Minute-Long Videos with Dual Parallelisms

Zeqing Wang[1], Bowen Zheng[1], Xingyi Yang[2], Zhenxiong Tan[1], Yuecong Xu[1], Xinchao Wang[1*]

[1]National University of Singapore
[2]The Hong Kong Polytechnic University
zeqing.wang@u.nus.edu, xinchao@nus.edu.sg

Abstract

Diffusion Transformer (DiT)-based video diffusion models generate high-quality videos at scale but incur prohibitive processing latency and memory costs for long videos. To address this, we propose a novel distributed inference strategy, termed **DualParal**. The core idea is that, instead of generating an entire video on a single GPU, we parallelize computation by partitioning both video frames and model layers across multiple GPUs. However, a naive parallel implementation is not feasible. Because all frames need to share the same noise level, they can't be processed independently. Instead, every step must wait for all others to finish, which cancels out the speed benefits of parallel processing. We overcome this obstacle with a *block-wise denoising* scheme. Namely, we segment the video into sequential blocks, each with a different noise level. As a result, we process them in a pipeline across the GPUs. Each GPU, holding a subset of the model layers, processes a specific block of frames and passes the results to the next GPU, enabling asynchronous computation and communication. To further optimize performance, we incorporate two key enhancements. Firstly, each GPU uses a feature cache technique to reduce the overhead of smooth transitions by reusing only features involved in cross-frame computation from the prior block, minimizing inter-GPU communication and redundant computation. Secondly, we employ a coordinated noise initialization strategy, ensuring globally consistent temporal dynamics by sharing initial noise patterns across GPUs. Together, these enable fast, artifact-free, and infinitely long video generation. Applied to the latest diffusion transformer video generator, our method efficiently produces 1,025-frame videos with up to 6.54× lower latency and 1.48× lower memory cost on 8×RTX 4090 GPUs.

Project Page — https://dualparal-project.github.io/
Code — https://github.com/DualParal-Project/DualParal

1 Introduction

Diffusion Transformer (DiT) (Peebles and Xie 2023) has significantly improved the scalability of video diffusion models (Kong et al. 2024; Zheng et al. 2024), enabling more realistic and higher-resolution video generation. Despite its benefits, large-scale DiT models suffer from their inherent computational inefficiency. This directly results in extended processing durations and memory demands.

Notably, this inefficiency is further exacerbated when generating long videos. Intuitively, longer videos increase the input sequence length. This has severe implications for latency: the attention mechanism, core to DiT (Fang et al. 2024a,b), exhibits time complexity that scales quadratically with sequence length. Concurrently, memory consumption escalates substantially due to the combination of a large number of model parameters and the extended video sequences. Therefore, enabling DiT-based models to efficiently generate high-quality long videos remains a formidable and pressing challenge.

Recently, parallelization has emerged as a promising solution for efficient long video generation. It uses multiple devices to produce video jointly, which scales memory and boosts processing speed.

Among existing strategies, *sequence parallelism* reduces latency by synchronously processing split hidden (Jacobs et al. 2023; Liu, Zaharia, and Abbeel 2023) or input (Tan et al. 2024; Kim et al. 2024) sequences using a full model replica on each device. However, they incurs high memory overhead due to the entire model on every device. In contrast, *pipeline parallelism* (Fang et al. 2024b) mitigates memory usage by partitioning the model across devices as a device pipeline (Huang et al. 2019; Li et al. 2021; Shoeybi et al. 2020). Therefore, an ideal solution would combine sequence parallelism with pipeline parallelism to maximize speed and minimize memory usage.

However, naively combining sequence and pipeline parallelism is fundamentally conflicting. The core issue stems from the inherent *synchronization property* of video diffusion models: all input tokens must pass through an entire layer together before any can move on. In pipeline parallelism, this means the full input must finish processing on one device (e.g., Device 1) before passing to the next (e.g., Device 2). This requirement directly contradicts sequence parallelism, which splits the input across devices. As a result, all distributed parts must be gathered back onto a single device for serialized processing on specific model layers. Only then can all parts enter the next pipeline stage, i.e. next device. This repeated gathering serializes computation and negates the benefits of sequence parallelism, reintroducing a serial bottleneck and significant communication overhead.

*Corresponding Author.

To address this conflict, we propose a novel distributed inference strategy, termed **DualParal**. At a high level, DualParal divides both the video sequence and model into chunks and applies parallel processing across both.

As discussed above, a naive combination presents challenges. Inspired by recent work on interpolating diffusion and autoregressive models (Arriola et al. 2025; Sand-AI 2025), we make this feasible by implementing a *block-wise denoising* scheme for video diffusion models. By the word *block-wise*, we refer to a strategy where, instead of denoising all frames at a uniform noise level, we divide the video into non-overlapping temporal blocks. Each block is assigned a different noise level according to its position in the video: blocks closer to the end have higher noise levels, while earlier blocks receive lower noise levels. During each inference step, the model processes all blocks asynchronously, incrementally reducing their respective noise levels. Crucially, because noise levels do not need to be synchronized across all frames, block-wise denoising resolves the inherent conflict between the two parallelism strategies.

Accordingly, we tailor this inference scheme for multi-device execution with our DualParal. Specifically, we organize video sequence blocks in a first-in-first-out (FIFO) queue (Kim et al. 2024), where noise levels decrease from tail to head. In each diffusion step, a new noisy block is appended to the tail, while a clean block is removed from the head. These blocks are then processed in reverse order, tail to head, through the device pipeline. In this setup, each device handles a specific video block and a model chunk, with denoised outputs passed asynchronously between GPUs. This distributed architecture enhances memory efficiency by distributing the model and achieves near-zero idle time through asynchronous block processing and communication.

