上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

SJTU

Communication-Efficient Diffusion Denoising Parallelization via Reuse-then-Predict Mechanism

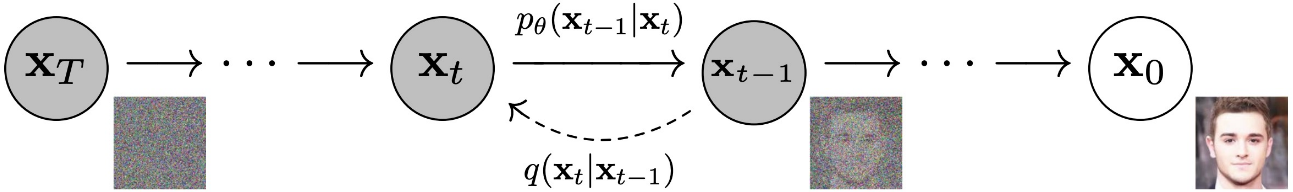
Communication-Efficient Diffusion Denoising Parallelization via Reuse-then-Predict Mechanism

Kunyun Wang · Bohan Li · Kai Yu · Minyi Guo · Jieru Zhao

Kunyun Wang

Shanghai Jiao Tong University

What is Diffusion Model

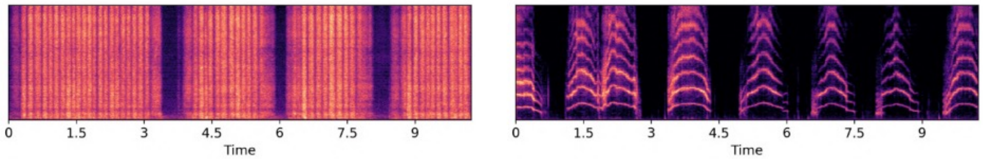


[1]

[2]

A pencil scribbling on a notepad.

A kitten mewing for attention.



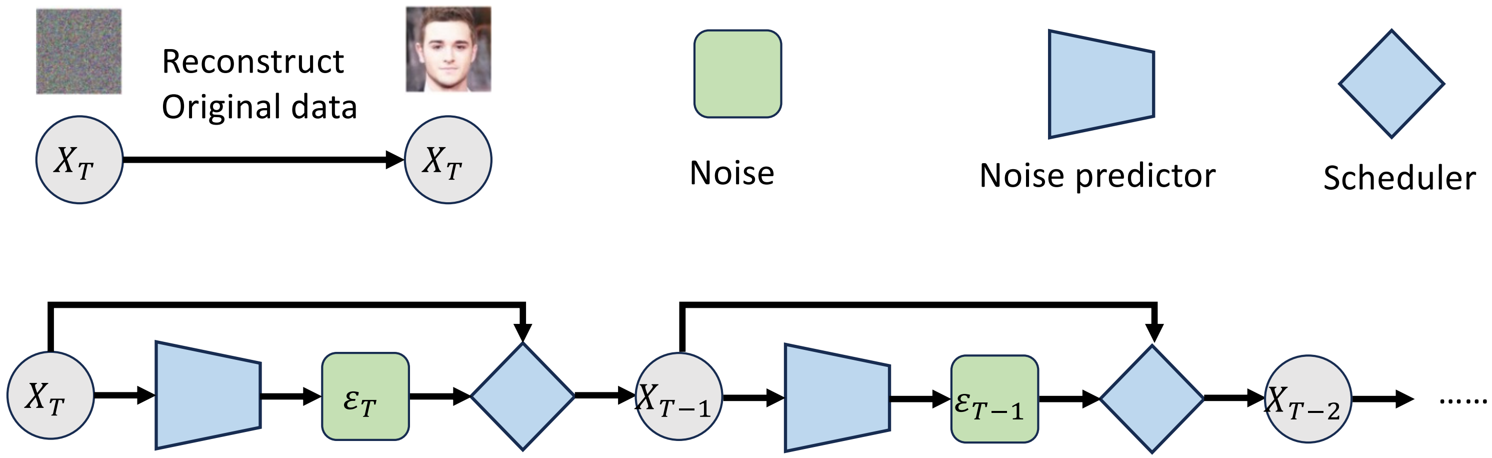
[3]

