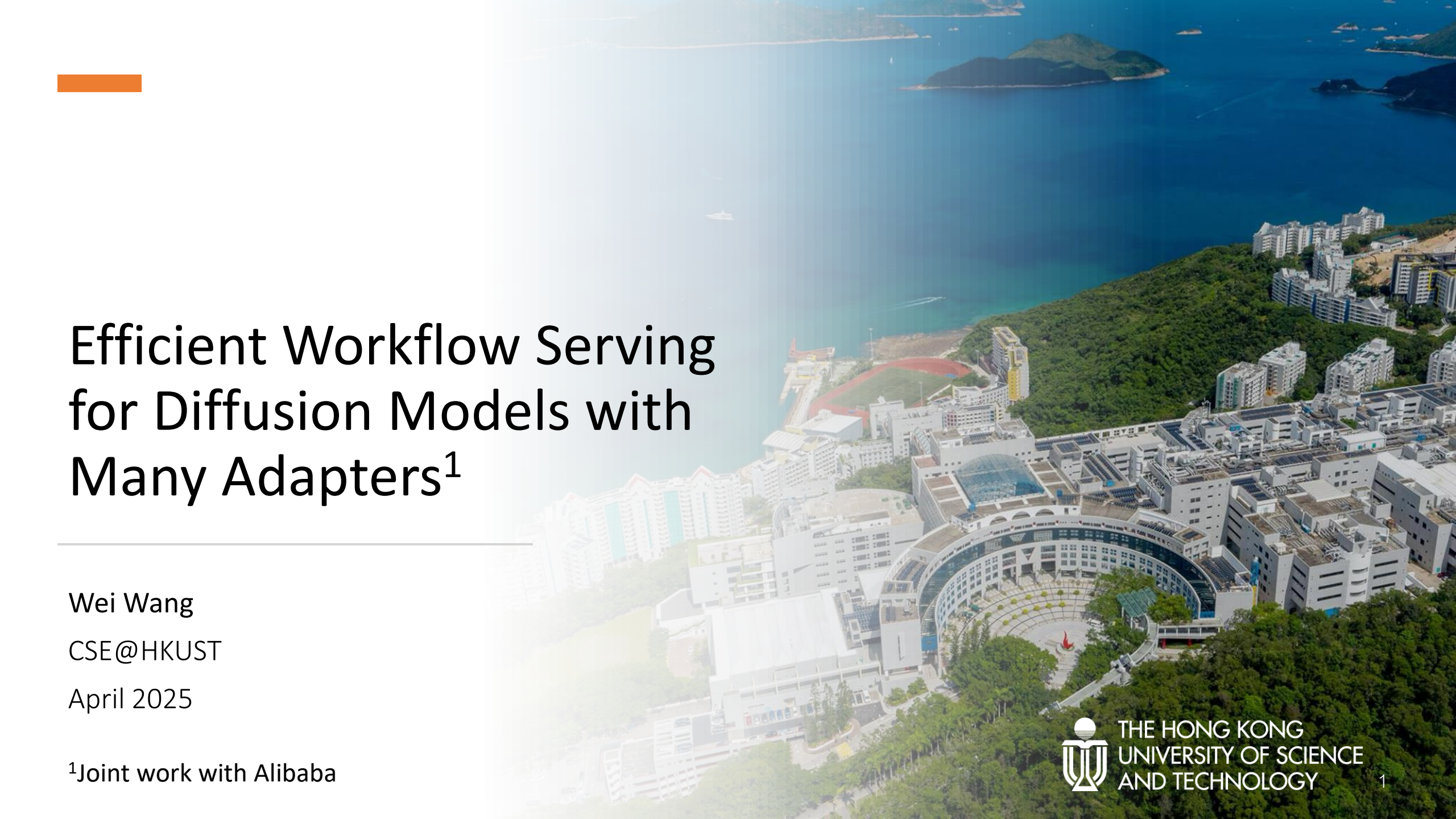
Efficient Workflow Serving for Diffusion Models with Many Adapters[1]

Wei Wang

CSE@HKUST

April 2025

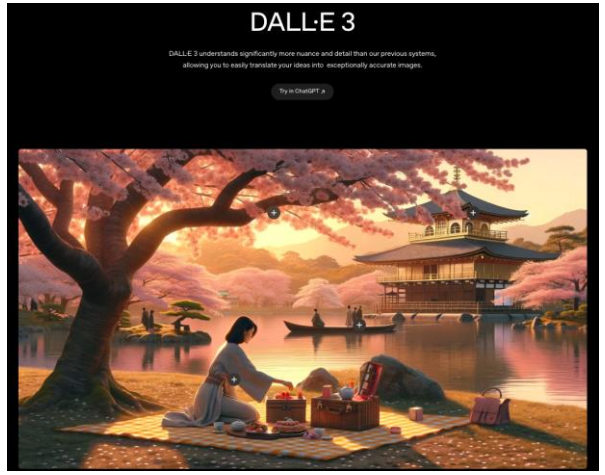
[1]Joint work with Alibaba



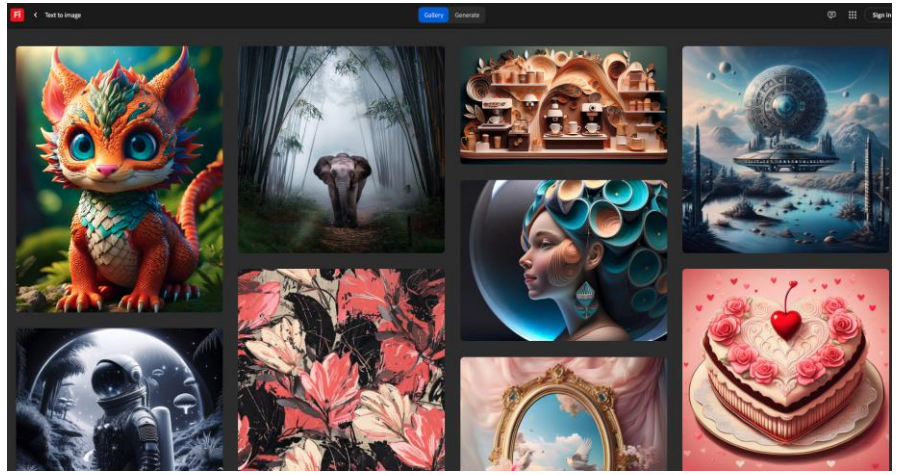
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Text-to-Image: A Blockbuster GenAI Service

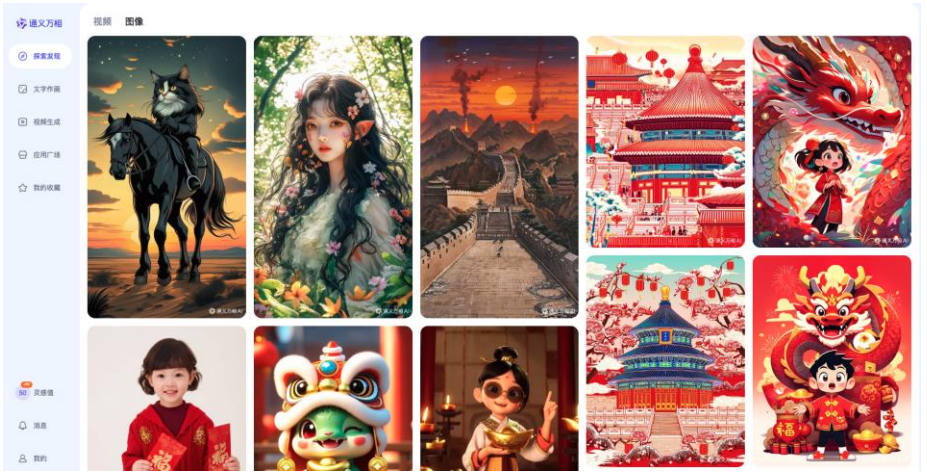
- Generate high-quality, contextually accurate images from textual prompts
  - Killer apps: e-commerce, advertisement, entertainment, and creative workflows…
- A blockbuster GenAI service in the cloud
  - OpenAI, Midjouney, Google, Adobe, Alibaba, Tencent, ByteDance…
  - Adobe Firefly has generated two billion images by 2023