Even more compelling, DualParal leverages its FIFO queue to enable long video generation. New blocks can be continuously appended to the queue, allowing for producing arbitrarily long videos. Because the number of frames within each block remains fixed, this approach again avoids quadratic increase in processing latency and high memory costs associated with extended video sequences.

To further optimize efficiency and maintain video quality, we introduce two key enhancements for DualParal. *Firstly*, to ensure a smooth transition between adjacent blocks, each block is concatenated with parts of previous and subsequent blocks before pipeline processing, incurring overhead. To reduce this, DualParal employs a feature cache technique on each GPU that reuses Key-Value (KV) features from the previous block without explicitly concatenating them. This reduces inter-GPU communication and redundant computation in non-cross-frame components, such as Cross-Attention and Feed-Forward Networks (FFN) (Vaswani et al. 2017), in DiT-based models. *Secondly*, to maintain global consistency across blocks without extra resource costs, each new block is initialized within a shared coordinated noise space. However, we observe that when concatenating adjacent blocks for coherence, repetitive noise from the same noise space in the concatenated block can degrade performance. To address this, we reorganize the initialization to ensure all denoised frames—including concatenated parts and the current block—originate from the same noise space without repetition. Together, these enable fast, artifact-free, and infinite-length video generation.

In summary, our contributions are summarized as follows: (1) We design an efficient distributed inference strategy by parallelizing both the video sequence and model layers, operating under a *block-wise denoising* scheme, to optimize both computation and memory costs. (2) We employ feature cache and coordinated noise initialization strategies to optimize parallel efficiency while preserving video quality. (3) Experiments show that DualParal reduces latency by up to 6.54× and memory cost by 1.48× compared to state-of-the-art parallel methods when generating 1,025-frame videos using 8×RTX 4090 GPUs on the representative DiT-based model Wan2.1 (Wan et al. 2025).

2 Preliminaries

Diffusion models in video generation. Video generation using diffusion models (Peebles and Xie 2023; Rombach et al. 2022; Bao et al. 2023; Tan et al. 2025a,b) involves progressively denoising frame latents x_t, where t denotes the noise level and ranges from T (most noisy) to 0 (cleanest). Here, T also represents the total number of denoising steps. The process starts with complete noisy latents x_T, and each step updates x_t to cleaner x_{t-1}. This continues until x_T is denoised to x_0, which is then decoded to generate the video. The key operation in updating x_t to x_{t-1} involves computing the noisy prediction $\epsilon_t = \mathcal{E}_\theta(x_t)$, where $\mathcal{E}_\theta$ represents the diffusion model. Subsequently, x_{t-1} is derived using $x_{t-1} = S(\epsilon_t, x_t, t)$, where S is certain updating scheduler.

Specifically, the noisy frame latent is defined as $x_t \in \mathbb{R}^{F\times H\times W\times C}$, where F is the number of frames, H and W are the height and width, and C is the number of channels. Note that after passing x_0 through the decoder to generate the final video X, the dimensions of X differ from those of x_0 due to the upsampling process in the decoder (Hong et al. 2023; Peng et al. 2025). For simplicity, we only use the dimensions of x to represent the video.

Parallelisms for DiT-based video diffusion models. Pipeline parallelism (Huang et al. 2019; Qi et al. 2024) involves evenly splitting the entire neural network across N devices, with each device responsible for a consecutive subset of the model, denoted as $\mathcal{E}_\theta = [\mathcal{E}_{\theta_1}, \mathcal{E}_{\theta_2}, \dots, \mathcal{E}_{\theta_N}]$. Since DiT-based video diffusion models (Peebles and Xie 2023; Wan et al. 2025) are generally composed of multiple similar DiT blocks, we define L as the total number of DiT blocks, with each device handling consecutive $\frac{L}{N}$ DiT blocks. Therefore, denoising x_t is represented as:

$$\begin{aligned}\epsilon_t &= \mathcal{E}_{\theta_N}(\mathcal{E}_{\theta_{N-1}}(\dots(\mathcal{E}_{\theta_1}(x_t))\dots)) \\ &= \mathcal{E}_{\theta_N}(\dots(\mathcal{E}_{\theta_j}(\epsilon_t^{j-1}))\dots),\end{aligned} \tag{1}$$

where $\epsilon_t^{j-1} \in \mathbb{R}^{p\times h}$ denotes the noisy prediction from the previous $(j-1)^{th}$ device. Here, p represents the sequence length and h denotes the hidden size. Specifically, $p = F' \times H' \times W'$, where F', H', and W' are the downsampled dimensions of F, H, and W, respectively. For the Wan2.1 model (Wan et al. 2025) used as the base in this paper, $F' = F$. Therefore, we define $p = F \times H' \times W'$.