Diffusion models perform denoising process to reconstruct original samples

Diffusion models exhibit strong performance in image, video and audio generation

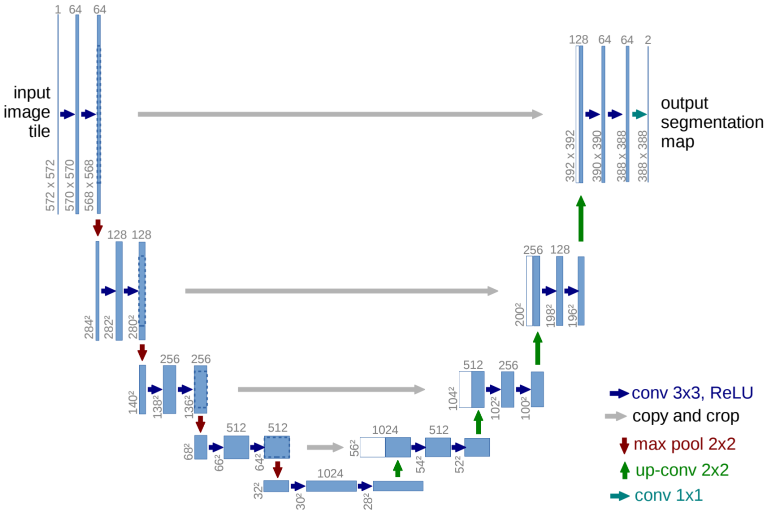
[1] Denoising Diffusion Probabilistic Models
[2] WAN: OPEN AND ADVANCED LARGE-SCALE VIDEO GENERATIVE MODELS
[3] AudioLDM 2: Learning Holistic Audio Generation with Self-supervised Pretraining

What is Diffusion Model

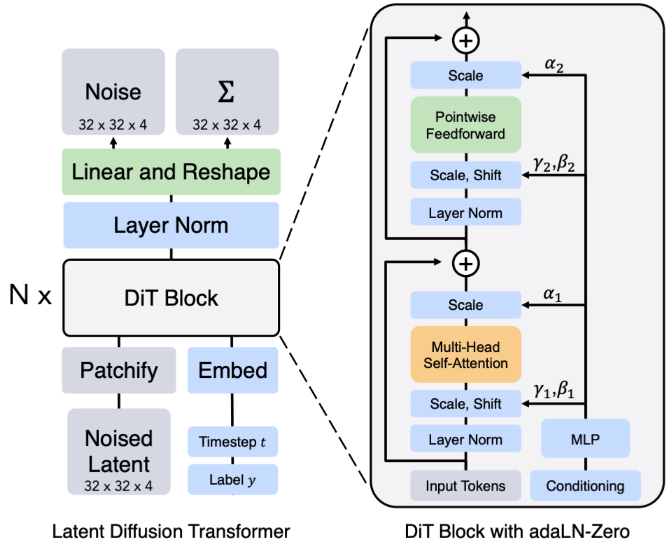


Tens or even hundreds of steps

What is Diffusion Model



[1] U-Net



[2] DiT

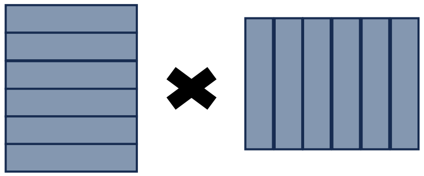
The noise predictor is typically implemented using architectures such as DiT or U-Net

[1] U-Net: Convolutional Networks for Biomedical Image Segmentation
[2] Scalable Diffusion Models with Transformers

