SAILOR:
fast, cost-effective ML training

Foteini Strati[1], George Manos[1], Zhendong Zhang[1], Qinghao Hu[2],
Tiancheng Chen[1], Berk Buzcu[3], Pamela Delgado[3], Ana Klimovic[1]

1 ETH Zurich
2 MIT
3 HES-SO

ongoing work

Motivation

- Large ML training workloads require a vast number of high-end GPUs

Meta engineers trained Llama 3 on computer clusters packing 24,576 NVIDIA H100 Tensor Core GPUs, linked with RoCE and NVIDIA Quantum-2 InfiniBand networks.

To further advance the state of the art in generative AI, Meta recently described plans to scale its infrastructure to 350,000 H100 GPUs.

GPT-4

- OpenAI utilized around 25,000 Nvidia A100 GPUs for training.

GPT4: Details leaked

Wide Open: NVIDIA Accelerates Inference on Meta Llama 3

Having all resources in one place is challenging

Having all resources in one place is challenging

- **From hyperscalers' perspective - Power Grid Limitations**

Microsoft Azure CTO claims distribution of AI training is needed as AI datacenters approach power grid limits

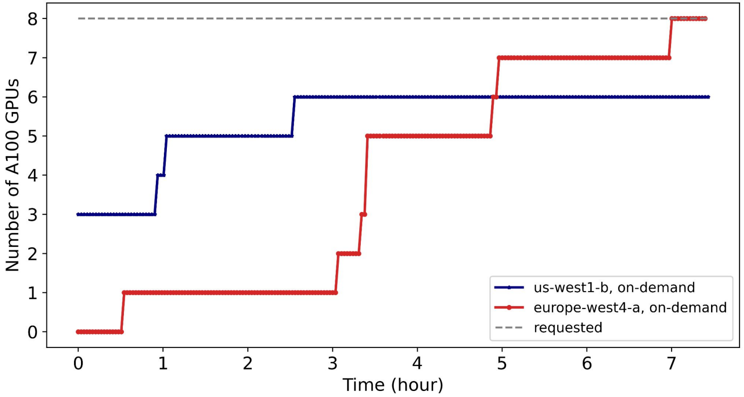
China has achieved a significant breakthrough in artificial intelligence by successfully training a generative AI model across multiple data centers and GPU architectures. This feat was revealed by Patrick Moorhead, Chief Analyst

September 4, 2024

Multi-Datacenter Training: OpenAI's Ambitious Plan To Beat Google's Infrastructure // Gigawatt Clusters, Telecom Networking, Long Haul Fiber, Hierarchical & Asynchronous SGD, Distributed Infrastructure Winners

Having all resources in one place is challenging

- **From simple users' perspective - Scarcity of (high-end) GPUs**



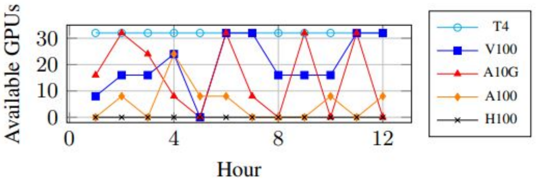


Figure 1. Hourly AWS GPU availability over 12-hour period.

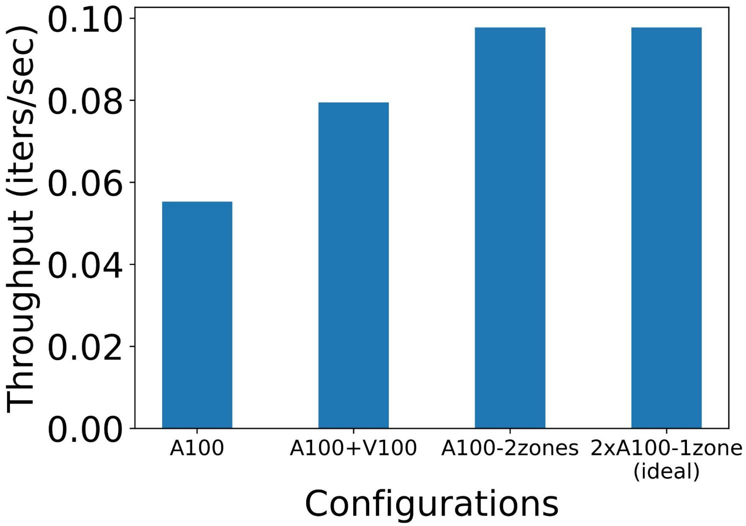
We found failover to be especially valuable for scarce resources (e.g., large CPU or GPU VMs). For example, depending on request timing, it took 3–5 and 2–7 location attempts to allocate 8 V100 and 8 T4 GPUs on AWS, respectively.

Yang et al, SkyPilot, NSDI'23
Guo et al, Cephalo: Harnessing Heterogeneous GPU Clusters for Training Transformer models
Strati et al, ML Training with Cloud GPU Shortages: Is Cross-Region the Answer?

Getting more resources

We can get more GPUs by allowing them to be:

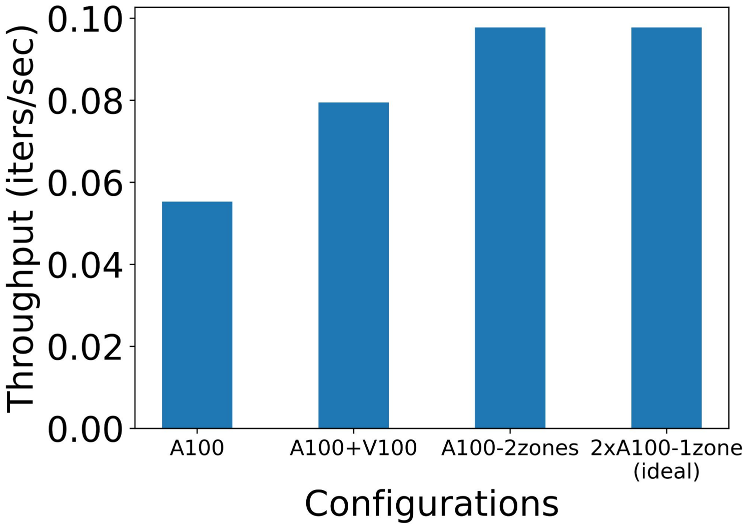
- **Heterogeneous**
- **Geo-distributed**
- **Preemptible - of varying availability**



Getting more resources

We can get more GPUs by allowing them to be:

- **Heterogeneous**
- **Geo-distributed**
- **Preemptible - of varying availability**