OpenAI DALL·E 3



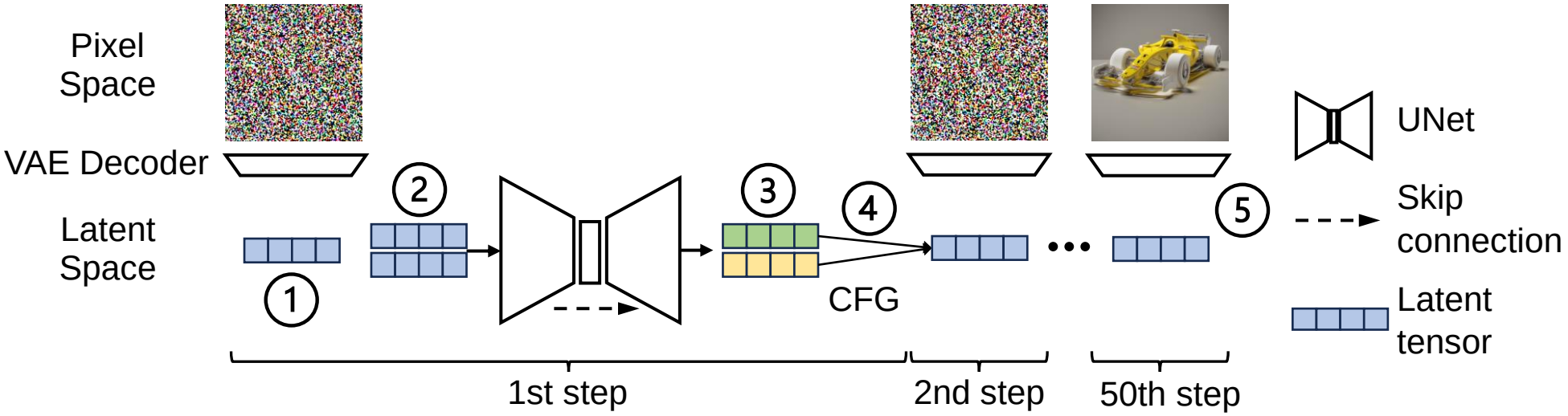
Adobe FireFly



Alibaba Tongyi Wanxiang (通义万相)

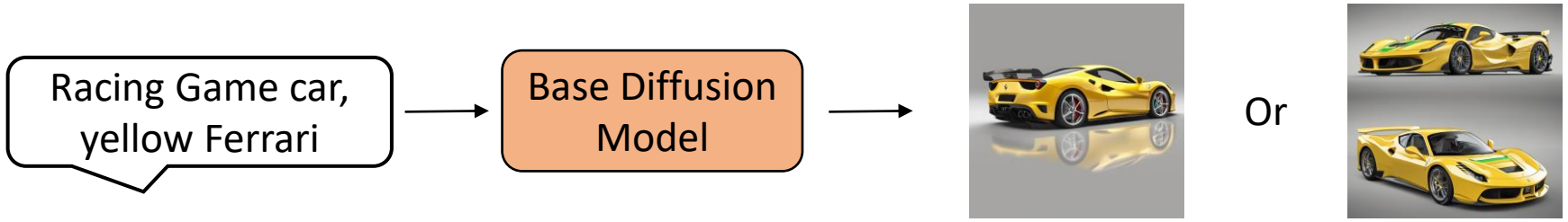
Diffusion-based Text-to-Image Generation

- The magic of generating an image from a noise latent with ***diffusion model***



Diffusion-based Text-to-Image Generation

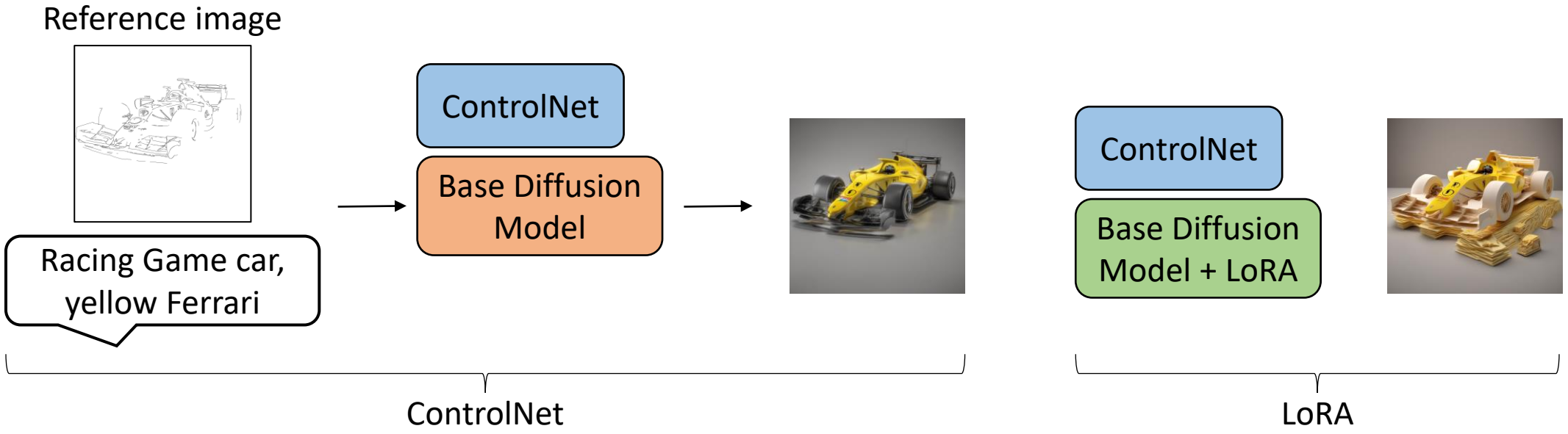
- The magic of generating an image from a noise latent with ***diffusion model***
- Text-to-image service is more than a ***base diffusion model***
  - Textual prompts alone are hard to precisely specify layouts, styles…



Hard to control image compositions

Text-to-Image Serving w/ Many Adapters

- Text-to-image service is more than a ***base diffusion model***
- Augment the base model with many ***adapters***
  - ***ControlNet:*** allow a referenece image to control compositions
  - ***LoRA:*** control the stylistic effects



ControlNets and LoRAs are prevalent in today’s T2I services

Characterization in a production platform

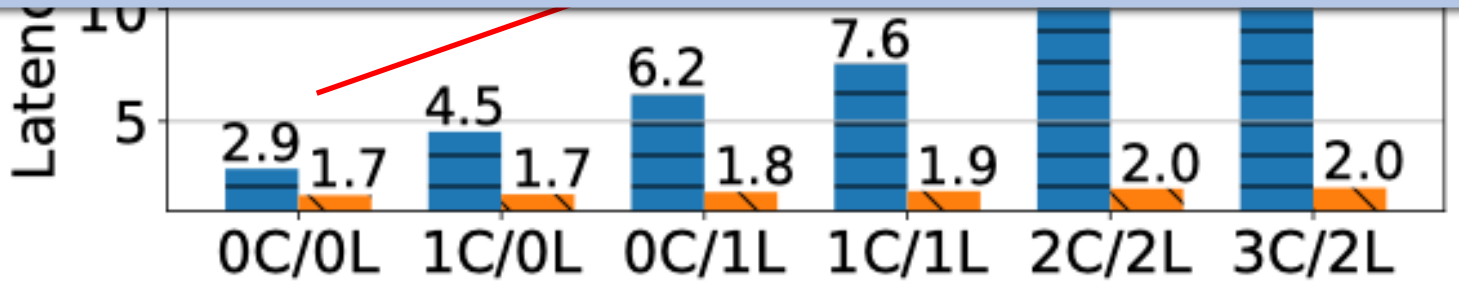
- **Prevalence of adapters**
  - A 20-day production trace collected in May and June 2024
  - 500k requests to two T2I core services in a production platform
  - A total number of 141 ControlNets and 14,371 LoRAs

Adapters	Number	Service A	Service B
ControlNet	0	0	1.9%
	1	30.5%	25.1%
	2	69.5%	69.9%
	3	0	3.1%
LoRA	0	0.2%	7.2%
	1	8.8%	73.6%
	2	91%	19.2%

Performance issues

- The use of adapters (e.g., ControlNets and LoRAs) introduces significant delays
  - Delays accumulate as more adapters are in use

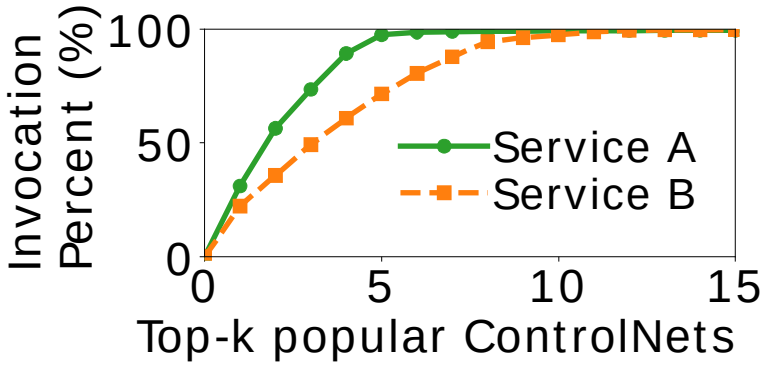
Where does the latency come from?