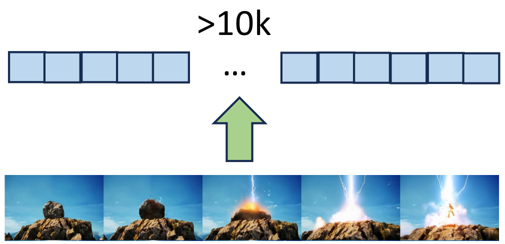
The high latency of Diffusion model

[1] SVD inference on 4090 for 51s for a 2s short video

[2] CogVideoX inference on 4090 for 91s for a 7s short video



Dense matmul



Bidirectional Attention

Thousands of tokens

Multi-step inference

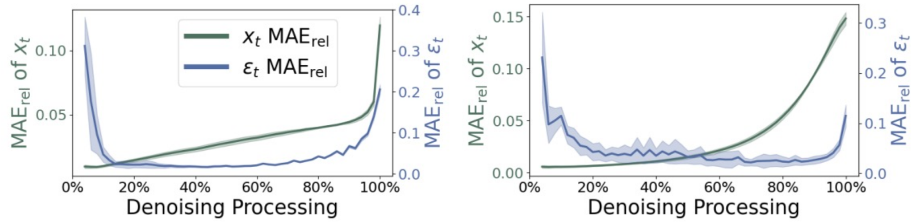
[1] Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets
[2] CogVideoX: Text-to-Video Diffusion Models with An Expert Transformer

Similarity across adjacent steps

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

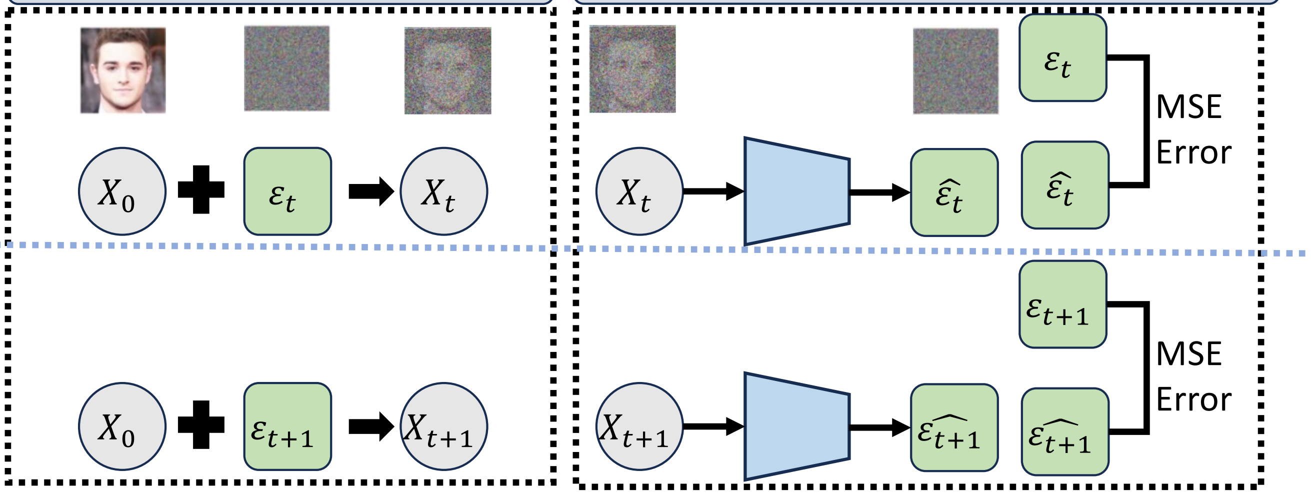
[1] The objective in training is to predict the added noise



(a) CogVideoX (b) HunyuanVideo

Adjacent denoising steps exhibit strong similarity, consistent with our analysis.