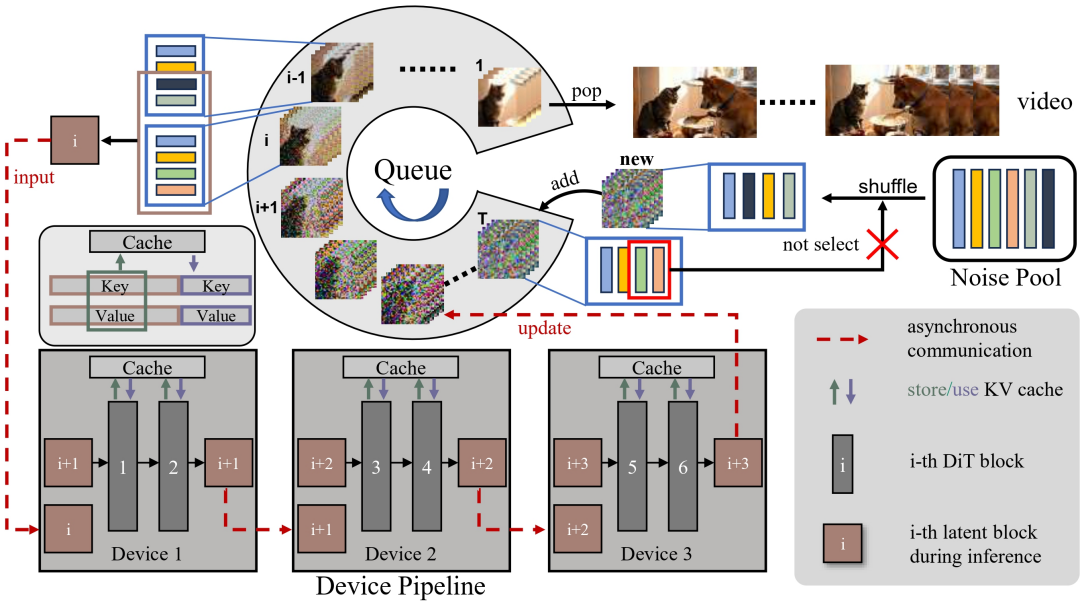

Figure 1: Overview of DualParal: DualParal partitions video frames into sequential blocks organized in a queue with noise levels increasing from tail to head, and distributes model layers across devices via a device pipeline. By feeding blocks into the pipeline in a reverse order (from tail to head), this *block-wise denoising* scheme significantly improves efficiency. To further improve performance, DualParal reuses Key-Value (KV) features from the previous block, requiring only the subsequent block to be concatenated. To preserve global consistency, each new block is initialized from a shared noise pool by shuffling noises, excluding the last $\frac{Num_c}{2}$ latents of the last block in queue.

Sequence parallelism divides input $x_t \in \mathbb{R}^{F\times H\times W\times C}$ into non-overlapping blocks, each denoted as $B_t \in \mathbb{R}^{Num_B\times H\times W\times C}$, where Num_B is the number of frames per block. To enhance temporal coherence across adjacent blocks, several methods (Kim et al. 2024; Tan et al. 2024) concatenate previous, subsequent, or global context frames with the current block during denoising, resulting in the extended block $B'_t \in \mathbb{R}^{(Num_B+Num_C)\times H\times W\times C}$, where Num_C denotes the number of concatenated context frames.

3 DualParal

At a high level, DualParal (Figure 1) achieves computational and memory efficiency through dual parallelisms applied across both video sequences and model layers, leveraging a *block-wise denoising* scheme. This section first details the architecture, then introduces two key enhancements, **Feature cache** and **Coordinated noise initialization**. Finally, we provide theoretical parallel performance analysis to quantify DualParal's efficiency gains.

3.1 Parallel architecture

Naively combining sequence and pipeline parallelism introduces an inherent conflict: synchronizing noise levels across frames requires all split sequences to be gathered and processed on a single device before proceeding to the next device in the pipeline. This conflict degrades both strategies into serialized processing, causing high device idle time in pipeline parallelism and disrupting parallel execution in sequence parallelism. Moreover, it incurs significant communication overhead due to repeated sequence gathering.

To address these issues, DualParal adopts dual parallelisms under a *block-wise denoising* mechanism. Namely, DualParal simultaneously processes block-wise frames with asynchronous noise levels across different model chunks. Since noise levels do not need to be synchronized across frames at each model segment, DualParal effectively resolves the conflict between dual parallelisms.

Specifically, as shown in Figure 1, DualParal comprises two key components: queue and device pipeline. In device pipeline, DiT blocks from the video diffusion model are evenly distributed across multiple GPUs. Within the queue, each element is a block of Num_B frame latents with same noise level, denoted as $B_i = [x_i, x_i, \ldots, x_i]$, where $x_i \in \mathbb{R}^{1\times H\times W\times C}$ represents a single frame latent at i^{th} noise level. Additionally, the queue is organized in a FIFO manner (Kim et al. 2024; Ruhe et al. 2024), with blocks arranged from tail to head in progressively decreasing noise from 1 to T. Formally, queue is described as $Q = [B_1, B_2, \ldots, B_T]$. During inference, blocks in the queue are continuously fed into the device pipeline in reverse order, from tail to head. After each diffusion step, all blocks in the queue shift forward by one position, i.e., $Q = [B_0, B_1, \ldots, B_{T-1}]$. A new noisy block B_T is then appended to the tail, while the clean block B_0 is removed from the head and passed to the decoder for video reconstruction. With this design, each device handles a specific video block and model segment, while denoised outputs are passed asynchronously between GPUs. This *block-wise denoising* scheme effectively resolves the serialization degradation caused by naively combining se-

quence and pipeline parallelism, thereby enabling true parallelization across both temporal frames and model layers.