Challenges of heterogeneity

- **Different specs: compute, memory, networking**

GPU type	FP16 TFLOPS	Memory
H100	67	80 GB
A100	19.5	40 GB
V100	14	16 GB
T4	8.1	16 GB

Traffic Between	Cost/GB ($)	Latency (ms)	Bandwdith (GB/sec)
Same AZ (US)	Free	<1	1.45
Diff. AZ, same region (US)	0.01	0.9	1.42
Diff. regions (US)	0.02	31	0.63
Diff. continents (US/EU)	0.05	102	0.18

Challenges of heterogeneity

- **Different specs: compute, memory, networking**

GPU type	FP16 TFLOPS	Memory
H100	67	80 GB
A100	19.5	40 GB
V100	14	16 GB
T4	8.1	16 GB

Traffic Between	Cost/GB ($)	Latency (ms)	Bandwdith (GB/sec)
Same AZ (US)	Free	<1	1.45
Diff. AZ, same region (US)	0.01	0.9	1.42
Diff. regions (US)	0.02	31	0.63
Diff. continents (US/EU)	0.05	102	0.18

=> Can create **stragglers** and **OOM effects**

=> We need to accurately *model* these effects

Challenges of heterogeneity

- **Heterogeneous GPU types + cloud regions create large search space**

=> We need to find **what resources** to allocate and **where**

=> We need to decide how to split our ML workload across these resources

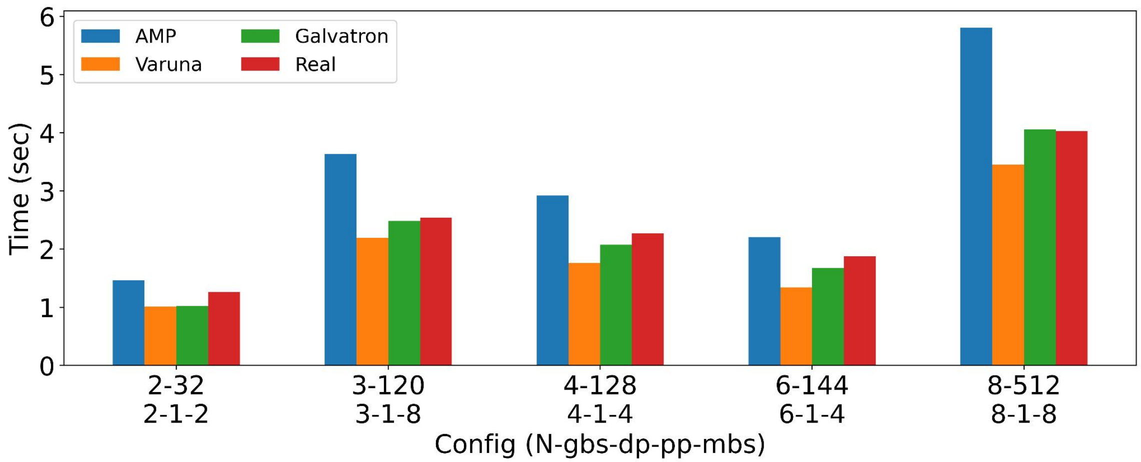
(+ extra decisions: microbatch sizes, optimizations to use, etc)

Key requirements for ML training framework

1. **Accurately model training time + memory footprint** under all possible allocation/partitioning scenarios
2. Find an optimal resource allocation + workload partitioning plan ***fast***
3. Be elastic + support heterogeneity in job configuration

Issues with related work

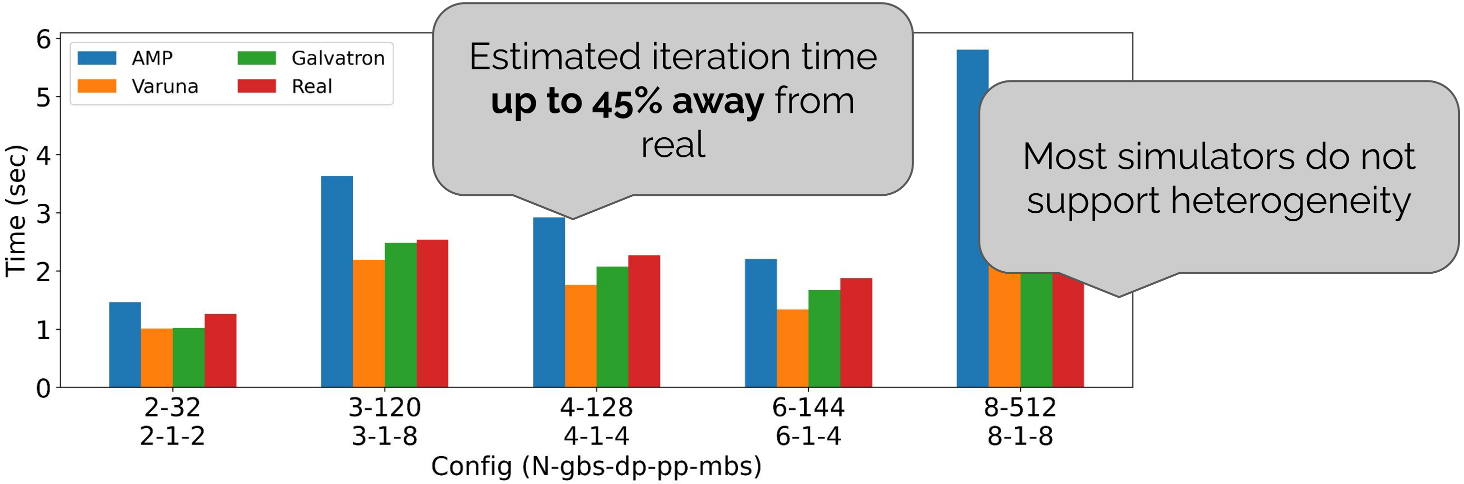
1. **Accurately model training time + memory footprint under all possible allocation/partitioning scenarios**



Training time + memory footprint modeling can be **inaccurate** (even on homogeneous environments)

Issues with related work

1. **Accurately model training time + memory footprint under all possible allocation/partitioning scenarios**



Training time + memory footprint modeling can be **inaccurate** (even on homogeneous environments)

Issues with related work

1. Accurately model training time + memory footprint under all possible allocation/partitioning scenarios
2. **Find an optimal resource allocation + workload partitioning plan *fast***

- **Most systems do not consider heterogeneity**
- Heterogeneous planners can be **very slow => cannot easily adapt to frequent resource changes**