Forward: Get noisy sample **Reverse:** Predict the added noise



ε_{t+1} is closed to ε_t in training

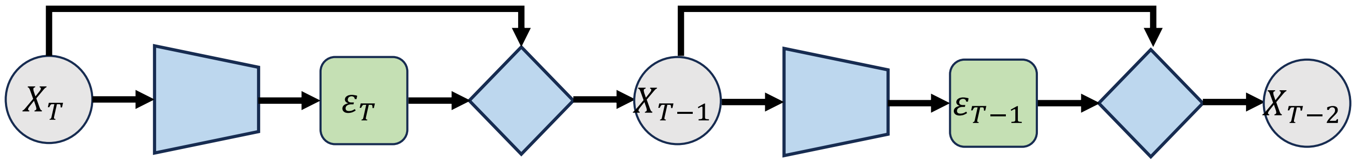
$\hat{\varepsilon}_t$ is closed to ε_t

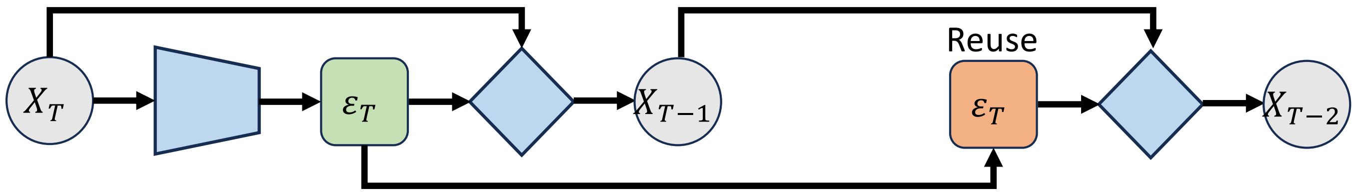
$\widehat{\varepsilon_{t+1}}$ is closed to ε_{t+1}

Therefor, $\hat{\varepsilon}_t$ is closed to $\widehat{\varepsilon_{t+1}}$

[1] Denoising Diffusion Probabilistic Models

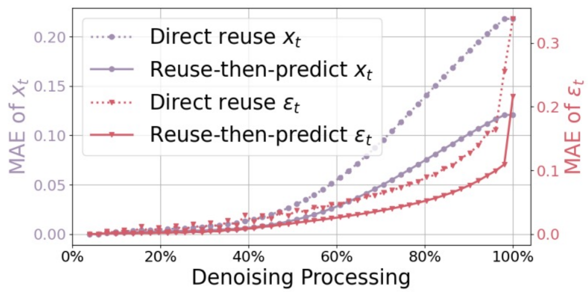
Direct Reusing cause cumulative deviation



