

StreamDiffusionV2: A Streaming System for Dynamic and Interactive Video Generation

Tianrui Feng^{1,*}, Zhi Li¹, Shuo Yang¹, Haocheng Xi¹, Muyang Li², Xiuyu Li¹, Lvmi Zhang³, Keting Yang⁴, Kelly Peng⁵, Song Han², Maneesh Agrawala³, Kurt Keutzer¹, Akio Kodaira¹, Chenfeng Xu^{1,6,†}

¹UC Berkeley, ²MIT, ³Stanford University, ⁴Independent Researcher, ⁵First Intelligence, ⁶UT Austin

Generative models are reshaping the live-streaming industry by redefining how content is created, styled, and delivered. Previous image-based streaming diffusion models have powered efficient and creative live streaming products but has hit limits on temporal consistency due to the foundation of image-based designs. Recent advances in video diffusion have markedly improved temporal consistency and sampling efficiency for offline generation. However, offline generation systems primarily optimize throughput by batching large workloads. In contrast, live online streaming operates under strict service-level objectives (SLOs): time-to-first-frame must be minimal, and every frame must meet a per-frame deadline with low jitter. Besides, scalable multi-GPU serving for real-time streams remains largely unresolved so far. To address this, we present **StreamDiffusionV2**, a *training-free* pipeline for interactive live streaming with video diffusion models. StreamDiffusionV2 integrates an SLO-aware batching scheduler and a block scheduler, together with a sink-token-guided rolling KV cache, a motion-aware noise controller, and other system-level optimizations. Moreover, we introduce a scalable pipeline orchestration that parallelizes the diffusion process across denoising steps and network layers, achieving near-linear FPS scaling without violating latency guarantees. The system scales seamlessly across heterogeneous GPU environments and supports flexible denoising steps (e.g., 1–4), enabling both ultra-low-latency and higher-quality modes. Without TensorRT or quantization, StreamDiffusionV2 renders the first frame within 0.5s and attains 58.28 FPS with a 14B-parameter model and 64.52 FPS with a 1.3B-parameter model on four H100 GPUs. Even when increasing denoising steps to improve quality, it sustains 31.62 FPS (14B) and 61.57 FPS (1.3B), making state-of-the-art generative live streaming practical and accessible—from individual creators to enterprise-scale platforms.

Project page: <https://streamdiffusionv2.github.io/>

[†]Project lead, corresponding to xuchenfeng@utexas.edu

*Work done when visiting UC Berkeley.

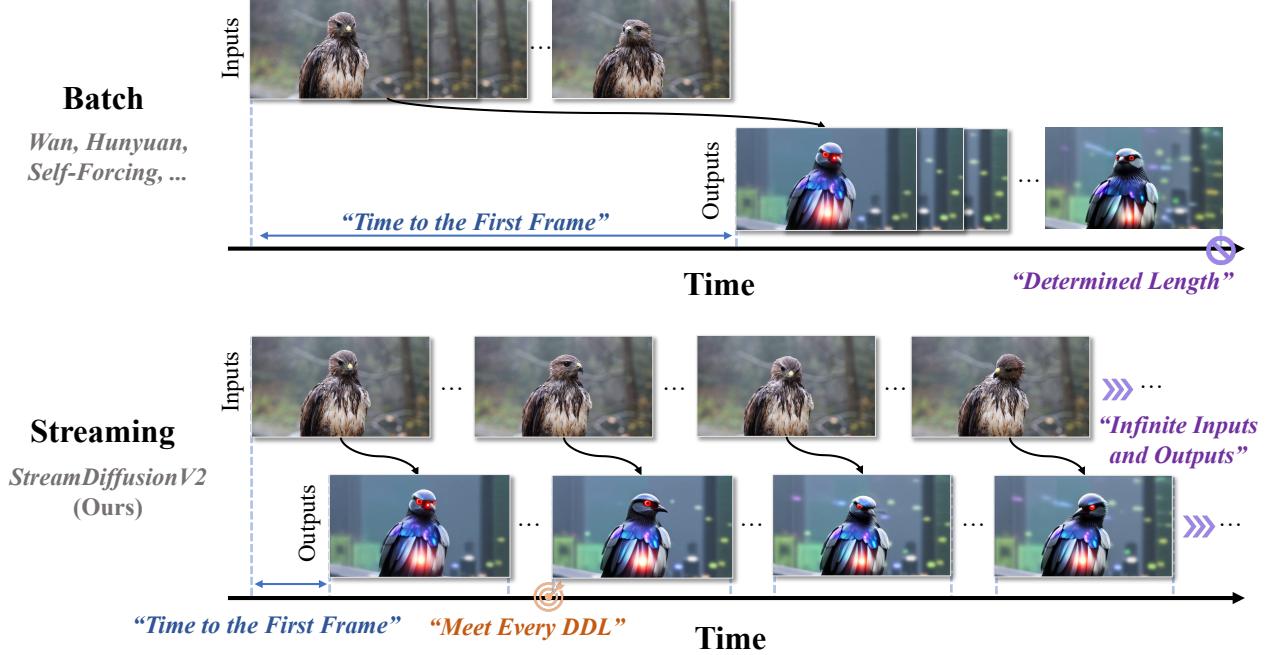
StreamDiffusion

1 Introduction

Recent advances in diffusion models have transformed the way live video content is created and rendered. Modern live AI streaming pipelines, such as Daydream¹ and TouchDesigner², are largely powered by image-diffusion-based methods (Kodaira et al., 2023; Liang et al., 2024) due to their flexible style control and responsiveness and ease of integration with real-time applications. With simple text prompts, creators can restyle scenes, replace backgrounds, add effects, and even animate virtual hosts that respond to live chat. These pipelines can also repurpose the same input footage into derivative formats such as short clips and thumbnails, dramatically reducing post-production effort. By combining low cost, high speed, and interactive visual fidelity, image-diffusion streaming has redefined traditional live-streaming workflows and enabled new creative use cases in gaming, social media, and live entertainment.

¹<https://blog.livepeer.org/introducing-daydream/>

²<https://derivative.ca/tags/streamdiffusion>



Prompt: A highly detailed futuristic cybernetic bird, blending avian elegance with advanced robotics.

Figure 1 Comparison between Batch and Streaming video generation. Different from generating a large batch of video, live-streaming video generation targets at cutting down the “time to the first frame” and producing continuous output with low latency.

