Inference Serving System for Stable Diffusion as a Service

Aritra Ray*, Lukas Dannull*, Farshad Firouzi*, Kyle Lafata*, Krishnendu Chakrabarty†

*Duke University †Arizona State University

***Abstract*—We present a model-less, privacy-preserving, low-latency inference framework to satisfy user-defined System-Level Objectives (SLO) for Stable Diffusion as a Service (SDaaS). Developers of Stable Diffusion (SD) models register their trained models on our proposed system through a declarative API. Users, on the other hand, can specify SLOs in terms of the style of the generated image for their input text, the requested processing latency, and the minimum requested text-to-image similarity (CLIP score) for inference through the user API. Assuming black-box access to the registered models, we profile them on hardware accelerators to design an inference predictor module. It heuristically predicts the required number of inference steps for the user-requested text-to-image CLIP score and the requested latency, for a specific SD model over a hardware accelerator, to satisfy the SLO. In combination with the inference predictor module, we propose a shortest-job first algorithm for our inference framework. Compared to traditional Deep Neural Network (DNN) and Large Language Model (LLM) inference scheduling algorithms, our proposed method outperforms on average job completion time, and the average number of SLOs satisfied in a user-defined SLO scenario.**

***Index Terms*—Inference Serving System, Stable Diffusion**

I. Introduction

To serve traditional DNN models in the cloud, the trained models are deployed on CPUs [1], [2], and one single forward pass cycle is initiated for every input query to generate a classification label. The optimizations are thereby focused on model switching [1], ease-of-use [3], and lower inference latency [2], to highlight a few. Unlike so, in SD models, the input text is mapped to the token embeddings as a representation of the input text, and starting with a random noisy latent image information array, the diffusion refines the information array such that the image decoder uses it to decode the final image. This process happens in a step-by-step fashion, with each diffusion step adding more relevant information to the latent array. With generative Artificial Intelligence (AI) models, particularly SD for text-to-image generation, being progressively deployed in the cloud [4], [5], and the striking difference in the inference process compared to DNNs motivates us to design a model-less, privacy-preserving, low-latency inference framework for SDaaS.

Developers are incrementally advancing the state-of-the-art SD models aimed at facilitating text-to-image conversion. For a variety of pre-trained SD models, each variant exhibits variations in resource footprints and processing time latencies across heterogeneous compute resources. In this paper, we present a model-less, privacy-preserving, low-latency inference framework to satisfy user-defined SLOs for SDaaS. Developers of SD models can register their models on our proposed system through a declarative API. Users, on the other hand, can specify SLOs in terms of the style of the generated image, the requested latency, and the minimum requested CLIP score for inference through the user API. Our proposed system manages model registration from the developers, and schedules volumes of user queries aimed to meet SLOs through an efficient deployment of the models onto hardware accelerators in the compute cluster.

The rest of the paper is organized as follows. Section II highlights our key findings to guide our inference framework development, followed by our proposed system design in Section III. Section IV delves into our evaluation results, and section V concludes our paper.

II. Motivation

We evaluate the runtime latencies for a specific prompt across 9 pre-trained SD models[1] for text-to-image generation. We specifically investigate the performance discrepancy between CPU and GPU for a single inference step in the diffusion process. The aggregated findings for a randomly selected prompt 'an astronaut riding a cow', are presented in Table I, utilizing a floating-point precision of 16 consistently across all SD models. We used an X86_64, employing 64-bit processing, powered by an Intel(R) Xeon(R) CPU @ 2.20GHz with an advertised frequency of 2.2000 GHz, and Tesla V100 as our CPU and GPU respectively. Our analysis reveals that for all the evaluated SD models, the average inference time on CPU is 92x higher, on average, compared to a GPU deployment. ❶This highlights the need for scheduling SD inferencing workloads on GPU to achieve significantly lower latency.

From the data presented in Table I, we observe the variability in inference times, when utilizing the same compute resource across a variety of SD models, for an identical query. ❷This elucidates that every SD model exhibits significant variations in their resource requirements.

