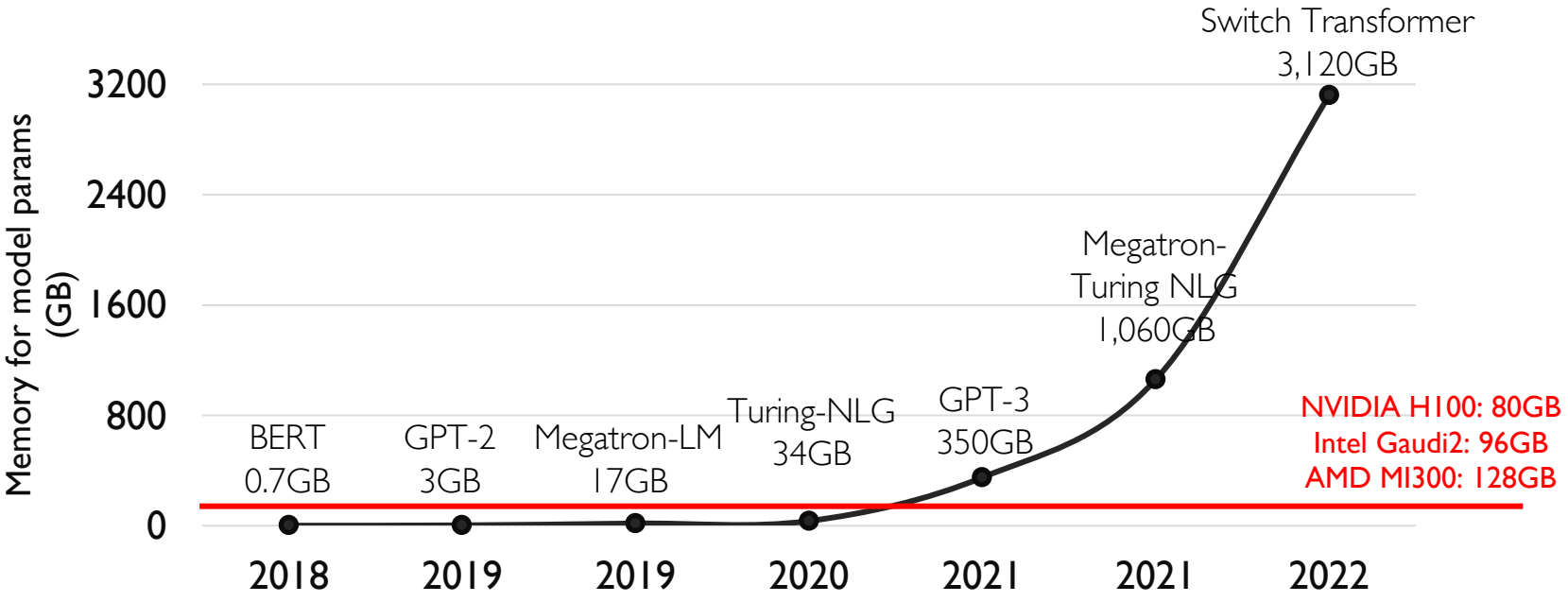
Oobleck

Resilient Distributed Training of Large Models Using Pipeline Templates

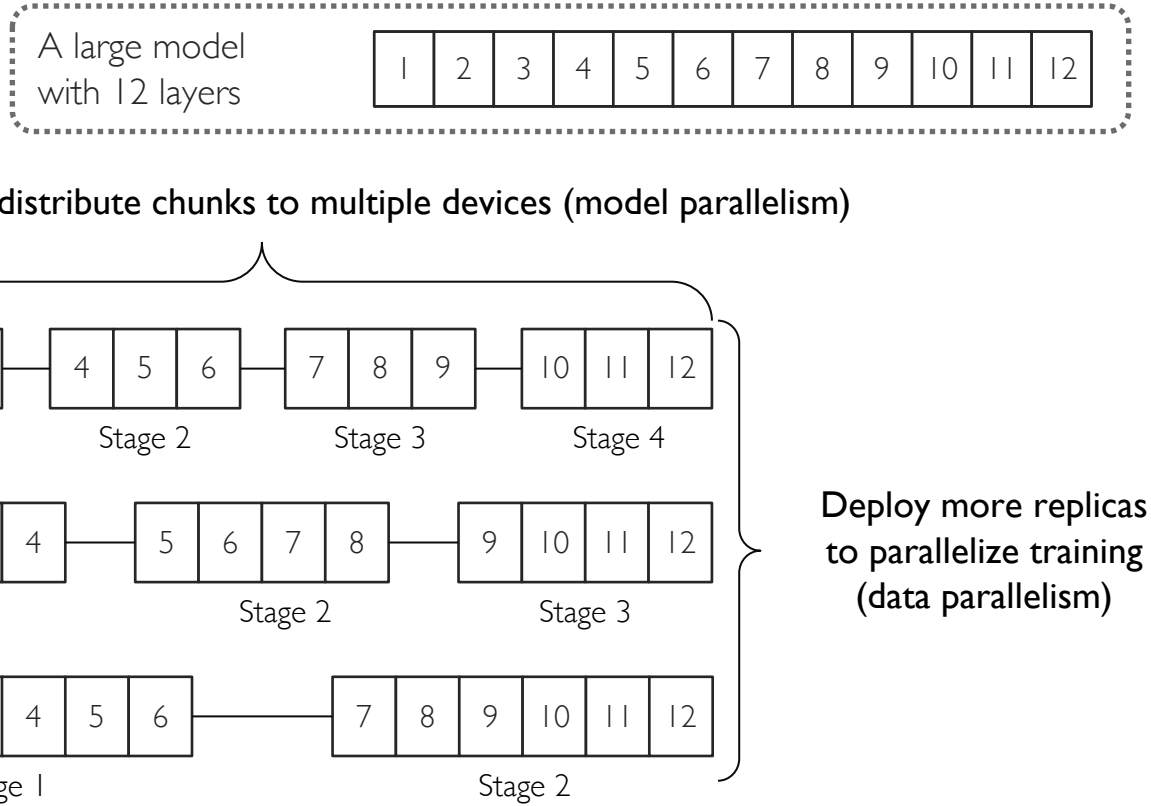
Insu Jang, Zhenning Yang, Zhen Zhang, Xin Jin, and Mosharaf Chowdhury

Models are Becoming Larger



Assumes 2 bytes per parameter.

Hybrid Parallelism is Becoming a Norm



Failures Getting Noticeable

Higher probability of GPU failing

We use more GPUs, each of which can fail independently

More GPUs are used

Higher cost of GPU failing

The entire training must pause, wasting all working GPUs due to synchronization

Reports about the impact of failures in training large models

- Meta AI training OPT: "Estimated 100+ host restarts due to hardware failures over the course of 2 months." [1]
- LAION training CLIP models: "Hardware issue is an annoying problem as if one GPU has an issue, all GPUs get stuck." [2]

[1] Susan Zhang et al. "OPT: Open Pre-trained Transformer Language Models". Arxiv'22
[2] Romain Beaumont, "Large Scale OpenCLIP: L/14, H/14 and G/14 Trained on LAION-2B". https://laion.ai/blog/large-openclip/

Resilient Training Requirements

	Bamboo [1]	Varuna [2]	Oobleck (ours)
Guaranteed fault tolerance	✗ No guarantee for ≥ 2 simultanoues failures	? No formal fault tolerance guarantee	✓
High throughput	✗ High computational overheads	? Getting slower when recovery overheads are higher	✓
Fast recovery	✓ Dynamic reconfiguration without restart	✗ Full restart from the last checkpoint	✓

[1] John Thorpe et al. "Bamboo: Making Preemptible Instances Resilient for Affordable Training of Large DNNs". NSDI'23
[2] Sanjith Athlur et al. "Varuna: Scalable, Low-cost Training of Massive Deep Learning Models". EuroSys'22