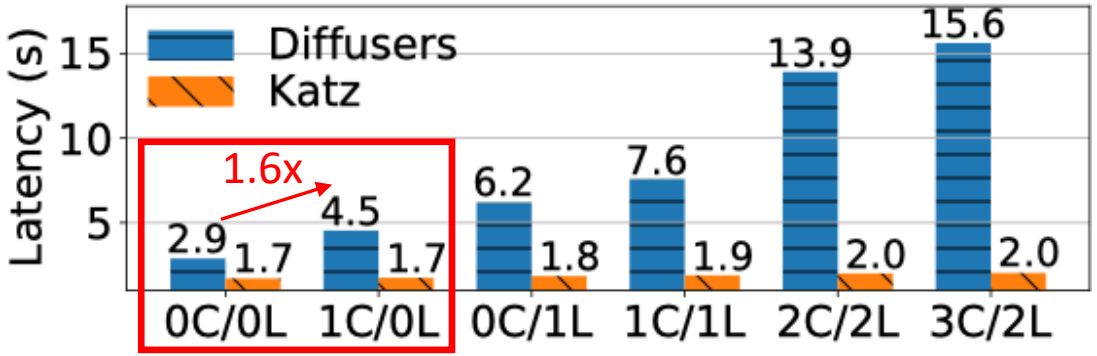


A base SDXL model augmented with ***m*** ControlNets and ***n*** LoRA (***m***C/***n***L) on H800 GPUs

ControlNet Characterization

- **Skewed popularity**
  - A few popular adapters contribute most invocations
  - Caching is effective
- **High loading overhead**
  - ControlNets are large in size, ~3 GiB each
  - Usually maintained in remote storage
- **Compute-heavy**
  - Each ControlNet adds extra 1.6s latency
  - Accumulates as more ControlNets are in use

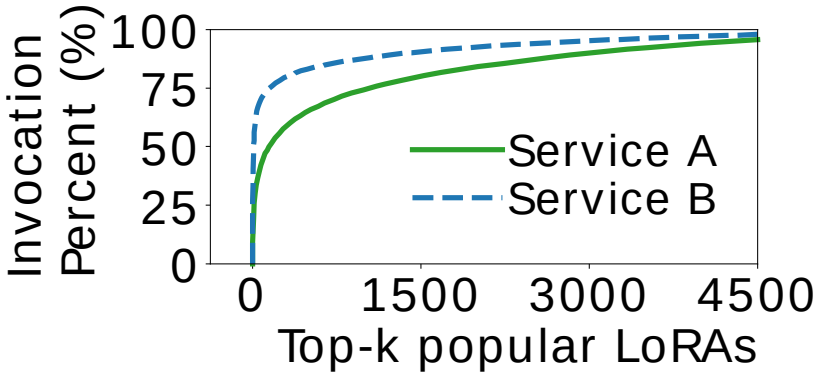


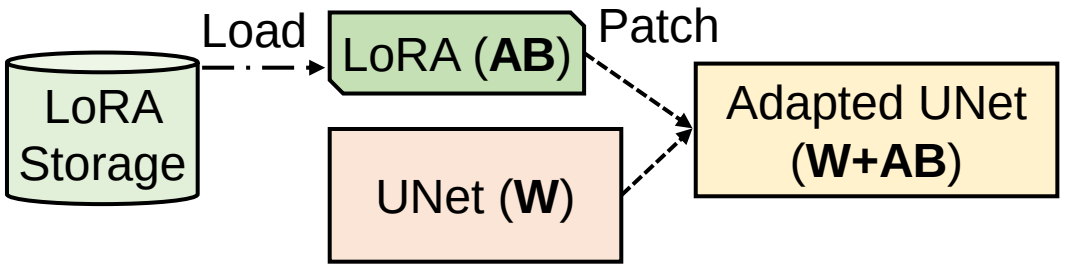


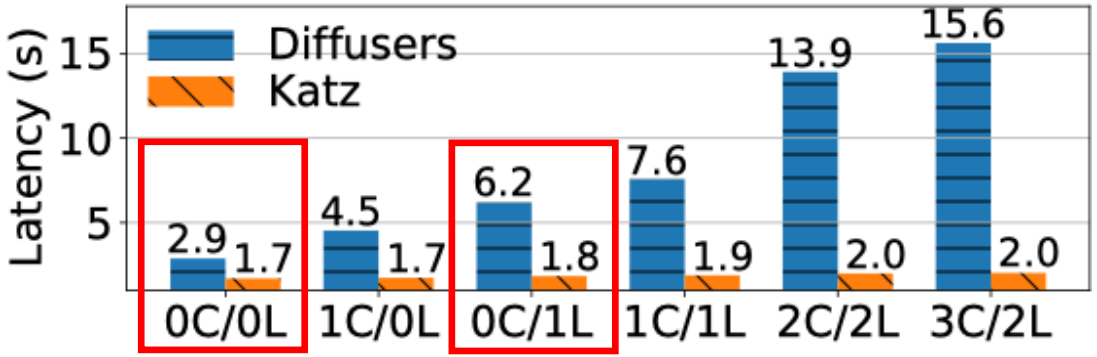
A base SDXL model with ***m*** ControlNets and ***n*** LoRA (***m***C/***n***L) on H800 GPUs

LoRA Characterization

- **Long-tailed popularity**
  - Many invocations contributed by less popular LoRAs
  - Ineffective caching
- **Compute-light**
- **High loading and patching overhead**
  - Loading + patching one LoRA takes >3s







A base SDXL model with ***m*** ControlNets and ***n*** LoRA (***m***C/***n***L) on H800 GPUs

How to efficiently serve a T2I workflow with many adapters ?

This Talk

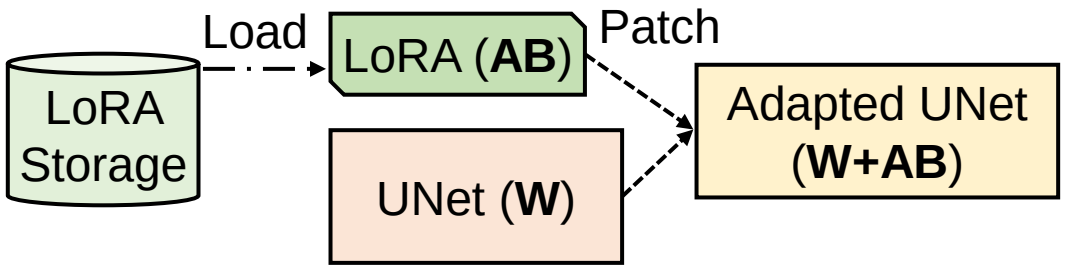
- Challenges
- Katz design
  - ControlNet-as-a-Service: Efficient ControlNet serving
  - Bounded Asynchronous Loading: Efficient LoRA loading
  - Optimized Base Model Execution
- Evaluation
- Conclusion

This Talk

- Challenges
- Katz design
  - ControlNet-as-a-Service: Efficient ControlNet serving
  - Bounded Asynchronous Loading: Efficient LoRA loading
  - Optimized Base Model Execution
- Evaluation
- Conclusion