Issues with related work

1. Accurately model training time + memory footprint under all possible allocation/partitioning scenarios
2. Find an optimal resource allocation + workload partitioning plan *fast*
3. **Be elastic + support heterogeneity**

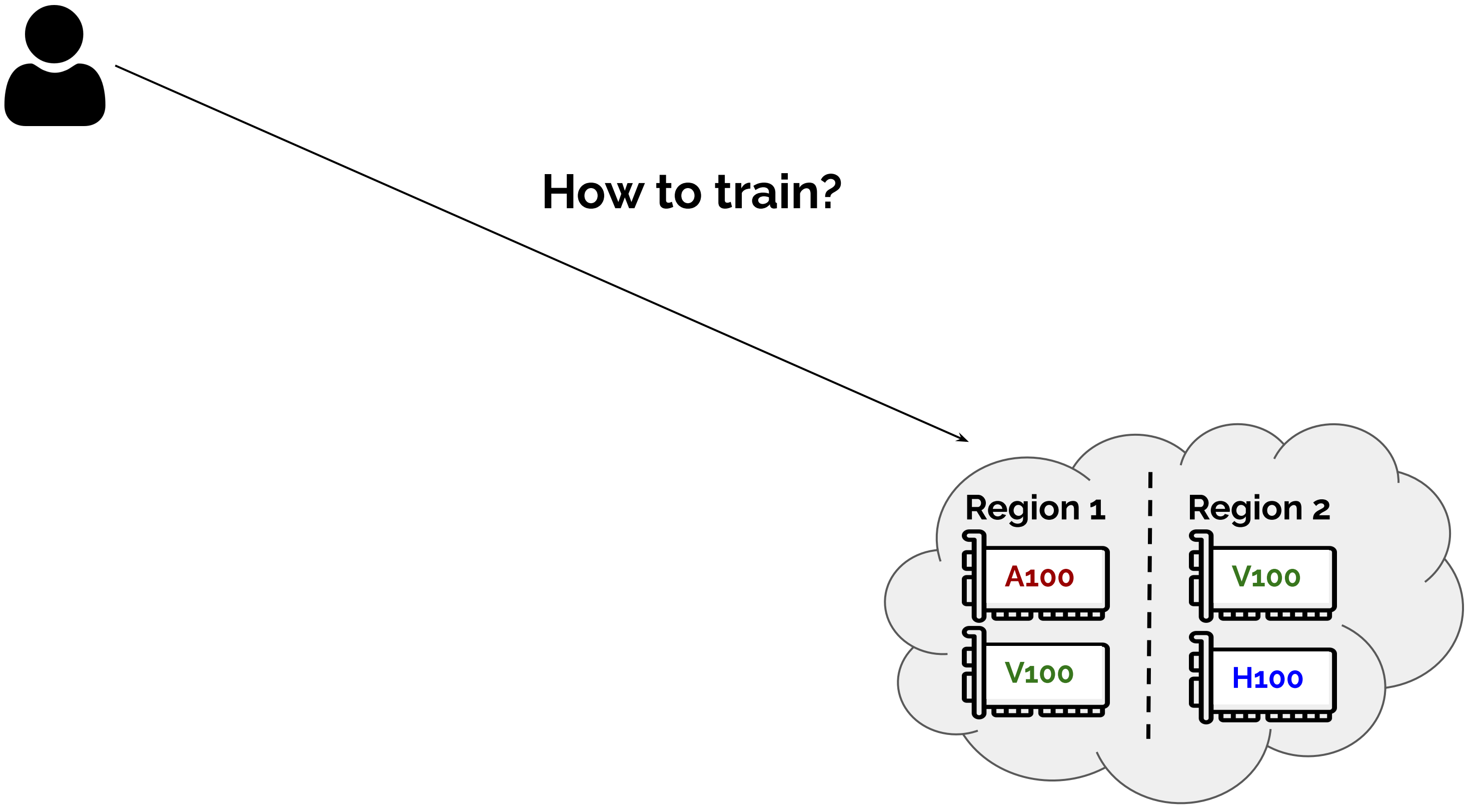
Highly optimized systems do not support heterogeneity and elasticity

System	Elasticity Support	Heterogeneity Support
DeepSpeed		
Megatron		
Varuna	✔	
Parcae	✔	
SDPipe		✔
Hetu	✔	