Next, we subject the identical inference workload for one SD model, across various hardware accelerators. In Table II, we present the average inference time for 10 inference steps, for the SD model 'dreamlike-art/dreamlike-anime-1.0',

[1]https://huggingface.co/

2024 IEEE Cloud Summit | 979-8-3503-7006-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/CLOUD-SUMMIT61220.2024.00009

979-8-3503-7006-5/24/$31.00 ©2024 IEEE
DOI 10.1109/Cloud-Summit61220.2024.00009

TABLE I: Single step inference latency for stable diffusion models on CPU and GPU.

Stable Diffusion Models	Inference Time (sec)	
	CPU	GPU
DGSpitzer/Cyberpunk-Anime-Diffusion	22.47	0.317
dreamlike-art/dreamlike-anime-1.0	77.41	0.576
CompVis/stable-diffusion-v1-4	27.21	0.309
stabilityai/stable-diffusion-2-1-base	21.77	0.315
stabilityai/stable-diffusion-2-1	86.64	0.734
SG161222/Realistic_Vision_V1.4	35.34	0.304
hakurei/waifu-diffusion	24.94	0.328
Nilaier/Waifu-Diffusers	22.58	0.314
kohbanye/pixel-art-style	27.85	0.302

while achieving a CLIP score of 0.37 on 'ViT-B/32'. ❸This elucidates the pronounced variability in processing latencies across hardware accelerators in the compute cluster for SD models.

To evaluate the influence of floating point precision in SD models, we present in Table III, the mean inference time over 10 inference steps, and CLIP score ('ViT-B/32') for the SD model 'dreamlike-art/dreamlike-anime-1.0', on the A100 PCIE GPU, for varying inference steps. With a higher floating point precision, the SD model achieves a marginally higher CLIP score in fewer inference steps, at the expense of higher latency per inference steps. ❹This elucidates that there exists a marginal trade-off between the floating point precision of the SD model to the associated inference time latency and CLIP score.

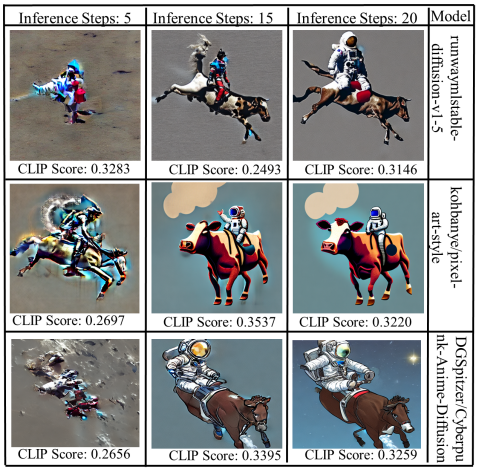


Fig. 1: Inference latency and CLIP Score ('ViT-B/32') for different styles of stable diffusion models, at seed 1024, for prompt 'an astronaut riding a cow'.

To illustrate the diversity of styles in the output image, we present a selection of generated images from 3 SD models: namely, 'runwaymlstable-diffusion-v1-5', 'kohbanye/pixel-art-style', and 'DGSpitzer/Cyberpunk-Anime-Diffusion', across variable inference steps along with their corresponding CLIP Scores ('ViT-B/32'), as shown in Fig. 1. Generally, we anticipate that as the number of inference steps increases, the quality of the images will improve, leading to a proportional increase in the CLIP score. However, it is noteworthy that the variation in generated features within the latent information array, in response to the input prompt, leads to a non-monotonic relationship between the number of inference steps and the CLIP score.

To second this further, we extract the text captions from the Flickr8k [6] dataset and feed 10% text prompts into SD models to evaluate the runtime latency and CLIP score over 3 hardware accelerators. ❺From Fig. 2, we observe a monotonic increase in inference latency with respect to the inference steps, but a non-monotonic relationship between the number of inference steps and the CLIP score.

III. System Design

In this section, we present the design of our proposed model-less, privacy-preserving, low-latency inference framework to satisfy user-centric SLOs for SDaaS. The components of our proposed system are outlined subsequently.

Design Principles. Developers of SD models can register their model on our proposed system through a declarative API. Users, on the other hand, can specify SLOs in terms of the style of the generated image, the requested latency, and the minimum requested CLIP score for inference through the user API. Our proposed system manages model registration from the developers, and schedules volumes of user queries to meet SLOs through an efficient deployment of the SD models to hardware accelerators in the compute cluster.

