DiTango: Cost-Effective Parallel Diffusion Generation with Selective Attention State Reuse

Yuyang Chen*
chen-yy20@sjtu.edu.cn
Shanghai Jiao Tong University,
Shanghai AI Laboratory
Shanghai, China

Runxin Zhong
zhongrx24@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Zan Zong
zongzan@ustb.edu.cn
University of Science and Technology Beijing
Beijing, China

Hengjie Li
lihengjie@pjlab.org.cn
Shanghai AI Laboratory
Shanghai, China

Yuyang Jin
jinyuyang@tsinghua.edu.cn
Tsinghua University
Beijing, China

Jidong Zhai
zhaijidong@tsinghua.edu.cn
Tsinghua University
Beijing, China

Abstract

Recent advances in AI-generated content have driven widespread adoption of Diffusion Transformers (DiTs) for high-resolution, long-duration content generation. While parallelization techniques accelerate diffusion inference, they face significant scalability challenges due to excessive communication overhead in multi-node environments.

We observe that sequence partitions in Context Parallelism (CP) exhibit distinct heterogeneity: spatially proximate partitions contribute more significantly to attention computation results. By mapping this heterogeneous pattern to hierarchical communication topology, we can access high-contribution partitions with reduced communication cost. This insight motivates our novel selective attention state mechanism that strategically balances partial attention computation and historical result reuse across denoising steps.

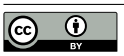
We present DiTango[1], an efficient parallel framework for DiT generation. DiTango features an anchor-guided state selection planner that optimizes computation-reuse decisions for each partition, complemented by a runtime that orchestrates efficient state-centric operations. This design achieves superior system efficiency while preserving generation quality.

Experimental evaluation on popular diffusion models demonstrates that DiTango achieves up to 1.9× end-to-end and 3.2× attention speedup with near-linear scaling in multi-node settings, while maintaining generation quality comparable to state-of-the-art approaches.

CCS Concepts

• **Computer systems organization → Parallel architectures**; • **Computing methodologies → Machine learning**.

*Work completed during an internship at Shanghai AI Laboratory.
[1]DiTango is available at https://github.com/thu-pacman/Chitu-Diffusion



HPDC '26, Cleveland, OH, USA

ACM ISBN 979-8-4007-2640-8/2026/07
https://doi.org/10.1145/3806645.3807581

Keywords

Diffusion, Parallelism, Long Context

ACM Reference Format:
Yuyang Chen, Runxin Zhong, Zan Zong, Hengjie Li, Yuyang Jin, and Jidong Zhai. 2026. DiTango: Cost-Effective Parallel Diffusion Generation with Selective Attention State Reuse. In *The 35th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '26), July 13–16, 2026, Cleveland, OH, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3806645.3807581

1 Introduction

Diffusion models have become a foundational paradigm for AI-generated content (AIGC), underpinning state-of-the-art image and video synthesis systems. Driven by the trend toward long-sequence generation (e.g., longer videos with higher spatial and temporal resolution), Diffusion Transformers (DiTs) [23] have emerged as a dominant backbone, enabling rapid progress in both closed-source models (e.g., Sora [1], Kling [13]) and open-source models (e.g., Wan 2.2 [28], HunyuanVideo [12], CogVideoX [32]).

Despite the impressive progress in generation quality, DiT-based generators remain prohibitively slow at inference time. For example, Wan2.2 [28] takes over one hour on a single NVIDIA A100 GPU to generate a 5-second 720p video. This latency is largely dominated by the full attention computation in DiT architectures: unlike autoregressive LLMs that amortize attention costs through KV caching, DiT must compute full attention over the entire spatiotemporal sequence in each denoising step.

From a systems perspective, parallel inference offers a promising approach to accelerate DiT-based diffusion generation. However, Data Parallelism (DP) and Pipeline Parallelism (PP) provide limited benefits due to the inherent single-sample nature of diffusion generation (batch size = 1). Consequently, practical deployments primarily leverage two strategies: Tensor Parallelism (TP)[25] and Context Parallelism (CP)[8, 11, 17]. TP shards model weights across GPUs and employs collective operations to assemble intermediate results, while CP[8, 11, 17] partitions tokens along the sequence dimension with communication for remote Key-Value partition access during attention. When attention computation dominates the runtime, these data transfers can be effectively overlapped with independent computation.

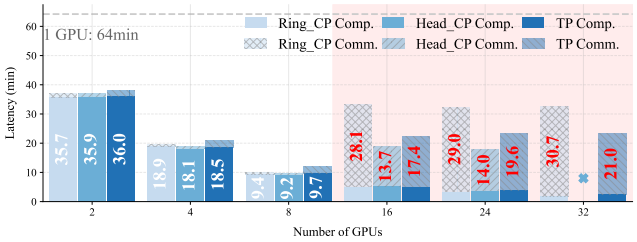


Figure 1: Parallel inference performance breakdown for multi-GPU deployments with 8 GPUs per node.

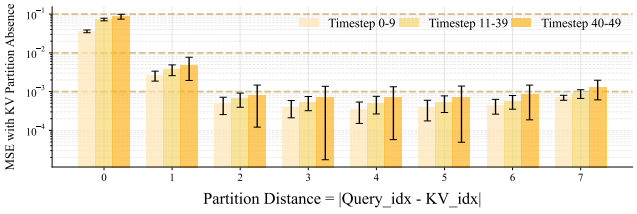


Figure 2: Attention computation contribution is highly unequal with distance-based decay across different CP partitions.

However, these parallel inference solutions still fall short of practical requirements. Our profiling of Wan-14B[28] generating a 81-frame 720p video (75,600 tokens) on NVIDIA H20s reveals that even with 8 GPUs in a single node, generation takes around 10 minutes. More critically, scaling to multiple nodes often degrades performance due to communication bottlenecks, as shown in Fig. 1.

This performance degradation stems from the communication-intensive nature of both TP and CP. TP requires frequent collective operations for sharded model weights, while CP variants either need All-to-All exchanges (Head-CP) or multiple rounds of P2P transfers (Ring-CP) to access global Key-Value sequence. In multi-node settings, limited inter-node bandwidth severely constrains both collective and ring communication patterns. Furthermore, the compute-communication overlap that proves effective within a single node becomes significantly less efficient across nodes, exacerbating the end-to-end performance impact.

To address the performance bottleneck in parallel inference, we identify a key pattern in CP-partitioned sequences for Diffusion inference: *attention contributions from different partitions exhibit strong spatial locality*. As shown in Fig. 2, we conduct a systematic analysis on different models and prompts under CP size=8 by measuring the Mean Squared Error (MSE) of attention outputs when excluding specific KV partitions. Our findings reveal that partitions contribute highly unequally to the final attention results. For instance, excluding the nearest partition (distance= 0) yields MSE of 10^{-2}, while distant partitions (distance$\geq$ 2) contribute merely 10^{-4}—a difference of 2-3 orders of magnitude. Attention contributions systematically decay with spatial distance that KV Partitions closer to the query (smaller |Q_idx − KV_idx|) consistently dominate the computation, demonstrating clear spatial locality. This spatial locality persists across diverse models, different DiT layers and various input prompts—indicating it is an intrinsic property of diffusion attention rather than an artifact of specific configurations.

This observation unveils a compelling optimization opportunity: by aligning locality priorities with the distributed system topology—mapping high-contribution partitions to nodes with efficient intra-node communication while placing low-contribution ones across nodes—we can strategically compute only the most cost-effective partitions and reuse historical results for less critical ones. This contribution-aware approach enables substantial efficiency gains without sacrificing generation quality.

In this paper, we propose DiTango, an efficient and scalable generation system in distributed DiT inference. DiTango exploits the heterogeneous contribution in CP-partitioned attention through a selective computation strategy: it prioritizes cost-effective KV pairs with high contribution and low communication costs, while reusing attention states (partial attention computation results) from previous denoising steps for less critical computations. As the two-partition example illustrated in Fig.3, the conventional full attention (Fig.3a) performs costly communication to access remote KV partition in step 1, despite their trivial contribution to the final result. In contrast, DiTango (Fig. 3b) selectively computes crucial partition locally while bypassing expensive communication by reusing attention states from previous denoising steps for trivial contributions. This design elegantly preserves the mathematical properties of attention computation while significantly reducing communication overhead.

However, implementing efficient attention state reuse presents two fundamental challenges:

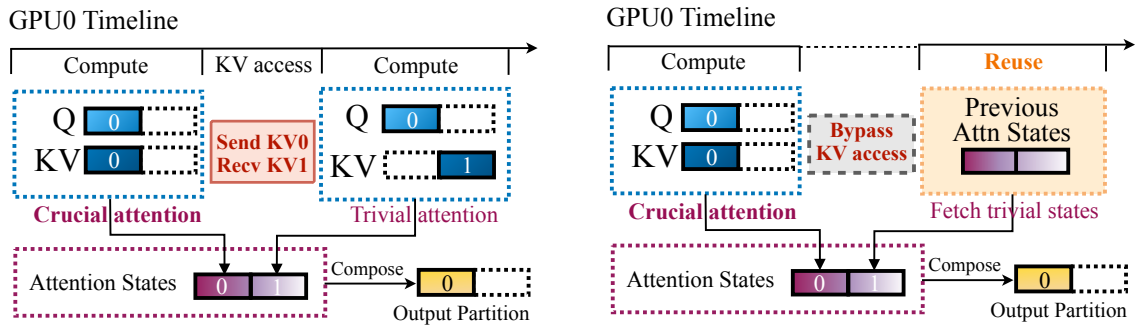
(1) **Complex Selection Space** - The reuse decision for each state must balance multiple factors: computational contribution, communication overhead, and reuse error. Making optimal selections across numerous states to achieve both good performance and quality becomes jointly complex.
(2) **Irregular Access Patterns** - Uneven selection disrupts the structured pattern in context parallelism, resulting in interleaved local memory accesses and heterogeneous communications that are inherently difficult to parallelize.

