KAIST

DATA MINING LAB

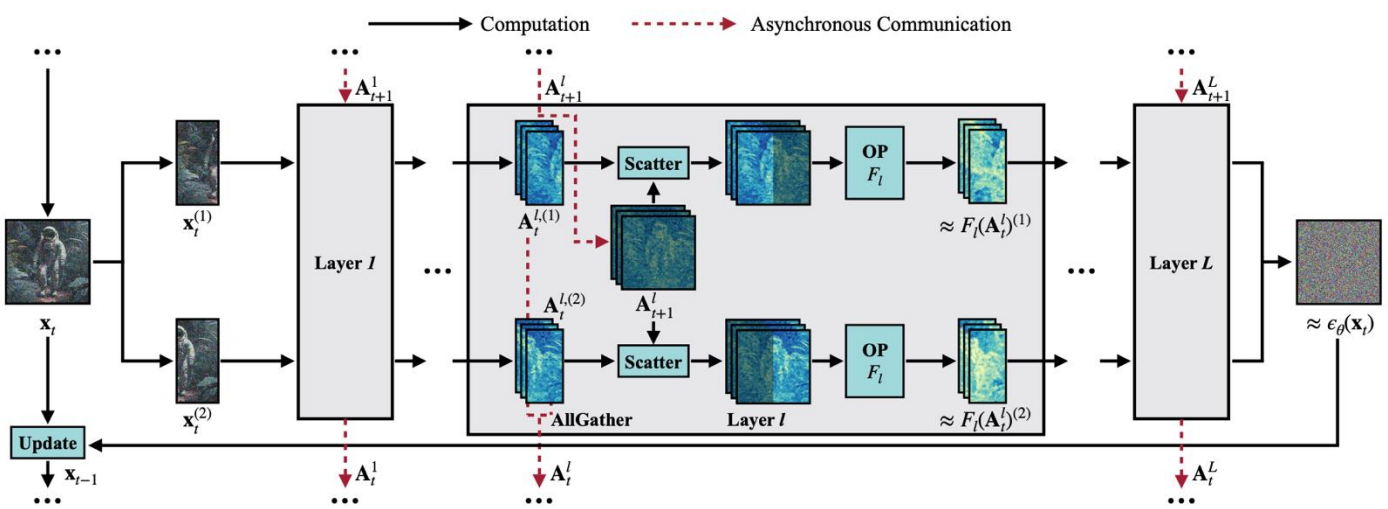
CVPR JUNE 3-7, 2026 DENVER COLORADO

Accelerating Diffusion via Hybrid Data-Pipeline Parallelism Based on Conditional Guidance Scheduling

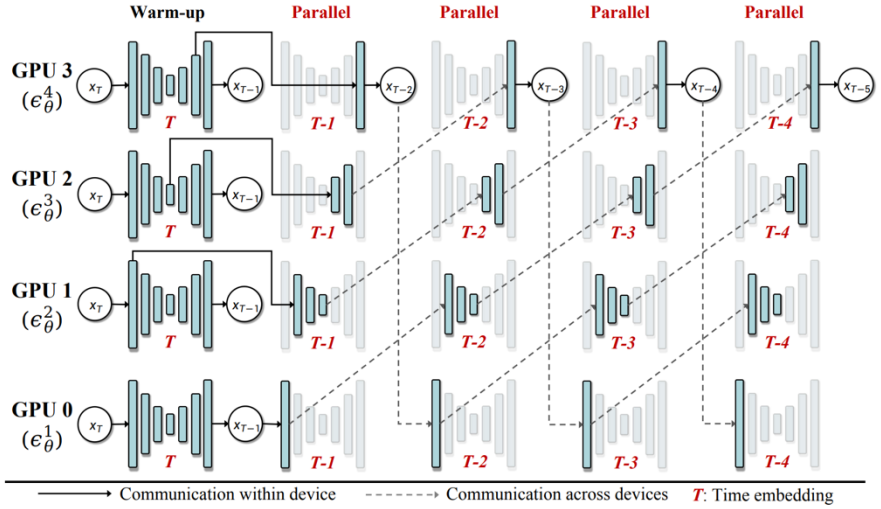
CVPR 2026 poster

Euisoo Jung, Byunghyun Kim, Hyunjin Kim, Seonghye Cho, Jae-Gil Lee

Accelerating Diffusion Inference via Multi-GPU



Overview of Distrifusion [1]