However, their image-centric design exposes a fundamental weakness: poor temporal consistency. Frame-by-frame generation introduces flicker and drift, causing noticeable instability across time. In practice, even commercial-grade systems still struggle to maintain coherent motion and appearance in continuous streams.

In parallel, video diffusion models (Huang et al., 2025; Yin et al., 2025; Wan et al., 2025) have achieved far stronger temporal consistency through explicit modeling of temporal dependencies. Recent efforts have pushed these models towards fast, even “real-time” generation (Xi et al., 2025; Yang et al., 2025b,a), making them promising candidates for streaming. A natural question arises: can these models replace image-based diffusion in real-time AI video pipelines?

Despite their efficiency improvements, current video diffusion systems remain fundamentally ill-suited for live streaming. Our systematic analysis identifies four key challenges:

- **Unmet real-time SLOs.** State-of-the-art video diffusion models (e.g., WAN (Wan et al., 2025)) and their efficient variants (Huang et al., 2025; Yin et al., 2025) are optimized for offline generation throughput by processing large, fixed inputs of shape $1 \times T \times H \times W$ per forward pass, where T often ranges from 81 to hundreds of frames. Such large chunks violate real-time service-level objectives (SLOs) (Sripanidkulchai et al., 2004; Zhang et al., 2005; Huang et al., 2008), which requires minimal time-to-first-frame (TTFF) and strict per-frame deadlines (see Fig. 1). The fixed input size also prevents adaptive scheduling under varying hardware load, making SLO compliance difficult across heterogeneous GPU environments.
- **Drift over unbounded horizons.** Existing video diffusion pipelines generate fixed-length clips using static configurations, including KV caches, sink tokens, and RoPE schedules, calibrated for bounded temporal contexts. These assumptions break down in continuous live streams, where temporal context, content statistics, and user prompts evolve dynamically. Over long horizons (e.g., hour-long session), these static components accumulate misalignment, leading to visual drift and degraded temporal coherence.
- **Motion tearing under high-speed dynamics.** Current video diffusion models are predominantly trained on slow-motion datasets (Bain et al., 2021; Chen et al., 2024c; Yang et al., 2024), leaving them poorly adapted to fast motion. Their inference pipelines often use large chunk sizes with rule-based noise

schedules to suppress inter-frame variation, which over-smooths motion and erodes fine details. As a result, live streams that feature rapid camera movement or action sequences suffer from blur, ghosting, and motion tearing.

- **Poor GPU scaling.** As AI-driven live streaming scales from individual creators to enterprise-grade deployments, multi-GPU scaling becomes critical. However, existing offline diffusion parallelization strategies do not translate to real-time workloads. Sequence parallelism suffers from unpredictable communication overhead, while naive pipeline parallelism yields minimal improvement in FPS. Because per-frame latency constraints dominate real-time workloads, achieving scalable, low-jitter performance remains an open system challenge.

These challenges call for a new inference system designed for real-time constraints rather than offline throughput.

We introduce **StreamDiffusionV2**, a training-free streaming system that adapts video diffusion models for interactive, low-latency generation. Our objective is stringent: *to satisfy live-streaming SLOs — low time-to-first-frame and strict per-frame deadlines (DDL) — while preserving temporal consistency and visual fidelity over long, dynamic sequences, and more importantly, to provide a scalable solution that serves users at different levels of compute capacity.*

StreamDiffusionV2 synergizes several techniques to achieve both efficiency and visual quality objectives. **Efficiency objectives:** **(1)** To satisfy live-streaming SLOs on a single GPU, instead of using a fixed input of $1 \times T \times H \times W$, we employ an SLO-aware *batching scheduler* that reformulates inputs as $B \times T' \times H \times W$. We intentionally keep T' small (e.g., only a few frames) to limit per-step latency and meet per-frame deadlines (DDL), while adapting the *stream batch size* B (Kodaira et al., 2023) to instantaneous hardware load to maximize utilization. **(2)** To deliver scalable performance across multiple GPUs, we introduce a dynamic scheduler that orchestrates the pipeline across both denoising steps and network stages. Naive pipeline parallelism alone does not yield linear FPS scaling with additional GPUs. We pair it with the SLO-aware batching scheduler above to keep all devices well utilized under SLO-aware workloads, thereby improving aggregate FPS while maintaining latency guarantees. **Vision quality objectives:** **(1)** To support unbounded live-streaming use cases, we continuously update the sink tokens to reflect the current prompt semantics and recent visual context, and we refresh anchor caches when topics or motion regimes change. We also reset RoPE offsets at chunk boundaries to avoid positional misalignment over long horizons. Together, these controls preserve appearance and motion semantics while retaining the latency benefits. **(2)** To handle high-speed motion, we introduce a motion-aware noise scheduler that estimates motion magnitude (e.g., via lightweight optical-flow proxies) and adapts the noise level and timestep schedule on a per-chunk basis. Specifically, fast motion receives more conservative denoising to suppress tearing and ghosting, while slow motion benefits from more aggressive refinement to recover fine details. This design substantially improves sharpness and temporal stability for high-speed content in live streams.

StreamDiffusionV2 integrates the aforementioned multiple system-level techniques into a cohesive, training-free pipeline that transforms efficient video diffusion models into real-time, stream-live applications. These components enable high FPS, high visual fidelity, and strong temporal consistency for AI-driven live streaming. Without relying on TensorRT or quantization, StreamDiffusionV2 is able to achieve time-to-first-frame within 0.5 seconds and is the first system to achieve 58.28 FPS with a 14B-parameter model and 64.52 FPS with a 1.3B-parameter model, both running on four H100 GPUs. Even when increasing the number of denoising steps to enhance generation quality, the system maintains 31.62 FPS (14B) and 61.57 FPS (1.3B). Beyond raw speed, StreamDiffusionV2 provides a tunable trade-off across resolution, denoising steps, and GPU scale, allowing users to balance quality, throughput, and resource constraints. This flexibility supports a wide range of deployment scenarios, from individual creators using a single GPU to enterprise-scale platforms operating large GPU clusters. StreamDiffusionV2 establishes a practical foundation for next-generation live generative media systems. We will release our implementation to promote open research and foster innovation in real-time interactive video generation.