To address these challenges, DiTango introduces two key components:

(1) **Anchor-guided Selection Planner** - By theoretically modeling attention state errors and designing anchor step-guided error prediction, we precisely identify states with minimal impact for reuse to optimize the performance-quality trade-off.
(2) **State-centric Parallel Runtime** - We schedule attention states with similar characteristics in groups and design a series of state-centric manipulations for efficient computation and retrieval. A specialized pipeline orchestrates these operations to maximize parallel computation and communication efficiency.

Our contributions are summarized as follows:

- We identify and analyze the spatial locality in attention contributions during DiT inference, revealing an opportunity to selectively reuse attention states based on their contribution significance and communication costs.



(a) Full-sequence block-wise attention (baseline): KV access incurs heavy communication yet contributes little.

(b) DiTango with selective sequence computation: reuse attention states from previous denoising steps to bypass expensive operations

Figure 3: Comparison between full-sequence attention and DiTango with selective computation via state reuse.

- We design and implement DiTango, a high-performance generation system that leverages theoretical error modeling to make precise reuse decisions while maintaining efficient execution through group-wise state management.
- Extensive evaluations on Wan2.1 [28] and HunyaunVideo [12] demonstrate that DiTango achieves near-linear scalability across 32 GPU cards, delivering up to 1.9× end-to-end speedup and 3.2× core attention speedup while maintaining superior generation quality compared to state-of-the-art frameworks [7, 15, 27, 38].

2 Background and Motivation

2.1 Background

2.1.1 Diffusion Transformers. Diffusion Transformers (DiTs)[23] generate content through an iterative denoising process, typically requiring 20-50 steps. In each step, the same transformer model processes latent representations to gradually refine them from random noise to high-quality content. Unlike traditional U-Net architectures, DiTs leverage transformer-based designs that excel at handling long sequences through their attention mechanisms. Recent DiT architectures[12, 20, 26, 28, 32] have widely adopted 3D full attention, which unifies spatial and temporal dimensions into a single sequence for processing. While this unified attention approach significantly enhances generation quality and temporal consistency, attention operations dominate computational cost, consuming over 70% of total generation time across all denoising steps.

2.1.2 Parallel Methods for Diffusion Generation. Not all parallel methods suit DiT inference. Data Parallelism (DP) and Pipeline Parallelism (PP) are ineffective for diffusion generation due to single-sample inference patterns that prevent batch-level parallelization. This leaves two primary strategies for DiT acceleration:

As shown in Fig. 4(a), **Tensor Parallelism** (TP) shards model weights across devices. Linear layers require All-Reduce operations to aggregate partial results, while attention modules need All-Gather communication to gather head-wise results. Though memory-efficient, TP suffers from frequent synchronization overhead.

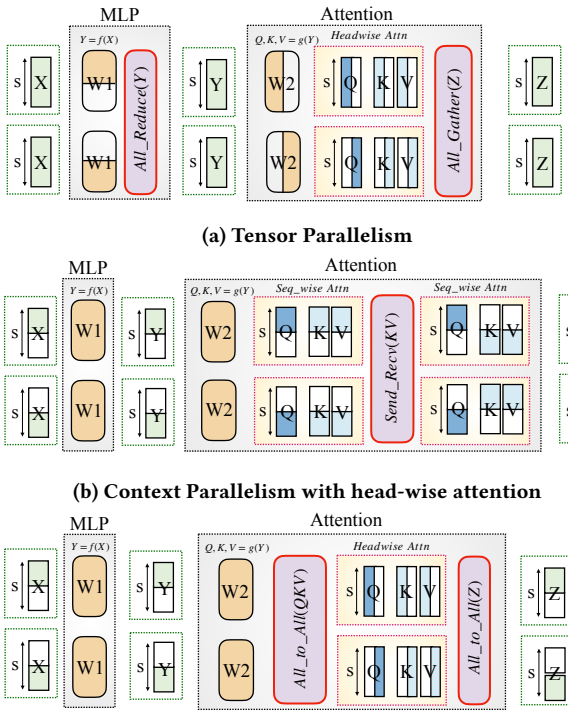
Context Parallelism (CP) better addresses long-sequence generation by partitioning sequences across devices, eliminating communication in linear layers while only requiring Key-Value (KV) remote access for attention. Two main CP variants have emerged: Head-CP [11] (Fig.4(b)) employs all-to-all communication to redistributes tensor layouts, enabling localized head computation. Ring-CP[17] (Fig.4(c)) maintains sequence-wise partitioning but exchanges KV blocks through ring communication, allowing each GPU to access the full sequence progressively. Recent work like Unified Sequence Parallelism [8] provides frameworks to dynamically switch between these strategies based on workload characteristics.

2.1.3 Diffusion Modules Feature Reuse. Feature reuse has emerged as a popular optimization technique in diffusion models, leveraging the observation that consecutive denoising steps often produce similar intermediate features. This similarity enables training-free but lossy acceleration through strategic caching and reuse of features from previous steps, effectively bypassing certain computational dependencies. Different approaches target various aspects of the model: Delta-DiT [2] reuses layer-wise computational features, Pyramid Attention Broadcast (PAB)[38] focuses on attention output reuse, while TeaCache[15] and TaylorSeer[18] dynamically reuse DiT's overall outputs based on input patterns. However, feature reuse inevitably introduces approximation errors, making the design of reuse strategies crucial for maintaining generation quality.

2.2 Motivation

2.2.1 Limitation: The Scalability-Quality Dilemma. Recent generation frameworks have pursued higher performance by combining parallel methods with feature reuse strategies. For example, VideoSys [27] integrates Dynamic-SP [37] parallelization with PAB [38]-based attention output reuse, while SGLang-Diffusion [39] combines Unified-SP [8] parallelization with Cache-DiT's TaylorSeer [18]-based reuse mechanism. These approaches demonstrate impressive acceleration within single-node environments. However, their performance deteriorates significantly in multi-node scenarios due to limited cross-node communication bandwidth.

This cross-node communication bottleneck creates a fundamental dilemma: to maintain reasonable performance scaling, these



(a) Tensor Parallelism

(b) Context Parallelism with head-wise attention

(c) Context Parallelism with sequence-wise attention

Figure 4: Parallel strategies for DiT inference.

frameworks must adopt more aggressive feature reuse strategies to reduce communication overhead. However, such aggressive reuse inevitably leads to substantial quality degradation. Conversely, attempting to preserve generation quality by limiting feature reuse results in poor scaling efficiency due to increased cross-node communication. This inherent trade-off between distributed scalability and generation quality highlights the limitations of treating parallelization and feature reuse as independent optimizations, suggesting the need for a more integrated approach to distributed generation acceleration.

2.2.2 Analysis: Spatial Locality of Partition Contribution. To understand this challenge, we analyze the computation contribution of 16 partitions by measuring output differences when omitting specific KV computations, averaged across DiT layers and timesteps using 50 prompts on Wan2.1. Fig. 5(a) reveals a striking pattern of spatial locality: *computational importance strongly correlates with spatial proximity*. Query-Key pairs from nearby partitions contribute significantly more to attention outputs, with importance diminishing as positional distance increases. This spatial locality pattern, consistent across different models and inputs, offers a natural opportunity for optimization when considering system topology.

Theoretically, this pattern demonstrates how fine-grained token-level attention sparsity studied in previous works [30], manifests as a simpler distance-dependent pattern when observed at the partition level.

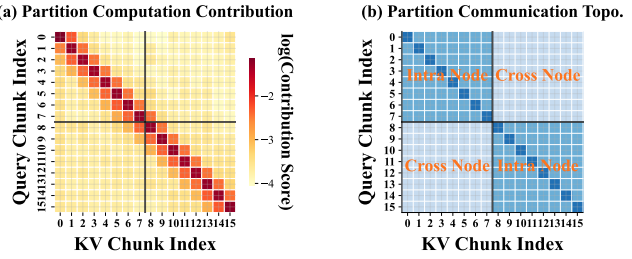


Figure 5: Spatial locality of CP attention in typical cluster: (a) KV partition contribution scores and (b) communication costs across partitions, demonstrating natural alignment between computational importance and system topology.

2.2.3 Insight: System-Pattern Alignment. This computational pattern naturally aligns with the hierarchical communication topology in distributed systems:

- **Local**: KV stored in local memory with negligible access overhead
- **Intra-node**: KV from different GPUs within the same node via high-bandwidth NVLink
- **Cross-node**: KV from remote nodes via low-bandwidth InfiniBand, incurring substantial cost

By mapping spatially adjacent partitions to hardware units with efficient communication, we can achieve perfect alignment between computation importance and communication efficiency. This alignment presents a compelling opportunity: *prioritize high-importance, low-cost computations while replacing expensive operations of minimal contribution with historical results*. This insight enables us to explore previously unreachable optimization space in the scalability-quality trade-off: by prioritizing computations with high contribution-to-cost ratios while reusing historical results for expensive yet less critical operations, we can maximize communication efficiency with minimal quality impact.