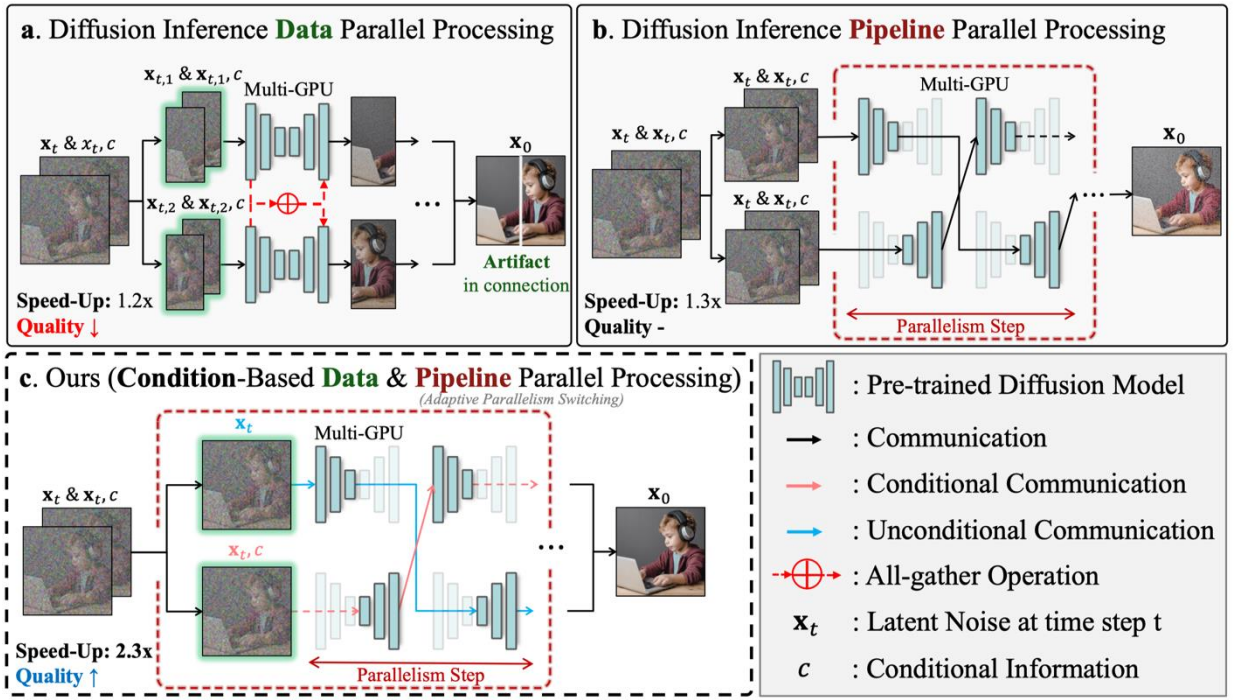
Overview of AsyncDiff [2]

- **Diffusion models** have become the backbone of image and video generation, but their **long inference latency** remains a key bottleneck, motivating extensive acceleration research
- Among these, **multi-GPU distributed inference** has emerged as a promising direction for accelerating generation
- Representative approaches include **Distrifusion** (patch-based data parallelism combined with tensor parallelism) and **AsyncDiff** (pipeline parallelism via asynchronous denoising)

[1] Li, Muyang, et al. "Distrifusion: Distributed parallel inference for high-resolution diffusion models." *CVPR*. 2024.

[2] Chen, Z., Ma, X., Fang, G., Tan, Z., & Wang, X. Asyncdiff: Parallelizing diffusion models by asynchronous denoising. *NeurIPS*. 2024.

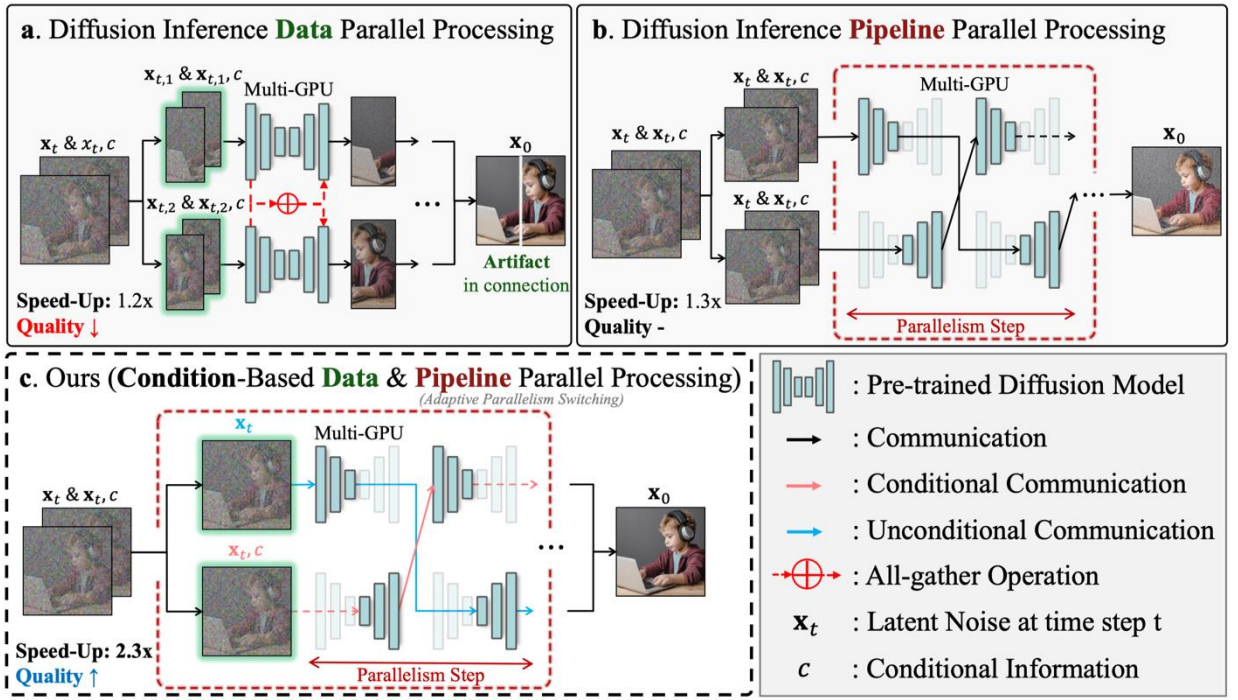
Prior Multi-GPU Inference Works Problems