3.2 Feature cache

Since whole video frames are divided into non-overlapping blocks, we concatenate previous and subsequent blocks with a total of Num_C frame latents to maintain transition smoothness. For simplicity, we assume all frame latents from both adjacent blocks are included, i.e., $Num_C = 2Num_B$, resulting in the denoising of an extended block $B'_i = [B_{i-1}, B_i, B_{i+1}]$. Note that B_{i-1} denotes the *subsequent* block, while B_{i+1} refers to the *previous* block under the reversed inference order. Therefore, denoising block B'_i is formally described as:

$$\begin{aligned} \epsilon_i &= \mathcal{E}_{\theta_N}(\mathcal{E}_{\theta_{N-1}}(\dots(\mathcal{E}_{\theta_1}(B'_i))\dots)) \\ &= \mathcal{E}_{\theta_N}(\dots(\mathcal{E}_{\theta_j}(\epsilon_i^{j-1}))\dots), \end{aligned} \tag{2}$$

where each intermediate output ϵ_i^{j-1} is transmitted from $(j-1)^{\text{th}}$ to j^{th} device using asynchronous peer-to-peer (P2P) communication, allowing communication and computation to overlap effectively.

However, this implementation introduces extra communication and computation overhead due to the concatenated parts. To mitigate this, we exploit a unique feature of DualParal and propose a feature cache technique. Specifically, since $B'_i = [B_{i-1}, B_i, B_{i+1}]$ is denoised after $B'_{i+1} = [B_i, B_{i+1}, B_{i+2}]$, B_{i+1} has already been processed during denoising B'_{i+1}. Leveraging this feature, we cache the KV features from the Self-Attention module of B_{i+1} during denoising B'_{i+1} and reuse them for B'_i. Consequently, the input block is reduced to $B'_i = [B_{i-1}, B_i]$, decreasing communication overhead between adjacent devices.

Moreover, among all model components, only those involving cross-frame interactions—such as the Self-Attention module in the Wan2.1 model (Wan et al. 2025)—are relevant. Therefore, we restrict the feature caching technique to the Self-Attention module while skipping components like Cross-Attention and FFN, which do not benefit from inter-frame information. This selective application effectively eliminates redundant computations.

3.3 Coordinated noise initialization

Although DualParal concatenates adjacent blocks to enhance transition smoothness, achieving global consistency remains challenging. Expanding global information concatenation is straightforward but incurs prohibitive communication, computation, and memory overhead. To avoid these costs, we explore noise latent reuse within the same noise space (Qiu et al. 2024; Zhang, Yang, and Lim 2025) as a more efficient alternative. This section analyzes different initialization strategies for achieving global consistency in DiT-based video diffusion models and reveals two observations: **1) Using the *complete* noise space preserves favorable global consistency. 2) Latents with repetitive noise throughout the *whole* denoising lead to significant performance degradation in DiT-based video diffusion.**

The first observation, illustrated in Figure 2, shows that using the complete noise space (a) yields better global consistency than using a subset (b) or adding new noise (c) to the original space. Based on this, we initialize blocks in DualParal using the complete noise space, with varying initialization orders. However, as shown in (d), directly using the same noise in Wan2.1, a DiT-based video diffusion model, leads to a significant performance degradation. This second observation arises from repetitive noises when concatenating the subsequent block in DualParal during *whole* denoising process. In contrast, the previous block only affects the Self-Attention module in Wan2.1, without degrading performance. To address this while retaining the benefits of the complete noise space, we propose a novel initialization strategy. As shown in Figure 1, when initializing a new block, we select noise from a pool excluding the last $\frac{Num_C}{2}$ latents of the final block B_T in the queue (e.g., $Num_C = 4$ in Figure 1). These selected noises are then shuffled and used to initialize the new block. Note that the first block uses the complete noise pool and contains $\frac{Num_C}{2} + Num_B$ frames. Therefore, this strategy ensures all noises are distinct across concatenated blocks during denoising process while still leveraging the complete noise pool.

3.4 Quantitative analysis of efficiency

This section provide quantitative analysis of parallel performance of DualParal in terms of bubble ratio, communication overhead, and memory cost.

The bubble ratio (Huang et al. 2019) measures the proportion of idle time on each device. We compute it for DualParal under the reverse (tail-to-head) denoising order, assuming $N \leq Block_{num}$, where $Block_{num}$ denotes the total number of blocks during generation. This assumption holds in practice, especially for minute-long videos. Under this setup, the bubble ratio is formally defined as:

$$Bubble = \frac{N^2 - 2 \times N - 1}{N^2 - 2 \times N - 1 + T \times Block_{num}}. \tag{3}$$

A detailed proof of Equation 3 is provided in Appendix D.1. To intuitively illustrate the bubble ratio, Figure 3 presents an example of pipeline scheduling in DualParal, exhibiting an approximate bubble ratio of 3.3%. Idle time occurs during brief warm-up and cool-down phases when the current number of blocks in the queue is less than the number of devices N (e.g., before step 4 and after step T in Figure 3). During these phases, some idle time and synchronization overhead may arise due to non-overlapping communication and computation. However, this overhead is negligible for long video generation. As $Block_{num}$ increases, the bubble ratio approaches 0%, indicating minimal device idling in the pipeline during long video generation. Thus, DualParal achieves high GPU utilization. Further bubble ratio analysis (proofs, varying denoising conditions) is in the Appendix D.