A straightforward approach would be caching KV partitions from previous steps. However, this faces a critical **memory bottleneck**: storing all KV partitions requires $2 \times L \times H \times D \times N \times C$ memory, where L=sequence length, H=heads, D=hidden size, N=layers, and C=CFG batch size. For Wan-14B ($2 \times 75600 \times 40 \times 128 \times 40 \times 2$ in BF16), this reaches 58GB per GPU, causing OOM when combined with model weights and activations.

2.2.4 Solution: Attention State Reuse. To leverage this system-pattern alignment while addressing the memory challenge, we introduce *attention state*, commonly used in partitioned parallel attention computation [5, 16, 17, 33], as our optimization medium. For sequence partition i, an attention state $\mathrm{AS}_t(i)$ at timestep t comprises output $\mathbf{OUT}_t(i)$ and log-sum-exp $\mathbf{LSE}_t(i)$:

$$\mathrm{AS}_t(i) = \begin{bmatrix} \mathbf{OUT}_t(i) \\ \mathbf{LSE}_t(i) \end{bmatrix}, \quad \begin{aligned} \mathbf{LSE}_t(i) &= \log \sum_{j \in \mathcal{I}_i} \exp(\mathbf{q}_t \cdot \mathbf{k}_j) \\ \mathbf{OUT}_t(i) &= \sum_{j \in \mathcal{I}_i} \frac{\exp(\mathbf{q}_t \cdot \mathbf{k}_j)}{\exp(\mathbf{LSE}_t(i))} \mathbf{v}_j \end{aligned} \tag{1}$$

where $\mathbf{q}$, $\mathbf{k}$, $\mathbf{v}$ are query, key, and value vectors.

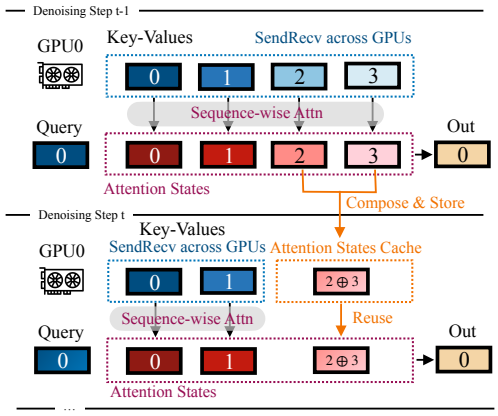


Figure 6: Selective attention state reuse mechanism (CP=4, rank=0).

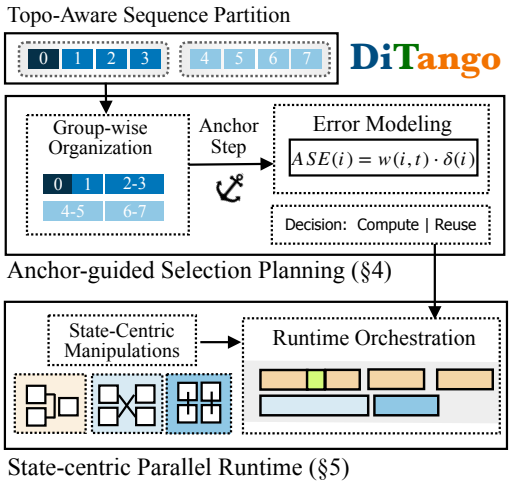


Figure 7: DiTango Overview

A key property of attention states is their **composability**—they can be composed associatively and commutatively. For partitions i and j:

$$\mathrm{AS}_t(i) \oplus \mathrm{AS}_t(j) = \left[\frac{e^{\mathrm{LSE}_t(i)}\mathrm{OUT}_t(i) + e^{\mathrm{LSE}_t(j)}\mathrm{OUT}_t(j)}{\log(e^{\mathrm{LSE}_t(i)} + e^{\mathrm{LSE}_t(j)})} \right] \tag{2}$$

This composability enables flexible partition-wise computation and reuse strategies.

Leveraging these properties, we design a selective compute and reuse mechanism. Taking CP=4 and rank=0, as an example (Fig. 6):

In denoising step $t-1$, after computing attention states from partitioned KVs through sequence-wise attention, we strategically compose states from remote partitions ($AS_{t-1}(2)$ and $AS_{t-1}(3)$) and cache the result.

In step t, we only compute fresh states for local and near partitions ($AS_t(0)$ and $AS_t(1)$), while reusing the cached composed state ($AS_{t-1}(2 \oplus 3)$) for remote partitions, effectively bypassing both KV transmission and attention computation overhead.

This attention state-based approach provides two key advantages over naive KV reuse:(1) Enhanced acceleration by eliminating both communication and computation costs for remote partitions. (2) Improved memory efficiency through both compact state representation (half of KV size) and flexible state composition.

3 DiTango overview

We propose DiTango, a parallel inference system that optimizes diffusion generation through selective attention state reuse. As illustrated in Figure 7, DiTango has two key components:

Anchor-guided selection planning (§4). Building on error propagation analysis of attention state computation, DiTango models the accumulated error introduced by reuse and employs periodic anchor steps for online decision-making. At anchor steps, the system performs fresh computation across all partitions to reset error accumulation, while using the observed attention weight distributions to predict reuse errors for subsequent steps. Through group-wise organization—partitioning the sequence into contiguous groups aligned with device topology—and error budget constraints, the planner generates compute/reuse decisions for each group that balance accuracy and efficiency.

State-centric parallel runtime (§5.2). To handle the heterogeneous computation and communication patterns arising from compute/reuse decisions, DiTango decouples attention operations into specialized state-centric primitives: Cross-Group Transfer consolidates required KV partitions across devices via symmetric P2P exchanges; Intra-Group Composition computes fresh attention states through ring-based communication within localized device groups; Dynamic Group Compose adaptively merges cached states when memory pressure is detected. The runtime orchestrates these operations across asynchronous computation and communication streams, leveraging dependency-free reuse operations to mask cross-node communication latency and maximize resource utilization throughout the pipeline.

Together, these components enable DiTango to exploit spatial locality in diffusion attention while maintaining accuracy guarantees and efficient parallel execution across distributed accelerators.

4 Anchor-guided Selection Planning

4.1 Error Modeling

4.1.1 Attention State Error Propagation. We first analyze how errors in attention states propagate through composition. Consider two attention states with errors $\delta\mathbf{OUT}_i$ and $\delta\mathbf{LSE}_i$ for $i \in \{1, 2\}$. Define normalized weights:

$$w_i = \frac{e^{\mathrm{LSE}_i}}{e^{\mathrm{LSE}_1} + e^{\mathrm{LSE}_2}} \tag{3}$$

The composition operation propagates errors as:

$$\begin{aligned} \delta\mathbf{OUT} &= w_1\delta\mathbf{OUT}_1 + w_2\delta\mathbf{OUT}_2 \\ &\quad + w_1 w_2(\mathbf{OUT}_1 - \mathbf{OUT}_2)(\delta\mathbf{LSE}_1 - \delta\mathbf{LSE}_2) \\ \delta\mathbf{LSE} &= w_1\delta\mathbf{LSE}_1 + w_2\delta\mathbf{LSE}_2 \end{aligned} \tag{4}$$

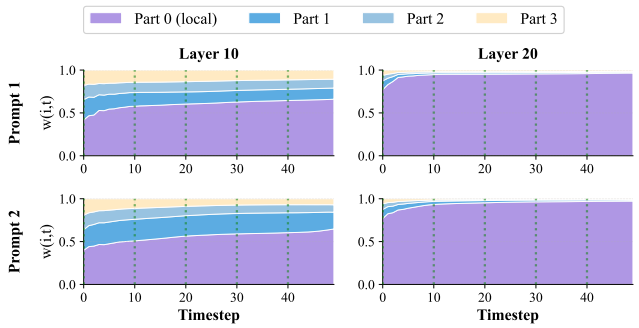


Figure 8: Partition importance $w(i,t)$ with different prompts and layers.

For n partitions, the accumulated output error satisfies:

$$\|\delta\mathbf{OUT}\|_2 \leq \sum_{i=1}^{n} w_i \|\delta\mathbf{OUT}_i\|_2 + \sum_{i<j} w_i w_j \|\mathbf{OUT}_i - \mathbf{OUT}_j\|_2 |\delta\mathbf{LSE}_i - \delta\mathbf{LSE}_j| \tag{5}$$

The second term is second-order and can be safely ignored: (1) the product $w_i w_j$ is small when one partition dominates, and (2) $|\delta\mathbf{LSE}_i - \delta\mathbf{LSE}_j|$ (difference of errors) is typically much smaller than individual errors. This yields:

$$\|\delta\mathbf{OUT}\|_2 \approx \sum_{i=1}^{n} w_i \|\delta\mathbf{OUT}_i\|_2 \tag{6}$$

4.1.2 Online Attention State Error Model. Based on Equation 6, we model the Attention State Error (ASE) of partition i's cached state $\mathrm{AS}_{t_c}(i)$ at current timestep t as:

$$\mathrm{ASE}(i,t,t_c) = w(i,t) \cdot \delta(i,t,t_c) \tag{7}$$

where t_c is the cache timestep, $w(i,t)$ captures partition importance, and $\delta(i,t,t_c)$ models temporal error growth over cache age $\tau = t - t_c$.