Comparison of parallel strategies for diffusion inference

- Multi-GPU distributed inference has established itself as a **training-free yet powerful acceleration** approach, but existing methods still suffer from several limitations
  - **Patch-based data parallelism** can degrade output quality due to **artifacts at patch connection boundaries**
  - **Pipeline parallelism** suffers from error accumulation caused by **heavy asynchronous communication**

Prior Multi-GPU Inference Works Problems

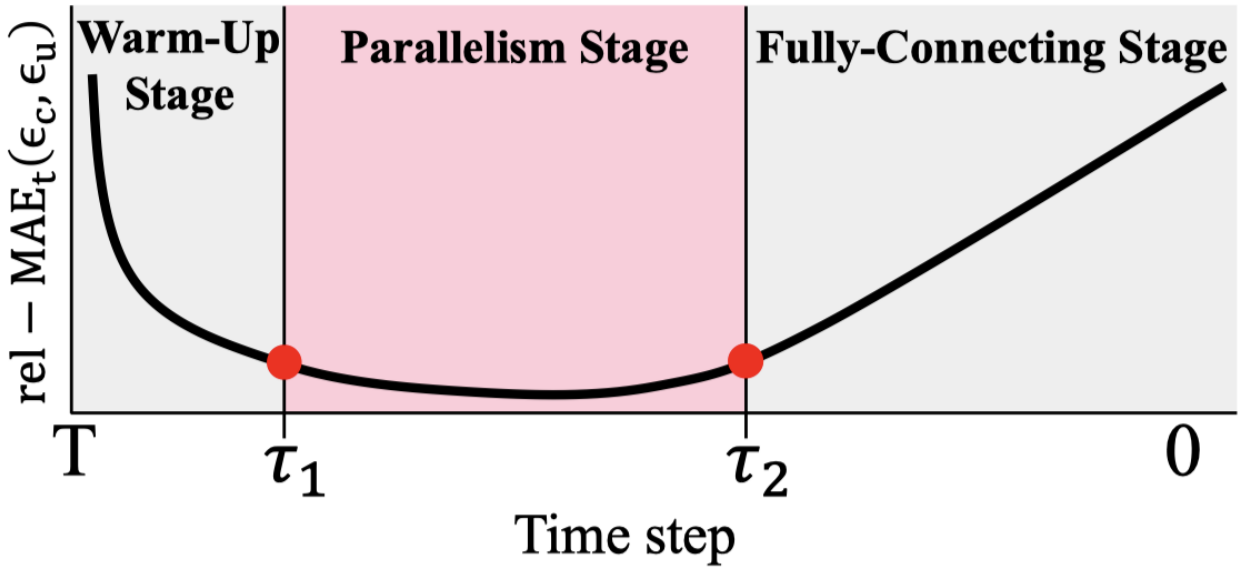


Comparison of parallel strategies for diffusion inference

✅ **Key Insight (TL;DR)** : The conditional & unconditional CFG paths form a natural, artifact-free data split for data parallelism;

combined with pipeline parallelism in the interval where their discrepancy is minimal, they constitute a hybrid parallelism that achieves beyond-linear 2.3× speed-up on 2 GPUs