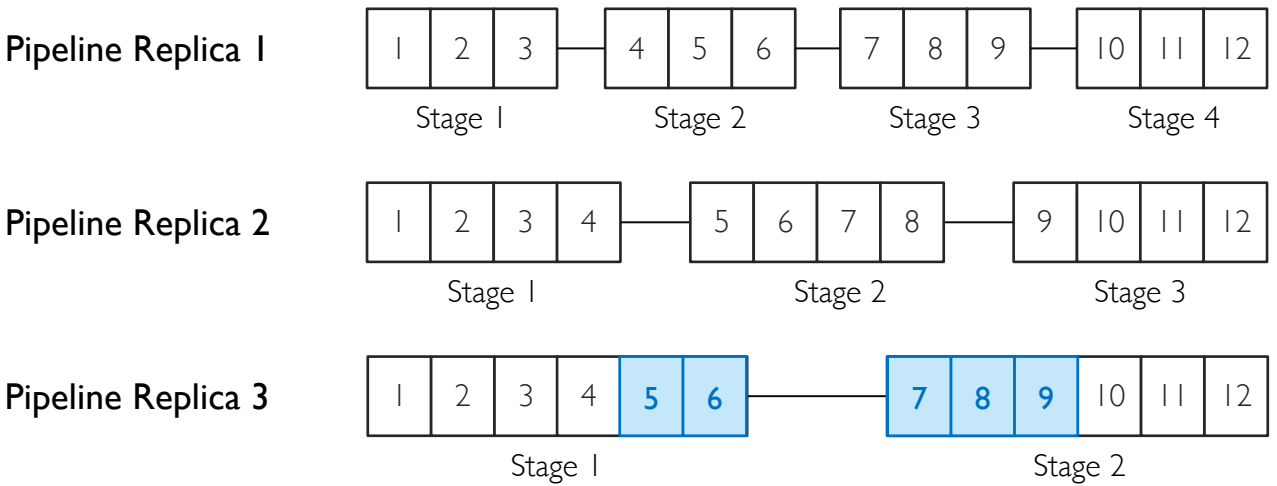
Oobleck: Overview

- **Guaranteed fault tolerance**
  - Hybrid parallelism has multiple replicas of a model
  - Utilize inherent redundancy in hybrid parallel execution

- **Introducing pipeline template**
  - Oobleck's core idea to achieve both **high throughput** and **fast recovery** simultanouesly
  - A specification of pipeline execution

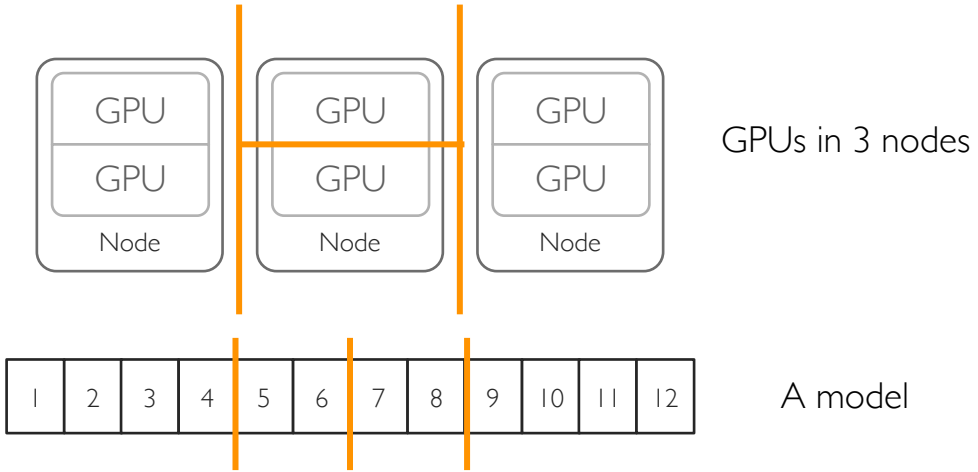
Fault Tolerance Guarantee

- Utilize inherent redundancy in hybrid parallel execution



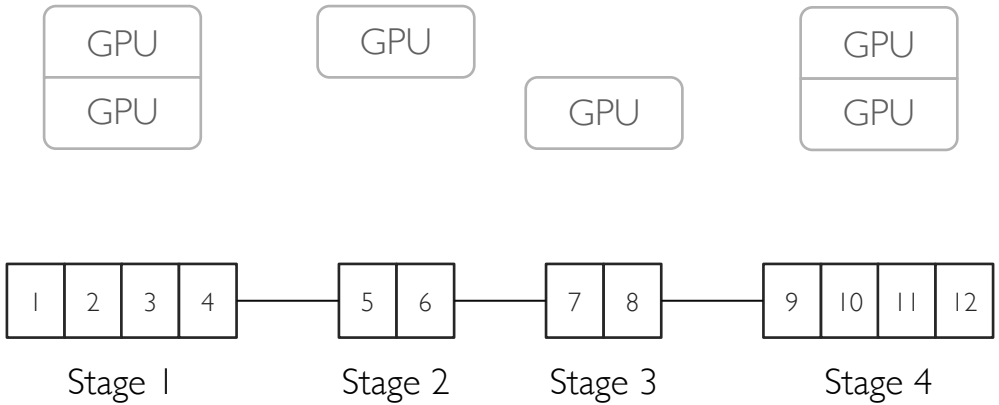
Pipeline Template

- Each template is **a pre-generated single pipeline execution specification** for specific number of nodes



Pipeline Template

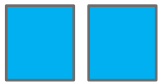
- Each template is a pre-generated single pipeline execution specification for specific number of nodes



A Pipeline Template for 3 Nodes

Parallel Execution Configuration

- Parallel execution plan is configured as **a linear combination of templates**
- Use all nodes & reduce search space but provide high throughput



Pipeline Template for 2 Nodes

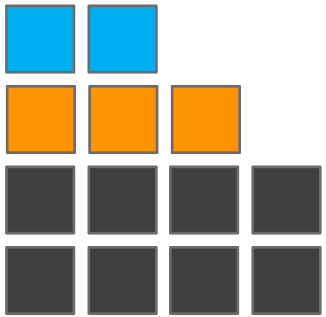
Pipeline Template for 3 Nodes

Pipeline Template for 4 Nodes

...

...

Pipeline Template for *k* Nodes

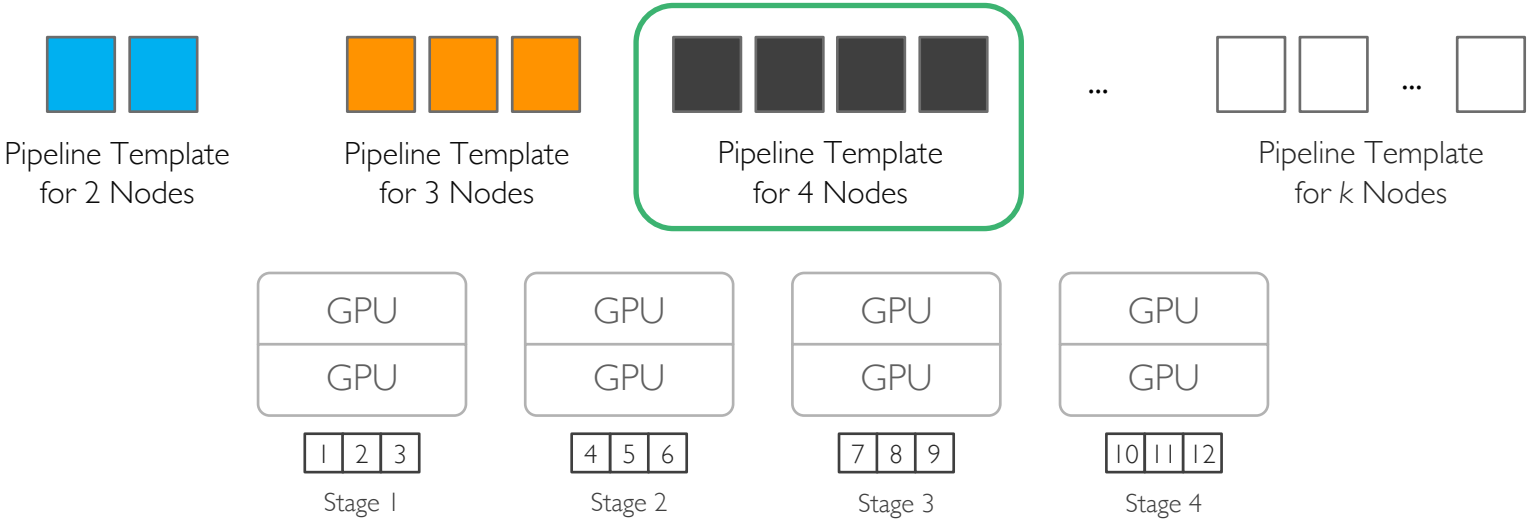


An example of parallel execution plan with 13 nodes

1× pipeline from the pipeline template for 2 nodes
1× pipeline from the pipeline template for 3 nodes
2× pipelines from the pipeline template for 4 nodes