Model-less interface for inference. The front-end interface of our proposed system involves model registration from the developers and the submission of user queries for inferencing on the registered models.

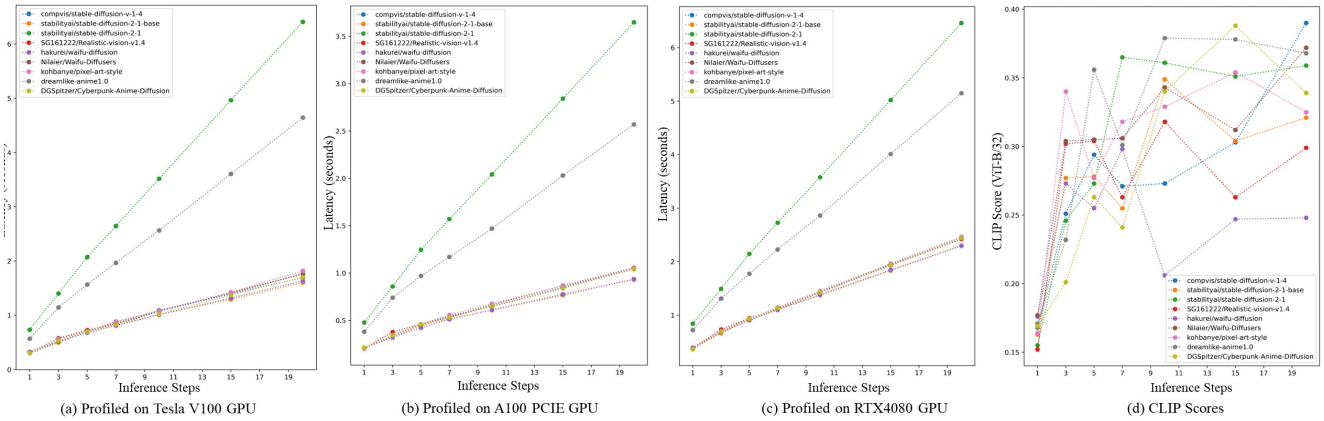
- *Model registration:* The developers can register their models using a declarative API. The API accommodates a model identifier designated by the developer and the trained weights of the SD model.
- *Query Submission:* Users can submit inference queries, mentioning the requested text prompt t_p to generate an output image, the style of the generated image s, and specify high-level performance criteria, like the requested CLIP score (SLO_{CLIP}). The user can also specify a requested SLO latency ($SLO_{latency}$) within which the request must be processed, alongside SLO_{CLIP}.

Architecture. The controller, depending on the specified s loads the appropriate model from the model repository, and forwards the inference query to a worker machine with the scheduling logic. The workers execute the inference queries to hardware accelerators and subsequently respond with the inference results to the user. If the user-defined SLO could not be satisfied for a particular query, the system returns a message appropriately to the user.

- **Controller** The centralized controller handles both model registration from developers, and SLO-based inference

TABLE II: Impact of inference latency for stable diffusion models on hardware accelerators.

GPU	RTX4090	RTX3090	A100PCIE	RTX4080	RTXA6000	RTXA4000	TESLAV100	RTX3060	A40	RTX3080
Inference Time (s)	5.473	5.952	3.684	7.417	5.591	9.601	6.804	13.997	6.214	7.211



(a) Profiled on Tesla V100 GPU (b) Profiled on A100 PCIE GPU (c) Profiled on RTX4080 GPU (d) CLIP Scores

Fig. 2: Processing latency for SD models, over (a) Tesla V100, (b) A100, (c) RTX4080 hardware accelerators. (d) Represents the CLIP score ('ViT-B/32') of the SD models as a function of inference steps.

TABLE III: Impact of inference time to CLIP score for floating point precision in stable diffusion models.