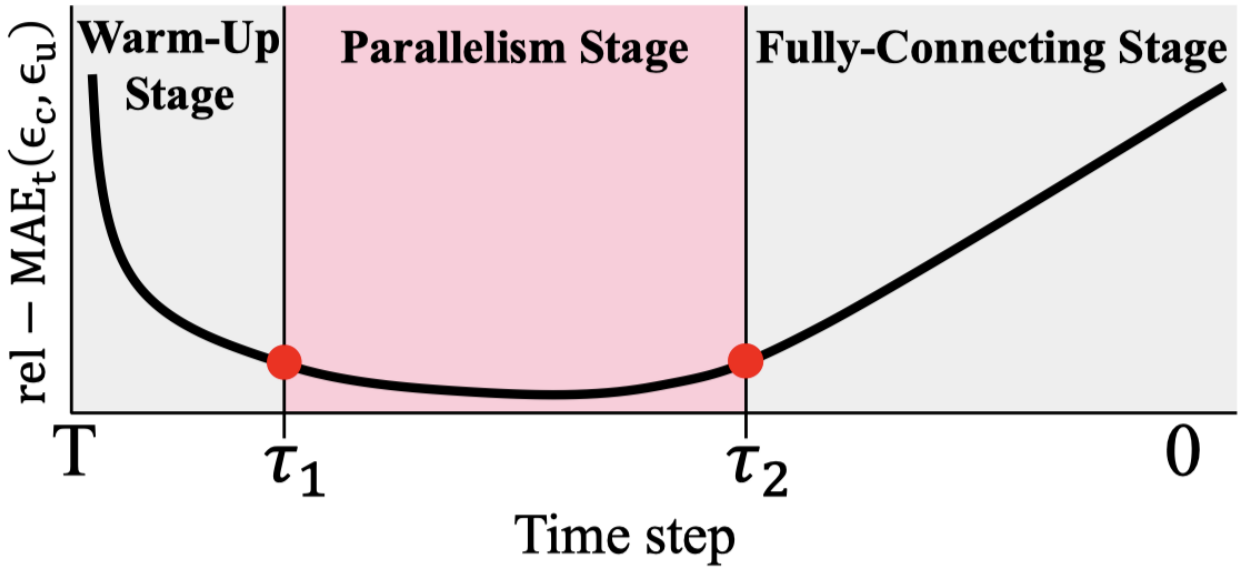
Classifier-Free Guidance (CFG) Analysis



relative − $\mathrm{MAE}_t(\epsilon_c, \epsilon_u)$ curve according to denoising step

- **CFG predicts two noises** at every step: **conditional (ϵ_c)** and **unconditional (ϵ_u)**
  We use their gap to decide when parallelism is safe
- **Denoising discrepancy**: relative − $\mathrm{MAE}_t(\epsilon_c, \epsilon_u) = \mathbb{E}[\|\epsilon_c - \epsilon_u\|_1]$ / $\mathbb{E}[\|\epsilon_u\|_1]$
  → How far the two paths diverge at step t
- Over 5,000 MS-COCO prompts, this curve forms a U-shape, splitting denoising into three stages:
  - **Warm-Up $[T, \tau_1]$: large gap** → branches run independently
  - **Parallelism (τ_1, τ_2): small / stable gap** → **parallelism activated** for max speed-up (branch merged)
  - **Fully-Connecting $[\tau_2, 0]$: gap rises again** → branches split for refinement

Classifier-Free Guidance (CFG) Analysis

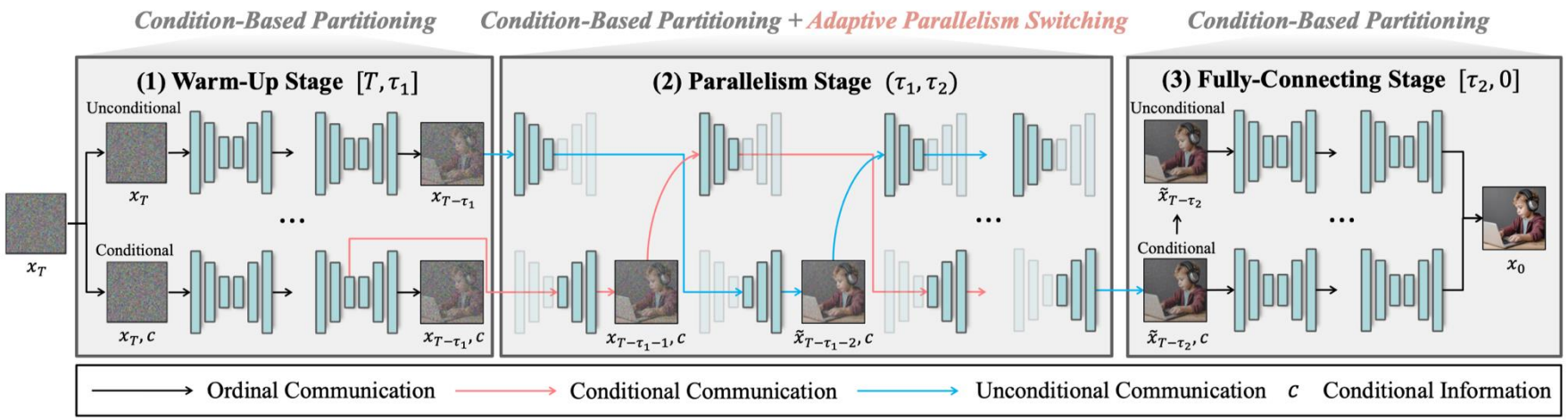


$\mathrm{relative}-\mathrm{MAE}_t(\epsilon_c,\epsilon_u)$ curve according to denoising step