Failure Recovery

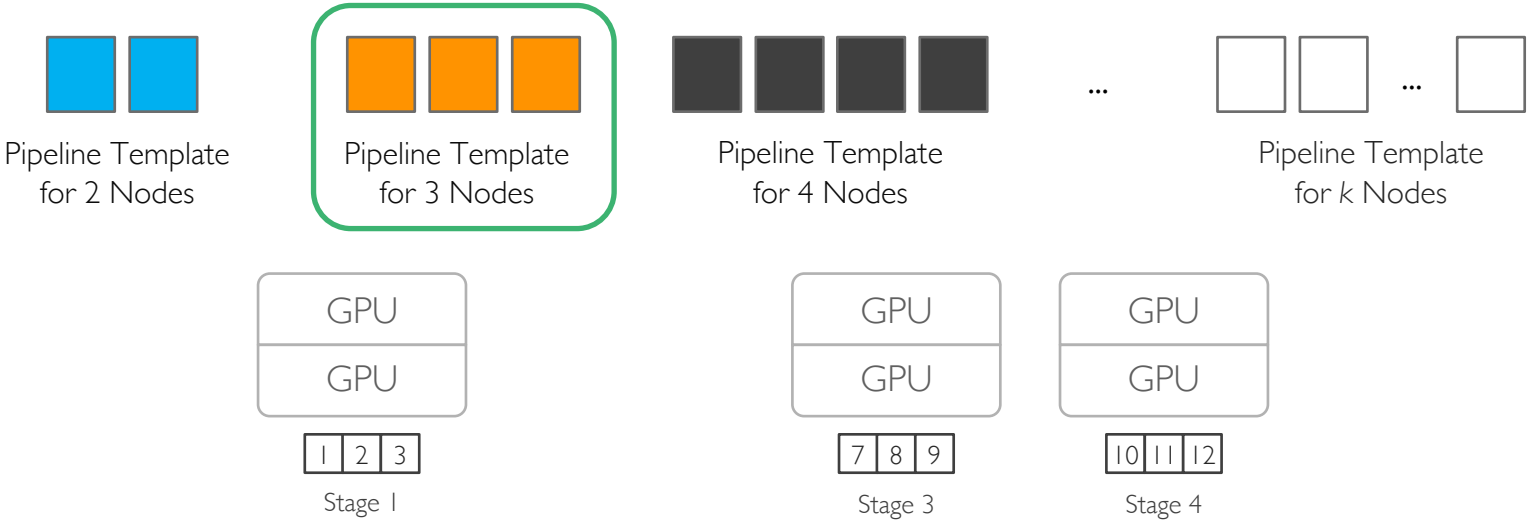
- **Quickly reinstantiate a new pipeline** from a template when failures happen



A pipeline instantiated from the template for 4 nodes

Failure Recovery

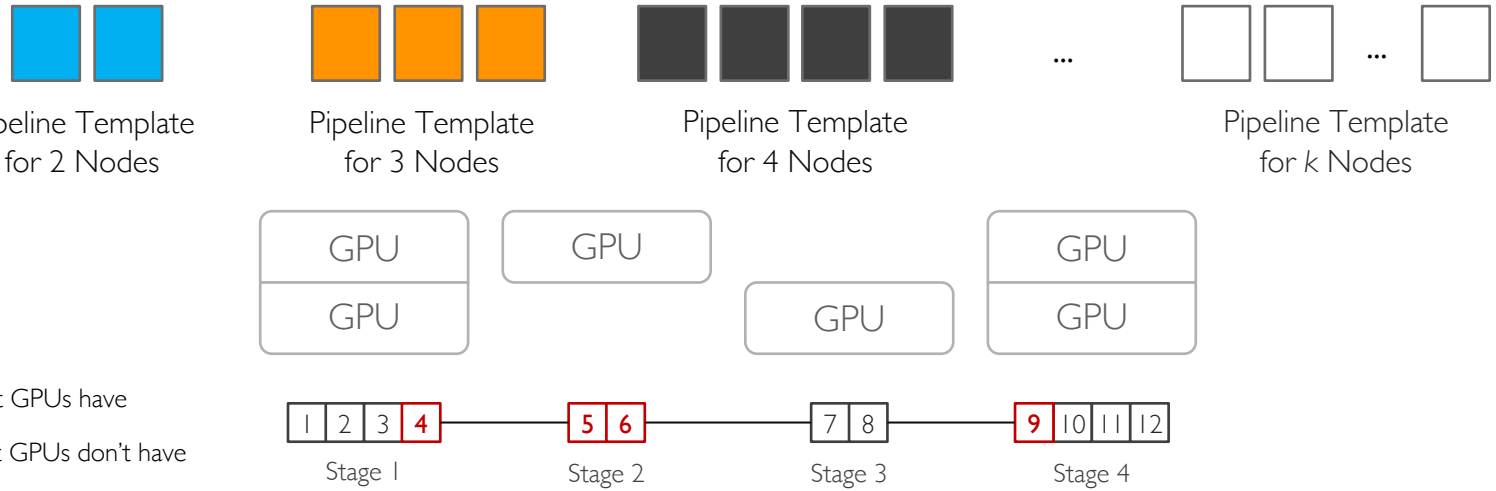
- **Quickly reinstantiate a new pipeline** from a template when failures happen



A node fails and GPUs are lost

Failure Recovery

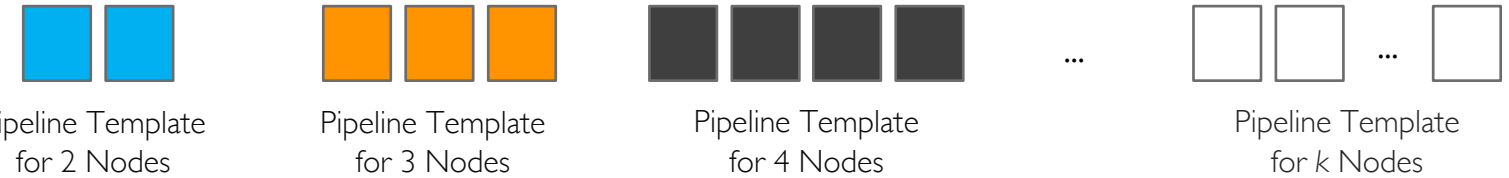
- **Quickly reinstantiate a new pipeline** from a template when failures happen