Partition Importance Weight. The weight $w(i,t)$ corresponds to w_i in Equation 6. Since computing exact $w(i,t) = e^{\mathrm{LSE}_t(i)} / \sum_k e^{\mathrm{LSE}_t(k)}$ requires full attention, we leverage the empirical observation that LSE distributions remain stable over $\tau \leq$ 5-10 steps (Figure 8).

We update $w(i,t)$ only at **anchor steps** t_a—timesteps with full attention computation:

$$w(i,t) = w(i,t_a) = \frac{e^{\mathrm{LSE}_{t_a}(i)}}{\sum_{k=1}^{n} e^{\mathrm{LSE}_{t_a}(k)}}, \quad \forall t \in (t_a, t_a + \tau_{\max}] \tag{8}$$

Age Penalty Error. Computing $\delta(i,t,t_c) = \|\mathrm{AS}_t(i) - \mathrm{AS}_{t_c}(i)\|_2$ directly contradicts our reuse goal. Instead, we exploit that the *local partition* i_{loc} (the diagonal partition, always computed) serves as a zero-cost error indicator.

At anchor step t_a, we compute the scale ratio:

$$\alpha(i) = \frac{\|\mathrm{AS}_{t_a}(i)\|_2}{\|\mathrm{AS}_{t_a}(i_{\mathrm{loc}})\|_2} \tag{9}$$

For timesteps $t > t_a$, we extrapolate partition i's error from the local state change:

$$\delta(i,t,t_a) = \alpha(i) \cdot \|\mathrm{AS}_t(i_{\mathrm{loc}}) - \mathrm{AS}_{t_a}(i_{\mathrm{loc}})\|_2 \tag{10}$$

Our evaluation guarantees that remote states evolve proportionally to the local state, scaled by their relative magnitudes. Since $\mathrm{AS}_t(i_{\mathrm{loc}})$ is computed every step, Equation 10 incurs nearly zero additional cost while achieving strong empirical accuracy ($R^2 > 0.95$, Section 6.5).

4.2 Group-wise Compute & Reuse Selection

4.2.1 Group-wise organization. With a CP size of p, KV tensors are distributed across GPUs, each corresponding to an attention stare. To achieve an efficient planning granularity, we group spatially adjacent states. This is motivated by the observed spatial locality that adjacent partitions exhibit similar computational contributions and communication costs, leading to consistent selection decisions. Consequently, the p states are divided into p/g groups, each containing g states. In particular, because the local state is handled separately, the intra-node group containing it has a reduced size of $g-1$.

Organizing partitions as group introduce following benefits: Grouping strategy naturally reduces cross-term errors in Equation 5 as spatially adjacent states have similar outputs. It also enables efficient unified scheduling and memory management through group-level attention state composition. Building upon this group organization and our error model, we now present the selection strategy that determines which attention state groups to compute or reuse at each timestep.

4.2.2 Strategy Formulation. For each timestep t, we maintain a set of attention state groups $\mathcal{G} = \{G_0, G_1, ..., G_{m-1}\}$ where $m = \lceil p/g \rceil$. Our goal is to determine for each group G_i:

$$\mathrm{Decision}(G_i,t) = \begin{cases} \texttt{COMPUTE} & \text{compute fresh state} \\ \texttt{REUSE}(t_{c,i}) & \text{reuse cached state from } t_{c,i} \end{cases} \tag{11}$$

where $t_{c,i}$ denotes the last computation timestep for group G_i. The strategy balances quality (bounded error) and efficiency (maximized reuse).

4.2.3 Error-based Selection. For each group G_i at timestep t, we compute its aggregated Attention State Error by summing errors from all partitions within the group:

$$\mathrm{ASE}(G_i,t) = \sum_{j \in G_i} w(j,t) \cdot \delta(j,t,t_{c,i}) \tag{12}$$

Given an error threshold ϵ, the selection decision follows:

$$\mathrm{Decision}(G_i,t) = \begin{cases} \texttt{COMPUTE} & \text{if } \mathrm{ASE}(G_i,t) > \epsilon \\ \texttt{REUSE}(t_{c,i}) & \text{otherwise} \end{cases} \tag{13}$$

4.2.4 Anchor Step Enforcement. To prevent unbounded error accumulation and ensure periodic weight updates (Eq. 8 and 9), we enforce **anchor steps** where all groups perform full computation. An anchor step is triggered when:

Algorithm 1 Attention State Selection Strategy

1: **Input:** timestep t, groups $\mathcal{G}$, cache ratio R, local threshold ϵ_{local}
2: Compute $\delta_{\text{local}}(t)$
3: **for** each group $G_i \in \mathcal{G}$ **do**
4: Compute $\text{ASE}(G_i, t) \leftarrow \sum_{j \in G_i} w(j,t) \cdot \delta(j, t, t_{c,i})$
5: **end for**
6: all_exceed $\leftarrow \forall G_i : \text{ASE}(G_i, t) > \epsilon$
7: local_drift $\leftarrow \delta_{\text{local}}(t) > \epsilon_{\text{local}}$
8: is_anchor $\leftarrow$ all_exceed $\vee$ local_drift
9: **if** is_anchor **then**
10: **for** each group G_i **do**
11: Decision$(G_i, t) \leftarrow$ `COMPUTE`
12: **end for**
13: Update ϵ based on R and anchor errors
14: Update $w(j,t)$ and $\alpha(j)$ for all partitions j
15: $t_{c,i} \leftarrow t$ for all G_i
16: **else**
17: **for** each group G_i **do**
18: **if** $\text{ASE}(G_i, t) > \epsilon$ **then**
19: Decision$(G_i, t) \leftarrow$ `COMPUTE`, $t_{c,i} \leftarrow t$
20: **else**
21: Decision$(G_i, t) \leftarrow$ `REUSE`$(t_{c,i})$
22: **end if**
23: **end for**
24: **end if**
25: **return** $\{\text{Decision}(G_i, t)\}$

$$\text{IsAnchor}(t) = \begin{cases} \text{True} & \text{if } \forall G_i : \text{ASE}(G_i, t) > \epsilon \\ \text{True} & \text{if } \delta_{\text{local}}(t) > \epsilon_{\text{local}} \\ \text{False} & \text{otherwise} \end{cases} \tag{14}$$

where $\delta_{\text{local}}(t)$ is the relative error of the local state. Unlike prior static thresholds, ϵ is adapted online as a percentile of the global anchor correction errors to match the target acceleration ratio, while the static $\tau_{\max}$ is replaced by this dynamic local drift condition. At anchor steps, we: (1) force all groups to compute, (2) update partition importance weights $w(j,t)$, (3) update scale ratios $\alpha(j)$, and (4) reset cache timestamps $t_{c,i} \leftarrow t$.

Algorithm 1 summarizes our selection strategy. At each timestep, we first compute local attention and predict errors for all groups (lines 2-4). Then we check anchor conditions (lines 5-7). If anchoring is triggered, all groups compute and metadata is updated (lines 8-13); otherwise, groups selectively compute or reuse based on error thresholds (lines 14-21).

5 State-Centric Attention Runtime

Given the compute/reuse decisions from selection (Sec. 4), the runtime must efficiently execute heterogeneous partition access patterns while maximizing computation-communication overlap and managing memory constraints. We design a suite of attention state-centric manipulations and orchestrate them in a pipelined runtime system.

5.1 State-Centric Manipulations

5.1.1 Cross-Group KV Transfer. When a group G_i is selected for computation (`COMPUTE`), we must gather its KV partitions from remote devices to enable local attention computation. At timestep t, let $C_t = G_i \mid \text{Decision}(G_i, t) = \texttt{COMPUTE}$ be the set of groups requiring fresh computation. Since our compute/reuse strategy is globally consistent across all devices, each GPU can independently determine its communication peer based on the partition distance within each group.

As illustrated in Figure 10(a), the Cross-Group KV Transfer operates through symmetric P2P communication. For each group $G_i \in C_t$, devices holding partitions of G_i exchange their KV pairs: each GPU performs a `SendRecv` operation, simultaneously sending its local KV partition to one peer while receiving a different partition from another peer. For example, in a group spanning Node 0 and Node 1, GPU 0 sends its partition to GPU 4 (cross-node) while GPU 4 sends to GPU 0, creating a symmetric exchange pattern.

The goal of this operation is to relocate each required KV partition to the device closest to where it will be consumed, consolidating all partitions of a compute-selected group within a localized device set. By organizing communication at the group granularity, we fully exploit hierarchical bandwidth: intra-node transfers leverage NVLink, while cross-node transfers use InfiniBand, ensuring efficient utilization of available interconnects.

5.1.2 Group State Computation. Once KV partitions are transferred, each group independently computes its fresh attention state through ring-based composed attention. As shown in Figure 10(b), devices within group G_i form a communication ring where KV partitions circulate while attention computation proceeds.

The computation consists of g sequential steps (where $g = |G_i|$ is the group size). At each step, every device: (1) computes partial attention using its current local KV partition and the query, (2) incrementally composes this result with its accumulated attention state using the composition operation from Equation 4, and (3) sends its KV partition to the next device in the ring while receiving a new partition from the previous device.