2 Related Works

Efficient Offline Video Generation. Diffusion-based video generation achieves impressive visual fidelity, but the inference latency of Video DiTs remains a key bottleneck. Prior work tackles this from two complementary angles: training-based and training-free methods. Training-based approaches—such as distillation (Salimans and Ho, 2022; Kim et al., 2023; Meng et al., 2023; Yin et al., 2024b,a; Lu and Song, 2024) and linear attention (Gao et al., 2024; Chen et al., 2024a; Dalal et al., 2025; Xie et al., 2024; Chen et al., 2025b; Po et al., 2025)—optimize the model to shorten the diffusion process or reparameterize attention for faster inference. In contrast, training-free methods—including cache reuse (Selvaraju et al., 2024; Chen et al., 2024b; Wimbauer et al., 2024; Liu et al., 2024; Zhou et al., 2025) and sparse attention (Xi et al., 2025; Yang et al., 2025b; Zhang et al., 2025b,a,c)—improve runtime by reusing latent features or sparsifying attention computation. Together, these techniques substantially reduce Video DiT latency in offline scenarios. However, despite strong offline speedups, they do not transfer directly to streaming, which requires unbounded (infinite) queues and strict low-latency, online generation.

Streaming Video Generation. Autoregressive video generation (Chen et al., 2025a; Teng et al., 2025; Kodaira et al., 2025; Yin et al., 2025; Huang et al., 2025; Yang et al., 2025a) is well aligned with our streaming setting: the forward-only, continuous process produces frames sequentially with significantly faster speed compared to offline generation. Among these approaches, CausVid (Yin et al., 2025) distills a student-initialized bidirectional DiT into a causal DiT, while Self-Forcing (Huang et al., 2025) employs self-rollout training to curb error accumulation. A series of works (Yang et al., 2025a; He et al., 2025; Kodaira et al., 2025; Valevski et al.) then extend the idea of a few-step autoregressive paradigm to support infinite-length synthesis for fast streaming generation. Despite substantial progress toward live-streaming, existing methods still fall short of streaming SLOs. More importantly, although fast, both CausVid and Self-Forcing struggle with high-speed motion: their training biases toward motion smoothing can introduce temporal over-smoothing and visible tearing during streaming. To address these gaps, StreamDiffusionV2 introduces a serving pipeline that adapts state-of-the-art efficient video-diffusion models to the streaming setting and is designed to meet strict SLOs.

Parallel Streaming Inference Serving. Live streaming is critical not only for end users but, more importantly, for broadcasters and platforms operating cloud-scale services, where parallelism is essential to unlock capacity and quality. To accelerate diffusion models on multi-GPU systems (Shih et al., 2023; Li et al., 2024; Fang et al., 2024a,b), two complementary strategies—Sequence Parallelism (SP) and Pipeline Parallelism (PP)—are widely used. DeepSpeed-Ulysses (Jacobs et al., 2023), Ring Attention (Liu et al., 2023), and their combination (Fang et al., 2024a) realize sequence-parallel attention by sharding tokens across GPUs: Ulysses employs all-to-all communication to parallelize attention, whereas Ring Attention circulates key-value blocks in a ring topology to reduce communication. Distrifusion (Li et al., 2024) further introduces an asynchronous variant that overlaps the computation between spatiotemporal patches and timesteps. PipeFusion (Fang et al., 2024b) applies pipeline parallelism with similar insights and achieves competitive efficiency. In our live-streaming framework, we find that barely utilizing either sequence-parallelism or pipeline parallelism leads to low FPS. We adopt a tailored parallelization scheme that explicitly balances compute and communication across heterogeneous hardware to meet strict latency and throughput targets.

3 Motivation & Bottleneck Analysis

Real-time video applications span diverse use cases with widely varying budgets for frame rate, resolution, latency, and motion. This heterogeneity shifts performance bottlenecks across different stages of the pipeline. We highlight four key bottlenecks below.

Fixed-size input cannot satisfy real-time SLOs. Existing streaming systems adopt a fixed-input strategy, processing tens to hundreds of frames per forward pass to maximize throughput. For instance, CausVid and Self-Forcing process 81 frames per step. While this large-chunk design improves average throughput in offline settings, it fundamentally conflicts with the requirements of real-time streaming.

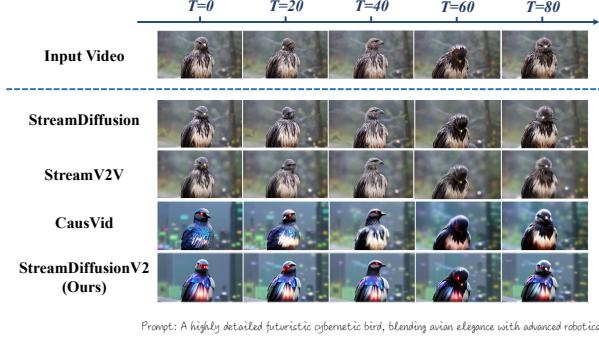


Figure 2 Generation results among various approaches. The examples above are frames picked from transferred videos among different methods, where the frame index is denoted as T .

Let B denote batch size, T the number of frames per forward pass, (H, W) the video resolution, P_{model} the model size (in parameters), ρ_{VAE} the pixel-to-token ratio, and C_{device} the average device throughput (FLOPs/s). Then, the time-to-first-frame (TTFF) can be approximated as:

$$\text{TTFF} \approx \frac{2 B T H W P_{\text{model}}}{C_{\text{device}} \rho_{\text{VAE}}}. \quad (1)$$

On a single H100 GPU, generating a 480p video with an 81-frame chunk using a 1.3B-parameter model yields a theoretical TTFF of 5.31s, which closely matches our measurements in Figure 10. This delay far exceeds typical live-streaming targets ($\approx 1\text{s}$ ³).

The excessive latency is due to the fixed large input size, which does not adapt to real-time constraints. Without adaptive input scheduling, such pipelines cannot satisfy the latency requirements of live-streaming SLOs.

Drift accumulation in long-horizon generation. Current “streaming” video systems are primarily adapted from offline, bidirectional clip generators. For example, CausVID Yin et al. (2025) and Self-Forcing Huang et al. (2025) are derived from Wan-2.1-T2V (Wan et al., 2025). These models are trained for short clips (5–10 seconds) and maintain coherence only within that range (Huang et al., 2025). Beyond this horizon, quality degrades rapidly (see Fig. 2), as the temporal context is treated *statically*: the sink tokens become stale, RoPE accumulates positional drift, and the fixed context windows do not adapt to evolving content statistics. Over long durations, such compounding errors make these architectures inherently unsuitable for continuous live streaming.