Inference Steps	Half-precision (16FP)		Single-precision (32FP)	
	Latency (s)	CLIP Score	Latency (s)	CLIP Score
1	0.123	0.163	0.283	0.165
2	0.195	0.168	0.386	0.175
3	0.241	0.341	0.493	0.348
4	0.293	0.296	0.604	0.309
5	0.339	0.269	0.704	0.295
6	0.376	0.327	0.805	0.334
7	0.411	0.322	0.892	0.329
8	0.447	0.314	0.985	0.324

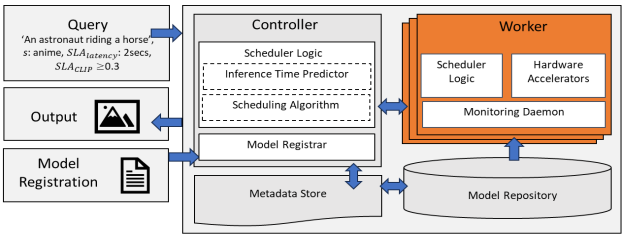


Fig. 3: Overview of our proposed system architecture.

queries from the users. It comprises of two modules: (a) *scheduler logic block*, and (b) *model registrar*. The *scheduler logic block* consists of the inference time predictor and the scheduling algorithm. Assuming black-box access to the models to preserve developer privacy, all registered models are profiled on the worker nodes (hardware accelerators) over multiple text-to-image prompts for varying inference steps. Based on the profiling results, the inference time predictor heuristically predicts the number of inference steps, for a user-defined SLA_{CLIP} and $SLA_{latency}$, for a specified SD model, over a hardware accelerator, to satisfy the SLO for the specific query. The other module, *model registrar* is responsible for managing model registration from the developers to the *model repository* through an API.

- **Worker** Worker nodes execute inference queries, following the scheduler logic, as directed by the controller, over the hardware accelerators. Hardware-specific execution daemons manage the deployment and execution of models.
- **Model Repository** The *Model Repository* functions as a high-capacity, persistent storage, containing the trained SD models registered by the developers. The worker nodes can access this storage to execute SD models on hardware accelerators based on the user inference query.
- **Metadata Store** Assuming black-box access to the models to preserve developer privacy, all registered models are profiled on the worker nodes over multiple text-to-image prompts for varying inference steps. The metadata store includes information concerning available models, and the profiled attributes of the number of inference steps to mean CLIP score and mean inference latency. The inference time predictor accesses this data to heuristically predict the number of inference steps, for the SLA_{CLIP} and $SLA_{latency}$ for a specified model, for a hardware accelerator, to satisfy the SLO for the specific query.

IV. Evaluation

We evaluated our inferencing framework over 3 SD models, namely 'DGSpitzer/Cyberpunk-Anime-Diffusion',

'runwayml/stable-diffusion-v1-5', and 'hakurei/waifu-diffusion' corresponding to styles (s) anime, regular diffusion, and waifu respectively. To simulate the text prompts t_p in user queries, we randomly sampled 100 prompts from Flickr8k dataset [6], with a randomly assigned requested style s (35 t_p with s: regular diffusion, 33 t_p with s: anime, 32 t_p with s: waifu). The requested CLIP score (SLA_{CLIP}) is sampled from a normal distribution with μ: 0.3, σ: 0.02, while the requested latency ($SLA_{latency}$) is sampled from normal distribution with μ: 10, σ: 2, in the user query. Due to a lack of publicly available traces for generative AI workloads, we opted for increasing workload pattern [7], for the job arrival rate. We set a job arrival rate of 0.8 jobs/second, increasing linearly to 0.85 jobs/second over 100 seconds. All experiments were performed over NVIDIA T4, NVIDIA A10G, at a fixed floating point precision 16 for SD model. The CLIP score was estimated with a pre-trained 'ViT-B/32' model.

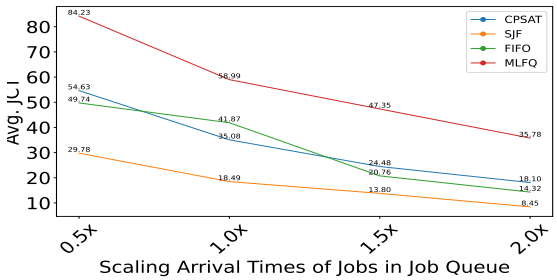


Fig. 4: Variation in job arrival rate to average job completion time. Our proposed inference time predictor in combination with SJF has the highest throughput.