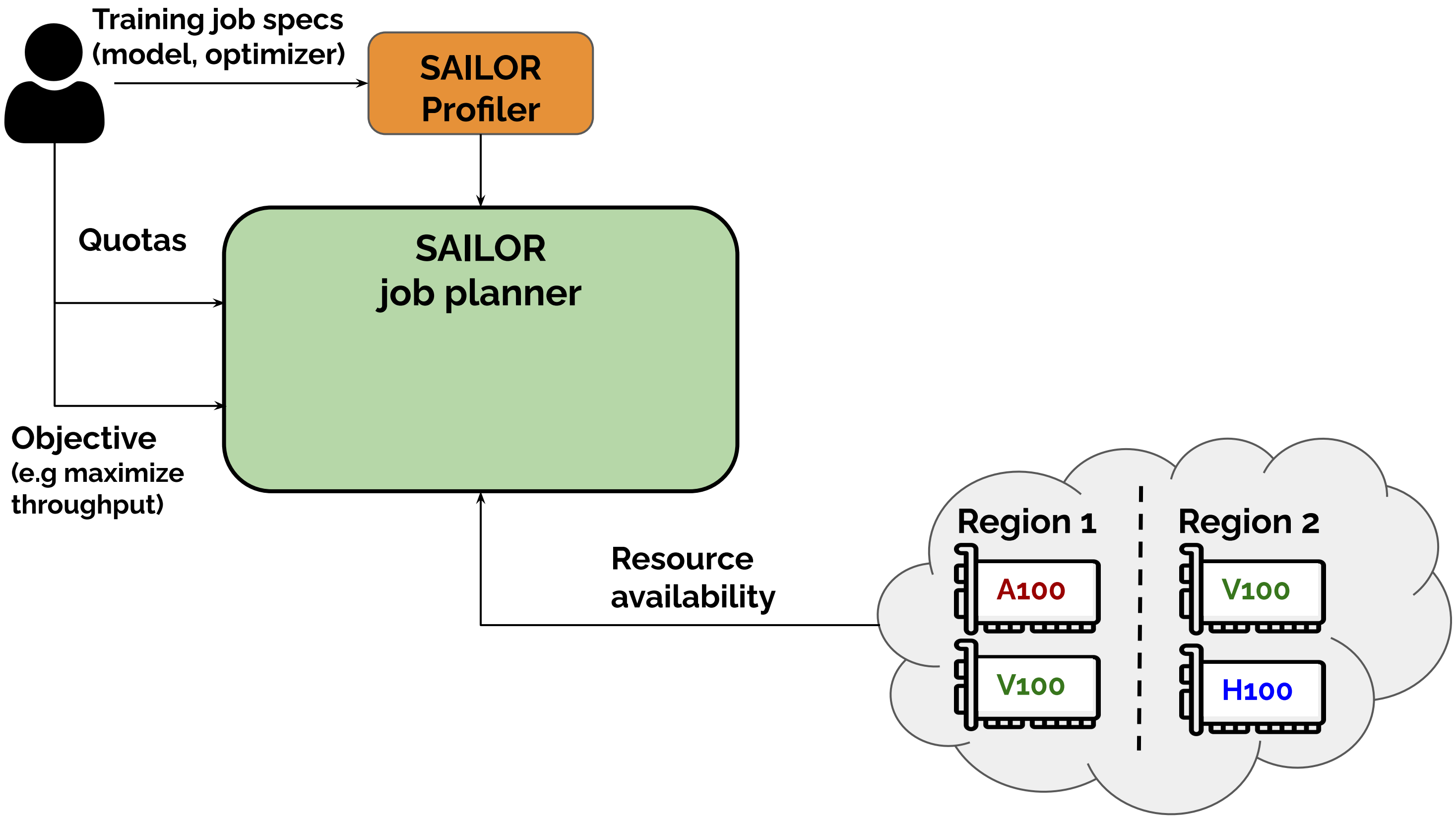
SAILOR

1. **Accurately model training time + memory footprint** under all possible allocation/partitioning scenarios
2. Find an optimal resource allocation + workload partitioning plan ***fast***
3. Be elastic + support heterogeneity