Quality degradation due to motion unawareness. Different motion patterns in the input stream impose distinct tradeoffs between latency and visual quality. Fast motion requires conservative denoising to prevent tearing, ghosting, and blur, whereas slow or static scenes benefit from stronger refinement to

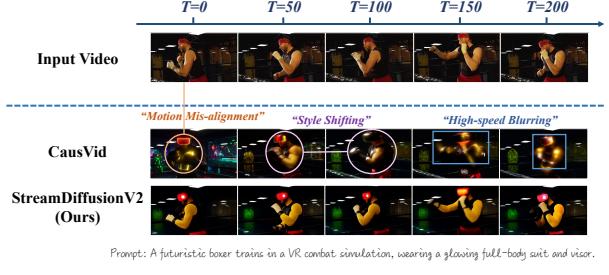


Figure 3 Generation results of CausVid and ours. The examples above are frames picked from transferred videos generated from CausVid and StreamDiffusionV2. We utilize the 1.3B model to produce the results.

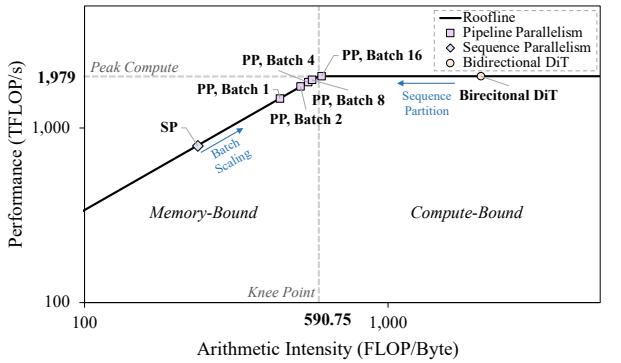


Figure 4 Roofline analysis of sequence parallelism and our pipeline orchestration. We compare the Sequence Parallelism and Pipeline Parallelism under varying batch sizes in the causal DiT, compared with the bidirectional DiT. The results demonstrate that our approach operates near the knee point of the roofline, effectively avoiding compute underutilization as seen in the bidirectional DiT and memory bandwidth limitations in Sequence Parallelism. The model is profiled on an NVIDIA H100 SXM GPU with a peak performance of 1,979 TFLOP/s and a knee point at an arithmetic intensity (AI) of 590.75 FLOP/Byte. The token length is 1,536 (a 4-frame chunk at 480P resolution.)

³<https://www.mux.com/docs/guides/data-startup-time-metric>

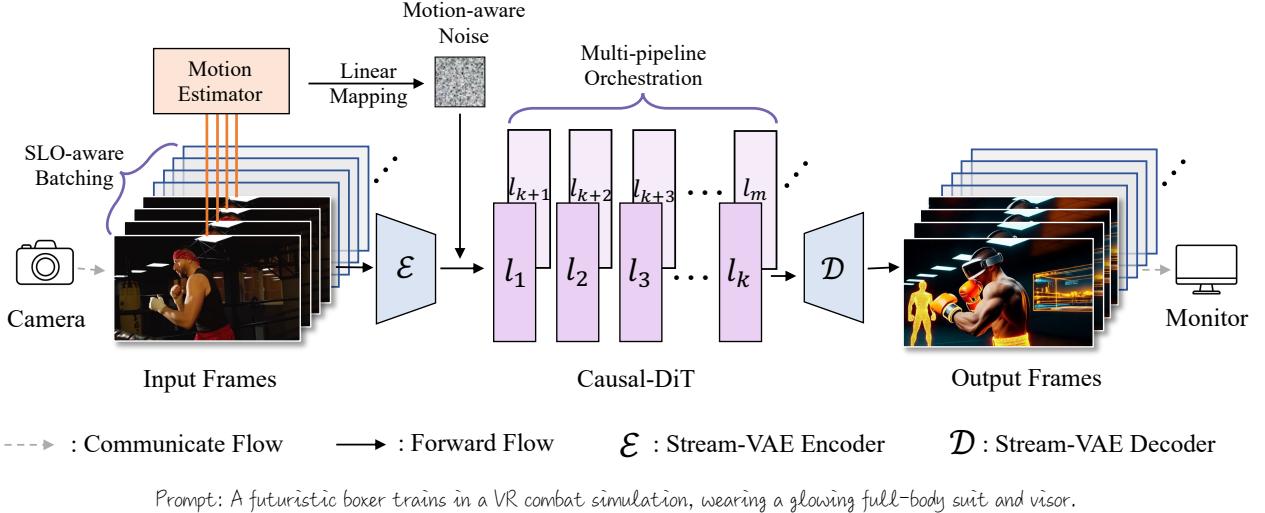


Figure 6 The overview pipeline of our StreamDiffusionV2. (1) **Efficiency.** We pair an SLO-aware batching scheduler (controlling input size) with a pipeline orchestration that balances latency and FPS, ensuring each frame meets its deadline and TTFF under strict service constraints. (2) **Quality.** We deploy a motion-aware noise controller to mitigate high-speed tearing, and combine adaptive sink tokens with RoPE refreshing to deliver high-quality user interaction and hours-level streaming stability.

recover details. Existing streaming pipelines rely on *fixed* noise schedules that ignore this variability, leading to temporal artifacts in high-motion regions and reduced visual fidelity in low-motion segments (see Fig. 3).

Poor GPU scaling under per-frame latency constraints. In live stream scenarios, strict per-frame deadlines hinder the scalability of conventional parallelization strategies for two key reasons: (i) communication latency in sequence parallelism significantly reduces potential speedup, and (ii) short-frame chunks drive the workload into a memory-bound regime, as shown in Fig. 4. These effects are further amplified in real-time streaming, where efficient causal DiTs operate on short sequences (e.g., 4 frames per step), reducing per-frame computation and making communication overhead proportionally heavier (see Fig. 5).

4 Methodology

In this section, we introduce StreamDiffusionV2, a training-free streaming system that achieves both real-time efficiency and long-horizon visual stability. At a high level, our design is based on two key layers of optimization: (1) **real-time scheduling and quality control**, which integrates SLO-aware batching, adaptive sink and RoPE refresh, and motion-aware noise scheduling to meet per-frame deadlines while maintaining long-horizon temporal coherence and visual fidelity; and (2) **scalable pipeline orchestration**, which parallelizes the diffusion process across denoising steps and network stages to achieve near-linear FPS scaling without violating latency guarantees. Additionally, we investigate several lightweight system-level optimizations, including DiT block scheduler, stream-VAE, and asynchronous communication overlap, which further enhance throughput and stability during long-running live streams.