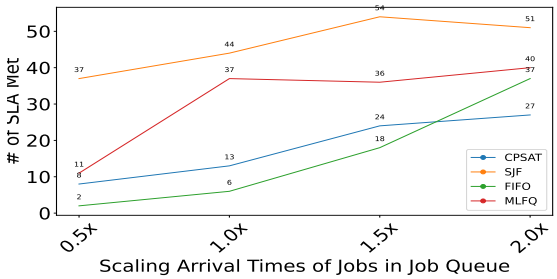


Fig. 5: Variation in job arrival rate to the number of SLOs met. Our proposed inference time predictor in combination with SJF satisfies the most number of SLOs.

Assuming a black-box access, we extract the meta-data of the runtime latencies and the associated CLIP score of the 3 SD models in a privacy-preserving fashion. Utilizing this data, we design the inference predictor to heuristically predict the number of inference steps, for a user-defined SLO_{CLIP} and $SLO_{latency}$, for a specific SD model, for a hardware accelerator, to satisfy the SLO for the specific query. As a baseline, we use First In First Out (FIFO), no preemption Multi-Level Feedback Queue (MLFQ) used to schedule LLMs [8] (without KV-caching), constrained Integer

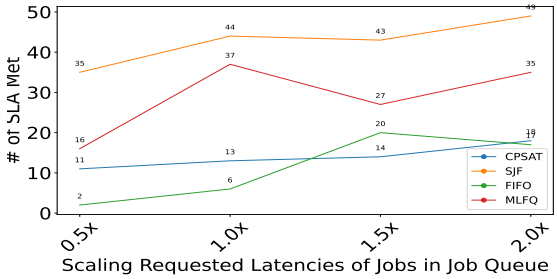


Fig. 6: Variation in $SLO_{latency}$ to the number of SLOs met. Our proposed inference time predictor in combination with SJF satisfies the most number of SLOs.

Linear Programming[2] (CP-SAT) used to schedule traditional DNNs (in our case, we utilize the inference predictor to predict the number of inference steps for processing latency), and our proposed Shortest Job First (SJF) algorithm in combination with the inference time predictor.

We first evaluate a scenario wherein the requested job queries are characterized by t_p, SLO_{CLIP} and s. We scale the job arrival rate to compute the average job completion time. Our results, as presented in Fig. 4 indicate the highest system throughput for our proposed inference time predictor module in combination with SJF. Next, we evaluate a scenario wherein the jobs queries are characterized by t_p, SLO_{CLIP}, $SLO_{latency}$ and s. On scaling the job arrival rate and $SLO_{latency}$, our results indicate our proposed inference time predictor in combination with SJF outperforms the baseline, as presented in Fig. 5, and 6 respectively.

V. Conclusion

Our proposed model-less, privacy-preserving, low-latency inferencing framework for SDaaS outperforms the baseline inference scheduling approaches. It would be essential to conduct evaluations on an expanded cloud trace, incorporating additional model variants and alternative job arrival patterns [7]. The inference time predictor can be dynamically updated to adapt to SLO needs.

References

[1] J. Zhang et. al, "Model-switching: Dealing with fluctuating workloads in machine-learning-as-a-service systems," in *USENIX HotCloud 2020*.

[2] J. R. Gunasekaran et. al., "Cocktail: A multidimensional optimization for model serving in cloud," in *USENIX NSDI 2022*.

[3] F. Romero et. al., "Infaas: Automated model-less inference serving," in *USENIX ATC 2021*.

[4] OpenAI, "Chatgpt," Large language model, 2024. [Online]. Available: https://chat.openai.com

[5] Microsoft, "Microsoft copilot," AI-powered code completion tool, 2022. [Online]. Available: https://developer.microsoft.com/en-us/copilot

[6] M. Hodosh et. al., "Framing image description as a ranking task: Data, models and evaluation metrics," *Journal of Artificial Intelligence Research 2013*.

[7] I. K. Kim et. al., "Forecasting cloud application workloads with cloudinsight for predictive resource management," *IEEE Transactions on Cloud Computing 2020*.

[8] B. a. Wu et. al., "Fast distributed inference serving for large language models," *arXiv preprint arXiv:2305.05920*, 2023.

[2]https://github.com/google/or-tools