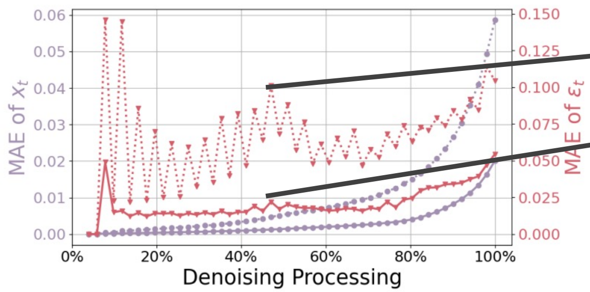


We can directly reuse the generated Noise ε_t

However, this can cause cumulative deviation



(a) Latte

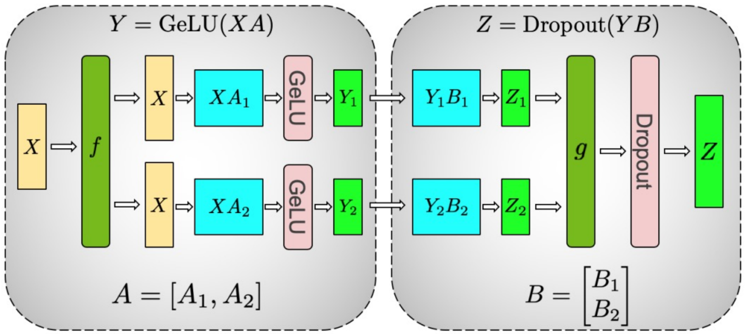


(b) HunyuanVideo

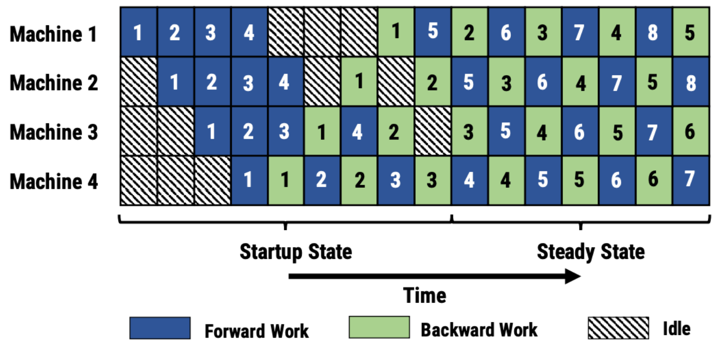
Direct reuse cause higher deviation

Our proposed method can alleviate this deviation

Can we use traditional parallelism to reduce Diffusion latency?



[1] Megatron-LM



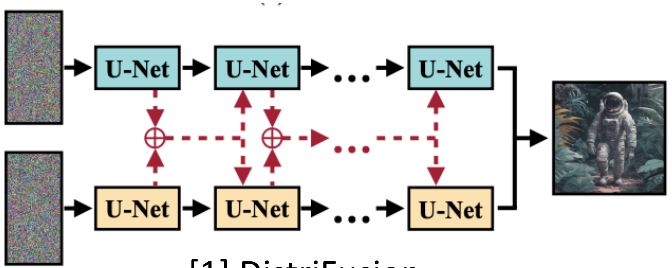
[2] PipeDream

Traditional parallelism is not suitable

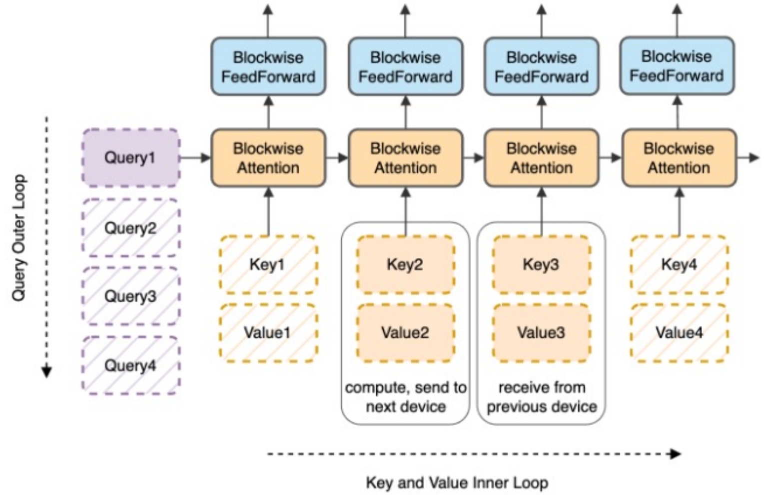
- ➢Data parallelism and Pipeline parallelism are primarily designed to improve throughput, NOT LATENCY
- ➢Tensor parallelism is less suitable for Diffusion models due to their large activation sizes, especially in commercial hardware.

[1] Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism
[2] PipeDream: Fast and Efficient Pipeline Parallel DNN Training