To compare DualParal with other parallel methods in communication and memory costs, we qualitatively evaluate it against DeepSpeed-Ulysses (Jacobs et al. 2023), Ring Attention (Liu, Zaharia, and Abbeel 2023), Video-Infinity (Tan et al. 2024), and FIFO (Kim et al. 2024). Following prior works (Fang et al. 2024a,b), we conduct a similar analysis of parallelism in video diffusion models, as summarized in Table 1. For DualParal, per-device communication cost is determined by the input and output of ϵ via asynchronous P2P

Figure 2: Examples of four different noise initializations for Wan2.1 model (Wan et al. 2025): (a) uses the complete noise space, (b) uses a subset of the noise space, (c) adds new noise to the original space, and (d) uses the complete noise space with the repetitive noise. In each initialization, the first image shows the standard video generated from the reference noise space, followed by two different orders of noise initialization based on the reference noise space.

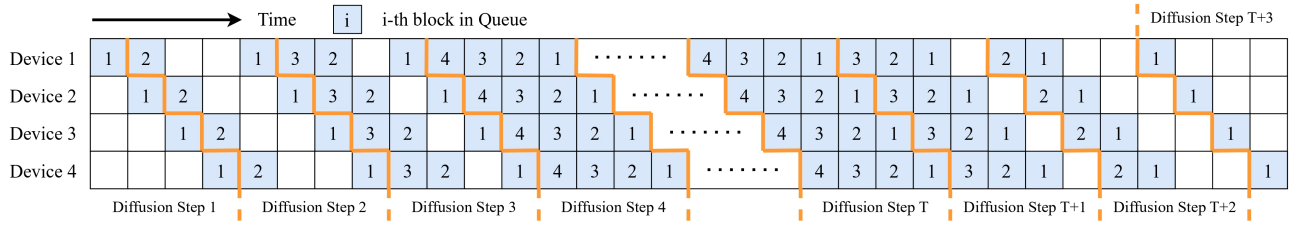

Figure 3: Pipeline schedule of DualParal with $N = 4$, $T = 50$, and $Block_{num} = 4$. Blocks are denoised in reverse order, from tail to head in the queue. After diffusion step T, the first clean block is popped from the queue, and all remaining blocks shift forward by one position, decrementing their indices accordingly.

communication. Though synchronous P2P occurs during warm-up and cool-down phases, its overhead is negligible for long videos. Furthermore, we employ feature cache technique to reduce this cost by caching the previous block, resulting in the transmission of only $Num_B + \frac{Num_C}{2}$ frames. For memory cost, pipeline parallelism distributes the model across N devices. The peak memory for KV activations is $(Num_B + Num_C)KV$, significantly lower than Ring Attention, DeepSpeed-Ulysses, and Video-Infinity in long video generation. This is because their fixed-length generation requires extending sequences at runtime to support long video generation. In contrast, DualParal and FIFO are infinite-length generation methods that process fixed-length frame blocks at each step and can generate long videos without increasing the number of frames per block. Compared to FIFO, DualParal has substantial memory advantages (model + KV activations) as DiT-based models scale.

Therefore, this quantitative analysis shows DualParal's superiority. Further details are in the Appendix E.

4 Experiments

4.1 Setups

Base model. In the experiments, the text-to-video model Wan2.1 (Wan et al. 2025) serves as the base model. Wan2.1 is a latest and representative DiT-based video foundation model renowned for its exceptional video generation performance. It is available in two versions: the Wan2.1-1.3B model, which generates 480p videos, and the Wan2.1-14B model, which can generate both 480p and 720p videos.

Metrics evaluation. To evaluate efficiency, we compare all methods in terms of generation latency and memory cost. Memory cost is measured as the peak memory overhead among all devices used during the diffusion process. For video performance, we apply VBench metrics (Huang et al. 2024) directly to long videos (Zhang, Yang, and Lim 2025). The metrics cover all indicators in the Video Quality category, including subject consistency, background consistency, temporal flickering, motion smoothness, dynamic range, aesthetic quality, and imaging quality.

Baselines. We compare DualParal with Ring Attention (Liu, Zaharia, and Abbeel 2023) and DeepSpeed-Ulysses (Jacobs et al. 2023), both officially supported by Wan2.1. We also evaluate it against Video-Infinity (Tan et al. 2024) and FIFO (Kim et al. 2024), two well-established parallel methods for long video generation.

Implementation details. By default, all diffusion parameters follow the original Wan2.1 inference settings, with 50 denoising steps. Experiments are conducted on Nvidia GeForce RTX 4090 (24GB) for Wan2.1-1.3B and Nvidia H20 (96GB, NVLink) for Wan2.1-14B. We use PyTorch's torch.distributed with Nvidia's NCCL backend for inter-GPU communication. We evaluate the efficiency of Wan2.1-1.3B (480p) and Wan2.1-14B (720p) in terms of latency and memory usage, and compare video performance across methods using Wan2.1-1.3B (480p). For DualParal, we perform one warmup step to ensure device communication.

4.2 Main results

Efficiency. We first evaluate all comparing methods on extremely long videos, followed by scalability analysis. For fair comparison, we set $Num_C = 8$ for DualParal, Video-Infinity and FIFO, and $Num_B = 8$ for DualParal and FIFO.