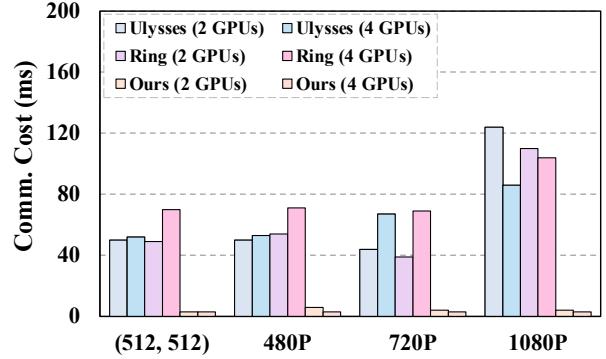


Figure 5 Communication consumption of various parallelism methods. We measure the communication latency by testing the parallel inference latency and theoretical latency (sequence or block partitioning without communication) on NVlink-connected H100 GPUs.

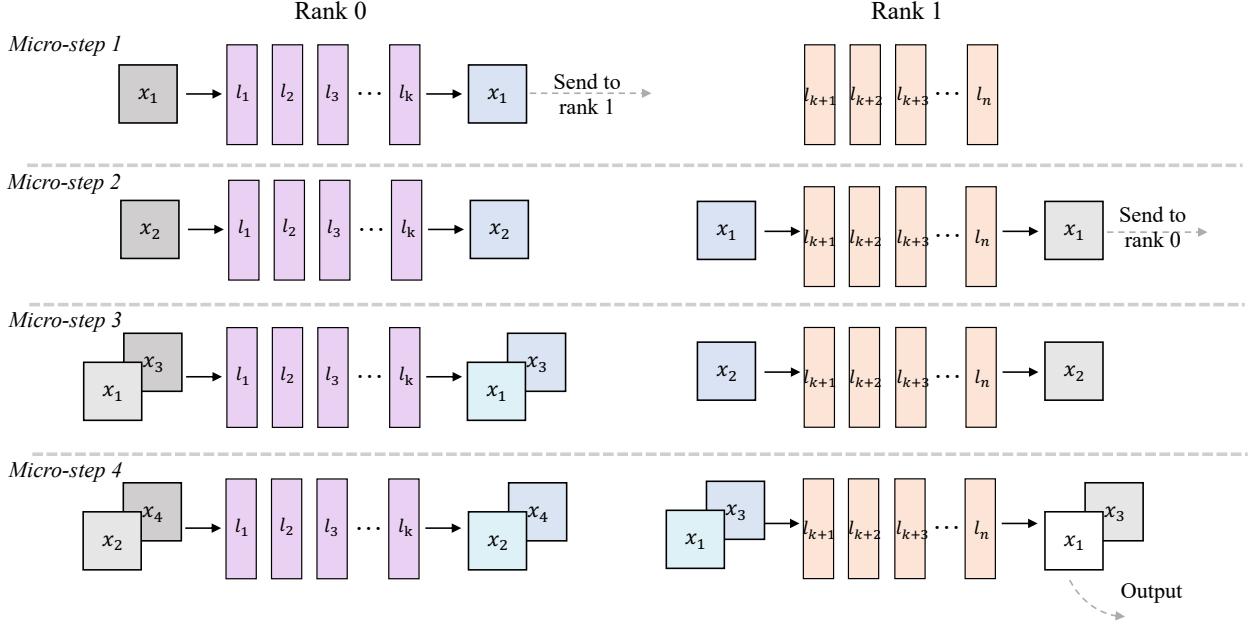


Figure 7 The detailed design of our Pipeline-parallel Stream-Batch architecture. The DiT blocks are distributed across multiple devices for pipeline parallelism, while the Stream-Batch strategy is applied within each stage. Different colors denote distinct latent streams, illustrating the communication structure, and the depth indicates the corresponding noise levels. Our implementation guarantees the generation of a clean latent at every micro-step during inference.

4.1 Real-time scheduling and quality control

As illustrated in Figure 6, StreamDiffusionV2 achieves real-time video generation through three key components: (1) an SLO-aware batching scheduler that dynamically adjusts the stream batch size to satisfy per-frame deadlines while maximizing GPU utilization; (2) an adaptive sink and RoPE refresh mechanism that mitigates long-horizon drift by periodically resetting temporal anchors and positional offsets; and (3) a motion-aware noise scheduler that adapts the denoising trajectory to motion magnitude, ensuring sharpness and temporal stability across diverse motion regimes.

SLO-aware batching scheduler. To satisfy the Service-Level Objective (SLO) while maximizing GPU utilization, we propose an SLO-aware batching scheduler that dynamically adjusts the batch size. Given a target frame rate f_{SLO} , the system processes T frames per iteration, with the overall inference latency depending on both the chunk size T and batch size B , denoted as $L(T, B)$. To ensure real-time processing, the product $B \cdot T$ must not exceed the number of frames already collected from the input stream. As analyzed in Section 3, the model operates in a memory-bound regime, and the inference latency can be approximated as:

$$L(T, B) \approx \frac{A(T, B) + P_{\text{model}}}{\eta \text{BW}_{\text{HBM}}}, \quad (2)$$

where $A(T, B)$ denotes the activation memory footprint, P_{model} represents the memory volume of the model parameters, and $\eta \text{BW}_{\text{HBM}}$ is the effective memory bandwidth with the utilization factor η ($0 < \eta \leq 1$). With FlashAttention (Dao et al., 2022), the activation term $A(T, B)$ scales linearly as $\mathcal{O}(BT)$, resulting in a proportional latency growth $L(T, B)$. The achieved processing frequency can therefore be expressed as $f = BT/L(T, B) \propto \frac{B}{1+B}$, which increases with larger batch size B as GPU utilization improves. As the system approaches the knee point of the roofline model (Fig. 4)—marking the transition from the memory-bound to the compute-bound regime—the scheduler adaptively converges to an optimal batch size B^* that maximizes throughput efficiency.