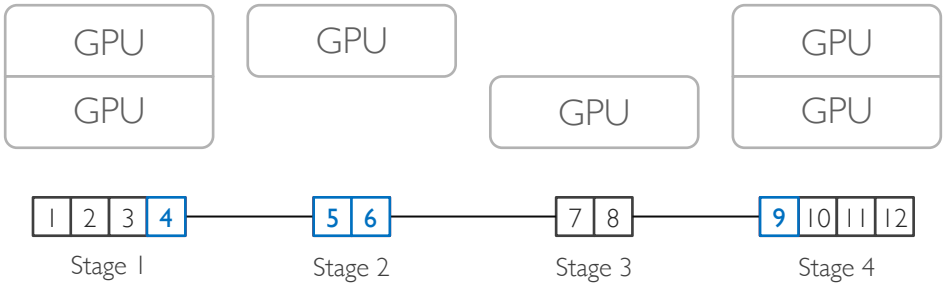


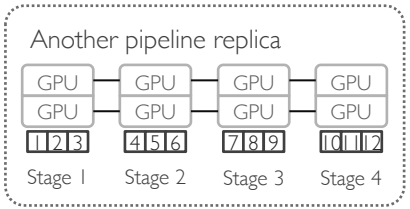
Instantiate a pipeline template for 3 nodes

Failure Recovery

- **Quickly reinstantiate a new pipeline** from a template when failures happen







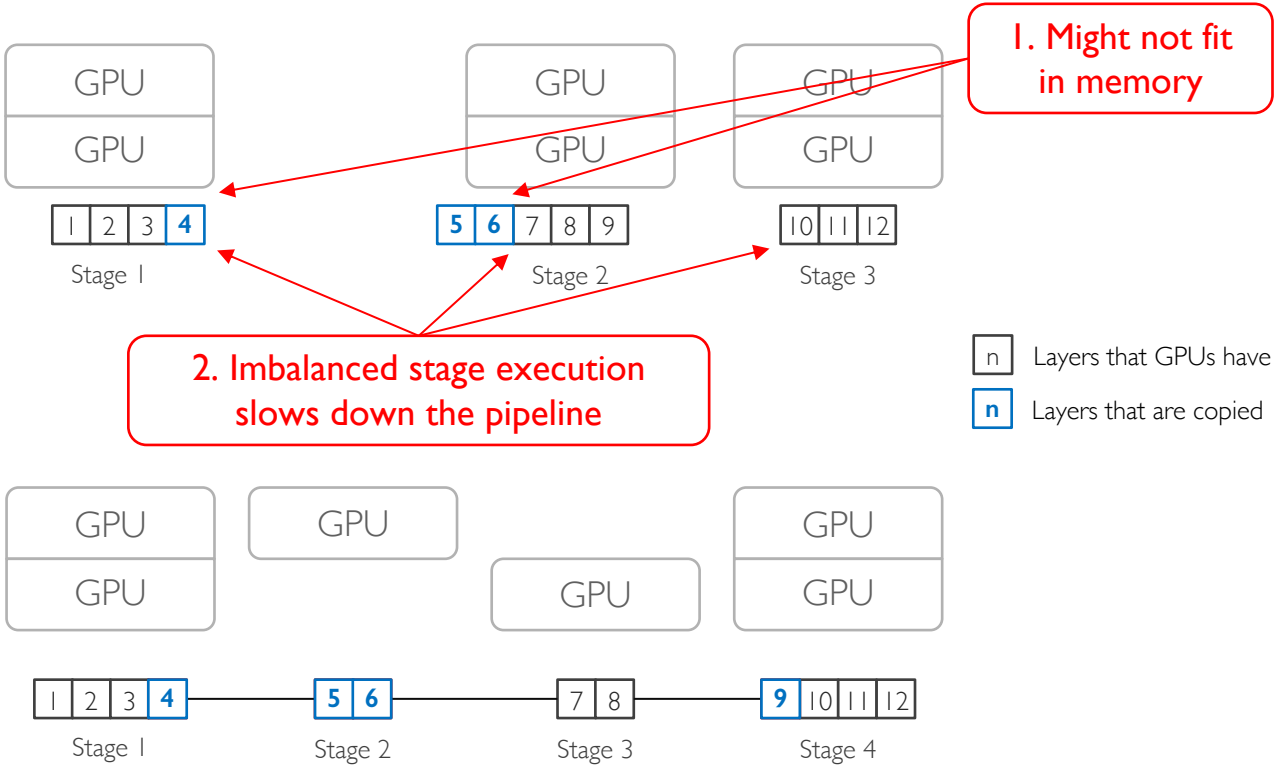
Copy missing layers from replica(s)

Reinstantiation vs Just Copying Layers

Copying lost layers to adjecent nodes without reinstantiation

VS

Pipeline reinstantiation



Issues in Using Pipeline Templates

1. Determining # pipeline templates and # nodes for each template
   → Node Specification

2. Determining number of pipelines to be instantiated from each template
   → Pipeline Instantiation

3. What if there is no feasible pipeline template to be instantiated?
   → Pipeline Merge

1. Node Specification

- No need to have a pipeline template for every possible # nodes

Train a model (required to have ≥ 2 nodes to train) with 13 nodes
How many pipeline templates do we need?

Pipeline Template for 2 Nodes	Pipeline Template for 3 Nodes	Pipeline Template for 4 Nodes	Pipeline Template for 5 Nodes
Pipeline Template for 6 Nodes	Pipeline Template for 7 Nodes	Pipeline Template for 8 Nodes	Pipeline Template for 9 Nodes
Pipeline Template for 10 Nodes	Pipeline Template for 11 Nodes	Pipeline Template for 12 Nodes	Pipeline Template for 13 Nodes

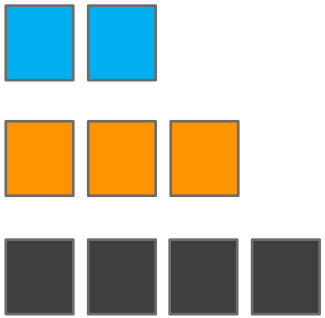
1. Node Specification

- Finding # templates & # nodes per template formulated as **a Frobenius problem**
- **Provable guarantee** that a linear combination of the set of pipeline templates use all nodes **even after failures**

Pipeline Template A (2 nodes)

Pipeline Template B (3 nodes)

Pipeline Template C (4 nodes)

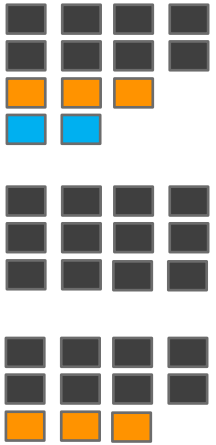


3 Heterogeneous Pipeline Templates

13 nodes

12 nodes

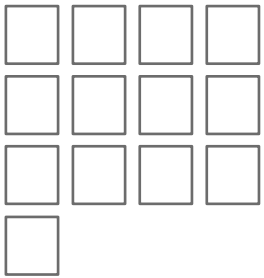
11 nodes



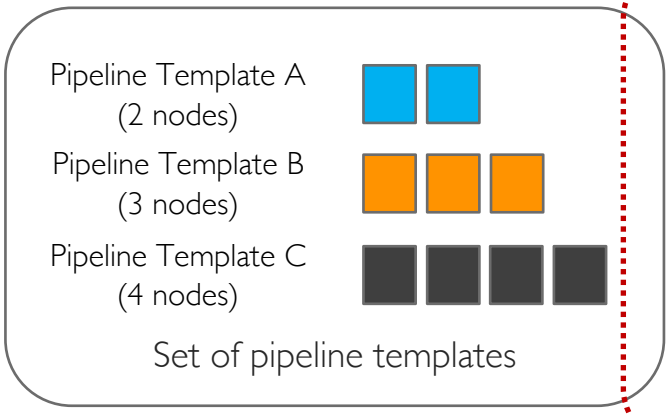
Any $2 \leq N \leq 13$ can be represented with the set of pipeline templates

2. Pipeline Instantiation

- Execution engine instantiates pipelines from pipeline templates that use all nodes