Method	Communication		Memory Cost	
	Cost	Overlap	Model	KV Activations
Ring Attention	$2O(p \times h)L$	✓	W	$F\frac{1}{N}KV$
DeepSpeed-Ulysses	$\frac{4}{N}O(p \times h)L$	✗	W	$F\frac{1}{N}KV$
Video-Infinity	$2O(Num_C \times H' \times W' \times h)L$	✗	W	$(F\frac{1}{N} + Num_C)KV$
FIFO	$2O((Num_B + Num_C) \times H \times W \times C)$	✓	W	$(Num_B + Num_C)KV$
DualParal (Ours)	$2O((Num_B + Num_C/2) \times H' \times W' \times h)$	✓	$\frac{1}{N}W$	$(Num_B + Num_C)KV$

Table 1: Comparison of parallel methods at a single diffusion step. 'Overlap' refers to degree of overlap between communication and computation. W is total memory cost of the model, while KV represents the memory cost for a single frame input.

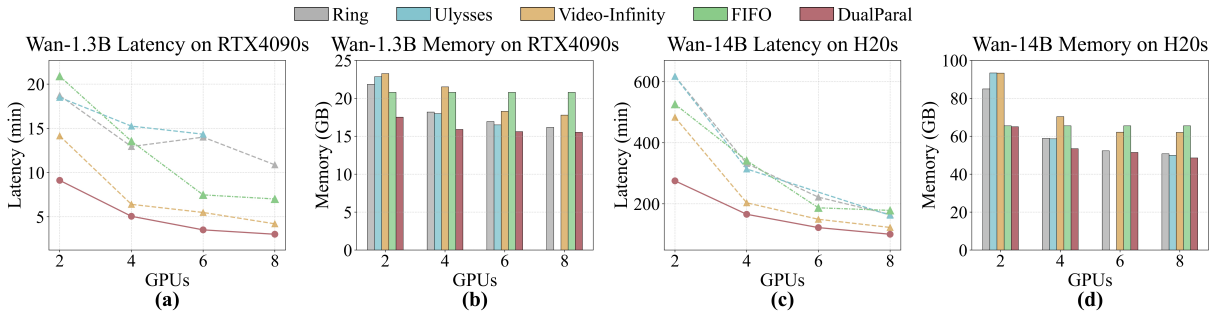

Figure 4: Scalability analysis in terms of latency and memory: (a) and (b) show the scalability of Wan2.1-1.3B (480p) across different methods on a 301-frame video, while (c) and (d) present the scalability of Wan2.1-14B (720p) on a 301-frame video.

Methods	Number of Frames	
	513	1025
Ring Attention	1328.3 / 17.66	3907.5 / 23.15
Video-Infinity	354.1 / 19.13	832.5 / 22.58
FIFO	565.3 / 20.59	1022.0 / 20.61
DualParal	**309.3 / 15.52**	**596.9 / 15.55**

Table 2: Efficiency evaluation on extreme-length video generation. Experiments are conducted on 8×RTX 4090 GPUs using Wan2.1-1.3B (480p). Results are reported as latency (s) / peak memory usage (GB).

For extremely long videos, as shown in Table 2, DualParal achieves great efficiency. Note that DeepSpeed-Ulysses is excluded due to incompatible attention head settings with 8 GPUs. Compared to static-length methods like Ring Attention and Video-Infinity, DualParal shows clear advantages at 513 frames and even more so at 1025 frames, achieving up to 6.54× lower latency and 1.48× lower memory usage. These gains stem from the fact that static-length generating methods require proportionally longer processing sequences as video length increases, leading to higher latency and memory consumption. In comparison to FIFO—another infinite-length video generation method—DualParal still achieves up to a 1.82× reduction in latency and a 1.32× reduction in memory usage at 513 frames.

To evaluate scalability, we measure generation latency and memory usage across multiple GPUs using various methods. Experiments are conducted on 301-frame video generation—the maximum length supported by 2 GPUs—and tested on Wan2.1-1.3B (480p) and Wan2.1-14B (720p). As shown in Figure 4, DualParal consistently outperforms all methods. For latency, shown in (a) and (c), DualParal achieves the lowest generation time across all tested GPU counts. Meanwhile, according to Equation 3, DualParal is expected to exhibit even better scalability for longer videos. For memory usage, shown in (b) and (d), DualParal maintains the lowest peak usage, with a steadily decreasing trend as the number of devices increases. This efficiency stems from DualParal's fixed memory footprint for KV activations and distributed model weights. In contrast, FIFO shows no memory scalability, posing challenges for large-scale video models. Although Ring Attention, DeepSpeed-Ulysses, and Video-Infinity benefit from reduced memory and latency with more devices, they still face scalability bottlenecks when generating longer videos, as shown in Table 2. Detailed analysis of Figure 4 is provided in the Appendix F.

Video quality. We compare the video quality generated by DualParal with that produced by DeepSpeed-Ulysses, Video-Infinity, and FIFO on Wan2.1-1.3B (480p). Table 3 presents a quantitative evaluation based on VBench (Huang et al. 2024). To ensure optimal video performance, we set $Num_C = 24$ with 16 local and 8 global paddings for Video-Infinity, $Num_C = 2$ and $Num_B = 2$ for FIFO, and $Num_C = 8$ and $Num_B = 8$ for DualParal.