This ring-attention pattern overlaps $(g-1)$ KV transfers with attention computation. The objective of this manipuation is to efficiently produce the complete composed attention state $\text{AS}_t(G_i)$ for group G_i on each participating device. After g steps, every device has processed all partitions in the group and obtained the full composed state, which is then stored in the attention state cache for future reuse. The group-based computation boundary again leverages hierarchical bandwidth, as ring circulation occurs within the localized device set established by the Cross-Group KV Transfer.

5.1.3 Dynamic Group State Compose. DiTango continuously monitors system memory usage. When memory consumption approaches a predefined threshold $M_{\max}$, the system triggers Dynamic Group State Compose to prevent out-of-memory failures. This manipulation merges existing attention states into coarser-grained groups, exponentially reducing memory overhead.

Given current group size g and a scaling factor k (typically $k = 2$), the operation:

$$\begin{aligned} &\text{GroupCompose}(g \to kg): \quad AS_t(G_i) \oplus AS_t(G_{i+1}) \oplus \\ &\ldots AS_t(G_{i+k-1}) \to AS_t(G'_{\lfloor i/k \rfloor}) \end{aligned} \tag{15}$$

This merging continues recursively until memory usage falls below $M_{\max}$ or an anchor step occurs (which recomputes all states

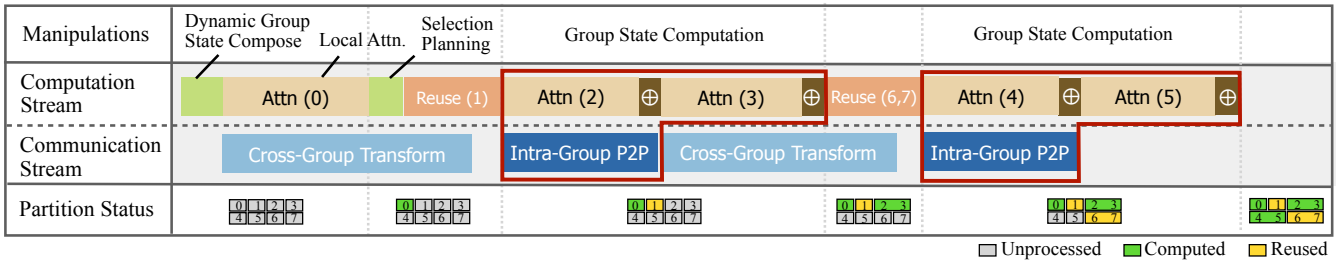


Figure 9: Runtime orchestration timeline. The computation and communication streams execute asynchronously, overlapping local attention with Cross-Group Transfer and pipelining multiple Group State Computations.

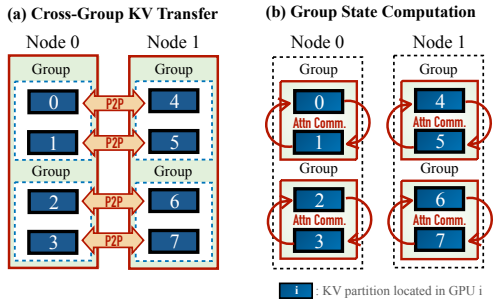


Figure 10: State-centric communication operations. (a) Cross-Group KV Transfer: symmetric P2P exchanges reorganize KV partitions across devices to co-locate required data. (b) Group State Computation: intra-group KV circulation enables composed attention state computation within each group.

and resets to the original group size g). The composed states remain valid for reuse due to the compositional properties of attention states (Eq. 4).

5.2 Runtime Orchestration

DiTango runtime orchestrates the state-centric manipulations across dual asynchronous streams—computation and communication—to maximize overlap while constraining memory usage. Figure 9 illustrates the pipeline execution flow.

Memory-aware initialization. At each timestep, the runtime first invokes *Dynamic Group State Compose* to monitor memory consumption. If the cached attention states approach the threshold $M_{\max}$, the system merges existing states into coarser groups (Sec. **??**), exponentially reducing memory overhead. This may trigger re-planning of compute/reuse decisions if the group size g changes, ensuring the selection strategy remains valid under the updated granularity.

Local attention and selection planning. The computation stream then executes local attention on the query's head partition, overlapped with the communication stream's first *Cross-Group KV Transfer*. Concurrently, the runtime performs selection planning for the next timestep, determining which groups to compute or reuse based on the accumulated error budget (Alg. 1). This planning-computation overlap hides the decision latency.

Group state computation pipeline. For each group $G_i \in C_t$ (the set of groups selected for computation), the runtime sequentially executes:

(1) Evict stale states: Remove $\mathrm{AS}(G_i)$ from cache, as it will be replaced by the fresh state $\mathrm{AS}_t(G_i)$.
(2) *Group State Computation*: Execute ring-based composed attention (Sec. 5.1.2) within the localized device set established by Cross-Group Transfer. The communication stream performs $(g-1)$ intra-group P2P transfers, overlapped with attention computation in the compute stream.
(3) Store fresh state: Cache $\mathrm{AS}_t(G_i)$ for future reuse.

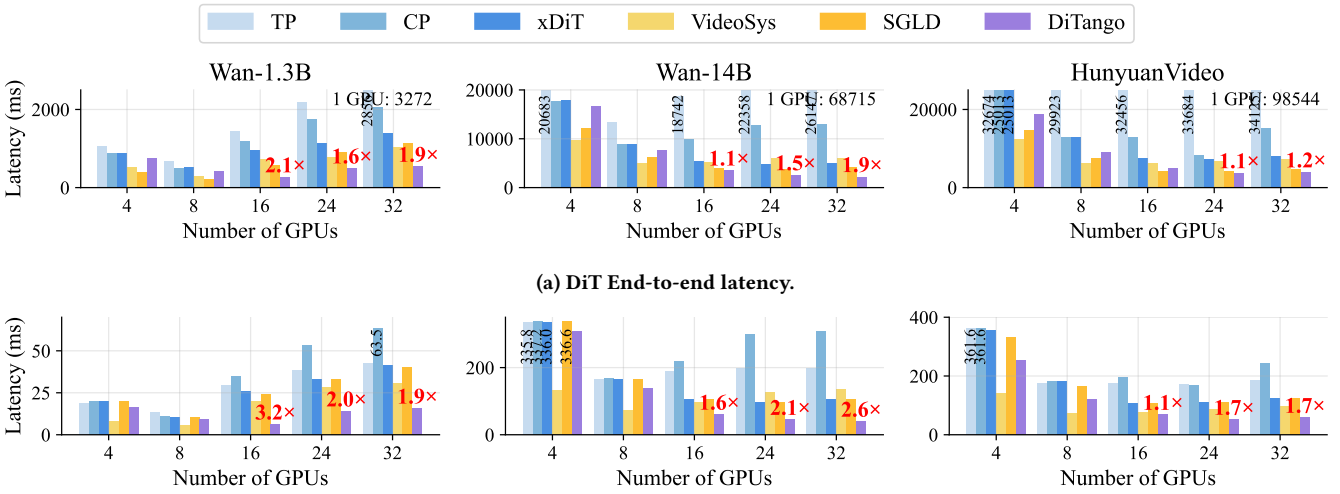
This pipeline repeats for all $|Ct|$ compute-selected groups. The runtime dynamically schedules *state reuse* operations—directly loading cached $ASt-1(G_j)$ for reused groups—to fill computation bubbles caused by cross-node communication latency in Cross-Group Transfer. This flexible interleaving ensures high GPU utilization throughout the pipeline.

Flexible reuse scheduling. A key design insight is leveraging *state reuse* operations to mask communication latency. Unlike Fresh Attention State Computation, which depends on prior KV access completing, reuse operations—directly loading cached $\mathrm{AS}_{t-1}(G_j)$ from local memory—have no data dependencies and can execute immediately. The runtime strategically schedules these lightweight reuse operations to fill computation bubbles caused by cross-node communication in Cross-Group Transfer, maintaining high GPU utilization throughout the pipeline.

The "Partition Status" visualization in Figure 9 illustrates the progress of DiTango attention: each partition transitions from unprocessed (gray) to either computed (green) via Group State Computation or reused (yellow) via cached state loading. In the example with 8 partitions, partitions 0, 2, 3, 4, 5 are computed while 1, 6, 7 are reused. Our selection algorithm (Alg. 1) guarantees that by the end of each timestep, all n partitions are covered through either fresh computation or cached reuse, ensuring completeness of the final attention output.

6 Evaluation

DiTango is implemented in 7K lines of Python (https://github.com/thu-pacman/Chitu-Diffusion), ensuring seamless integration with modern video generation models. We evaluate both its performance and quality improvements, followed by detailed ablation studies to dissect the key factors driving these gains.



(a) DiT End-to-end latency.

(b) Core Attention latency.

Figure 11: Performance evaluation of DiTango. Bars for baselines of too long running time are truncated, and their execution times are marked on the bars. The numbers above DiTango's bars show speedups over the best baseline in multi-node inference.

6.1 Evaluation Setup

Platform. We evaluate DiTango on 4 nodes with 8 NVIDIA H20 GPUs per node (148 TFLOPS FP16, 96GB), with NVLink (900GB/s) for intra-node communication and InfiniBand (400Gbps) for inter-node communication. Our software stack consists of PyTorch 2.5.0, Python 3.10.16, CUDA 12.4, and GCC 11.4.