- Over 5,000 MS-COCO prompts, this curve forms a U-shape, splitting denoising into three stages:
  - **Warm-Up $[T,\tau_1]$: large gap** → branches run independently
  - **Parallelism (τ_1,τ_2): small / stable gap** → **parallelism activated** for max speed-up (branch merged)
  - **Fully-Connecting $[\tau_2,0]$: gap rises again** → branches split for refinement

→ Need to find τ_1,τ_2 dynamic & automatically

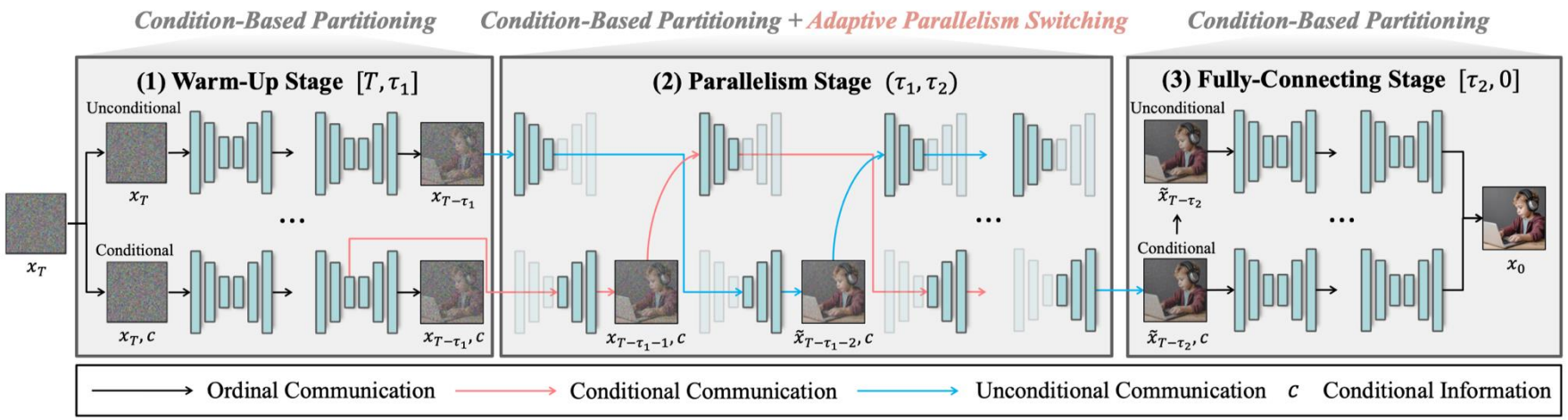
Method: Hybrid Data-Pipeline Parallelism



Overview of proposed hybrid data-pipeline parallelism

- *Condition-Based Partitioning*
  - Split by CFG branch (condional & uncondinal)
    → Enable data parallelism **w/o artifacts**
- *Adaptive Parallelism Switching*
  - Engage pipeline parallelism only in $[\tau_1, \tau_2]$ where two branches similar
    → Enable **hybrid data-pipeline parallelism** w/o minimize the quality degradation

Method: Hybrid Data-Pipeline Parallelism



Overview of proposed hybrid data-pipeline parallelism

- Determining τ_1
  1. $\mathrm{M_t} = \mathrm{relative} - \mathrm{MAE_t}(\epsilon_c, \epsilon_u)$
  2. $\mathrm{G_t} = \frac{\mathrm{M_t} - \mathrm{M_{t-L}}}{\mathrm{L}}$
  3. $\tau_1^{'} = \min\{\mathrm{t} \mid 0 \le \mathrm{G_t} < \mathrm{g_{slope}}\}$
  4. $\tau_1 = \min(\tau_1^{'}, \tau_{\mathrm{cap}})$

- Determining τ_2
  1. $\tau_2 = \tau_1 + \mathrm{k}, \quad \mathrm{k} \in \mathbb{N}, 1 \le \mathrm{k} < \mathrm{T} - \tau_1$