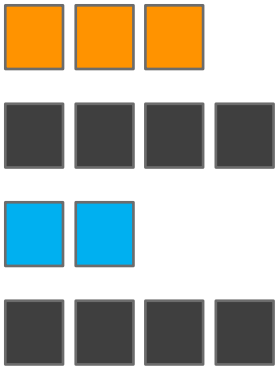


13 nodes



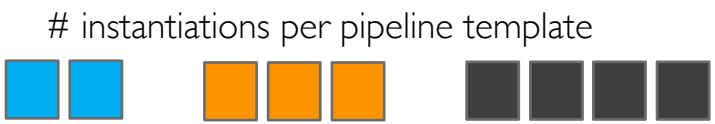
×1
×1
×2

The plan is not unique



2. Pipeline Instantiation

- Use **dynamic programming** to enumerate all possible instantiation plans
- Estimate iteration time of every plans and pick the best one



	# instantiations per pipeline template			Total # nodes used
Plan 1	x1	x1	x2	13
Plan 2	x0	x3	x1	13
Plan 3	x5	x1	x0	13

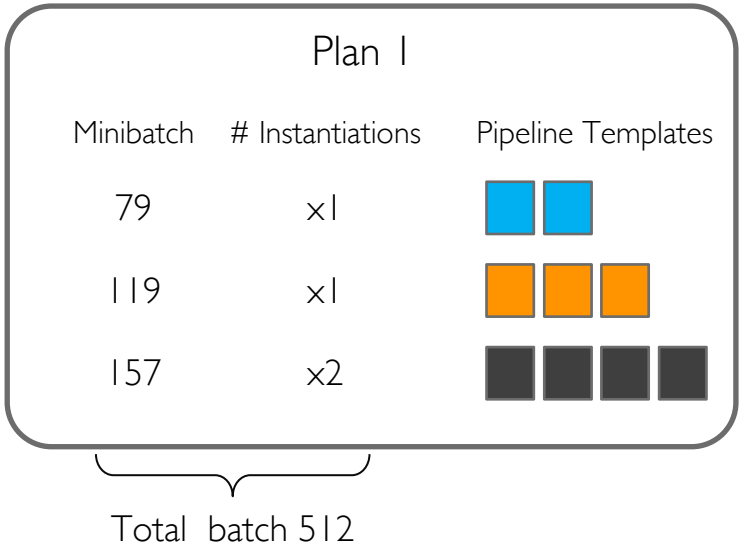
more options

Batch Distribution

- Need to know batch size per pipeline to estimate iteration time
- Formulate finding batch distribution that minimizes overall iteration time as an integer optimization problem

Global batch
512

"Find batch size of each pipeline"



Batch Distribution

- Estimate iteration time of every plans and pick the best one

Plan 1

Minibatch	# Instantiations	Pipeline Templates
79	x1	
119	x1	
157	x2	

Plan 2

N/A	x0	
118	x3	
158	x1	

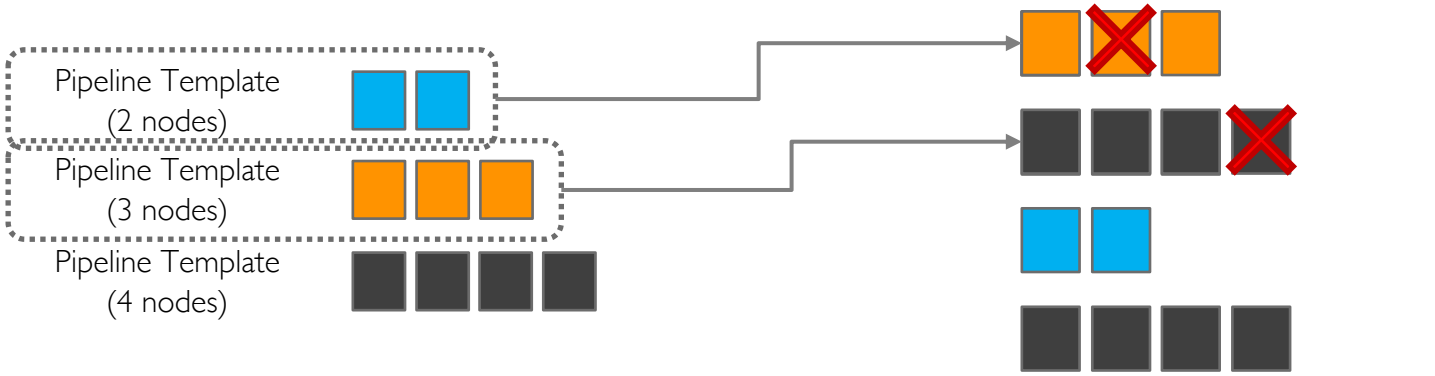
Plan 3

79	x5	
117	x1	
N/A	x0	

more options ...

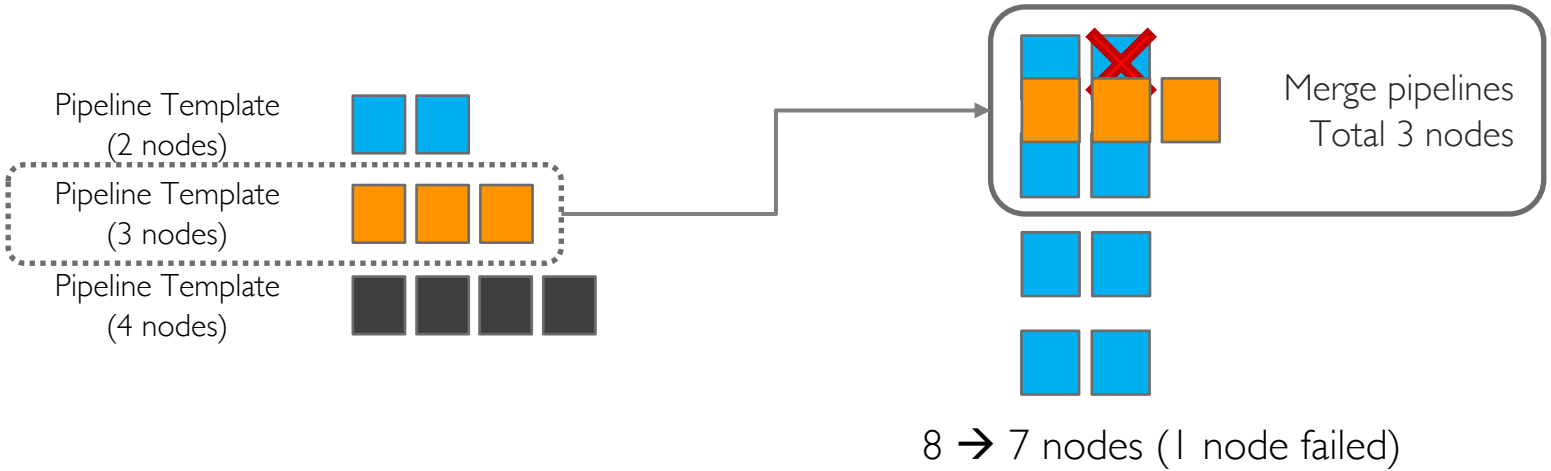
3. Pipeline Merge

- **Reinstantiate a new pipeline** from another pipeline template when failures happen