Workloads. We evaluate DiTango using the best open-source models on VBench Leaderboard [10]: Wan2.1-14B, Wan2.1-1.3B [28], and HunyuanVideo [12]. Table 1 summarizes their key characteristics. The *Tensor Shape* is defined as *[batch size, sequence length, head number, hidden size]*. Wan2.1 models utilize Classifier-Free Guidance[9], with a latent batch size set to 2. For memory-efficient inference, we offload T5 encoder to CPU in Wan2.1-14B.

Table 1: Evaluated Model Specifications

Model	Size	Type	Frames	Reso.	Tensor Shape
Wan2.1-14B	14B	FP16	81	720×1280	[2, 76K, 40, 128]
HunyuanVideo	13B	BF16	129	720×1280	[1, 119K, 48, 64]
Wan2.1-1.3B	1.3B	FP16	81	832×480	[2, 33K, 12, 128]

Baselines. We evaluate DiTango against five state-of-the-art baselines: Tensor-parallelism [25], the best performing Context Parallel variant between Seq-wise CP [17] and Head-wise CP [11], Hybrid CP [8] implemented via xDiT [7], and two feature reuse-based lossy acceleration systems - VideoSys [27, 38] with PAB (Pyramid Attention Broadcast, warm-up/cool-down=4, broadcast range=3) and SGLang-Diffusion [39] (denoted as SGLD) which integrates Cache-DiT [6].

6.2 DiT End-to-End Performance

We conduct comprehensive evaluations on both end-to-end DiT inference latency and core attention mechanism performance. In terms of end-to-end performance, DiTango demonstrates significant efficiency improvements, achieving up to 1.9× speedup compared to state-of-the-art baselines when performing Wan-14B [28] inference across 32 H20 GPUs (Fig.11a). While existing parallel methods such as TP, CP, and xDiT maintain model accuracy through full attention computation, DiTango achieves superior performance through selective operation bypassing without compromising generation quality.

In single-node scenarios (≤ 8 GPUs), DiTango exhibits slightly longer processing times compared to traditional lossy acceleration frameworks VideoSys and SGLang-Diffusion, primarily due to its higher computation ratio designed to preserve generation quality. However, in multi-node deployments, DiTango outperforms all baselines by effectively bypassing cross-node communication while maintaining high-quality generation results. This demonstrates DiTango's particular effectiveness in distributed computing environments where communication overhead typically becomes a significant bottleneck.

While DiTango exhibits strong scalability on Wan-14B and HunyuanVideo, its performance on Wan-1.3B deteriorates beyond 24 GPUs as the limited computation is insufficient to hide intra-node communication latency, resulting in communication-bound behavior.

6.3 Core Attention Performance & Efficiency

Core Attention Latency. In our evaluation of core attention latency (Fig.11b), DiTango demonstrates remarkable efficiency, achieving an average 2× speedup compared to all baselines. Notably, this performance advantage extends even over VideoSys [27], despite its aggressive optimization strategy of bypassing approximately one-third of attention computations. SGLang-Diffusion [39] exhibits comparable core attention performance to xDiT [8] due to

their shared parallel attention implementation, with its optimization is limited to bypassing coarse-grained DiT steps rather than fine-grained attention operations.

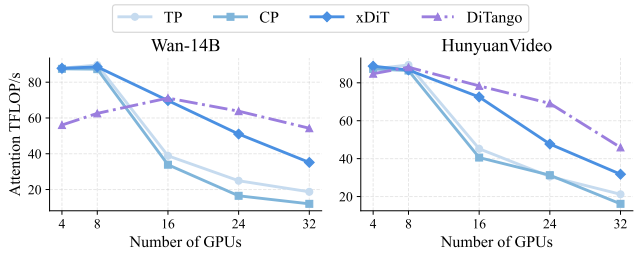


Figure 12: Core attention computational efficiency measured in FLOP/s across different GPU configurations. Note that VideoSys and SGLD is omitted as they utilize the same underlying attention mechanisms with their base implementations (CP and xDiT respectively).

FLOP/s. To further demonstrate the efficiency of DiTango attention, we employ FLOP/s metrics to evaluate core attention of different frameworks. Without any kernel-level optimizations, DiTango achieves superior computational efficiency in multi-node scenarios through its communication-optimized design, maintaining high FLOP/s where traditional parallel approaches suffer from communication bottlenecks (Fig. 12). In single-node settings, while DiTango shows lower FLOP/s due to extra state-centric operations overhead, its overall latency advantage stems from reduced attention computations compared to traditional approaches that achieve high FLOP/s through computation-communication overlap.

6.4 Generation Quality & Tradeoff

We comprehensively evaluate the generation quality of DiTango against existing frameworks using multiple metrics: PSNR, SSIM, and LPIPS for fidelity comparison against original model outputs, and VBench Score [10] for overall generation quality assessment. As shown in Table 2, DiTango consistently achieves superior performance across most quality metrics while delivering significantly higher speedup across all evaluated models. Notably, baseline methods including TP, CP, and xDiT maintain original model performance through lossless parallelization, thus serving as quality references with optimal VBench Scores but without comparative PSNR, SSIM, or LPIPS measurements.

We conduct an in-depth analysis of the factors contributing to DiTango's superior generation performance. As illustrated in Fig. 13(a), DiTango achieves higher generation quality under equivalent computation budgets through two key mechanisms: (1) its contribution-aware approach that precisely identifies and preserves the most crucial computations, and (2) its communication-efficient design that enables more effective computation within given time constraints. These architectural advantages result in a significantly better Pareto frontier in the quality-performance trade-off space, as shown in Fig. 13(b), where DiTango consistently outperforms existing approaches in balancing generation quality against computational efficiency.

Method	Latency	Visual Quality			
	Speedup ↑	LPIPS ↓	SSIM ↑	PSNR ↑	VBench(%) ↑
Wan-1.3B	1.20×	-	-	-	82.34
VideoSys	3.13×	0.288	0.676	14.08	81.37
SGLD	2.87×	0.220	0.728	15.77	81.92
DiTango	**6.08×**	**0.214**	**0.734**	**16.34**	**82.21**
Wan-14B	7.02×	-	-	-	81.53
VideoSys	11.25×	0.215	0.743	16.36	80.14
SGLD	16.51×	0.261	0.707	15.02	79.86
DiTango	**31.15×**	**0.137**	**0.806**	**18.55**	**80.79**
HunyuanVideo	6.26×	-	-	-	78.22
VideoSys	13.28×	0.174	**0.631**	21.15	77.45
SGLD	21.00×	0.178	0.558	20.70	77.12
DiTango	**24.87×**	**0.166**	0.621	**21.40**	**77.89**

Table 2: Performance and Quality comparison across different models. Speedup ratios compare 32-GPU parallel inference against single-GPU baseline.

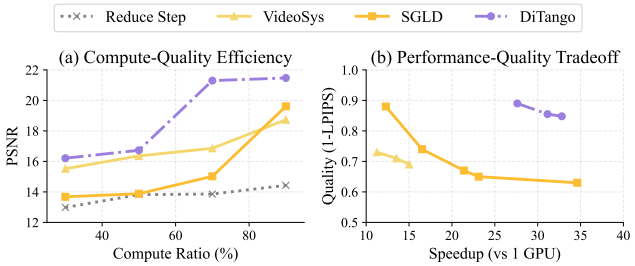


Figure 13: Quality analysis with 32-GPU inference on Wan-14B: (a) Quality comparison under different computation ratios; (b) Quality-speedup tradeoff under different reuse configurations.

6.5 Performance Breakdown

We analyze the effectiveness of key components in DiTango.

Error Modeling Accuracy. To validate the effectiveness of our error modeling approach, we evaluate the prediction accuracy across diverse samples spanning different timesteps, layers, and prompts. As shown in Fig. 14, our error model demonstrates remarkable accuracy with an R^2 value of 0.9792 and a Pearson correlation coefficient of 0.9899, indicating a strong linear relationship between predicted and actual errors. This high correlation validates two key design principles of DiTango: (1) the attention state errors across different ranks are indeed highly correlated with local state errors, and (2) our anchor-guided selection mechanism effectively captures error propagation patterns within temporal windows. The low RMSE (32.46) and MAE (21.88) values further confirm that our model can reliably predict error magnitudes, enabling informed decisions in the selection planner for balancing computation efficiency and generation quality.

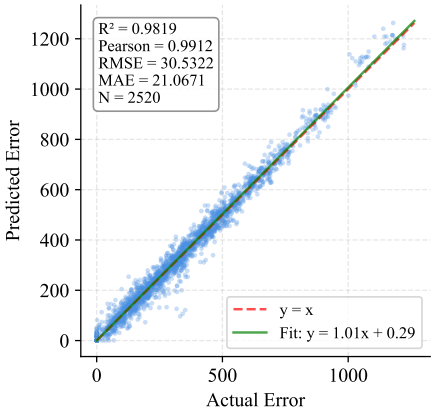


Figure 14: DiTango error modeling accuracy.

Runtime Efficiency. We analyze the runtime efficiency of DiTango's state-centric attention mechanism across different state group sizes in a 32-GPU distributed setting. As shown in Fig. 16, DiTango consistently achieves higher computation ratios compared to baseline approaches like CP and SGL-Diffusion. This superior computational efficiency is attributed to our effective computation-communication overlap strategy and hierarchical communication design. With group sizes of 4 and 8, DiTango demonstrates optimal performance by minimizing cross-node communication. However, at group size 16, while still maintaining better efficiency than baselines, performance is slightly impacted due to necessary inter-group state communications. Notably, the overhead from DiTango's additional components (planner execution and memory operations) remains minimal, as indicated by the small "Others" portion in the breakdown.