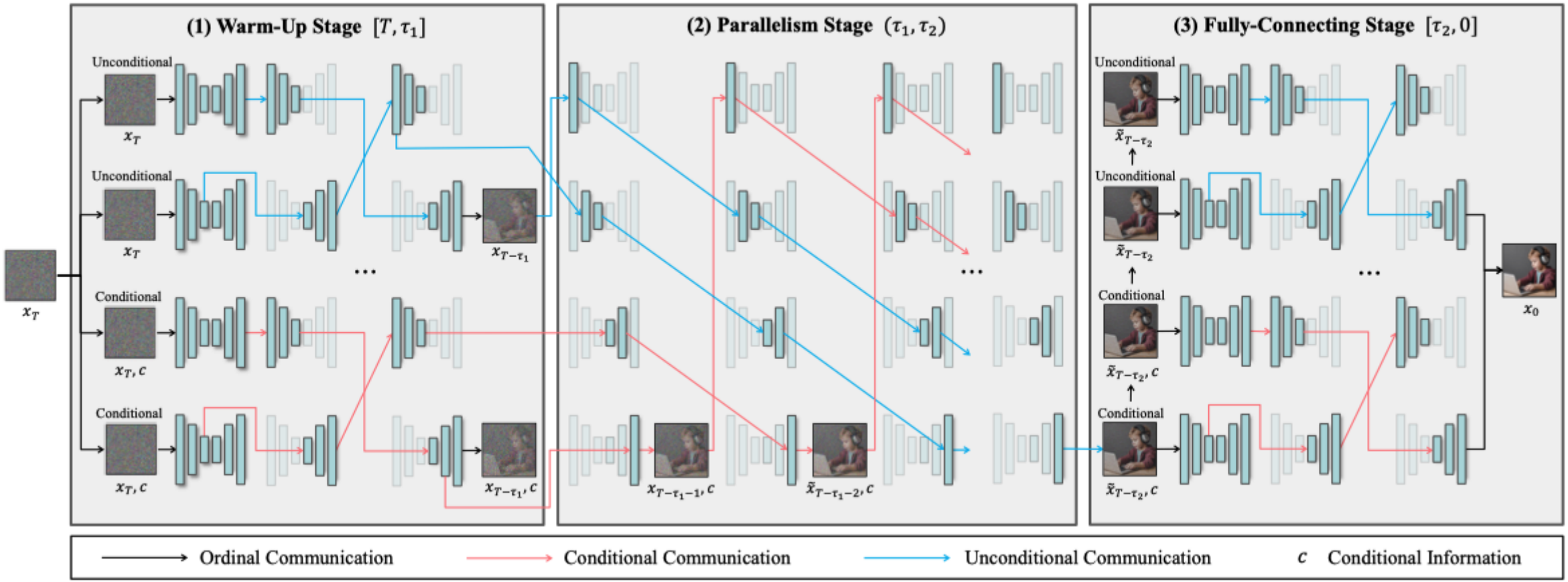
Detailed Algorithm

Algorithm 1 Adaptive Parallelism Switching via Denoising Discrepancy

Require: latent noise $\mathbf{x}_t$, prompt c, steps T, window L, slope threshold g, safety-cap τ_{cap}, interval k

$\tau_1, \tau_2 \leftarrow \varnothing$
for $t = T, T-1, \ldots, 1$ **do**
  $\epsilon_c, \epsilon_u \leftarrow \epsilon_\theta(\mathbf{x}_t, c, t),\ \epsilon_\theta(\mathbf{x}_t, t)$
  $M_t \leftarrow \frac{\mathbb{E}_{x,\epsilon}\|\epsilon_c - \epsilon_u\|_1}{\mathbb{E}_{x,\epsilon}\|\epsilon_u\|_1}$ ▷ rel-MAE$_t(\epsilon_c, \epsilon_u)$
  $G_t = \frac{M_t - M_{t-L}}{L}$
  if $\tau_1 = \varnothing$ **and** $t > L$ **and** $0 \le G_t < g$ **then**
    $\tau_1 \leftarrow \min(t, \tau_{\text{cap}});\quad \tau_2 \leftarrow \tau_1 + k$
  Denoise:
  if $t \ge \tau_1$ **then**
    WARM-UP
  else if $t > \tau_2$ **then**
    PARALLELISM
  else
    FULLY-CONNECTING
  end if
  $x_{t-1} \leftarrow$ STEP DENOISE$(\mathbf{x}_t, \epsilon_c, \epsilon_u, t)$
end for
return $x_0,\ (\tau_1, \tau_2)$