Adaptive sink and RoPE refresh. To address the *drifting issue* discussed in Section 3, we introduce an adaptive sink token update and a RoPE refresh policy that jointly maintain long-horizon stability during

continuous video generation. Unlike prior methods such as Self-Forcing (Huang et al., 2025), which fix the sink set throughout generation, StreamDiffusionV2 dynamically updates the sink tokens based on the evolving prompt semantics. Let $\mathcal{S}_t = \{s_1^t, \dots, s_m^t\}$ denote the sink set at chunk t . Given a new chunk embedding \mathbf{h}_t , the system computes similarity scores $\alpha_i = \cos(\mathbf{h}_t, s_i^{t-1})$ and refreshes the least similar sinks: $s_i^t = s_i^{t-1}$ if $\alpha_i \geq \tau$, and $s_i^t = \mathbf{h}_t$ otherwise, where τ is a similarity threshold. In practice, we find that τ should be set large to ensure alignment with the evolved text. To prevent positional drift caused by accumulated RoPE offsets over long sequences, we periodically reset the RoPE phase once the current frame index t exceeds a threshold T_{reset} , i.e., $\theta_t = \theta_t$ if $t \leq T_{\text{reset}}$ and $\theta_t = \theta_{t-T_{\text{reset}}}$ otherwise.

Motion-aware noise scheduler. To handle diverse motion dynamics in live-streaming videos, we propose a motion-aware noise scheduler that adaptively regulates the denoising noise rate according to the estimated motion magnitude of recent frames. As illustrated in Fig. 8, we estimate the motion magnitude between consecutive frames using a frame-difference metric. Given consecutive latent frames $\mathbf{v}_t, \mathbf{v}_{t-1} \in \mathbb{R}^{C \times H \times W}$, the motion intensity d_t is

$$d_t = \sqrt{\frac{1}{CHW} \|\mathbf{v}_t - \mathbf{v}_{t-1}\|_2^2}.$$

To stabilize this measurement over a short temporal window of k frames, we normalize it by a statistical scale factor σ and clip it into $[0, 1]$:

$$\hat{d}_t = \text{clip}\left(\frac{1}{\sigma} \max_{i \in \{t-k, \dots, t\}} d_i, 0, 1\right).$$

The normalized \hat{d}_t determines how aggressively the system should denoise the current chunk. A higher \hat{d}_t (fast motion) corresponds to a more conservative denoising schedule, while a lower \hat{d}_t (slow or static motion) allows stronger refinement for sharper details. Finally, we smooth the noise rate s_t using an exponential moving average (EMA) to ensure gradual temporal transitions:

$$s_t = \lambda \left[s_{\max} - (s_{\max} - s_{\min}) \hat{d}_t \right] + (1 - \lambda) s_{t-1},$$

where $0 < \lambda < 1$ controls the update rate, and s_{\max} and s_{\min} denote the upper and lower bounds of the noise rate.

4.2 Scalable pipeline orchestration

Multi-Pipeline orchestration scaling. To improve system throughput on multi-GPU platforms, we propose a scalable pipeline orchestration for parallel inference. Specifically, the DiT blocks are partitioned across devices. As illustrated in Fig. 7, each device processes its input sequence as a micro-step and transmits the results to the next stage within a ring structure. These enable consecutive stages of the model to operate concurrently in a pipeline-parallel manner, achieving near-linear acceleration for DiT throughput.

Notably, the pipeline-parallel inference adds inter-stage communication, which, together with activation traffic, keeps the workload memory-bound. To cope with this and still meet real-time constraints, we extend the SLO-aware batching mechanism to the multi-pipeline setting and combine it with the batch-denoising strategy. Concretely, we produce a fine-denoised output at every micro-step (Fig. 7), while treating the n denoising steps as an effective batch multiplier, yielding a refined latency model $L(T, nB)$. The scheduler continuously adapts B to the observed end-to-end latency so that the per-stream rate satisfies f_{SLO} , while the aggregate throughput approaches the bandwidth roofline.

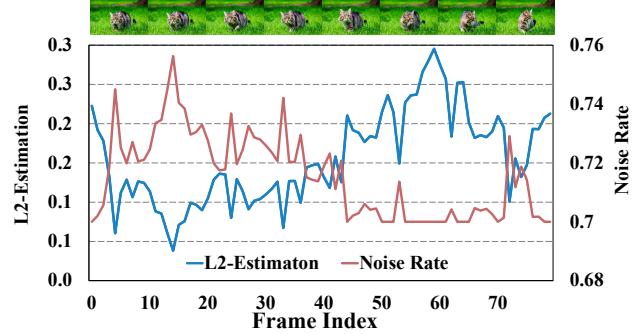


Figure 8 Example of motion estimation and dynamic noise rate. The curves indicate the L2-estimation and its corresponding noise rate of the video.

4.3 Efficient system-algorithm co-design

DiT Block Scheduler. Static partitioning often produces unbalanced workloads because the first and last ranks handle VAE encoding and decoding in addition to DiT blocks, as shown in Fig.13(a). This imbalance leads to pipeline stalls and reduced utilization (Tsung et al.). We introduce a lightweight inference-time *DiT block scheduler* that dynamically reallocates blocks between devices based on measured execution time. The scheduler searches for an optimal partition that minimizes per-stage latency, as illustrated in Fig. 13(b), significantly reducing overall pipeline bubbles.

Stream-VAE. StreamDiffusionV2 integrates a low-latency Video-VAE variant designed for streaming inference. Instead of encoding long sequences, Stream-VAE processes short video chunks (e.g., 4 frames) and caches intermediate features within each 3D convolution to maintain temporal coherence.

Asynchronous communication overlap. To further reduce synchronization stalls, each GPU maintains two CUDA streams: a computation stream and a communication stream. Inter-GPU transfers are executed asynchronously, overlapping with local computation to hide communication latency. This double-stream design aligns each device’s compute pace with its communication bandwidth, effectively mitigating residual bubbles and sustaining high utilization across the multi-GPU pipeline.

5 Experiments

5.1 Setup

Models. The StreamDiffusionV2 model is built on Wan 2.1 (Wan et al., 2025) and CausVid (Yin et al., 2025). The proposed method is training-free. In the Appendix, we describe how we can improve the visual quality by efficient finetuning with REPA (Yu et al., 2025) to cater to the user applications.

Efficiency metrics. We benchmark throughput under varying methods and configurations. For delay analysis, we report FPS (frames per second) and time-to-First-Frame (TTFF) for end-to-end latency. We also report the acceleration rate = (baseline inference time) / (optimized inference time), which directly quantifies overall speedup.