Specified parallel methods for reducing diffusion latency



[1] DistriFusion



[2] Ring Attention

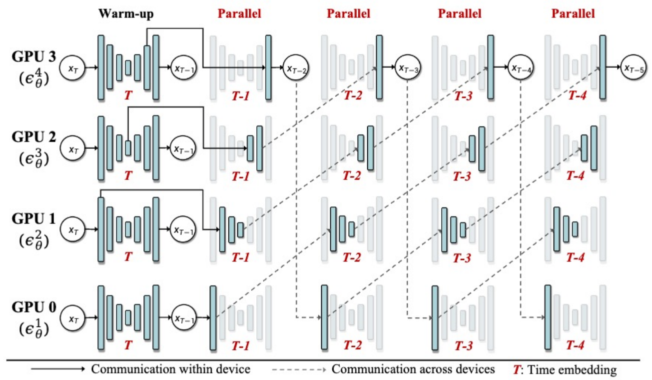


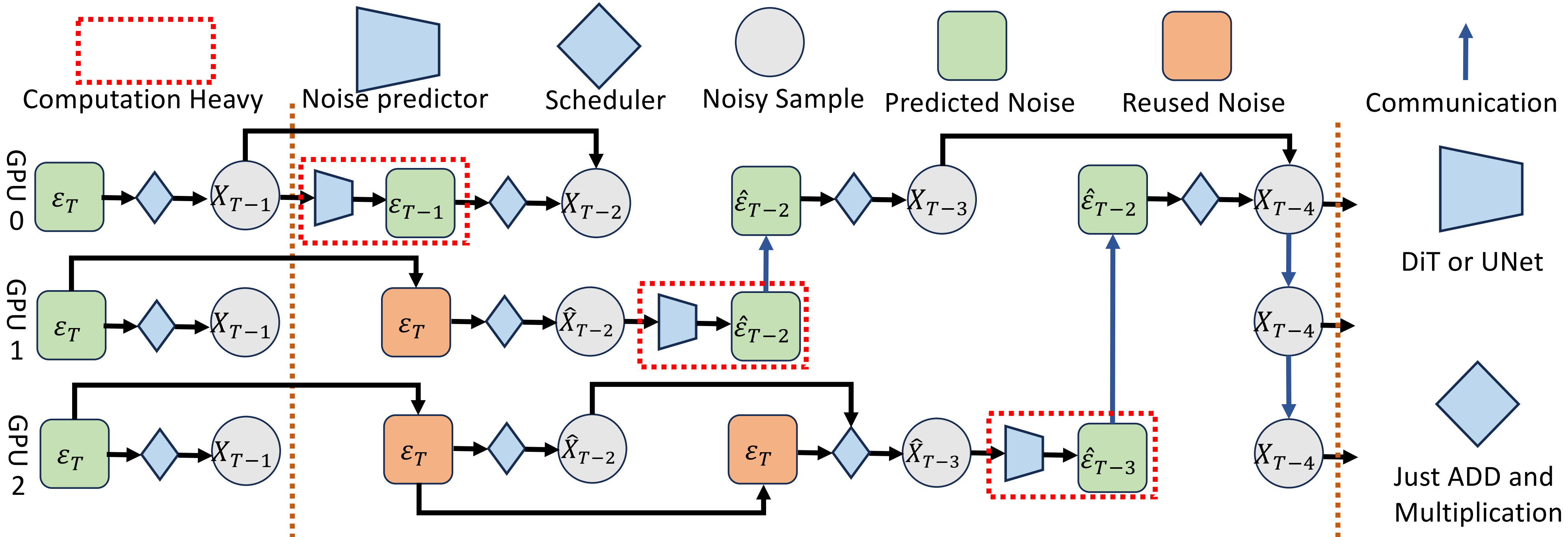
Figure 3: Overview of the asynchronous denoising process. The denoising model ϵ_θ is divided into four components $\{\epsilon_\theta^n\}_{n=1}^4$ for clarity. Following the warm-up stage, each component's input is prepared in advance, breaking the dependency chain and facilitating parallel processing.

[3] AsyncDiff

These methods perform **operation-wise**, **layer-wise**, or **stage-wise communication** to exchange essential tensors, which incurs **substantial communication cost.**

[1] DistriFusion: Distributed Parallel Inference for High-Resolution Diffusion Models
[2] Ring Attention with Blockwise Transformers for Near-Infinite Context
[3] AsyncDiff: Parallelizing Diffusion Models by Asynchronous Denoising

Parallel method based on Reuse-then-Predict

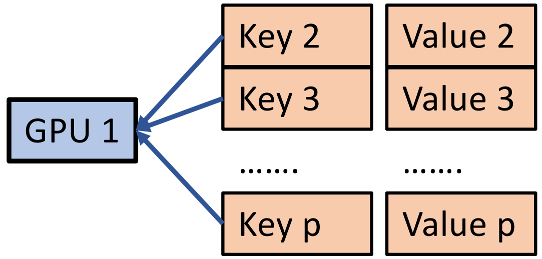


- Use Reuse-then-Predict process to reduce the deviation caused by Direct Reuse
- Through careful arrangement, reuse-then-predict can be computed in parallel.
- Step-wise, low volume communication