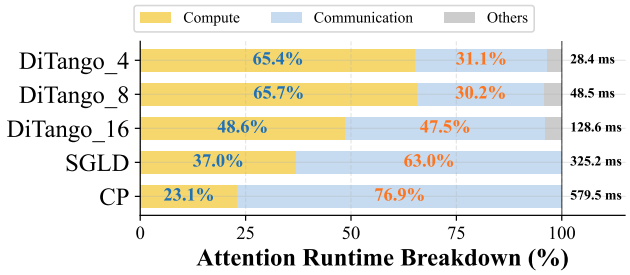


Figure 15: Runtime breakdown analysis showing the proportion of computation, communication, and other operations across different approaches and group sizes in 32-GPU distributed inference.

Extra Overhead. Despite the complex scheduling logic, DiTango introduces minimal system overhead. Computationally, the anchor-guided selection planner incurs negligible overhead. During anchor steps, the additional statistic analysis is performed on norm-compressed representations rather than full-precision tensors, adding

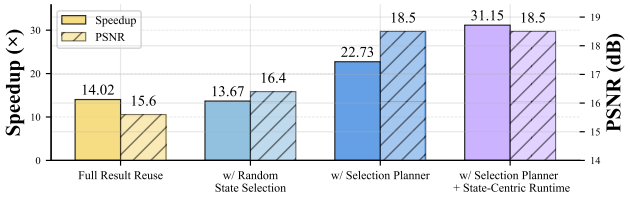


Figure 16: Ablation study showing the impact of different DiTango components on speedup and generation quality (PSNR) for Wan-14B inference on 32 GPUs.

less than 1% to the overall computation compared to a standard full-compute pass. On the memory side, maintaining the attention state cache does require additional capacity. For instance, caching the necessary attention states at full precision consumes approximately 30,GB of extra memory for large models like HunyuanVideo and Wan2.1-14B. However, unlike large language models, DiTs possess relatively small parameter footprints. Consequently, this additional memory overhead is well within the capacity of modern GPU architectures. Memory-aware management in section 5.2 is robust enough to ensure practical deployability without out-of-memory constraints.

6.6 Ablation Study

We conduct an ablation study to analyze the synergistic effects of DiTango's key components on performance and generation quality. Using Wan-14B model inference on 32 GPUs as our experimental setting, we evaluate four configurations with increasing system complexity. The baseline approach of full attention result reuse demonstrates the limitations of coarse-grained reuse strategies, showing both modest speedup and generation quality. Introducing fine-grained state reuse with random selection shows limited improvements, as its effectiveness is hampered by unstructured communication patterns and uncontrolled error accumulation. The addition of our selection planner marks a significant improvement in generation quality while also enhancing speedup through more informed state selection. Finally, the complete DiTango implementation with both selection planner and state-centric runtime orchestration achieves optimal performance while maintaining high generation quality, demonstrating how these components work in concert to overcome the scalability-quality trade-off.

7 Related Work

Parallelism for Diffusion. Traditional 3D parallelism strategies like Megatron-LM [25] exhibit limited efficacy for single-sample diffusion generation. Since the batch size is inherently 1, neither Data Parallelism nor Pipeline Parallelism can be efficiently utilized. While AsyncDiff [4] introduces a timestep-wise pipeline, it fundamentally relies on multi-sample batching, making it inapplicable to single-sample scenarios.

Context Parallelism (CP) [8, 11, 17] targets long sequence generation but incurs significant communication overhead when lacking

proper state decomposition and topology-aware scheduling. Conversely, Classifier-Free Guidance (CFG) Parallelism [9], which parallelizes the positive and negative prompt branches, is orthogonal to our approach and can be seamlessly integrated with DiTango.

Recently, some approaches have exploited stepwise feature similarity to enhance parallel efficiency. DistriFusion [14] leverages asynchronous all-gather for KV cache reuse to mitigate communication in U-Net architectures; however, this design yields sub-optimal results for DiTs due to their differing attention patterns. PipeFusion [7, 29] treats sequence chunks as micro-batches to enable pipeline parallelism, yet it encounters severe consistency degradation, particularly in video generation tasks.

Lossy Diffusion Acceleration. Diffusion generation is inherently computation-intensive yet robust to noise, making lossy acceleration a mainstream technique for achieving significant speedups with negligible visual degradation. Model-level methods like distillation [19, 22, 24] can reduce inference steps by an order of magnitude, but they require extensive extra data and training. At the operator level, quantization [35, 36] and sparsity [30, 31] kernels provide efficient hardware-level acceleration; these techniques are orthogonal to DiTango and can be seamlessly integrated with it.

Caching-based methods [2, 3, 15, 21, 34, 38] exploit temporal redundancy across diffusion steps, skipping corresponding computations by reusing similar intermediate features. Compared to distillation, they are training-free and easy to deploy. However, their coarse-grained reuse strategies and reliance on heuristic configurations severely limit their acceleration potential. This limitation becomes particularly pronounced in long video generation, where feature dynamics shift rapidly and require more fine-grained adaptivity.

8 Conclusion

In essence, DiTango is a parallel caching framework that strikes an elegant balance between algorithmic design and system orchestration, delivering substantial performance gains without compromising generation quality. It addresses the scalability bottlenecks caused by limited bandwidth in distributed diffusion inference. By selectively reusing attention states, DiTango mitigates expensive partition access overhead while preserving critical computations. At its core, the combination of an online selection planner and an efficient runtime engine enables fine-grained compute-reuse control over sequence partitions. Our findings highlight that this granularity is crucial for achieving both system scalability and generation fidelity, which is validated by significant speedups with minimal quality loss over state-of-the-art methods on typical GPU clusters.

Looking ahead, DiTango demonstrates strong potential for large-scale generative model serving. It is particularly effective in heterogeneous clusters, where it can reduce generation latency to the second-level, paving the way for real-time and interactive generation applications.

Despite these advancements, we acknowledge two current limitations of DiTango. First, in high-bandwidth environments (e.g., single-node setups) where cross-node communication is no longer the primary bottleneck, methods focusing purely on computation reduction may yield higher throughput; thus, a hybrid deployment strategy combining both paradigms is optimal in practice. Second, although DiTango incorporates memory-aware management, the attention state cache still consumes substantial capacity, which can constrain GPU utilization in production environments. Future work will focus on addressing this memory footprint through cache compression techniques or intelligent CPU offloading mechanisms to further unleash the potential of parallel diffusion systems.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work is partially supported by the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM910), NSFC for Distinguished Young Scholar under Grant 62225206, National Natural Science Foundation of China under Grants 62532006, U23A6007, and Beijing Natural Science Foundation under Grant L242017. Jidong Zhai is the corresponding author of this paper.

References

[1] Tim Brooks, Bill Peebles, Connor Holmes, Amos Storkey, Alexei A. Efros, Andreas Terzis, Abhinav Gupta, Devi Parikh, Douwe Kiela, Gabriel Synnaeve, Hannaneh Hajishirzi, Ilya Sutskever, James Zung, Joelle Pineau, Luke Metz, Mira Murati, Pranav Shyam, Rohun Kulkarni, Ruth Fong, Vedant Misra, Yufei Guo, Adrià Recasens, Alexander Papiez, Alexandre Lebrun, Arthur Mensch, Avel Guénin, Bowen Baker, Brandon Houghton, Brian Tanner, Briane Paul V. Samson, Chang Chen, Christopher Clark, Cory McLean, David Martinez-Rubio, David Schnurr, Desu Narendra, Eli Moser, Ethan Perez, Igor Mordatch, J.R. Harris, Joey Faulkner, John Baumgartner, Karl Cobbe, Liam Fedus, Madeleine Thompson, Mark Chen, Mayur Mudigonda, Mimee Xu, Noemi Dreymann, Teddy Lee, Theresa Yoon, Timothy Lillicrap, Torrey Fellbaum, Trevor Darrell, Yuntao Bai, and Yutian Chen. 2024. *Video generation models as world simulators.* Technical Report. OpenAI. https://openai.com/research/video-generation-models-as-world-simulators

[2] Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. 2024. Delta-DiT: A Training-Free Acceleration Method Tailored for Diffusion Transformers. arXiv:2406.01125 [cs.CV] https://arxiv.org/abs/2406.01125

[3] Yuyang Chen, Linqian Zeng, Yijin ZHou, Hengjie Li, and Jidong Zhai. 2026. Jano: Adaptive Diffusion Generation with Early-stage Convergence Awareness. arXiv:2603.00519 [cs.CV] https://arxiv.org/abs/2603.00519

[4] Zigeng Chen, Xinyin Ma, Gongfan Fang, Zhenxiong Tan, and Xinchao Wang. 2024. AsyncDiff: Parallelizing Diffusion Models by Asynchronous Denoising. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (Eds.). http://papers.nips.cc/paper_files/paper/2024/hash/ad15848baa3932c0d2deabf0e11d1dcd-Abstract-Conference.html

[5] Tri Dao. 2024. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.* OpenReview.net. https://openreview.net/forum?id=mZn2Xyh9Ec

[6] vipshop.com DefTruth. 2025. cache-dit: A PyTorch-native and Flexible Inference Engine with Hybrid Cache Acceleration and Parallelism for DiTs. https://github.com/vipshop/cache-dit.git Open-source software available at https://github.com/vipshop/cache-dit.git.

