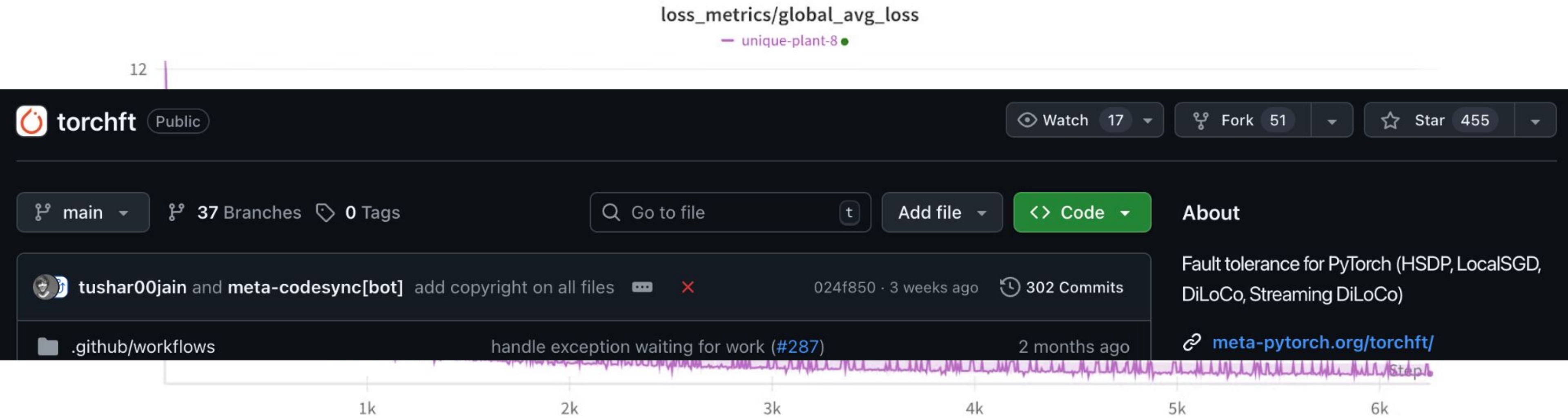
But what about fault-tolerance?!



Training loss across 1200 failures with no checkpoints.

NOTE: Each small spike is a non-participating worker recovering which affects the metrics but not the model

But what about fault-tolerance?!



Training loss across 1200 failures with no checkpoints.

NOTE: Each small spike is a non-participating worker recovering which affects the metrics but not the model

Don’t Try This at Home

Don’t Try This at Home

TorchFT ProcessGroup issues

Don’t Try This at Home

TorchFT ProcessGroup issues

Explosion of egress costs

→ Need to customize collective communication ops to be geography-aware

Don't Try This at Home

TorchFT ProcessGroup issues

→ We use Tunnels for inter-container networking

→ ProcessGroup requires

Explosion of egress costs

→ Need to customize collective communication ops to be geography-aware

So, back to inference and evolutionary algorithms we go!

Thanks!

We’re hiring: https://modal.jobs, or stop by Kiosk 9 in Expo Hall B!

$30 free Modal credits: https://modal.com/signup

Modal