Table 3 reports quantitative results for video generation at 129 and 257 frames. Since DeepSpeed-Ulysses and Ring Attention operate on full-length sequences without segmentation, we include only DeepSpeed-Ulysses as a representative. At 129 frames, DeepSpeed-Ulysses achieves

Method	Frames	Subj. cons.	Bg. cons.	Temp. flick.	Motion smth.	Dyn. deg.	Aesth. qual.	Img. qual.	Overall Score
DeepSpeed-Ulysses	129	**93.43**%	92.50%	98.88%	**98.57**%	**62.50**%	**62.96**%	**64.13**%	**81.85**%
Video-Infinity	129	82.35%	88.46%	98.41%	97.15%	59.72%	57.69%	63.91%	78.24%
FIFO	129	80.46%	90.13%	95.04%	95.30%	0%	48.28%	58.13%	75.89%
DualParal	129	92.92%	**95.68**%	**99.46**%	97.28%	59.72%	58.86%	62.59%	80.93%
DeepSpeed-Ulysses	257	**93.45**%	**95.07**%	98.05%	**98.45**%	23.61%	53.87%	55.86%	74.05%
Video-Infinity	257	86.49%	89.41%	98.36%	97.63%	**52.78**%	**58.88**%	**63.01**%	78.08%
FIFO	257	71.69%	85.41%	95.42%	95.19%	0%	48.61%	58.18%	64.93%
DualParal	257	89.15%	91.80%	**99.34**%	96.82%	50.00%	57.35%	62.73%	**78.17**%

Table 3: The comparison of various video generation methods, as benchmarked by VBench.

the best performance by preserving the full sequence and maintaining Wan2.1's original quality. However, its performance drops sharply at 257 frames due to exceeding Wan2.1's supported video length. In comparison, DualParal outperforms other distributed methods—including FIFO and Video-Infinity—at 129 frames and achieves the highest overall score at 257 frames. In Appendix G, we provide more statistical analysis on VBench and evaluation results on the VBench-Long version. More visualizations of video examples are provided in the Appendix B.

4.3 Ablation

Method	Latency (s)	Memory (GB)	Infinite Length
Queue	621.37	18.81	✓
Device Pipeline	472.25	21.18	✗
DualParal $_{\text{w/o cache}}$	182.71	16.76	✓
DualParal	**142.62**	**15.52**	✓

Table 4: Ablation study on DualParal. All settings are evaluated on a 129-frame video with 8×4090s.

Parallel ablation. The parallel architecture of DualParal consists of two main components: queue and device pipeline. By continuously feeding blocks from the queue into the device pipeline, supported by a feature cache mechanism, DualParal ensures efficient and seamless dual parallelization across devices. To evaluate the individual contributions of each component—queue, device pipeline, and feature cache—we conduct an ablation study focusing on latency, memory usage, and the ability to support infinite-length video generation. As shown in Table 4, using only the queue with a single GPU results in high latency and memory consumption. In contrast, relying solely on the device pipeline cannot support infinite-length generation and remains inefficient due to underutilization of GPUs when processing a single input. The complete DualParal architecture, integrating both the queue and device pipeline without cache, successfully addresses these limitations, achieving superior efficiency. With the addition of the feature cache, efficiency is further enhanced, enabling the generation of minute-long videos with ease.

Effectiveness of coordinated noise initialization. Through the two key observations for DiT-based video models discussed in Section 3.3, we utilize the complete noise space and avoid using the same noise throughout the process to maintain global consistency among non-overlapping blocks. In this part, we verify the effectiveness of this approach. As shown in Figure 5, without applying noise initialization, DualParal fails to maintain temporal consistency, as seen in (a) and (b). Although (b) incorporates neighboring blocks to partially mitigate the inconsistency, the effect is limited. In contrast, after introducing noise initialization, as shown in (c), DualParal achieves significantly improved temporal consistency across frames. In Appendix H, we provide a further evaluation of different noise initialization strategies on VBench.

Prompt: A cat and a dog baking a cake together in kitchen. The cat is measuring flour, while the dog is stirring the batter with a wooden spoon. The kitchen is cozy, with sunlight streaming through the window.

Figure 5: Video frames under different conditions: (a) $Num_C = 0$ without noise initialization; (b) $Num_C = 8$ without noise initialization; (c) $Num_C = 8$ with coordinated noise initialization.

5 Conclusion

In this paper, we propose **DualParal**, a novel distributed inference strategy for DiT-based video diffusion models. By implementing a *block-wise denoising* scheme, DualParal successfully parallelizes both temporal frames and model layers across GPUs, resulting in high efficiency for long video generation. To further enhance efficiency and video quality, DualParal reuses KV features via a feature cache strategy for reducing communication and computational redundancy, and applies coordinated noise initialization to ensure global consistency without extra cost. These designs together enable efficient generation of minute-long videos.

Acknowledgments

This project is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award Number: MOE-T2EP20122-0006), and by the Presidential Young Scholars Scheme (Project ID: IDP0058232) from The Hong Kong Polytechnic University.

References

Arriola, M.; Sahoo, S. S.; Gokaslan, A.; Yang, Z.; Qi, Z.; Han, J.; Chiu, J. T.; and Kuleshov, V. 2025. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Bao, F.; Nie, S.; Xue, K.; Cao, Y.; Li, C.; Su, H.; and Zhu, J. 2023. All Are Worth Words: A ViT Backbone for Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Fang, J.; Pan, J.; Sun, X.; Li, A.; and Wang, J. 2024a. xDiT: an Inference Engine for Diffusion Transformers (DiTs) with Massive Parallelism. *arXiv preprint arXiv:2411.01738*.

Fang, J.; Pan, J.; Wang, J.; Li, A.; and Sun, X. 2024b. PipeFusion: Patch-level Pipeline Parallelism for Diffusion Transformers Inference. *arXiv preprint arXiv:2405.14430*.