[7] Jiarui Fang, Jinzhe Pan, Xibo Sun, Aoyu Li, and Jiannan Wang. 2024. xDiT: an Inference Engine for Diffusion Transformers (DiTs) with Massive Parallelism. *CoRR* abs/2411.01738 (2024). doi:10.48550/ARXIV.2411.01738 arXiv:2411.01738

[8] Jiarui Fang and Shangchun Zhao. 2024. A Unified Sequence Parallelism Approach for Long Context Generative AI. *arXiv preprint arXiv:2405.07719* (2024).

[9] Jonathan Ho and Tim Salimans. 2022. Classifier-Free Diffusion Guidance. *CoRR* abs/2207.12598 (2022). doi:10.48550/ARXIV.2207.12598 arXiv:2207.12598

[10] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. 2024. VBench: Comprehensive Benchmark Suite for Video Generative Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

[11] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. 2023. DeepSpeed Ulysses:

System Optimizations for Enabling Training of Extreme Long Sequence Transformer Models. *CoRR* abs/2309.14509 (2023). doi:10.48550/ARXIV.2309.14509 arXiv:2309.14509

[12] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Daquan Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. 2024. HunyuanVideo: A Systematic Framework For Large Video Generative Models. *CoRR* abs/2412.03603 (2024). doi:10.48550/ARXIV.2412.03603 arXiv:2412.03603

[13] Kuaishou Technology. 2025. Kling - Community for short video & livestream. https://kling.kuaishou.com/en Accessed: 2025-03-07.

[14] Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Kai Li, and Song Han. 2024. DistriFusion: Distributed Parallel Inference for High-Resolution Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*. IEEE, 7183–7193. doi:10.1109/CVPR52733.2024.00686

[15] Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. 2024. Timestep Embedding Tells: It's Time to Cache for Video Diffusion Model. *CoRR* abs/2411.19108 (2024). doi:10.48550/ARXIV.2411.19108 arXiv:2411.19108

[16] Hao Liu and Pieter Abbeel. 2023. Blockwise Parallel Transformers for Large Context Models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/1bfd87d2d92f0556819467dc08034f76-Abstract-Conference.html

[17] Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring Attention with Blockwise Transformers for Near-Infinite Context. *CoRR* abs/2310.01889 (2023). doi:10.48550/ARXIV.2310.01889 arXiv:2310.01889

[18] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. 2025. From Reusing to Forecasting: Accelerating Diffusion Models with TaylorSeers. *arXiv preprint arXiv:2503.06923* (2025).

[19] Qiang Liu. 2022. Rectified Flow: A Marginal Preserving Approach to Optimal Transport. arXiv:2209.14577 [stat.ML] https://arxiv.org/abs/2209.14577

[20] Guoqing Ma, Haoyang Huang, Kun Yan, Liangyu Chen, Nan Duan, Shengming Yin, Changyi Wan, Ranchen Ming, Xiaoniu Song, Xing Chen, Yu Zhou, Deshan Sun, Deyu Zhou, Jian Zhou, Kaijun Tan, Kang An, Mei Chen, Wei Ji, Qiling Wu, Wen Sun, Xin Han, Yanan Wei, Zheng Ge, Aojie Li, Bin Wang, Bizhu Huang, Bo Wang, Brian Li, Changxing Miao, Chen Xu, Chenfei Wu, Chenguang Yu, Dapeng Shi, Dingyuan Hu, Enle Liu, Gang Yu, Ge Yang, Guanzhe Huang, Gulin Yan, Haiyang Feng, Hao Nie, Haonan Jia, Hanpeng Hu, Hanqi Chen, Haolong Yan, Heng Wang, Hongcheng Guo, Huilin Xiong, Huixin Xiong, Jiahao Gong, Jianchang Wu, Jiaoren Wu, Jie Wu, Jie Yang, Jiashuai Liu, Jiashuo Li, Jingyang Zhang, Junjing Guo, Junzhe Lin, Kaixiang Li, Lei Liu, Lei Xia, Liang Zhao, Liguo Tan, Liwen Huang, Liying Shi, Ming Li, Mingliang Li, Muhua Cheng, Na Wang, Qiaohui Chen, Qinglin He, Qiuyan Liang, Quan Sun, Ran Sun, Rui Wang, Shaoliang Pang, Shiliang Yang, Sitong Liu, Siqi Liu, Shuli Gao, Tiancheng Cao, Tianyu Wang, Weipeng Ming, Wenqing He, Xu Zhao, Xuelin Zhang, Xianfang Zeng, Xiaojia Liu, Xuan Yang, Yaqi Dai, Yanbo Yu, Yang Li, Yineng Deng, Yingming Wang, Yilei Wang, Yuanwei Lu, Yu Chen, Yu Luo, Yuchu Luo, Yuhe Yin, Yuheng Feng, Yuxiang Yang, Zecheng Tang, Zekai Zhang, Zidong Yang, Binxing Jiao, Jiansheng Chen, Jing Li, Shuchang Zhou, Xiangyu Zhang, Xinhao Zhang, Yibo Zhu, Heung-Yeung Shum, and Daxin Jiang. 2025. Step-Video-T2V Technical Report: The Practice, Challenges, and Future of Video Foundation Model. arXiv:2502.10248 [cs.CV] https://arxiv.org/abs/2502.10248

[21] Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. 2024. Learning-to-Cache: Accelerating Diffusion Transformer via Layer Caching. arXiv:2406.01733 [cs.LG]

[22] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. 2022. On Distillation of Guided Diffusion Models. *arXiv e-prints*, Article arXiv:2210.03142 (Oct. 2022), arXiv:2210.03142 pages. doi:10.48550/arXiv.2210.03142 arXiv:2210.03142 [cs.CV]

[23] William Peebles and Saining Xie. 2023. Scalable Diffusion Models with Transformers. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*. IEEE, 4172–4182. doi:10.1109/ICCV51070.2023.00387

[24] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. 2023. Adversarial Diffusion Distillation. arXiv:2311.17042 [cs.CV] https://arxiv.org/abs/2311.17042

[25] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *CoRR* abs/1909.08053 (2019). arXiv:1909.08053 http://arxiv.org/abs/1909.08053

[26] Genmo Team. 2024. Mochi 1. https://github.com/genmoai/models.

[27] VideoSys Team. 2024. VideoSys: An Easy and Efficient System for Video Generation. https://github.com/NUS-HPC-AI-Lab/VideoSys

[28] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. 2025. Wan: Open and Advanced Large-Scale Video Generative Models. *arXiv preprint arXiv:2503.20314* (2025).

[29] Jiannan Wang, Jiarui Fang, Aoyu Li, and PengCheng Yang. 2024. PipeFusion: Displaced Patch Pipeline Parallelism for Inference of Diffusion Transformer Models. *CoRR* abs/2405.14430 (2024). doi:10.48550/ARXIV.2405.14430 arXiv:2405.14430

[30] Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. 2025. Sparse VideoGen: Accelerating Video Diffusion Transformers with Spatial-Temporal Sparsity. *CoRR* abs/2502.01776 (2025). doi:10.48550/ARXIV.2502.01776 arXiv:2502.01776

[31] Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, et al. 2025. Sparse VideoGen2: Accelerate Video Generation with Sparse Attention via Semantic-Aware Permutation. *arXiv preprint arXiv:2505.18875* (2025).

[32] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Xiaotao Gu, Yuxuan Zhang, Weihan Wang, Yean Cheng, Ting Liu, Bin Xu, Yuxiao Dong, and Jie Tang. 2024. CogVideoX: Text-to-Video Diffusion Models with An Expert Transformer. *CoRR* abs/2408.06072 (2024). doi:10.48550/ARXIV.2408.06072 arXiv:2408.06072

[33] Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. 2025. FlashInfer: Efficient and Customizable Attention Engine for LLM Inference Serving. *CoRR* abs/2501.01005 (2025). doi:10.48550/ARXIV.2501.01005 arXiv:2501.01005

[34] Zhihang Yuan, Hanling Zhang, Lu Pu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. 2024. DiTFastAttn: Attention Compression for Diffusion Transformer Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=51HQpkQy3t

[35] Jintao Zhang, Haofeng Huang, Pengle Zhang, Jia Wei, Jun Zhu, and Jianfei Chen. 2025. Sageattention2: Efficient attention with thorough outlier smoothing and per-thread int4 quantization. In *International Conference on Machine Learning (ICML)*.

[36] Jintao Zhang, Jia Wei, Pengle Zhang, Jun Zhu, and Jianfei Chen. 2025. SageAttention: Accurate 8-Bit Attention for Plug-and-play Inference Acceleration. In *International Conference on Learning Representations (ICLR)*.

[37] Xuanlei Zhao, Shenggan Cheng, Zangwei Zheng, Zheming Yang, Ziming Liu, and Yang You. 2024. DSP: Dynamic Sequence Parallelism for Multi-Dimensional Transformers. *CoRR* abs/2403.10266 (2024). doi:10.48550/ARXIV.2403.10266 arXiv:2403.10266

[38] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. 2024. Real-Time Video Generation with Pyramid Attention Broadcast. arXiv:2408.12588 [cs.CV] https://arxiv.org/abs/2408.12588

[39] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. SGLang: Efficient Execution of Structured Language Model Programs. arXiv:2312.07104 [cs.AI] https://arxiv.org/abs/2312.07104