Serving with Spot GPUs in the Sky

Ziming Mao and Tyler Griggs

Introduction

Problem

Large generative AI applications are rapidly growing in popularity. However, serving these models is challenging because they have strict availability requirements and are expensive to operate. We identify one primary component of the model serving stack that exacerbates these problems: GPUs.

GPUs are notoriously ***expensive*** and ***scarce***. Expensive GPUs inflate the cost of model serving, and GPU scarcity forces organizations to horde GPUs when available in order to meet service demand, leading to resource inefficiency and even higher serving costs.

Approach

In this work, we seek to ***reduce model serving cost*** while achieving ***high service availability*** by pursuing two directions that exploit the characteristics of cloud GPU instances to make more cost-efficient use of GPUs:

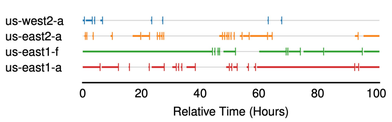
(1) Spot Instances: reduced-cost, preemptible GPU instances
(2) Mixing Accelerators: mix accelerator types in same serving pool

Background

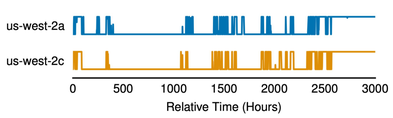
Spot Instances

	A100	V100	T4	K80
AWS	10×	4–12×	6–8×	4–8×
Azure	2×	4×	10×	10×
GCP	3×	3×	5-7×	10×

Cost Savings (Spot GPUs)

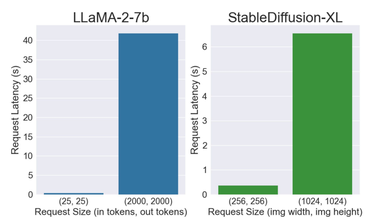


Spot instances preemptions

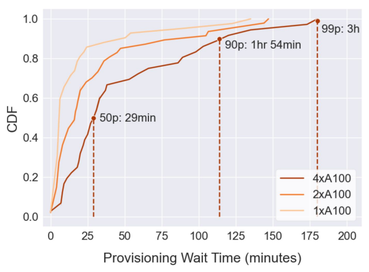


Correlated Preemptions within a Region

Mixed Accelerators



Heterogeneity in Generative Models

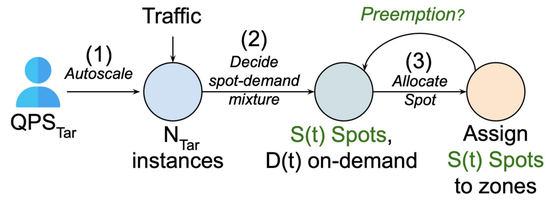


Accelerator Unavailability

Spot

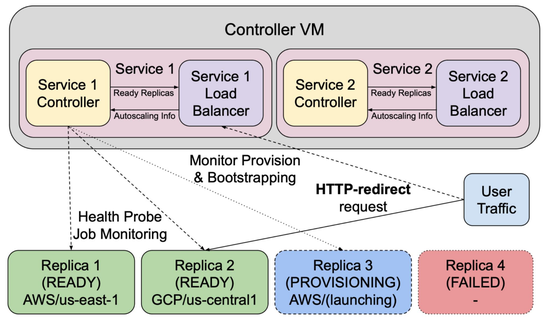
Policy Overview

(1) Overprovision cheap spot instances to mitigate preemptions
(2) De-correlate preemptions by placing spots on different regions / clouds
(3) Fallback to on-demand instances when spot becomes unavailable



System Overview

(1) Launch service replicas on different regions and clouds; Redirect user traffic to different service replicas
(2) Service controller handles replica life cycles (provisioning, health probe, job monitoring, and termination)
(3) Implement policy (autoscaler, spot mixture, spot placer, as above)



Evaluation

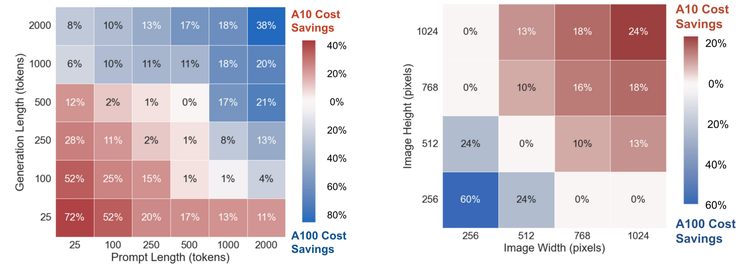
(1) Spot preemption and availability traces, covering 15 zones, 8 regions, 2 clouds, up to 2 months
(2) Three workloads: Poisson, ChatbotArena, and Azure Functions.
(3) Simulation, replay traces for reproducibility, multi-node simulation to measure latency percentiles.
(4) Real end-to-end evaluation on prototype system. Running System on actual AI serving workloads with real-time preemptions.

Summary of Results

(1) Guarantee high resource availability
(2) Saves cost (44% - 55%) and improves service latency (3x P99, 1.5x P50) compared to using on-demand instances w/o overprovisioning. More cost saving compared to on-demand w/ overprovisioning.
(3) Evaluating System on real AI workloads show that compared to using on-demand, System saves 38-43% of cost while reducing P99, P90 latencies by 55%, 74% respectively.

Mixed Accelerators

Policy #1: Request Size



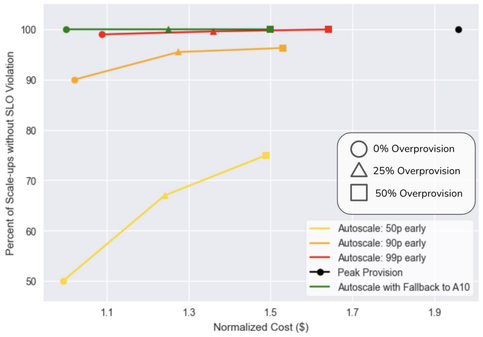
Observation: Both LLaMA2-7b (left) and StableDiffusion-XL (right) have large request size space wherein different GPUs are cost-optimal.
Policy: Serve each model request on the GPU that is most cost-efficient for the request's size.

Policy #2: Request Rate

Observation: At lower request rates, high-end GPUs are underutilized.
Policy: If a low request rate is causing GPU underutilization, switch to using a cheaper and less powerful GPU.

Policy #3: Availability

Observation: Autoscaling with a single GPU type is challenging because GPUs may be unavailable when needed to meet demand. However, the GPU shortage is mostly experienced by popular high-end GPUs.



Policy: When a preferred GPU is unavailable, fall back to plentiful, highly-available GPUs.

On left, autoscaling with a mix of GPUs (Policy #3) achieves higher availability and lower cost compared to single-GPU autoscaling and provisioning for peak.

System

We combined the policies into one cohesive auto-scaler and load balancer that provisions a cost-optimal mix of accelerators given the observed (or predicted) workload.

Evaluations

Simulation Setup: Replay production traces (ChatbotArena, Azure Functions) using real LLM datasets (ChatbotArena, ShareGPT)
Preliminary Results:

- Compared to single-GPU autoscaling: 26-36% cost reduction while achieving higher availability
- Compared to provisioning for peak: same high availability at 45-70% reduced cost