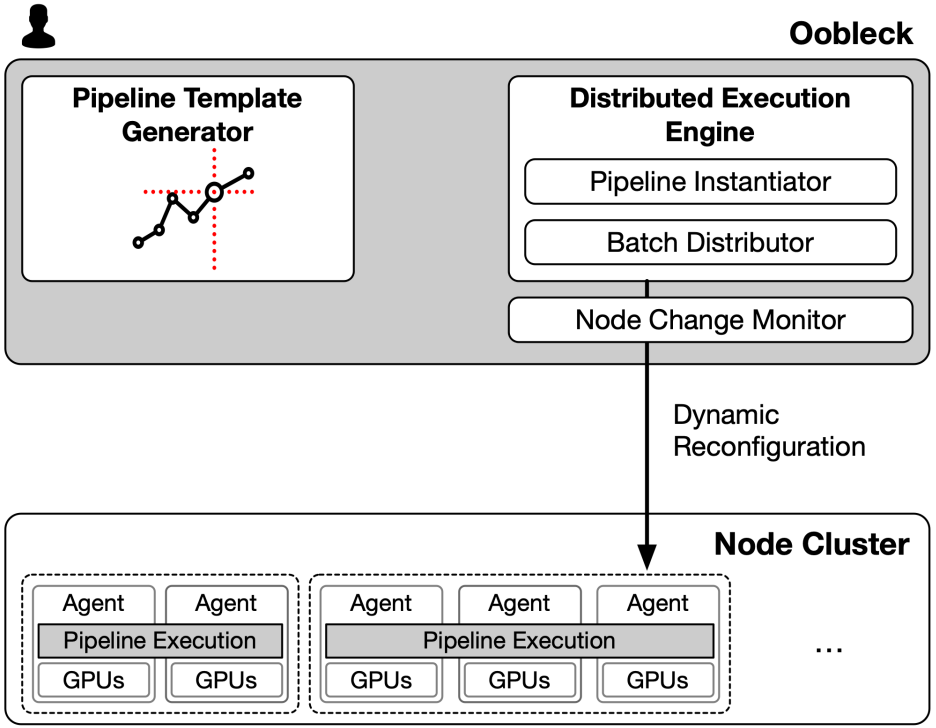


3. Pipeline Merge

- When no feasible pipeline template: **merge pipelines**
- **Provable guarantee** that Oobleck always has a template for merged pipeline



Oobleck Architecture Workflow



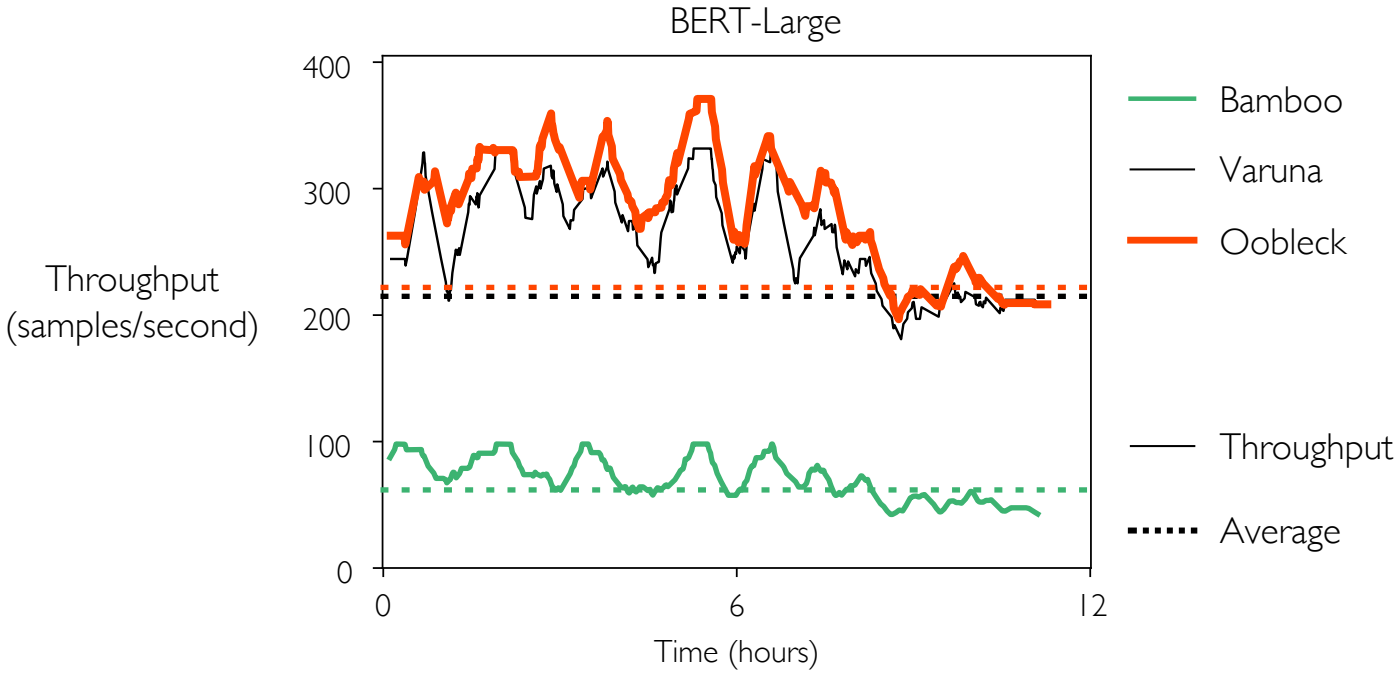
1. Generate pipeline templates
2. Instantiate pipelines from the pipeline templates
3. Pipeline reinstantiation when failures detected

Evaluation

- Setup
  - Compare Bamboo, Varuna, and Oobleck
  - 30 NVIDIA A40 GPUs with 200Gbps Infiniband
  - Various size of models from BERT-Large (345M) to GPT-3 6.7b (6.7B)

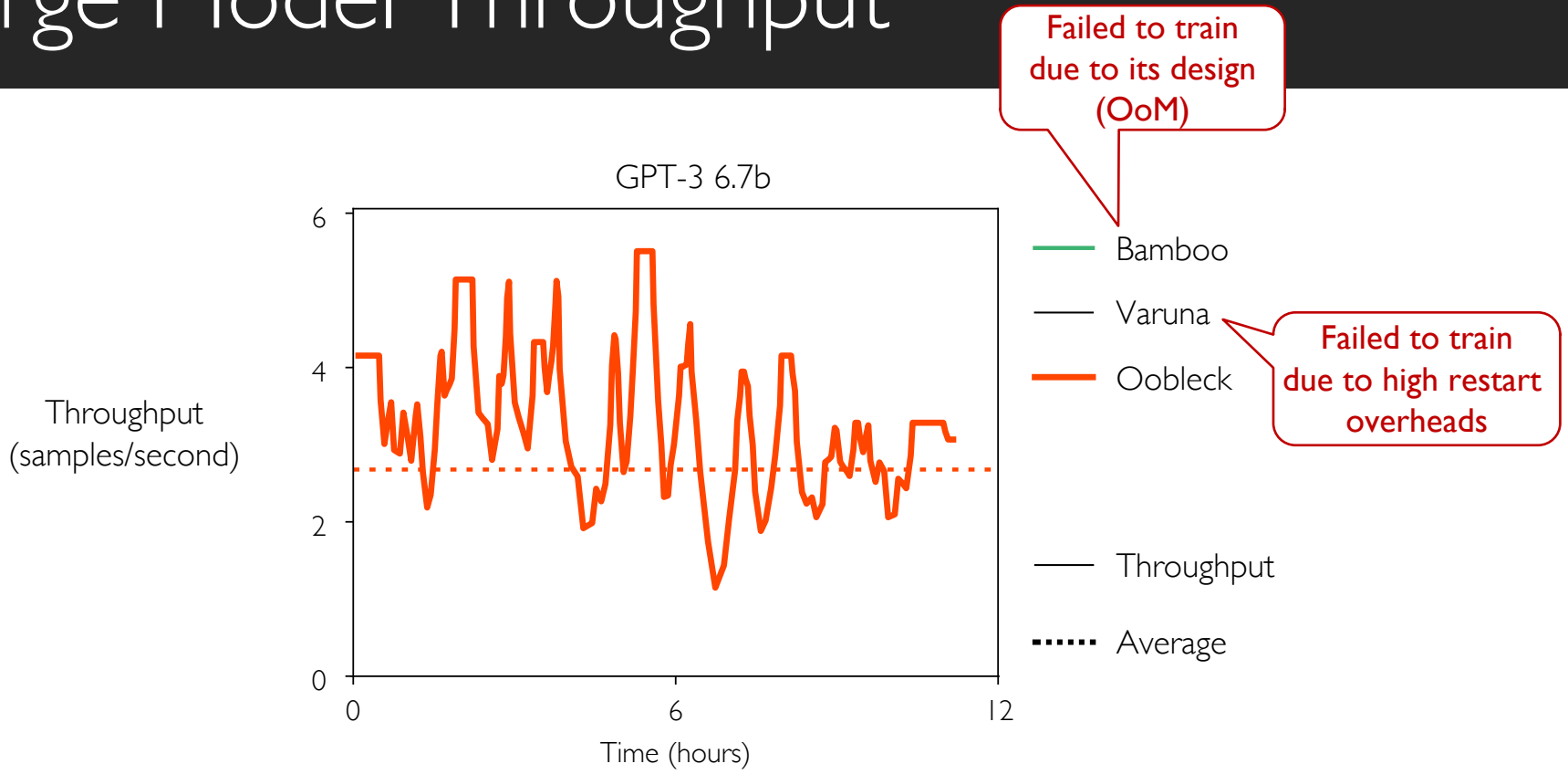
- Questions
  - How much is Oobleck better than SOTAs (Bamboo and Varuna)?
  - Why Oobleck is better?