Hong, W.; Ding, M.; Zheng, W.; Liu, X.; and Tang, J. 2023. CogVideo: Large-scale Pretraining for Text-to-Video Generation via Transformers. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Huang, Y.; Cheng, Y.; Bapna, A.; Firat, O.; Chen, D.; Chen, M.; Lee, H.; Ngiam, J.; Le, Q. V.; Wu, Y.; and Chen, z. 2019. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Huang, Z.; He, Y.; Yu, J.; Zhang, F.; Si, C.; Jiang, Y.; Zhang, Y.; Wu, T.; Jin, Q.; Chanpaisit, N.; Wang, Y.; Chen, X.; Wang, L.; Lin, D.; Qiao, Y.; and Liu, Z. 2024. VBench: Comprehensive Benchmark Suite for Video Generative Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jacobs, S. A.; Tanaka, M.; Zhang, C.; Zhang, M.; Song, S. L.; Rajbhandari, S.; and He, Y. 2023. DeepSpeed Ulysses: System Optimizations for Enabling Training of Extreme Long Sequence Transformer Models. *arXiv preprint arXiv:2309.14509*.

Kim, J.; Kang, J.; Choi, J.; and Han, B. 2024. FIFO-Diffusion: Generating Infinite Videos from Text without Training. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Kong, W.; Tian, Q.; Zhang, Z.; Min, R.; Dai, Z.; Zhou, J.; Xiong, J.; Li, X.; Wu, B.; Zhang, J.; et al. 2024. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*.

Li, Z.; Zhuang, S.; Guo, S.; Zhuo, D.; Zhang, H.; Song, D.; and Stoica, I. 2021. TeraPipe: Token-Level Pipeline Parallelism for Training Large-Scale Language Models. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

Liu, H.; Zaharia, M.; and Abbeel, P. 2023. Ring Attention with Blockwise Transformers for Near-Infinite Context. *arXiv preprint arXiv:2310.01889*.

Peebles, W.; and Xie, S. 2023. Scalable Diffusion Models with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Peng, X.; Zheng, Z.; Shen, C.; Young, T.; Guo, X.; Wang, B.; Xu, H.; Liu, H.; Jiang, M.; Li, W.; Wang, Y.; Ye, A.; Ren, G.; Ma, Q.; Liang, W.; Lian, X.; Wu, X.; Zhong, Y.; Li, Z.; Gong, C.; Lei, G.; Cheng, L.; Zhang, L.; Li, M.; Zhang, R.; Hu, S.; Huang, S.; Wang, X.; Zhao, Y.; Wang, Y.; Wei, Z.; and You, Y. 2025. Open-Sora 2.0: Training a Commercial-Level Video Generation Model in 200k. *arXiv preprint arXiv:2503.09642*.

Qi, P.; Wan, X.; Huang, G.; and Lin, M. 2024. Zero Bubble (Almost) Pipeline Parallelism. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Qiu, H.; Xia, M.; Zhang, Y.; He, Y.; Wang, X.; Shan, Y.; and Liu, Z. 2024. FreeNoise: Tuning-Free Longer Video Diffusion via Noise Rescheduling. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. O. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ruhe, D.; Heek, J.; Salimans, T.; and Hoogeboom, E. 2024. Rolling Diffusion Models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Sand-AI. 2025. MAGI-1: Autoregressive Video Generation at Scale.

Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; and Catanzaro, B. 2020. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. arXiv:1909.08053.

Tan, Z.; Liu, S.; Yang, X.; Xue, Q.; and Wang, X. 2025a. OminiControl: Minimal and Universal Control for Diffusion Transformer.

Tan, Z.; Xue, Q.; Yang, X.; Liu, S.; and Wang, X. 2025b. OminiControl2: Efficient Conditioning for Diffusion Transformers. *arXiv preprint arXiv:2503.08280*.

Tan, Z.; Yang, X.; Liu, S.; and Wang, X. 2024. Video-Infinity: Distributed Long Video Generation. *arXiv preprint arXiv:2406.16260*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wan, T.; Wang, A.; Ai, B.; Wen, B.; Mao, C.; Xie, C.-W.; Chen, D.; Yu, F.; Zhao, H.; Yang, J.; Zeng, J.; Wang, J.; Zhang, J.; Zhou, J.; Wang, J.; Chen, J.; Zhu, K.; Zhao, K.; Yan, K.; Huang, L.; Feng, M.; Zhang, N.; Li, P.; Wu, P.; Chu, R.; Feng, R.; Zhang, S.; Sun, S.; Fang, T.; Wang, T.; Gui, T.; Weng, T.; Shen, T.; Lin, W.; Wang, W.; Wang, W.; Zhou, W.; Wang, W.; Shen, W.; Yu, W.; Shi, X.; Huang, X.; Xu, X.; Kou, Y.; Lv, Y.; Li, Y.; Liu, Y.; Wang, Y.; Zhang, Y.; Huang, Y.; Li, Y.; Wu, Y.; Liu, Y.; Pan, Y.; Zheng, Y.; Hong,

Y.; Shi, Y.; Feng, Y.; Jiang, Z.; Han, Z.; Wu, Z.-F.; and Liu, Z. 2025. Wan: Open and Advanced Large-Scale Video Generative Models. *arXiv preprint arXiv:2503.20314*.

Zhang, S.; Yang, H.; and Lim, S.-N. 2025. VideoMerge: Towards Training-free Long Video Generation. *arXiv preprint arXiv:2503.09926*.

Zheng, Z.; Peng, X.; Yang, T.; Shen, C.; Li, S.; Liu, H.; Zhou, Y.; Li, T.; and You, Y. 2024. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*.