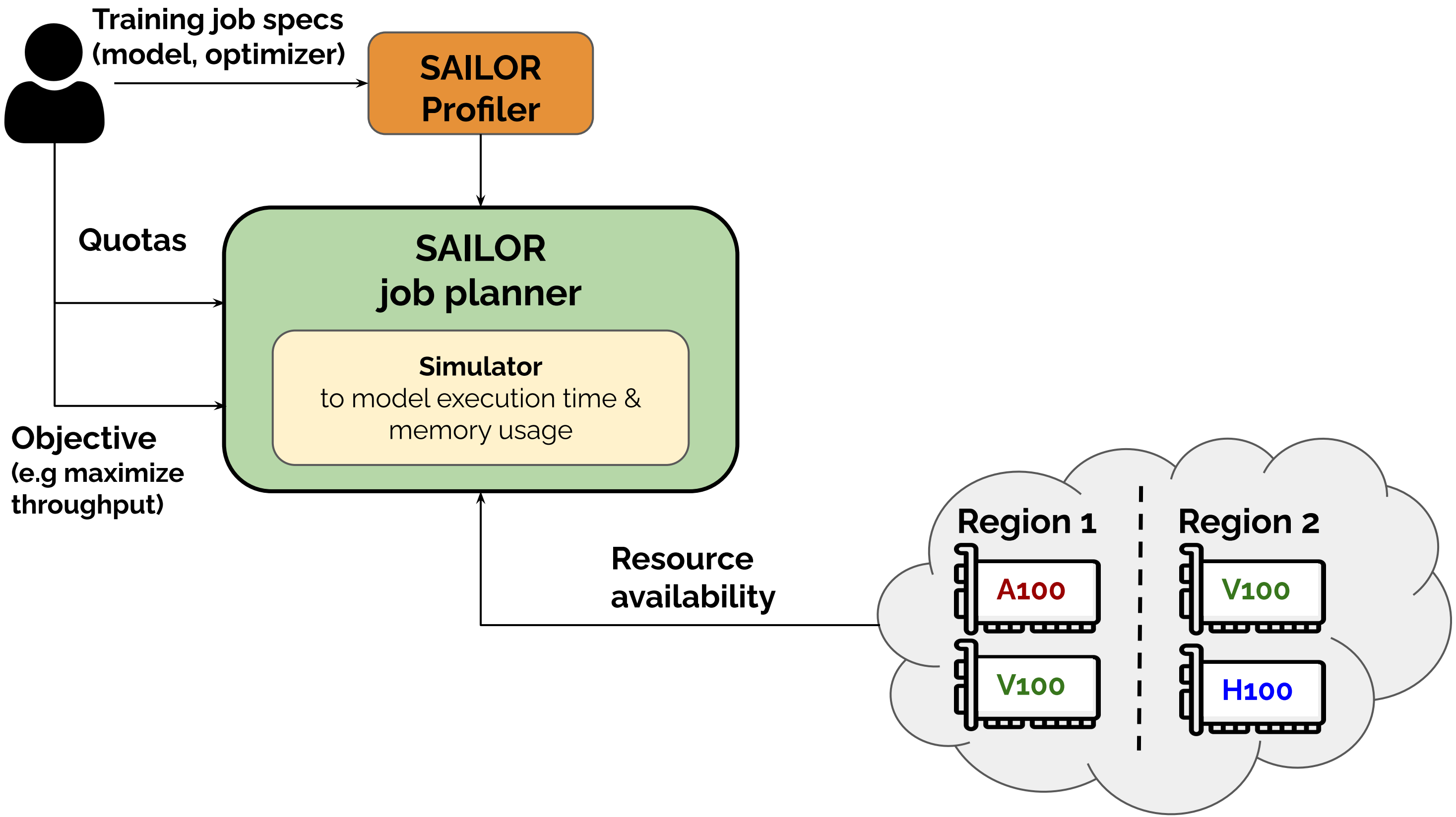
SAILOR overview



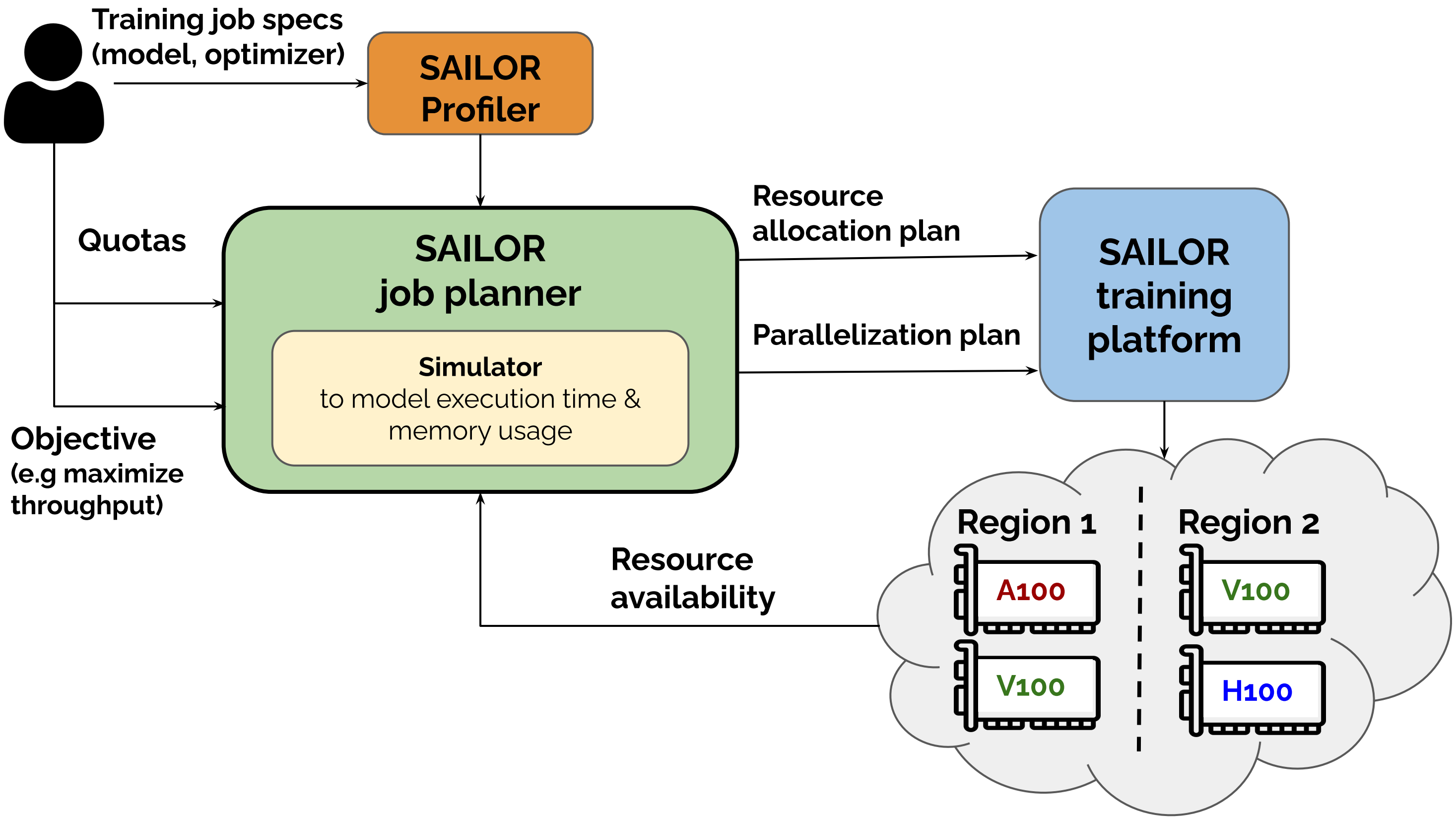
SAILOR overview



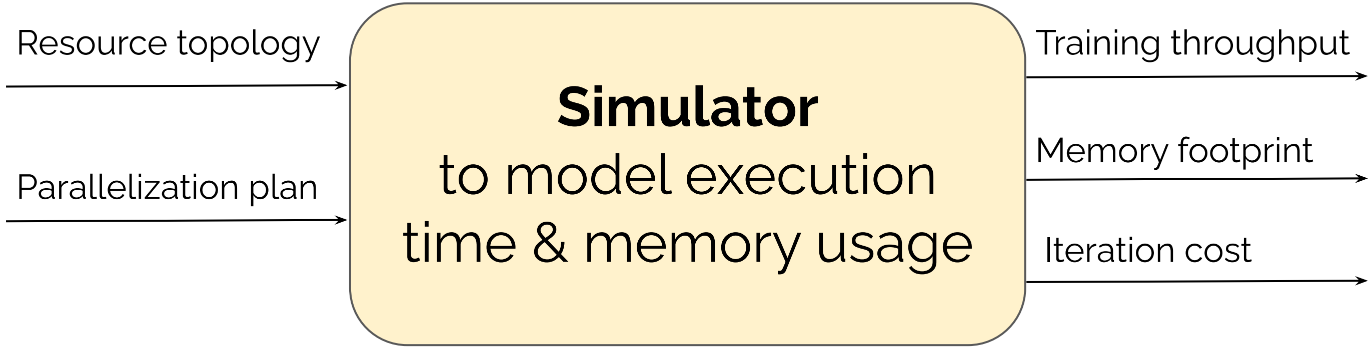
SAILOR overview



SAILOR overview



SAILOR simulator



Memory requirements estimation

Estimate memory taking into account all sources of memory consumption

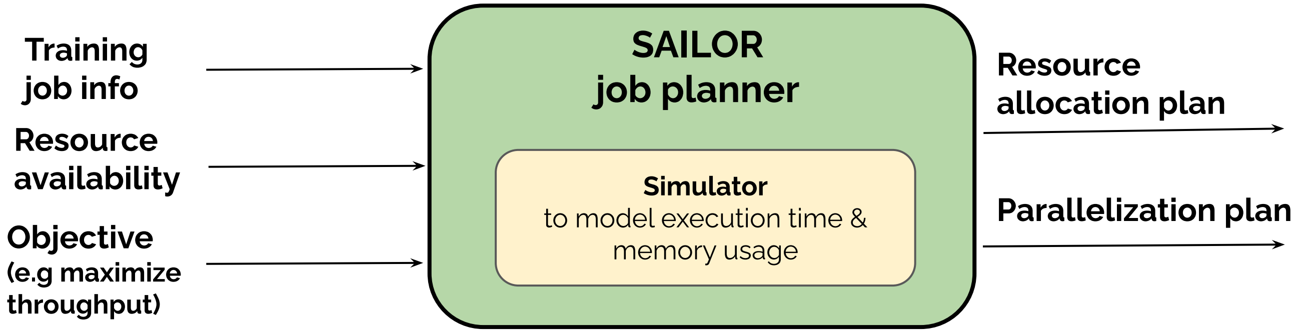
For example, assuming training with full precision and Adam optimizer:

- 1 copy of parameters for the model
- 2 copies of parameters for the optimizer
- 1 copy for communication
- Activations
- Gradients

+ **Fragmentation**

[Rajbhandari et al. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models]

SAILOR Planner



Planner requirements

- Consider different combinations of heterogeneous GPUs and zones/regions
- Prune search space efficiently

Planner key solutions

- **Consider different combinations of heterogeneous GPUs and zones/regions**
  - Dynamic-programming based approach
  - Allow different degrees of tensor parallelism per stage/per replica

Planner key solutions

- Consider different combinations of heterogeneous resources
- **Prune search space efficiently to save search time**
  - Constrain tensor parallelism within a node
  - Early-stop of cases that would lead to OOM
  - Maximum data parallelism based on scaling and all-reduce overheads
  - Constrain data parallel communication within a region
  - Topological sorting based on network bandwidth

Evaluation

Planner evaluation

2 setups:

1. **Homogeneous setup:** only A100 GPUs, one cloud zone

   => SAILOR leads to higher throughput due to better modeling

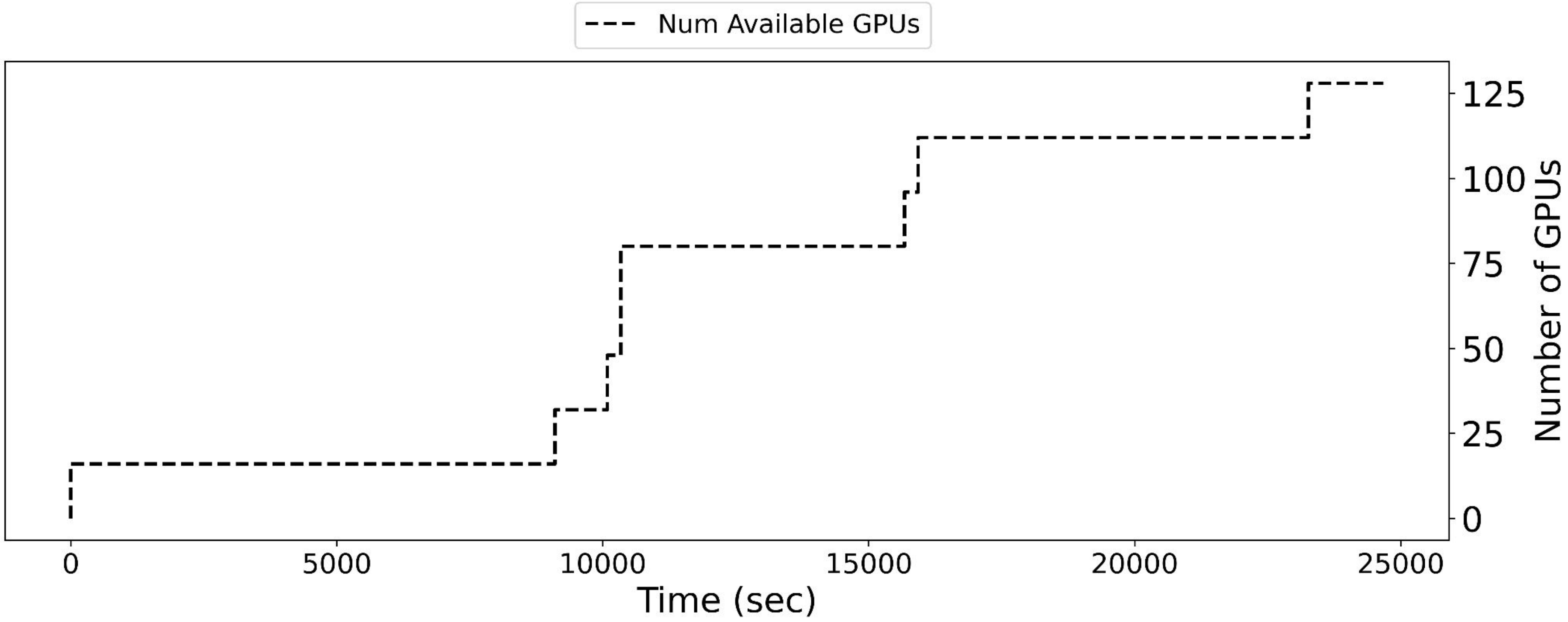
2. **Heterogeneous setup:** A100 + V100 GPUs, 4 cloud zones

   => SAILOR leads to higher throughput due to using more GPUs

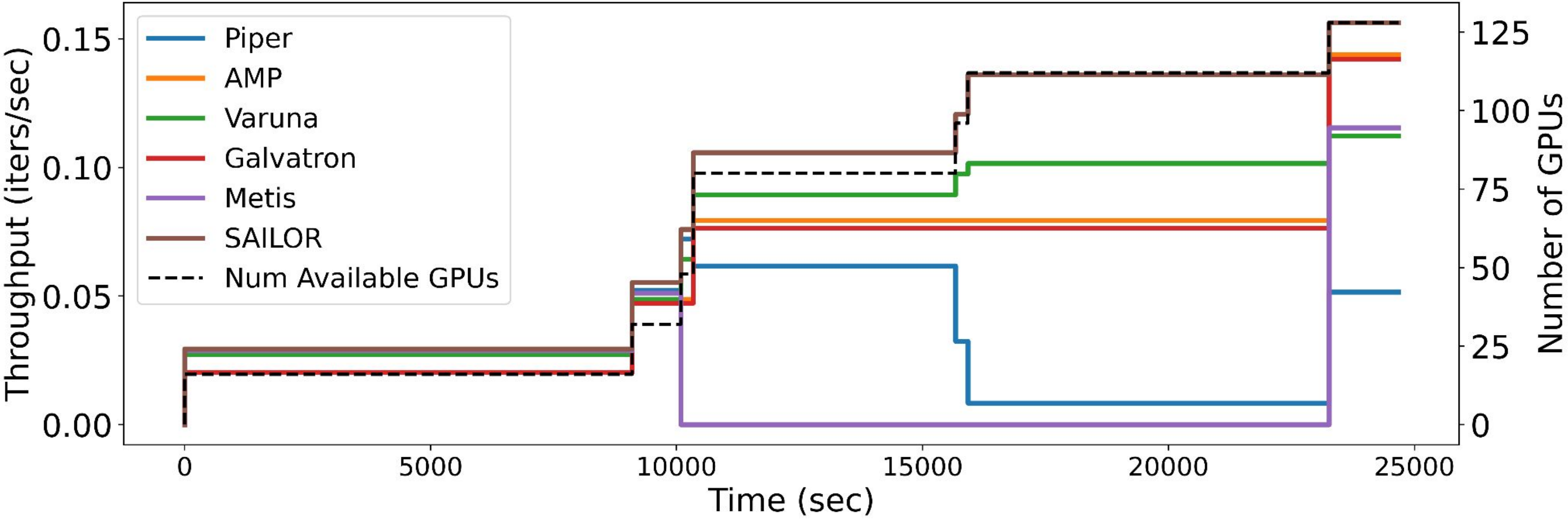
   => Short search time due to efficient planning algorithm