Small Model Throughput



Numbers in Bamboo EC2 p3 spot instance traces.

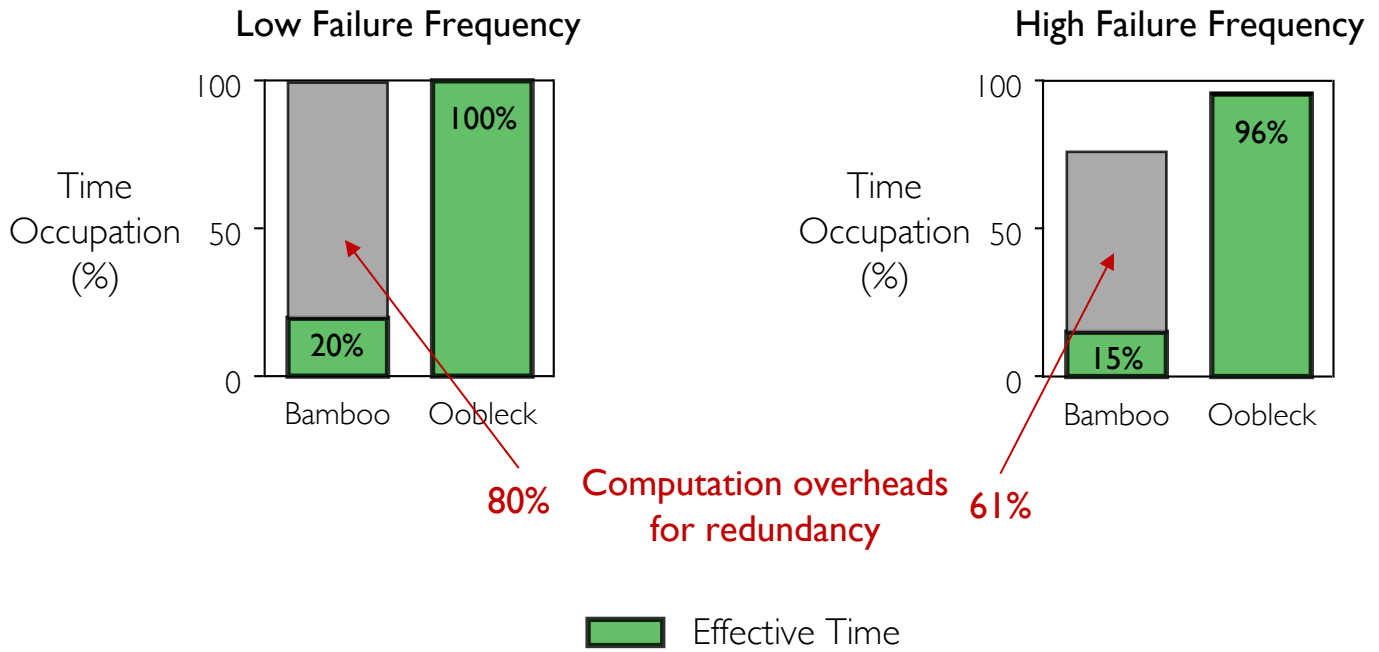
Large Model Throughput



Numbers in Bamboo EC2 p3 spot instance traces.

Throughput vs Bamboo

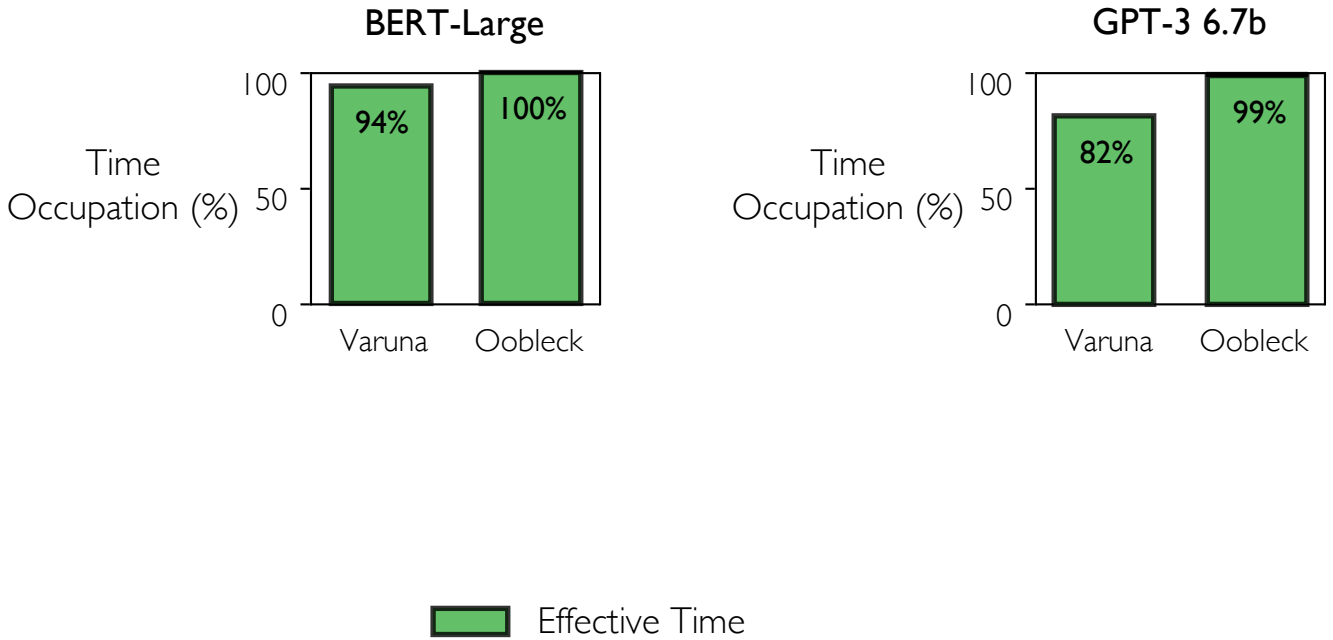
Model: BERT-Large



Numbers in failiure-controlled environment.