Communication analysis

Hidden States — Message size: M
GPU — p
Key 1 — Message size: M/p
Model: layernum L

Ring Attention

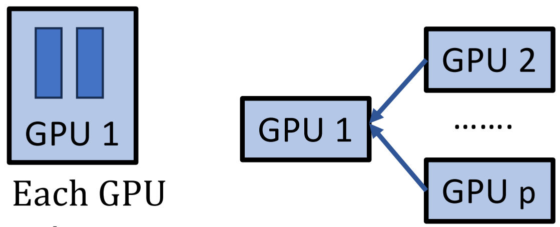


For GPU0, $\frac{M}{p}$ for single key, $2\frac{(p-1)M}{p}$ for KVs

For GPU0, $2\frac{(p-1)M}{p}L$ for total model

For all GPUs, total volume per step $\boxed{2L(p-1)M}$

AsyncDiff

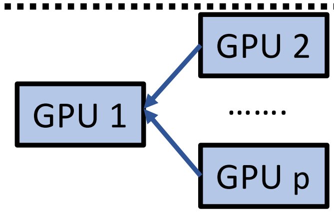


Each GPU contains a stage

GPU0/stage0 performs a broadcast: $(p-1)M$

For all GPUs, total volume per step $\boxed{p(p-1)M}$

ParaStep



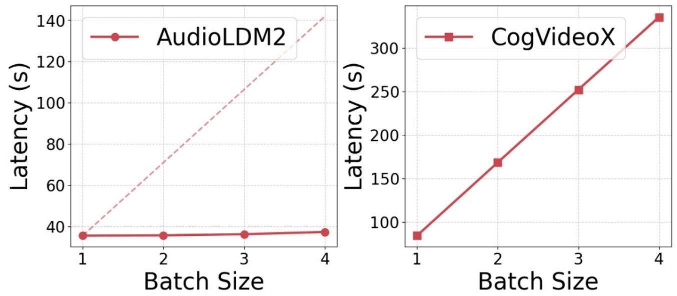
For **p steps**, GPU2…GPU p send predicted noise to GPU0: $(p-1)M$

For **p steps**, GPU0 performs a broadcast: $(p-1)M$

For all GPUs, total volume **per step:** $\boxed{2\frac{(p-1)M}{p}}$

SJTU

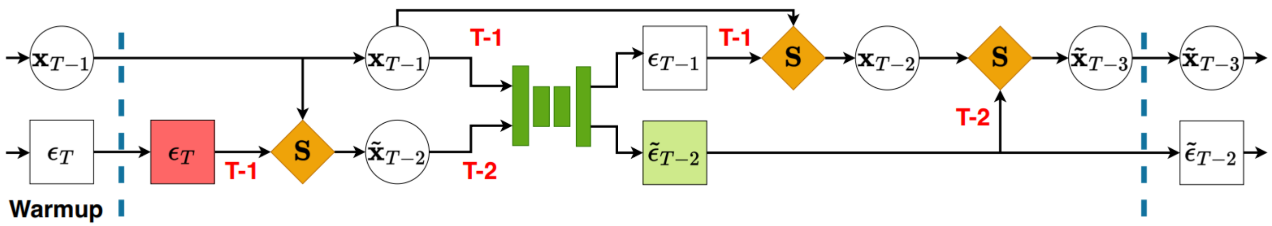
BatchStep: transform parallelism into batched inference



(a) AudioLDM2 (b) CogVideoX

➢ AUDIO: Increasing the batch size does NOT lead to a linear increase in latency — indicating a good batching effect.

➢ VIDEO: Increasing the batch size leads to a linear increase in latency — indicating a poor batching effect.



➢ Transform the parallel execution of the noise predictor into a batched inference process on a single device

➢ The good batching effect contributes to acceleration.

Experiments： Latency and performance

- For Image generation: ParaStep achieves the highest speedup while maintaining competitive quality metrics.