Homogeneous setup

Only A100, 1 zone
OPT-350M

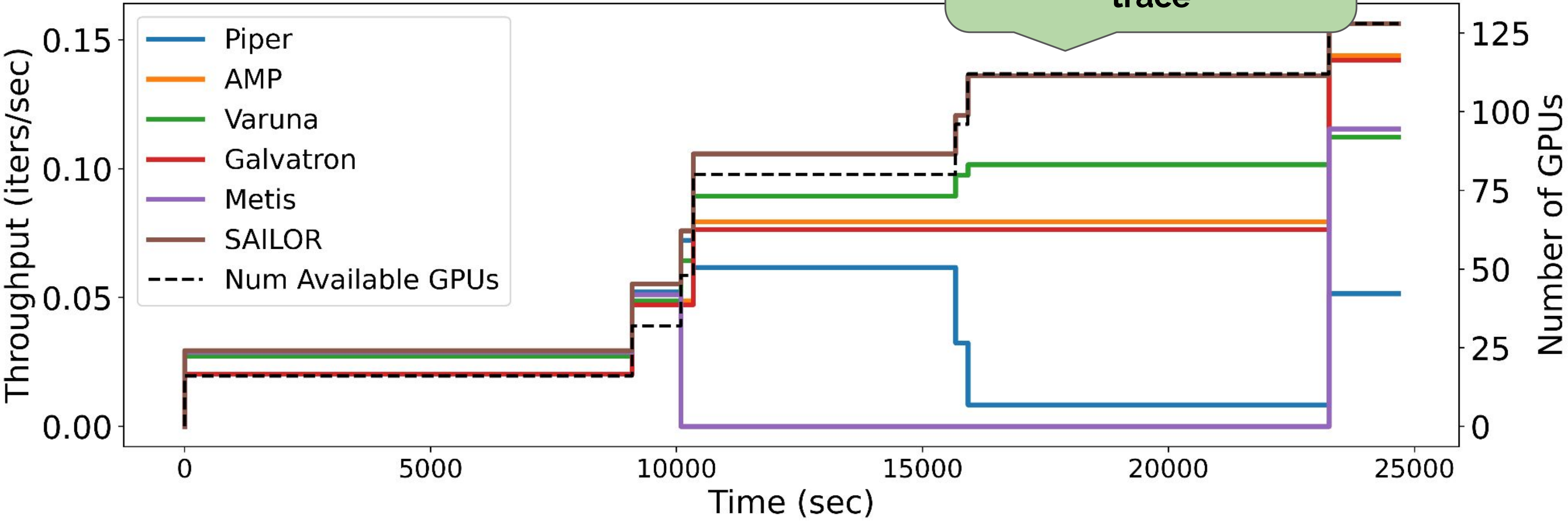


Homogeneous setup



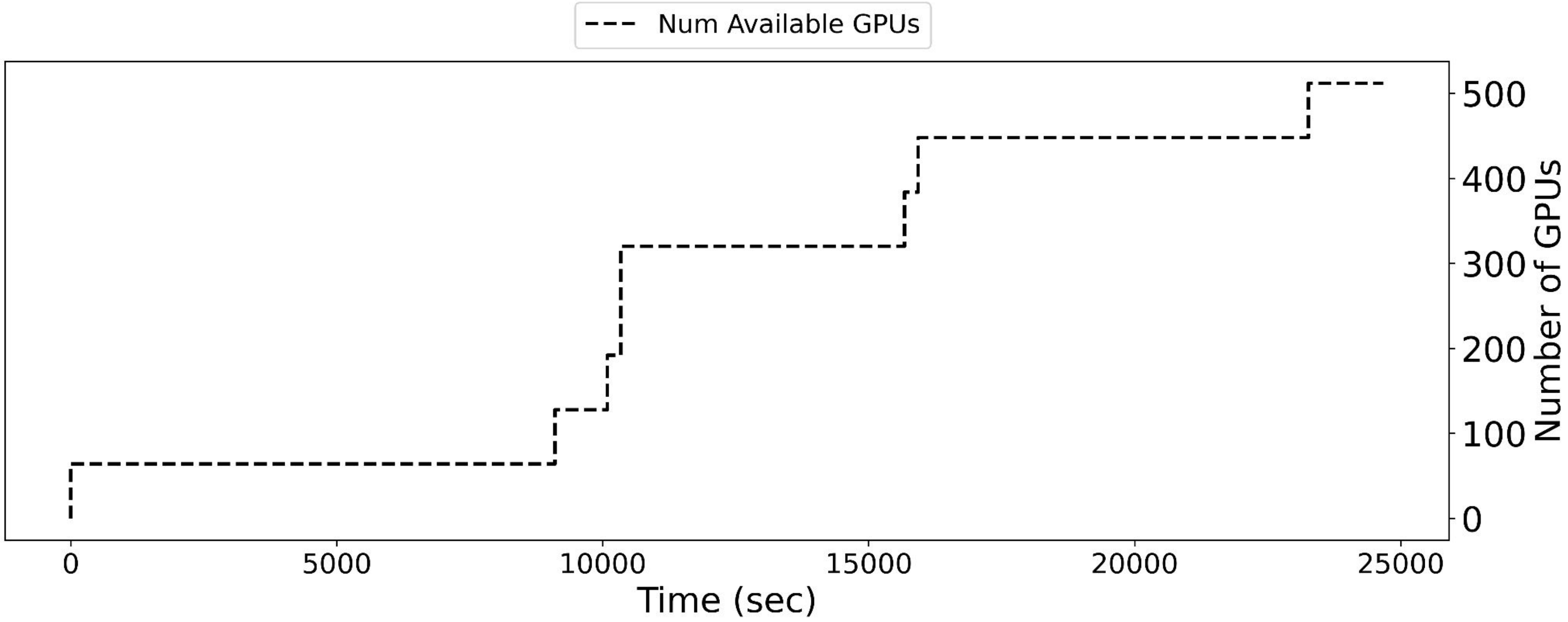
Homogeneous setup

SAILOR is consistently better in all parts of the trace

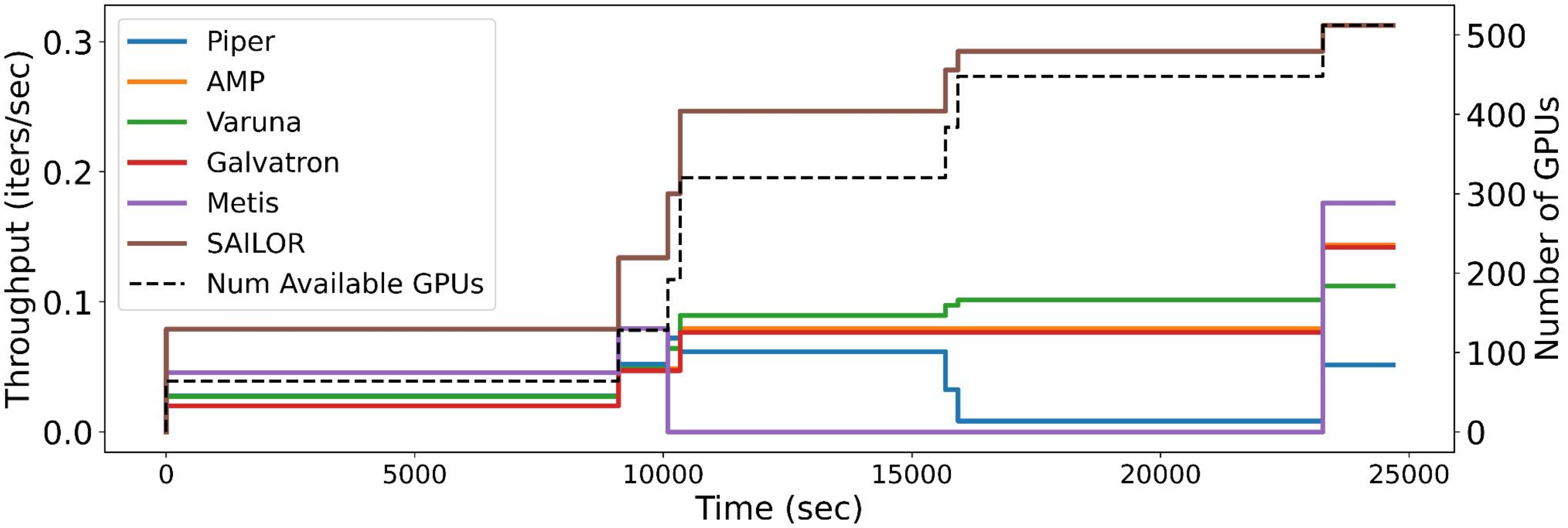


A100+V100, 4 zones
OPT-350M

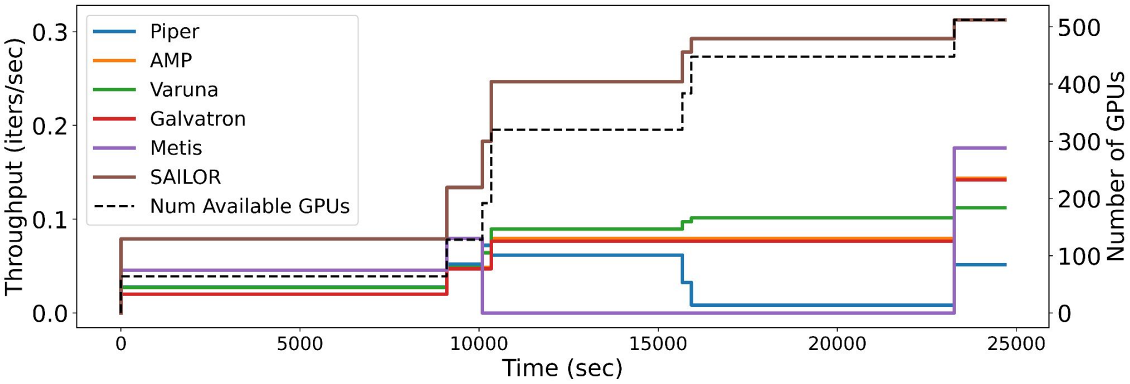
Heterogeneous setup



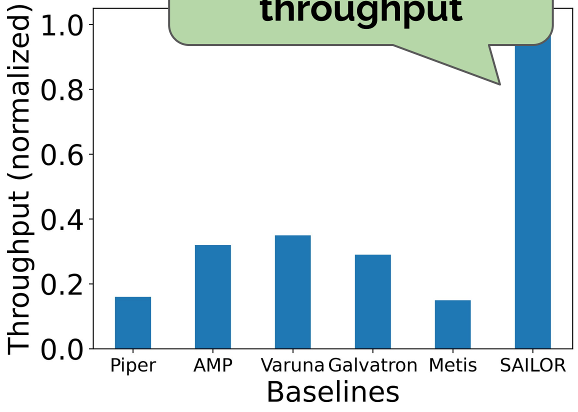
Heterogeneous setup



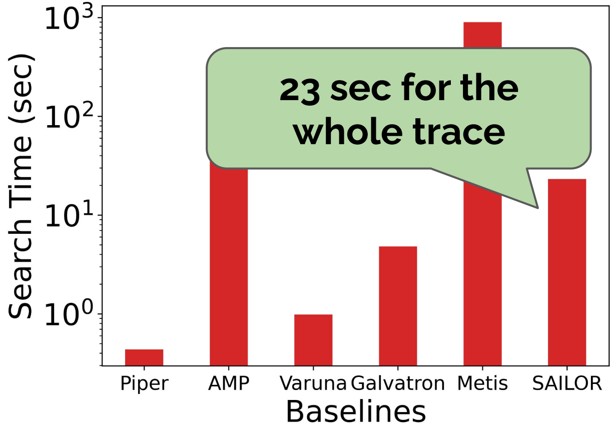
Heterogeneous setup



2.88-6.5x higher throughput



23 sec for the whole trace



Summary

- We are building **SAILOR**, a system to automate training and fine-tuning of large models on heterogeneous environments
- 3 major components:
  - A **simulator** to accurately estimate:
    - training time
    - memory footprint under all possible scenarios
  - A **planner** to find resource allocation and parallelization plans *fast*
  - An elastic **training system** with heterogeneity support