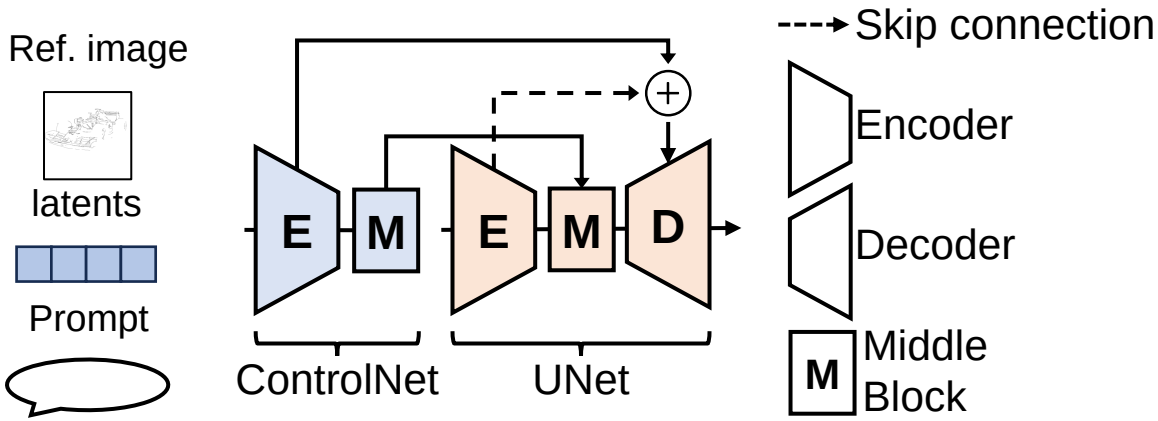
Challenge #1

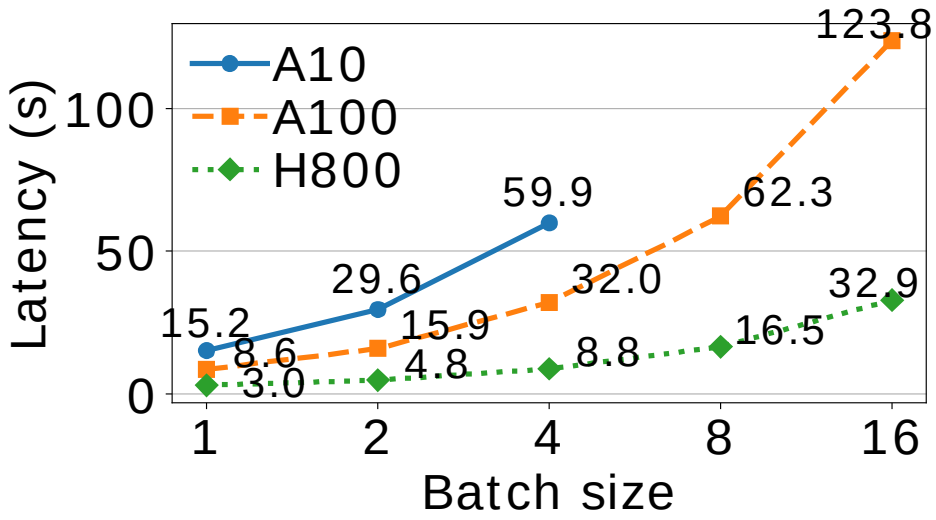
- Adapter loading (and patching)
  - Desired ***ControlNets*** and ***LoRAs*** vary across requests
    - On average, each request undergoes 1 ControlNet and 1 LoRA loading.
    - Accounts for **37%** of end-to-end serving latency
  - Naïve pre-caching all adapters is ***infeasible***
    - 141 ControlNets (~3GiB each) and 14,371 LoRAs (hundreds of MiB each) for SDXL



Challenge #2

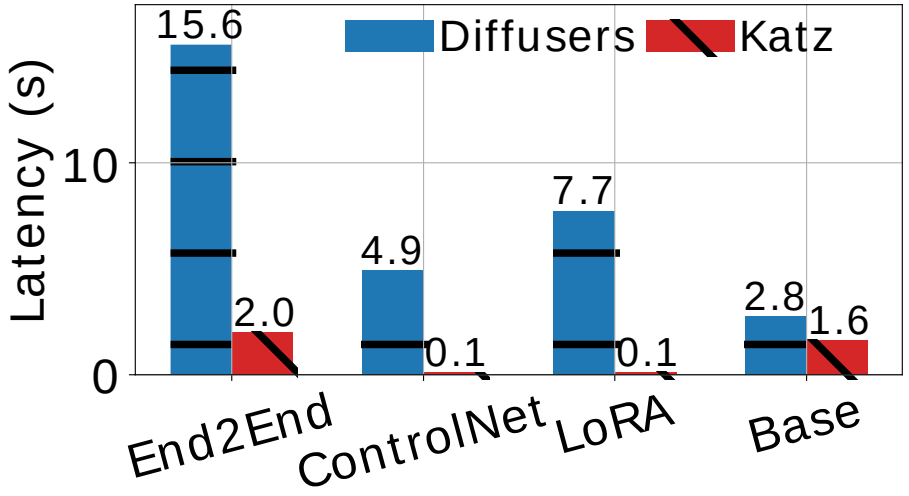
- Computation
  - ***ControlNet*** is compute-intensive
    - Using one ControlNet increase serving latency by **1.6x**
  - ***Base model serving*** is compute-heavy
    - Limited to none performance gains offered by batching

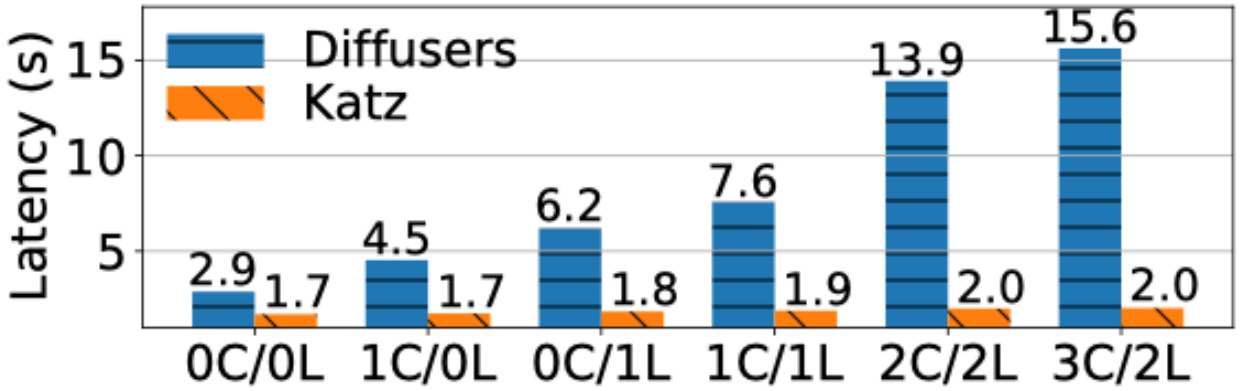




Challenges

- Adapter loading & patching
- ControlNet and base model are compute-heavy





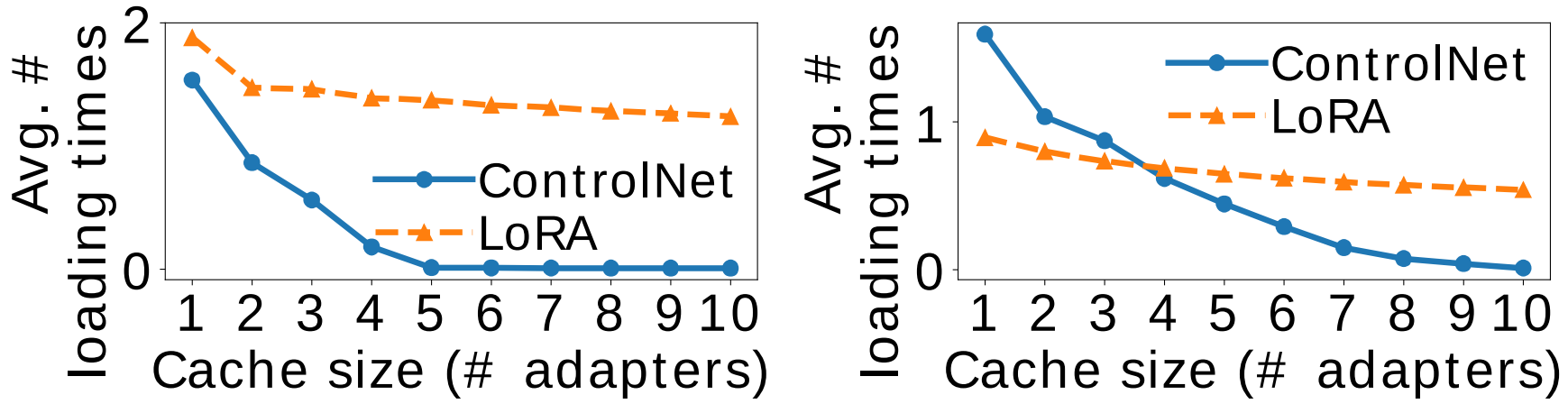
A base SDXL model augmented with ***m*** ControlNets and ***n*** LoRA (***m***C/***n***L) on H800 GPUs

This Talk

- Challenges
- Katz design
  - ControlNet-as-a-Service: Efficient ControlNet serving
  - Bounded Asynchronous Loading: Efficient LoRA loading
  - Optimized Base Model Execution
- Evaluation
- Conclusion