Quality metrics. Following prior work (Liang et al., 2024), we compute the **CLIP Score** as the cosine similarity between the CLIP (Radford et al., 2021) features of the generated and reference frames. We further measure **Warp Error** by estimating optical flow between consecutive inputs with RAFT (Teed and Deng, 2020) and warping the corresponding generated frames; lower is better. These metrics capture semantic- and pixel-level consistency, respectively.

Baselines. For efficiency, we compare with sequence-parallel approaches: Ring-Attention (Liu et al., 2023) and DeepSpeed-Ulysses (Jacobs et al., 2023). For generation quality, we include StreamDiffusion (Kodaira et al., 2023), StreamV2V (Liang et al., 2024), and a video-to-video variant of CausVid (implemented via a naïve noising–denoising scheme).

Implementation details. We evaluate StreamDiffusionV2 on enterprise- and consumer-grade GPUs— $4 \times$ H100 (80 GB, NVLink) and $4 \times$ RTX 4090 (24 GB, PCIe). All runs use **bf16** with no TensorRT or quantization. Results are reported at 512×512 and 480p (832×480) with 1–4 denoising steps.

5.2 Efficiency Evaluation

5.2.1 TTFF Results

We compare the Time-to-First-Frame (TTFF) results of Wan-T2V-1.3B, CausVid, and our proposed StreamDiffusionV2 during video-to-video generation. The time-to-first-frame (TTFF) depends on the configured input/output FPS and frame chunk size, and is calculated through the sum of the frame buffering delay and

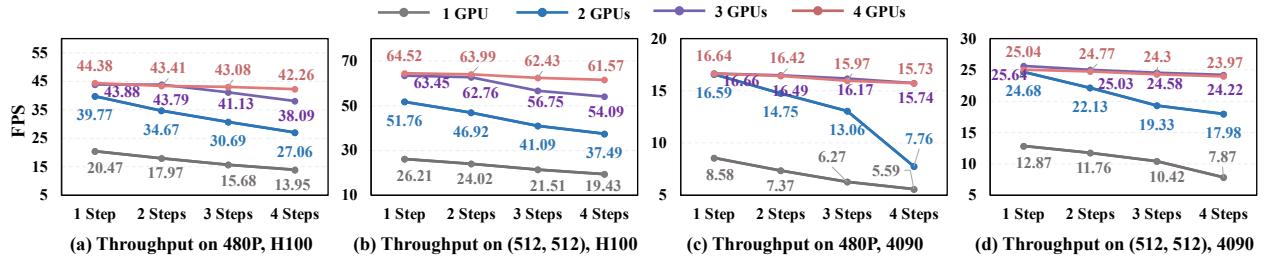


Figure 9 The throughput results of the 1.3B model on H100 GPUs (with NVLink) and 4090 GPUs (with PCIe) among different denoising steps and various resolutions. We report the result without batching on 4090 because of the memory limitation.

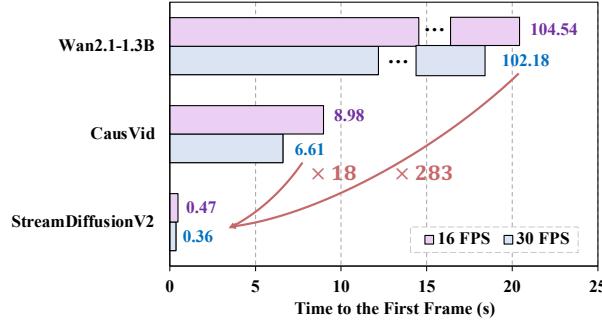


Figure 10 Time to the first frame on H100 GPU. We present the 2-step denoising results of CausVid and StreamDiffusionV2, 50 steps denoising results on Wan2.1-T2V-1.3B.

the processing latency. Taking advantage of the SLO-aware input and streaming VAE design, StreamDiffusionV2 achieves a substantial reduction in TTFF, reaching 0.47s and 0.37s at 16 FPS and 30 FPS video throughput, respectively. Specifically, at 30 FPS, CausVid and Wan2.1-1.3B exhibit 18 \times and 280 \times higher TTFF than our pipeline, respectively. The TTFF metric indicates the actual latency in a live-streaming application, which has demonstrated the capacity of the proposed pipeline in interactive real-time generation.

5.2.2 FPS Results

Fig. 9 presents the speed under different resolutions and GPU configurations, with the Wan-T2V-1.3B models. On the H100 platform, which benefits from high-bandwidth NVLink interconnects, StreamDiffusionV2 achieves 42.26 FPS at 480P and 61.57 FPS at 512 \times 512 with a 1-step model, as shown in Fig. 9 (a) and (b), respectively. Even when the denoising steps increase to four, the system still produces more than 40 FPS at 480P and 60 FPS at 512 \times 512, showing stable performance under heavier diffusion workloads. Moving toward custom devices such as 4090 GPUs with PCIe connections, we still achieve nearly 16 FPS in 480P and 24 FPS in 512 \times 512, respectively.

Moreover, we evaluate our method on a 14B parameter configuration to assess scalability for large diffusion backbones, as shown in Fig. 11. Despite the substantial model size, the proposed Pipeline-parallel Stream-batch design achieves 39.24 FPS at 480P and 58.28 FPS at 512 \times 512 across 4 GPUs, showing that the system remains compute-efficient and communication-balanced under heavy workloads. In particular, our pipeline achieves comparable throughput on the 14B-parameter model. This is mainly because both the 1.3B and 14B models share the same VAE weights, while the VAE accounts for approximately 30% of the total inference time. As a result, the VAE’s processing time remains constant and the increased computational cost only impacts the DiT component. The time consumption caused by VAE also leads to a deviation between the ideal and actual throughput gains for the whole pipeline.

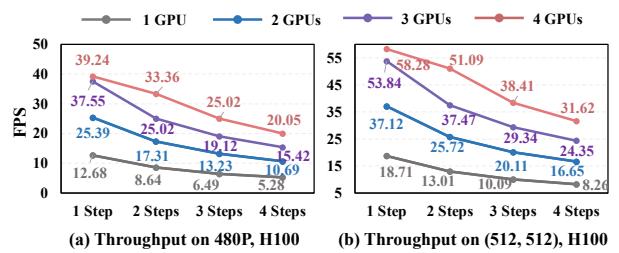


Figure 11 The throughput results of the 14B model on H100 GPUs (communicate through NVLink) among different denoising steps and various resolutions.