Method: Hybrid Data-Pipeline Parallelism



Overview of proposed hybrid data-pipeline parallelism in over 2 GPUs

Main Results: Quantitative Results

Base Model	Devices	Methods	Latency (s) ↓	Speed-Up ↑	Comm. (GB) ↓	FID ↓		LPIPS ↓		PSNR ↑	
						w/ G. T.	w/ Orig.	w/ G. T.	w/ Orig.	w/ G. T.	w/ Orig.
Stable Diffusion XL	**1**	Original Model	16.49	-	-	23.977	-	0.797	-	9.618	-
	2	DistriFusion [12]	13.53	1.22×	0.525	24.164	4.864	0.7978	0.146	9.597	24.634
		AsyncDiff [2] (stride=1)	12.54	1.31×	9.830	23.941	4.103	0.797	0.108	9.586	26.387
		Ours (k=5)	**7.12**	**2.31**×	**0.516**	**23.831**	**4.100**	**0.796**	**0.107**	**9.665**	**26.640**
	4	DistriFusion	7.13	2.31×	**0.486**	24.197	5.772	0.798	0.183	9.578	23.046
		AsyncDiff (stride=1)	9.45	1.74×	10.531	24.225	5.829	0.795	0.147	9.626	24.776
		AsyncDiff (stride=2)	6.57	2.51×	4.795	24.140	6.572	0.803	0.192	9.557	23.070
		Ours (k=5)	**4.83**	**3.41**×	0.751	**24.087**	**5.544**	**0.788**	**0.113**	**9.888**	**25.233**
Stable Diffusion 3	**1**	Original Model	19.36	-	-	33.433	-	0.810	-	8.086	-
	2	AsyncDiff [2] (stride=1)	9.82	1.97×	1.290	33.379	2.032	0.813	0.052	8.155	27.812
		xDiT-Ring [4]	14.31	1.35×	121.646	33.356	1.909	0.809	0.047	8.085	27.857
		Parastep [33]	9.98	1.94×	**0.032**	33.340	3.350	0.810	0.112	8.091	22.917
		Compact-2bit [21]	13.96	1.39×	9.049	33.560	4.712	0.810	0.196	7.999	19.242
		Ours (k=5)	**9.33**	**2.07**×	0.189	**33.322**	**1.878**	**0.780**	**0.046**	**8.229**	**27.875**
	4	AsyncDiff (stride=1)	6.51	2.97×	3.774	33.909	3.458	0.887	0.177	8.033	26.540
		AsyncDiff (stride=2)	**4.86**	**3.98**×	1.241	34.434	8.939	0.895	0.272	7.956	21.087
		xDiT-Ring	17.96	1.08×	130.792	33.374	2.232	0.861	0.127	8.001	27.101
		Parastep	6.29	3.08×	**0.038**	33.211	2.581	0.910	0.134	7.990	27.089
		Compact-2bit	9.25	2.09×	13.593	33.420	5.461	**0.811**	0.248	7.926	17.490
		Ours (k=5)	5.53	3.50×	0.572	**33.113**	**2.109**	0.857	**0.122**	**8.046**	**27.110**

Main quantitative results

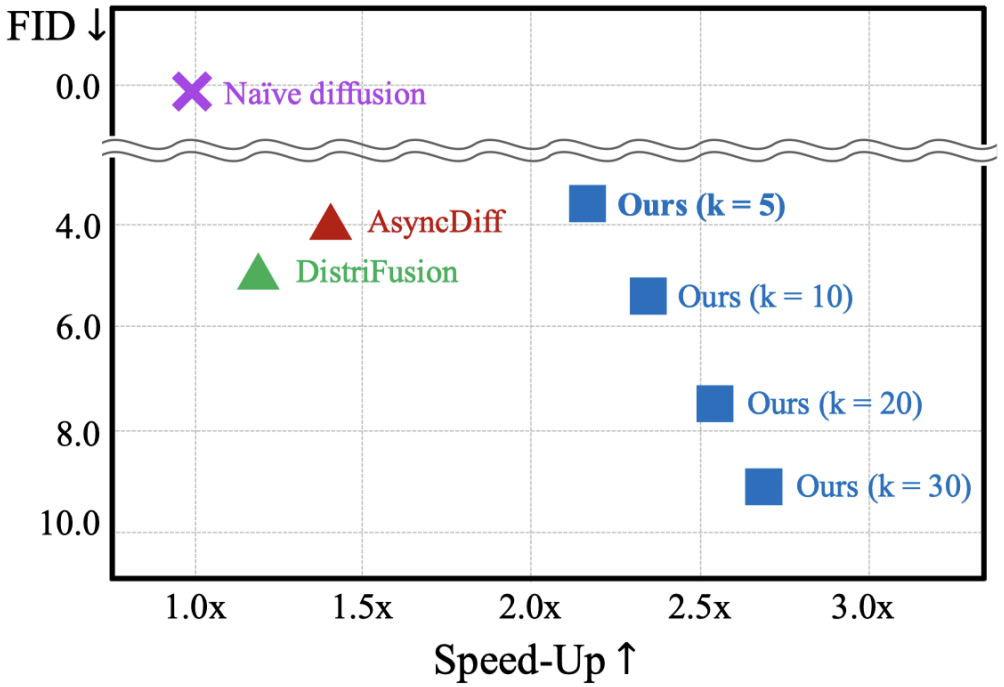
- On **2 GPUs**, Ours achieves **2.31× (SDXL)** and **2.07× (SD3)** speed-up
  → outperforming DistriFusion and AsyncDiff while preserving FID/LPIPS/PSNR
- **Communication cost** is up to **19.6×** lower than AsyncDiff, and the gain scales to **3.41× (SDXL)** and **3.50× (SD3)** on **4 GPUs** → confirming generality across **U-Net and DiT** architectures.