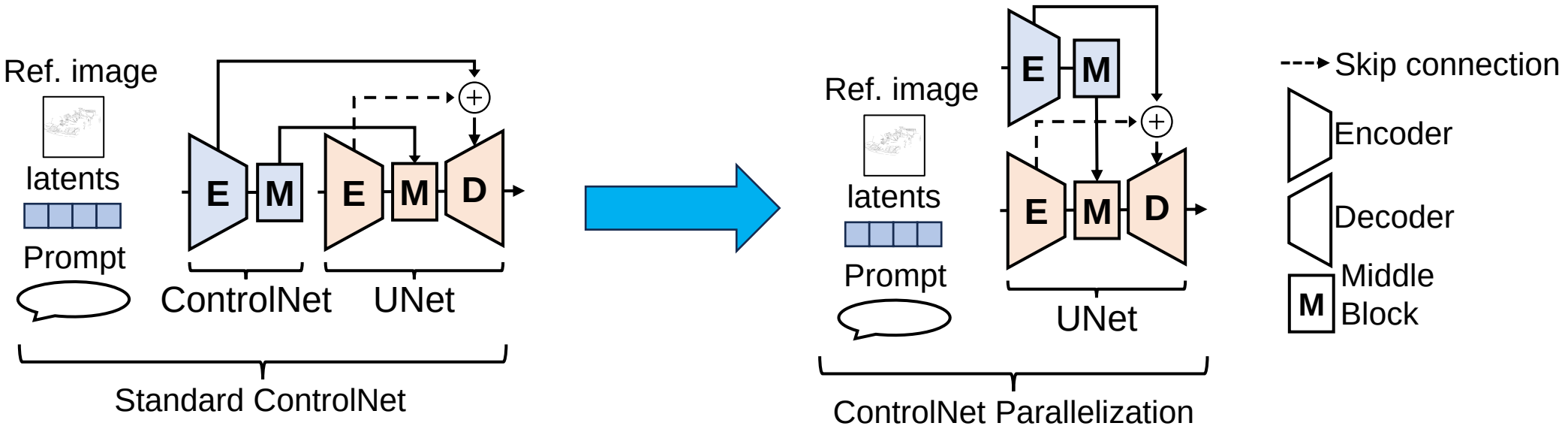
Optimizing ControlNet: Opportunity #1

- Skewed popularity of ControlNets
  - Service-A: Top-5 most popular ControlNets account for 98% invocations
  - Service-B: Top-8 account for 95% invocations
- Caching a few popular ControlNets in GPU can largely eliminate loading overhead



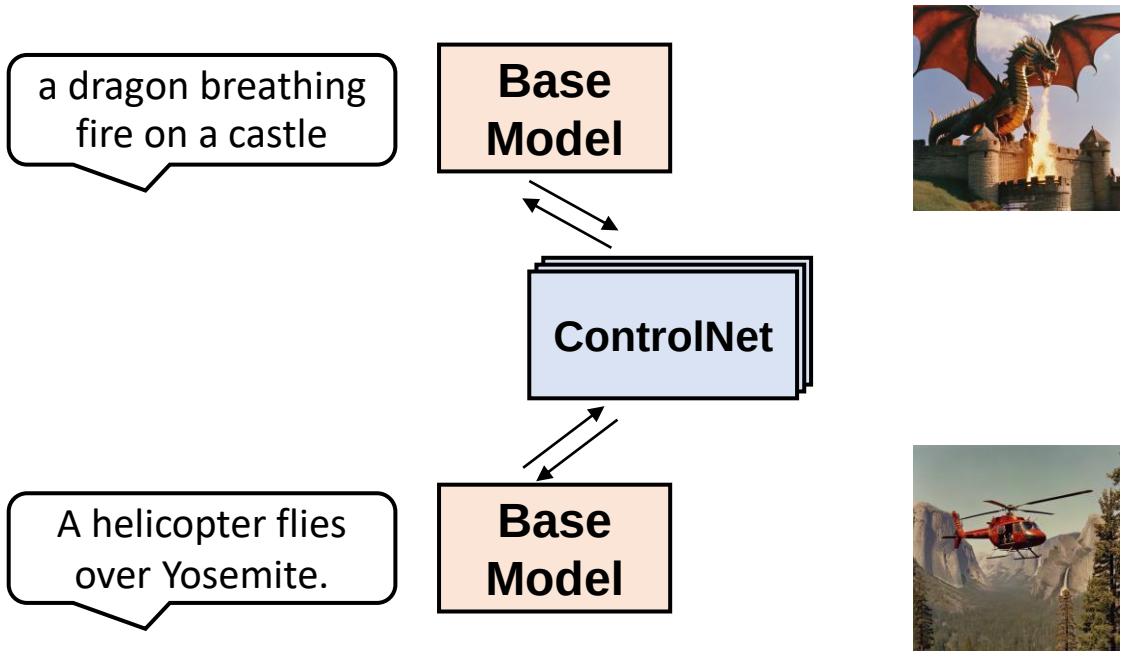
Optimizing ControlNet: Opportunity #2

- ControlNet parallelization
  - Concurrently execute ControlNet(s) with base model on multiple GPUs



Putting It Together: ControlNet-as-a-Service

- **ControlNet-as-a-Service:** deploy ControlNets as a separate, independently scaled service on dedicated GPUs
  - Caching popular ControlNets
  - ControlNet parallelization
  - Shared ControlNet service among workflows