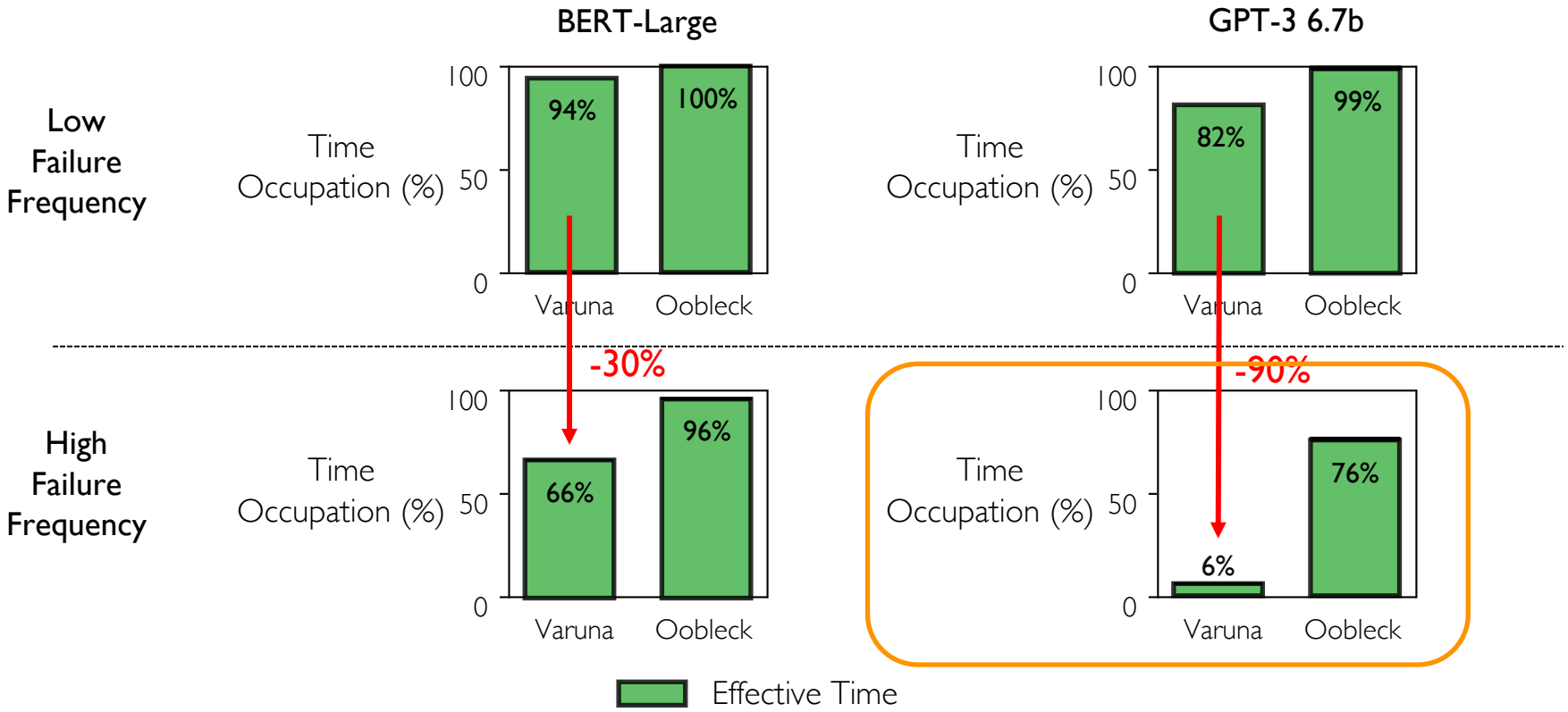
Throughput vs Varuna

Low Failure Frequency



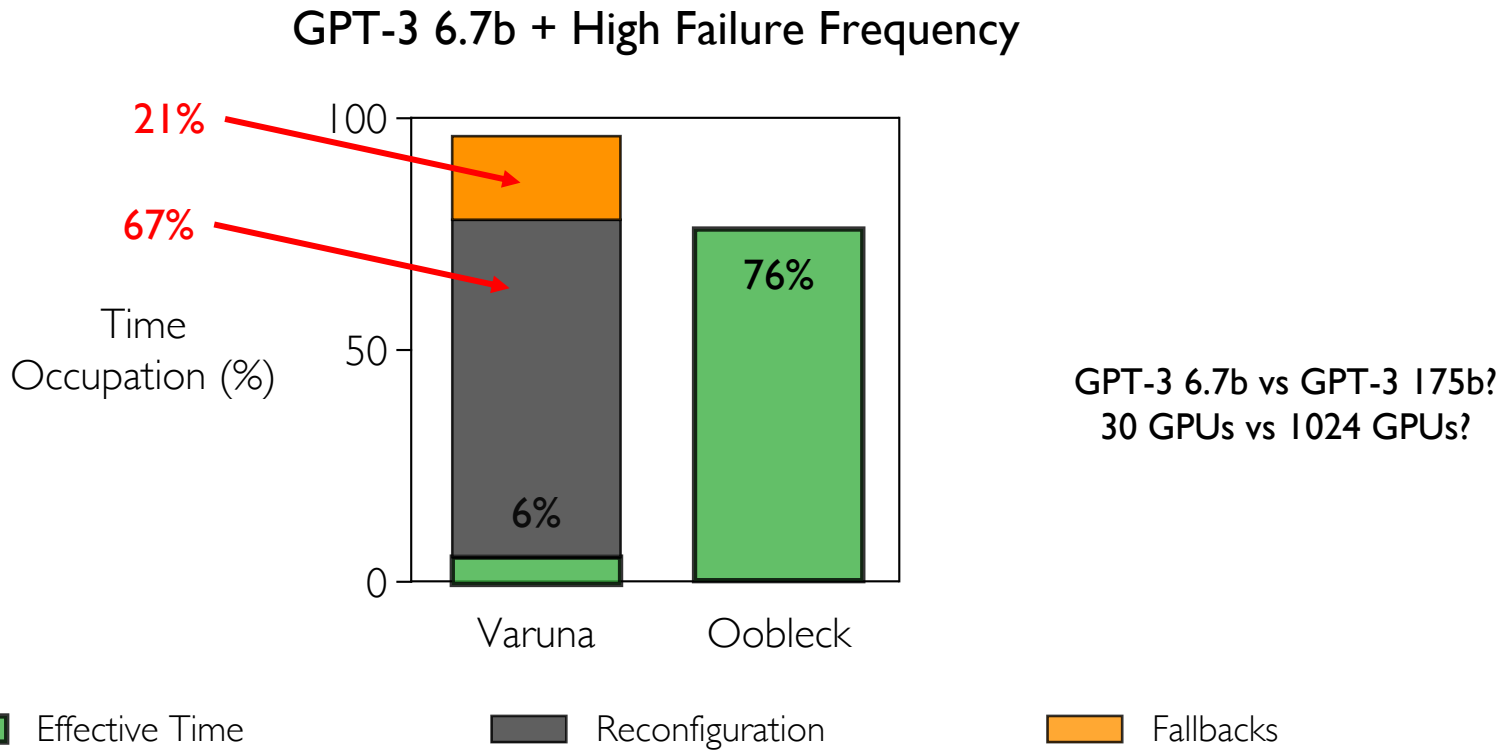
Numbers in failiure-controlled environment.

Throughput vs Varuna



Numbers in failiure-controlled environment.

Throughput vs Varuna

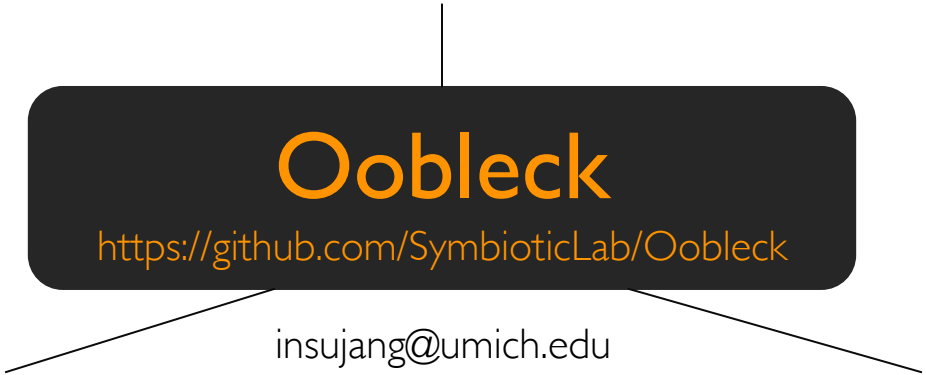


GPT-3 6.7b vs GPT-3 175b?
30 GPUs vs 1024 GPUs?

Numbers in failiure-controlled environment.

Fault tolerance guarantee

- Utilize model replicas as redundancy



High throughput

- Utilize all available resources
- Avoid stragglers in heterogeneous pipeline execution

Fast recovery from failures

- Dynamic reconfiguration without restart
- Reuse pre-generated pipeline templates