Main Results: Qualitative Results

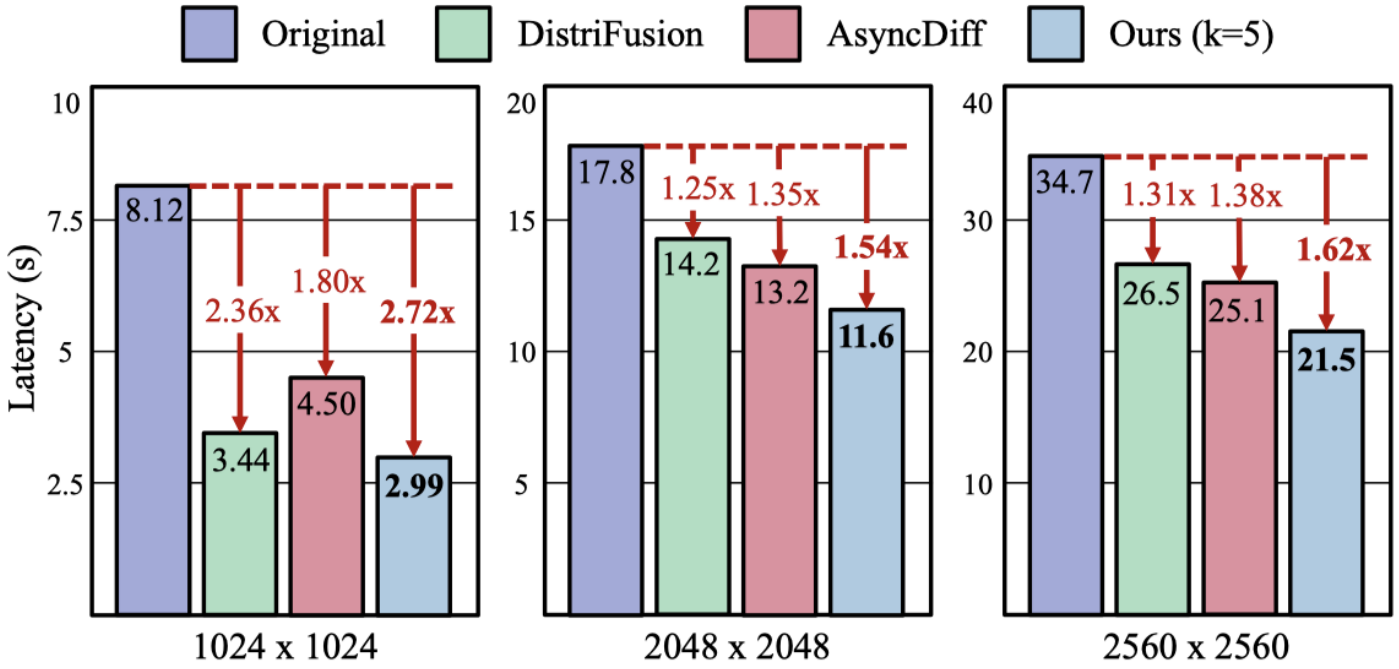
Main qualitative results

- DistriFusion shows boundary **artifacts** and AsyncDiff exhibits **spatial inconsistency**, while Ours preserves **global coherence** and **fine details** closest to the original
- Despite being the **fastest (2.32×)**, Ours also yields the **best FID (4.14)**
  → demonstrating that condition-based partitioning avoids the quality trade-off seen in prior methods

Sensitivity Analysis: k Values & High Resolution



Visualization of different k values



Results of high resolution tasks (H200 GPUs)

- **k controls the speed–quality trade-off**: smaller k preserves fidelity, larger k boosts speed; **k=5** sits on the Pareto frontier, dominating both prior methods
- **High-resolution scalability**: Ours maintains the best acceleration across **1024 x 1024 (2.72×)**, **2048 x 2048 (1.54×)**, and **2560 x 2560 (1.62×)** on H200 GPUs

Summary

- **Method**: Hybrid parallelism combining condition-based partitioning (ϵ_c, ϵ_u as two data paths) with adaptive parallelism switching (τ_1, τ_2 set by denoising discrepancy)
- **Results**: 2.31× (SDXL) and 2.07× (SD3) speed-up on 2 GPUs with preserved fidelity, scaling to 3.41× / 3.50× on 4 GPUs
- **Contribution**: A **unified, training-free** parallelism paradigm that generalizes across U-Net, DiT, and high-resolution settings with minimal communication cost.

https://github.com/kaist-dmlab/Hybridiff

Codes are available via online & right QR code

Thank you!