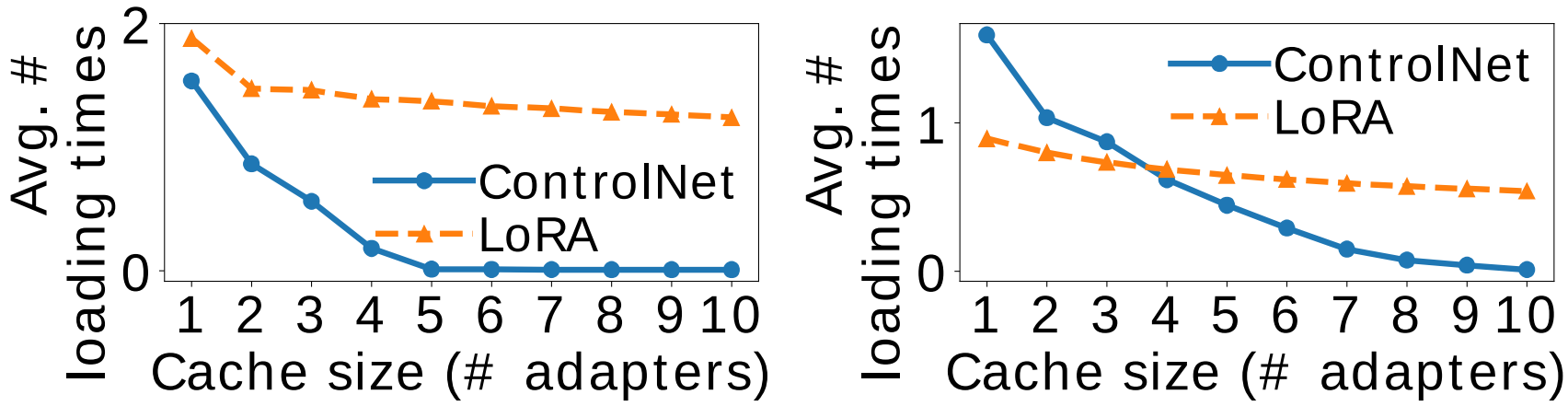


This Talk

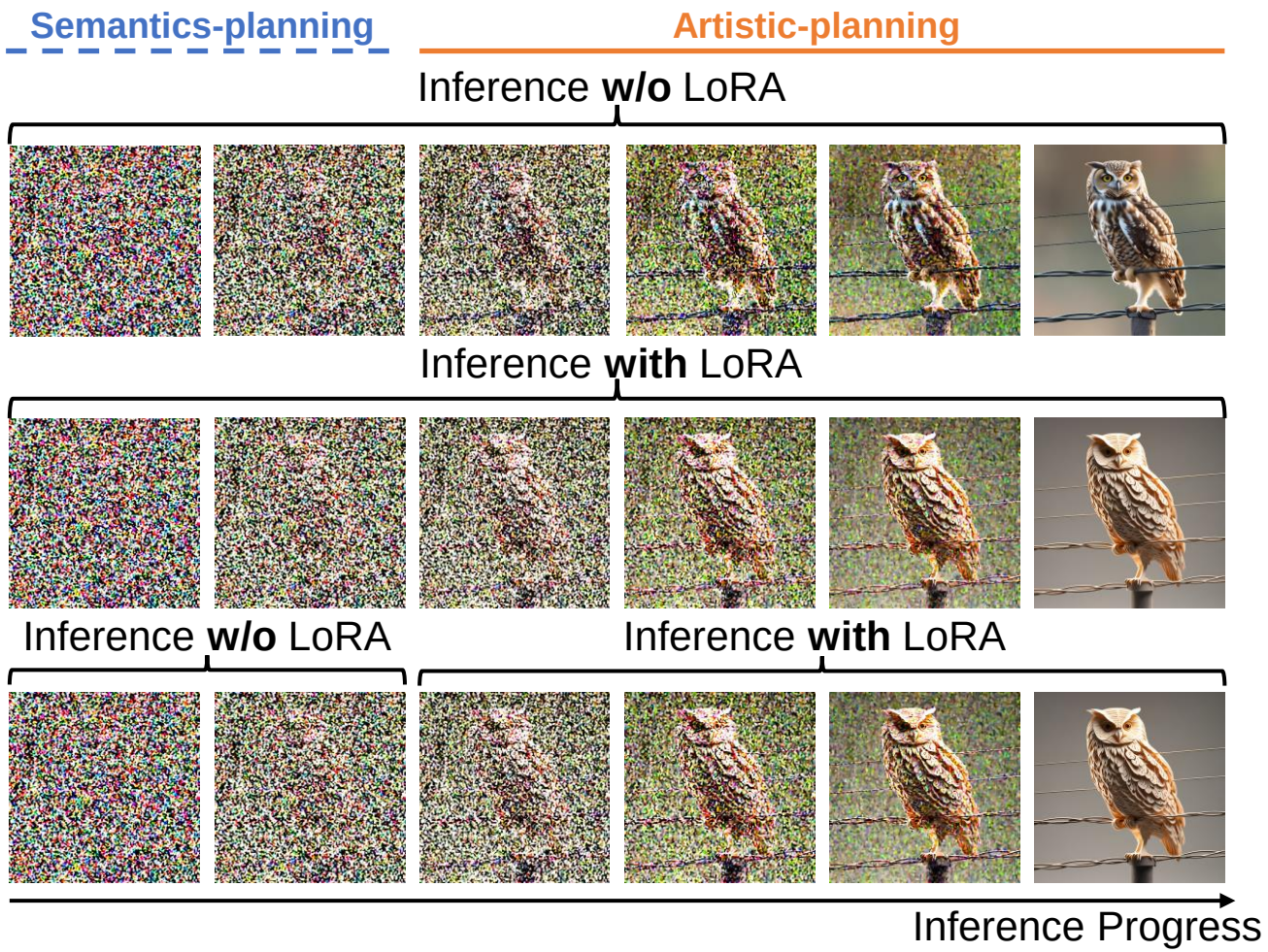
- Challenges
- Katz design
  - ControlNet-as-a-Service: Efficient ControlNet serving
  - Bounded Asynchronous Loading: Efficient LoRA loading
  - Optimized Base Model Execution
- Evaluation
- Conclusion

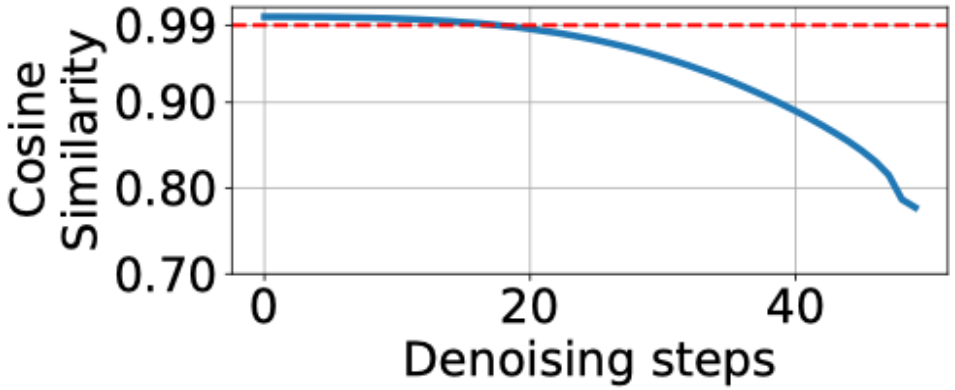
The Loading Bottleneck of LoRA Serving

- In production, LoRA is fetched from a remote storage or disk
  - Fetching LoRAs of size 800 MiB takes more than 1 second, delaying serving latency by **34%**
- LoRA caching is ineffective
  - LoRA population follows a long-tail distribution



The Magic of LoRA

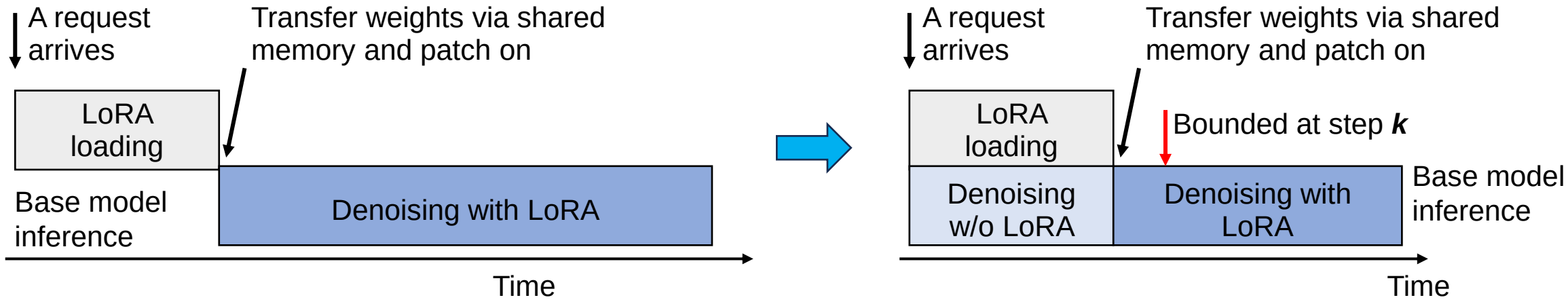




Cosine similarities between the latents generated with LoRA and those without LoRA at each denoising step.

Bounded Async LoRA Loading (BAL)

- Overlap LoRA loading and base model execution in the initial stage
- Impose an asynchrony bound ***K*** to ensure good image quality
- Engineering optimizations: use shared mem and in-place LoRA patching

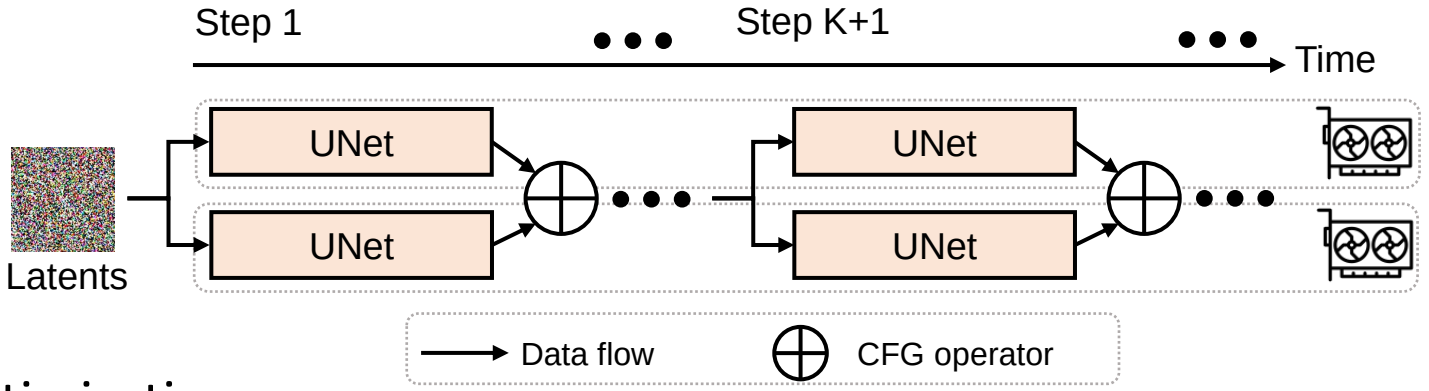


This Talk

- Challenges
- Katz design
  - ControlNet-as-a-Service: Efficient ControlNet serving
  - Bounded Asynchronous Loading: Efficient LoRA loading
  - Optimized Base Model Execution
- Evaluation
- Conclusion