Method	Efficiency		Visual Quality			
	Speedup ↑	Latency (s) ↓	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑
SD3 ($T = 50$)	1	18.75	-	-	-	-
AsyncDiff ($p = 2$)	1.61	11.62	7.11	0.2141	16.47	0.7290
xDiT-Pipe ($p = 2$)	1.50	12.50	6.20	0.1943	16.68	0.7498
ParaStep ($p = 2$)	**1.68**	**11.16**	**5.01**	**0.1362**	**18.61**	**0.8157**

Table 1: Speedup and generation quality on image model SD3, with a resolution of 1440×1440. T is the number of inference steps, and p is the degree of parallelism.

Experiments： Latency and performance

- For Video generation: ParaStep achieves the highest speedup while maintaining competitive quality metrics.

Method	Efficiency		Visual Quality			
	Speedup ↑	Latency (s) ↓	VBench ↑	LPIPS ↓	PSNR ↑	SSIM ↑
SVD (14 frames, 1024×576)						
SVD ($T = 50$)	1	51.04	**42.00**	-	-	-
AsyncDiff ($p = 2$)	1.37	37.36	41.75	0.0790	25.99	0.8366
AsyncDiff ($p = 4$)	1.80	28.43	41.56	0.1285	23.13	0.7599
ParaStep ($p = 2$)	1.69	30.27	41.90	**0.0278**	**33.35**	**0.9347**
ParaStep ($p = 4$)	**2.49**	**20.49**	41.74	0.0732	26.99	0.8433
Latte (16 frames, 512×512)						
Latte ($T = 50$)	1	32.56	73.68	-	-	-
xDiT-Ring ($p = 2$)	1.01	32.18	**73.87**	**0.0424**	32.79	**0.9273**
xDiT-Ring ($p = 4$)	1.07	30.50	73.81	0.0431	**32.80**	0.9266
ParaStep ($p = 2$)	1.43	22.80	73.76	0.0432	32.51	0.9258
ParaStep ($p = 4$)	**1.82**	**17.91**	73.76	0.0533	31.10	0.9115
CogVideoX-2b (45 frames, 512×720)						
CogVideoX ($T = 50$)	1	91.89	76.95	-	-	-
xDiT-Ring ($p = 2$)	0.86	106.68	76.79	0.0570	32.24	0.9284
xDiT-Ring ($p = 4$)	0.98	93.44	76.66	0.0817	29.53	0.9028
ParaStep ($p = 2$)	1.47	62.50	**76.97**	**0.0213**	**37.57**	**0.9642**
ParaStep ($p = 4$)	**1.93**	**47.66**	76.74	0.0359	34.34	0.9505

Experiments： Latency breakdown

- ParaStep exhibits significantly lower communication latency compared to AsyncDiff
- Lower communication directly contributes to its higher overall speedup.

Method	**Total (s)**	**Comp. (s)**	**Comm. (s)**
SVD	51.04	51.04	-
AsyncDiff ($p = 2$)	39.30	30.71	8.59
ParaStep ($p = 2$)	30.52	30.42	0.10
AsyncDiff ($p = 4$)	31.30	21.01	10.29
ParaStep ($p = 4$)	**20.08**	19.94	0.14

Experiments： BatchStep

- ParaStep achieves high speedup with minor quality drop.
- BatchStep achieves competitive speedup without additional computation resources.

Method	Latency (s) ↓	FAD ↓
AudioLDM2 ($T = 200$)	34.80	1.6653
ParaStep ($p = 2$)	18.86	**1.6651**
ParaStep ($p = 4$)	9.86	1.6716
BatchStep ($s = 2$)	17.74	1.6671
BatchStep ($s = 4$)	**9.45**	1.6699

Table 3: Generation latency and FAD on AudioLDM2-large. p is the degree of parallelism, s is the cycle length in BatchStep.

Broader impacts

- Our method can be applied in commercial settings to accelerate compute-intensive diffusion models, because it doesn’t need high bandwidth
- For non-compute-intensive models, our proposed variant BatchStep enables speedup on a single device

Thanks