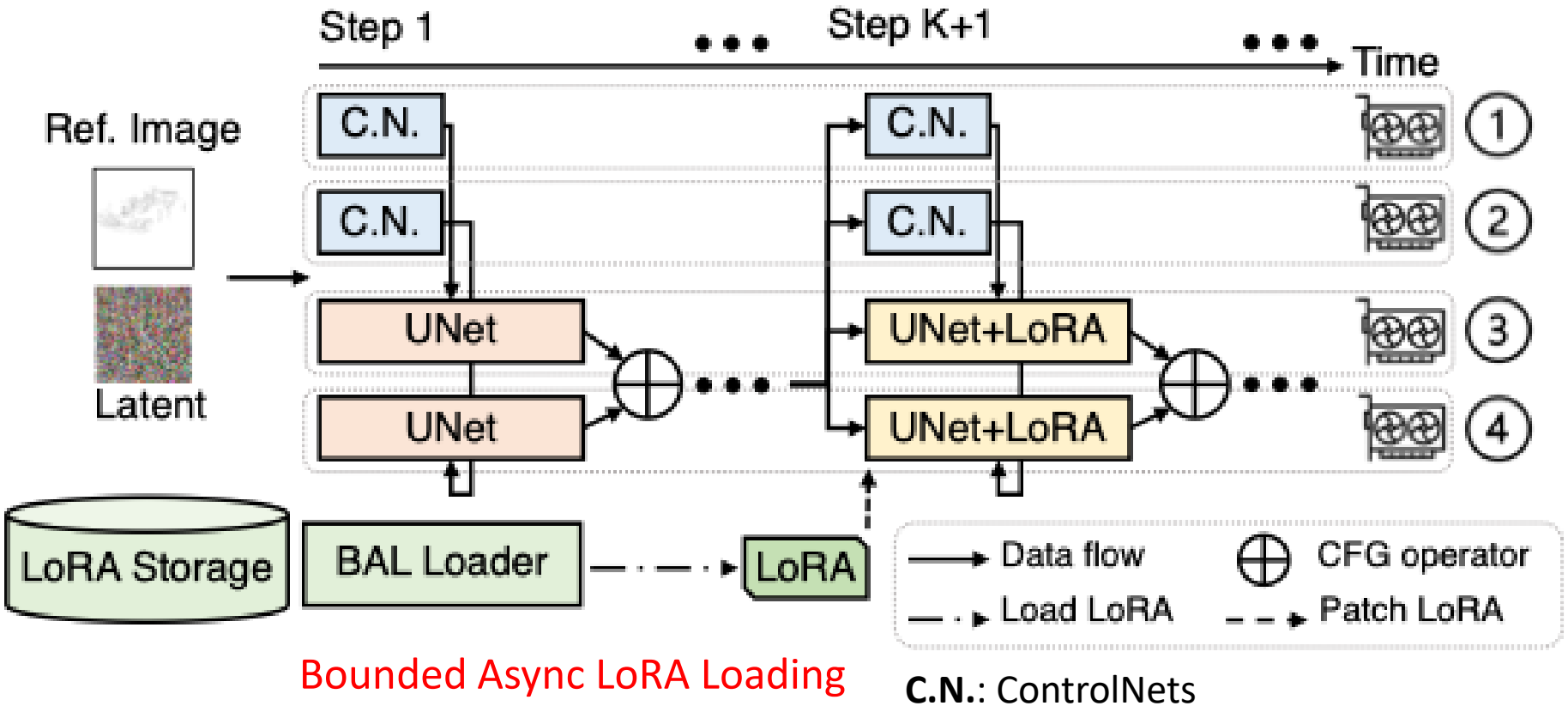
Optimized Base Model Execution

- CFG computation accounts for 90% of base model execution time
- Latent parallelism
  - Parallelize the CFG computation in image generation
  - Accelerate base model execution with multiple GPUs



- Kernel-level optimization
  - CUDA graph
  - Kernel optimization specific to UNet in SDXL

Katz: Putting It Altogether



ControlNet-as-a-Service

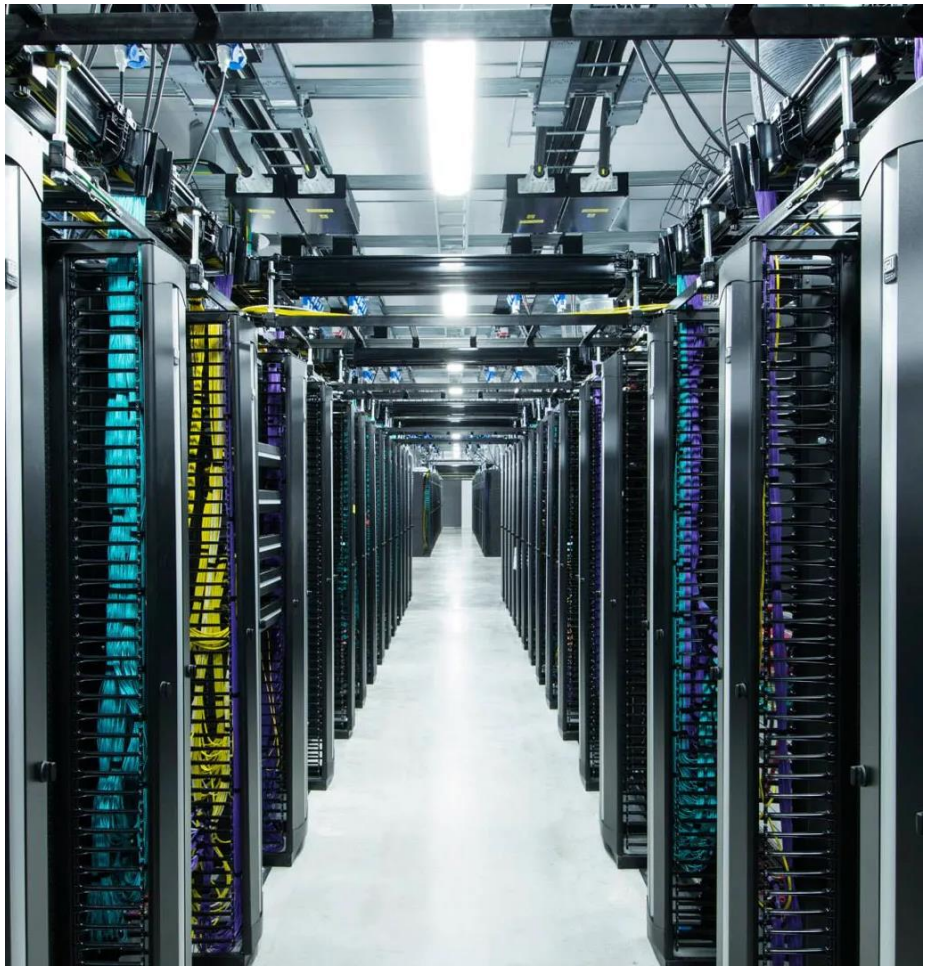
Latent parallelism

Bounded Async LoRA Loading

C.N.: ControlNets

This Talk

- Challenges
- Katz design
  - ControlNet-as-a-Service: Efficient ControlNet serving
  - Bounded Asynchronous Loading: Efficient LoRA loading
  - Optimized Base Model Execution
- Evaluation
- Conclusion



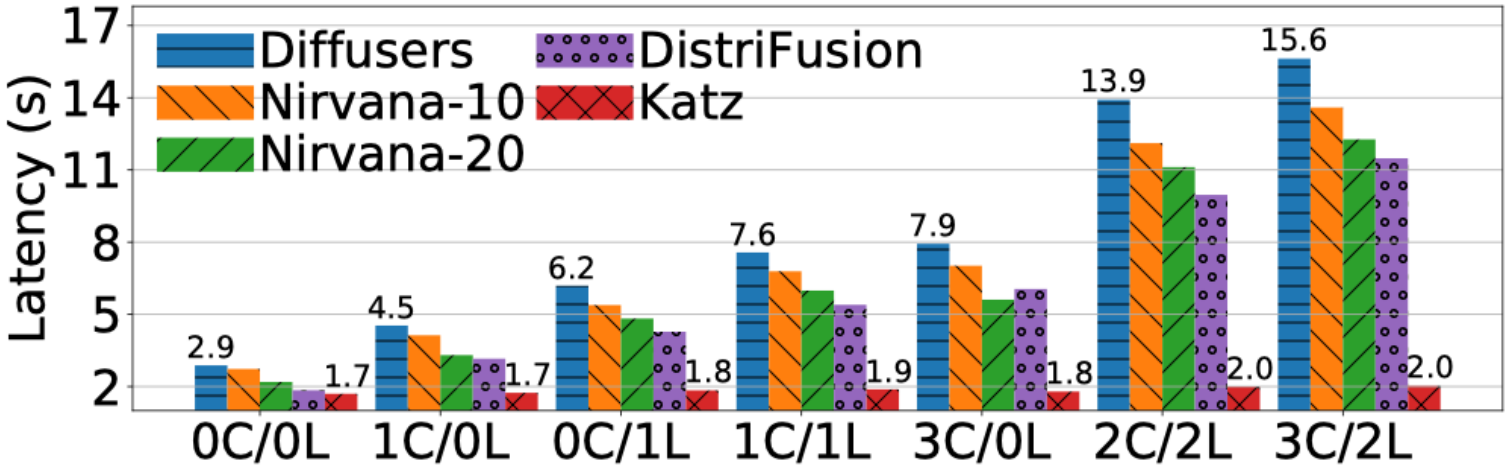
Methodology

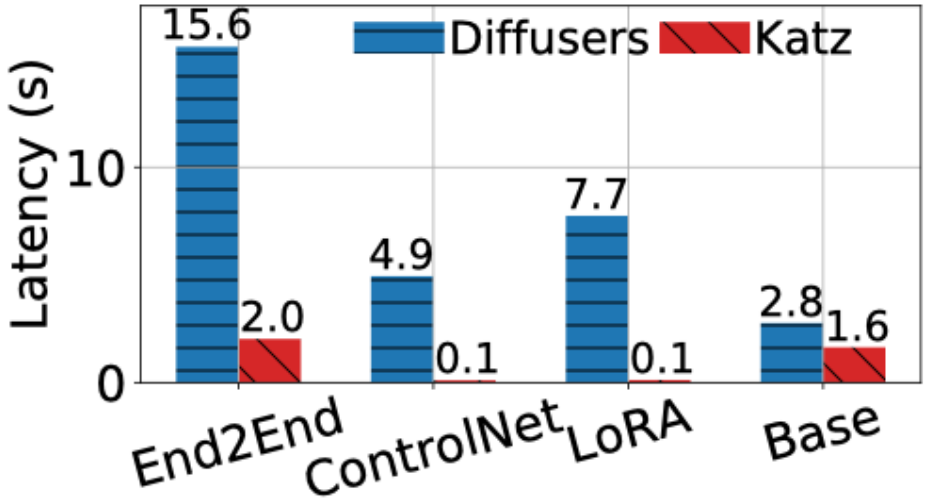
- **Testbed**
  - NVIDIA H800 SuperPOD, 400 Gbps IB
  - Base model: SDXL[1]
- **Baselines**
  - ***Diffusers****:* standard workflow; image quality upper bound
  - ***Nirvana*** *[NSDI'24]:* accelerate image generation by skipping **κ** steps
  - ***DistriFusion*** *[CVPR'24]:* accelerate image generation using multiple GPUs
- **Serving metrics:**
  - Serving latency
  - Image quality
    - Quantitative: CLIP(↑), FID(↓), SSIM(↑)
    - Qualitive: User study

[1]Our design can generalize to DiT-based models with details in our paper.

Evaluation: Serving Latency

- Up to **7.8x speedup** in end-to-end latency of generating an image of 1024x1024
- Up to **1.7x** per GPU throughput improvement
- End-to-End latency and component breakdowns for a 3C/2L request, using *Diffusers* and *Katz*
  - Overhead associated with adapters are virtually eliminated.



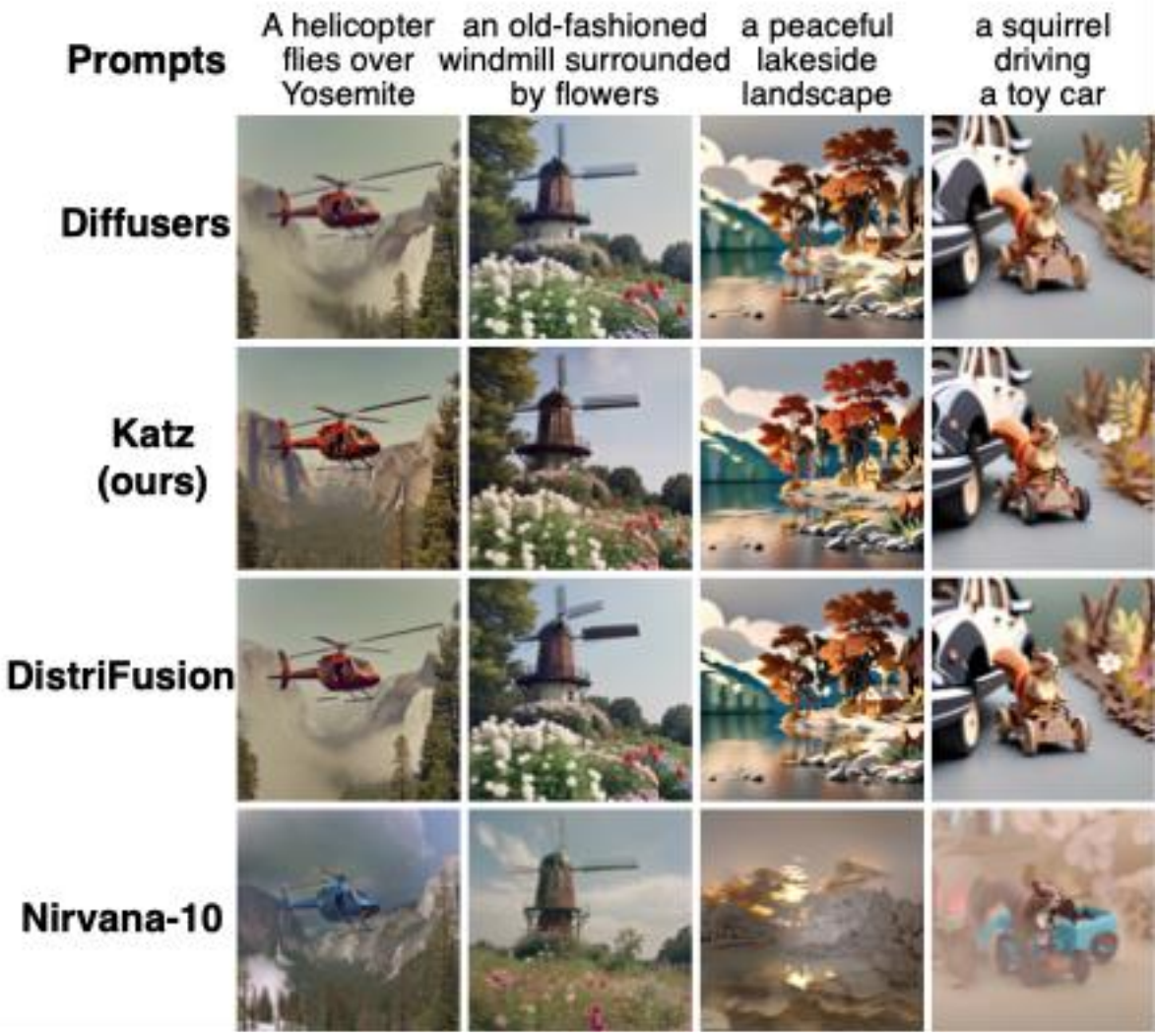


Evaluation: Image quality

- Quantitative

LoRA Setting	System	CLIP(↑)	FID (↓)	SSIM (↑)
One LoRA: Papercut [46]	DIFFUSERS	34.1	-	-
	NOLORA	32.9	11.4	0.63
	NIRVANA-10	33.5	9.5	0.45
	NIRVANA-20	33.7	10.9	0.44
	DISTRIFUSION	34.0	1.7	0.86
	KATZ (ours)	34.1	2.1	0.83
Two LoRAs: Filmic [45] + Photography [47]	DIFFUSERS	34.2	-	-
	NOLORA	31.3	13.4	0.67
	NIRVANA-10	33.3	9.0	0.51
	NIRVANA-20	32.8	9.4	0.47
	DISTRIFUSION	34.1	2.9	0.86
	KATZ (ours)	34.1	3.1	0.78

- Qualitive
  - Collect **1.2k data** points from **75 human participants**
  - No image quality loss compared with Diffusers



This Talk

- Challenges
- Katz design
  - ControlNet-as-a-Service: Efficient ControlNet serving
  - Bounded Asynchronous Loading: Efficient LoRA loading
  - Optimized Base Model Execution
- Evaluation
- Conclusion

Conclusion

- First comprehensive characterization study of text-to-image serving workflows
  - Adapters are ***effective*** and ***prevalent*** in production workloads.
  - Adapters poses new performance challenges: ***loading*** and ***computation***
- ControlNet-as-a-Service
  - ***Caching popular ControlNets; ControlNets parallelization; ControlNets multiplexing***
- Bounded async LoRA loading
  - ***Overlapping LoRA loading and base model execution in the initial image generation stage***
- Optimized base model execution
  - ***Latent parallel***
  - ***Kernel-level optimizations***

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Alibaba Group 阿里巴巴集团