Table 1 Quantitative metrics comparison. We report the CLIP consistency and prompt score, and warp error to indicate the consistency of generated videos.

	StreamDiffusion	StreamV2V	CausVid	StreamDiffusionV2
CLIP Score \uparrow	95.24	96.58	98.48	98.51
Warp Error \downarrow	117.01	102.99	78.71	73.31

Table 2 Quantitative metrics for ablation studies. We report the CLIP score and warp error for ablation studies. The \checkmark indicates adding the corresponding module to the baseline models.

Sink Token	Dynamic Noising	CLIP Score \uparrow	Warp Error \downarrow
-	-	98.38	79.51
-	\checkmark	98.36	75.71
\checkmark	-	98.47	73.64
\checkmark	\checkmark	98.51	73.13

5.3 Generation Quality Evaluation

5.3.1 Comparison of Video Quality Metrics

As shown in Tab. 1, approaches based on image diffusion models, such as StreamDiffusion (Kodaira et al., 2023) and StreamV2V (Liang et al., 2024), exhibit noticeable temporal inconsistency, resulting in lower CLIP scores and higher Warp Errors. For CausVid (Yin et al., 2025), we implement a naive video-to-video generation baseline for fair quality evaluation. It achieves a comparable CLIP score but exhibits a higher Warp Error than our proposed method. These results indicate that our style-preserving and motion-aware strategies effectively enhance pixel-level temporal consistency, while maintaining comparable semantic similarity, as both methods share the same model weights.

5.3.2 Comparison of Generation Results

We also present the generation results in Fig. 2 for visualization comparison. Compared to previous approaches, which are based on image diffusion models, the proposed method achieves superior video transfer capabilities, consistent motion, and a broader range of results. Moreover, without sink tokens for style preservation, CausVid exhibits significant style fading as generation progresses. When input videos contain high-speed motion, as shown in Fig. 12, CausVid produces motion-misaligned transferred frames, because of the over-smooth training data. In contrast, our proposed method leverages the sink token strategy to maintain visual style and employs a motion-aware noise controller for dynamic noising, resulting in a temporally consistent appearance and accurate motion structure.

5.4 Analysis of Efficiency and Generation Quality

5.4.1 Effectiveness of Sink Token and Motion-Aware Noise Controller

Following Sec. 5.3, we evaluate the proposed modules using **CLIP Score** and **Warp Error**. As reported in Table 1, augmenting the baseline (live-stream–customized CausVid) with the *Motion-Aware Noise Controller* slightly reduces CLIP Score but noticeably improves Warp Error. This matches intuition: the controller adaptively lowers noise intensity in proportion to motion frequency, trading a small amount of semantic consistency

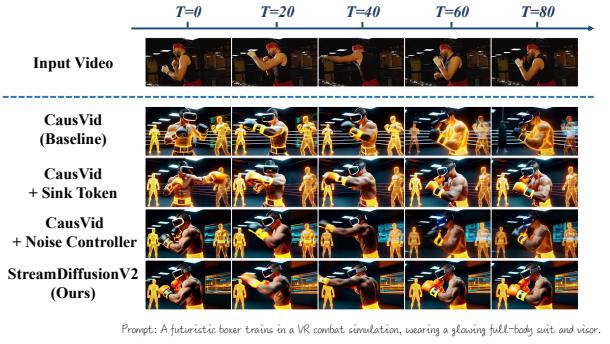


Figure 12 High-speed input video generation results across various configurations from 14B models.

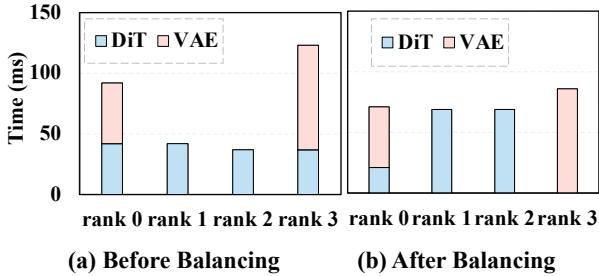


Figure 13 Time consumption before and after the balancing schedule. (a) Time consumption among various devices before balancing. (b) Time consumption after balancing. We present the 4-step denoising results on NVLink-connected H100 GPUs.

for better pixel-level alignment. When combined with the *Sink Token*, the pipeline achieves state-of-the-art performance on both metrics.

To disentangle the contributions, Fig. 12 shows a high-speed motion example where vanilla CausVid exhibits severe degradation. The *Sink Token* stabilizes global style across frames (see the target character and background), while the *Motion-Aware Noise Controller* preserves motion structure between generated and reference frames, mitigating temporal misalignment.

5.4.2 Effectiveness of the Dynamic DiT-Block Scheduler

Balanced workload is critical for efficient pipeline-parallel inference. We profile DiT vs. VAE cost and measure latency at 480p resolution with 4 denoising steps on a 1.3B model. Figure 13(a) shows that Video VAE (our streaming implementation following the baseline designs (Wan et al., 2025)) contributes substantially to the runtime and can induce stage imbalance. Our *dynamic scheduler* partitions DiT blocks to equalize per-device time, markedly reducing imbalance and improving throughput; see Fig. 13(b).

5.4.3 Sequence Parallelism vs. Pipeline Orchestration

Sequence parallelism is a widely used efficiency technique. Here we compare it with our pipeline orchestration along two axes: (i) *communication cost*, and (ii) the *performance-bound regime*.

Communication cost. As shown in Fig. 5, we measure communication overhead by subtracting the ideal single-device latency from the observed distributed latency. Across resolutions, both DeepSpeed-Ulysses (Jacobs et al., 2023) and Ring-Attention (Liu et al., 2023) incur $\sim 40\text{--}120$ ms cross-device latency—about $20\text{--}40\times$ higher than our approach.

Performance-bound regime. To isolate algorithmic scaling, we evaluate the *theoretical* latency of sequence parallelism and pipeline parallelism with communication removed (Fig. 14). Our method, built on pipeline parallelism, attains a near-ideal acceleration by partitioning DiT blocks. In contrast, sequence parallelism shows clear gains only at high resolutions; at moderate and low resolutions, it shifts the workload into a memory-bound regime, yielding little to no latency reduction and underutilizing compute.

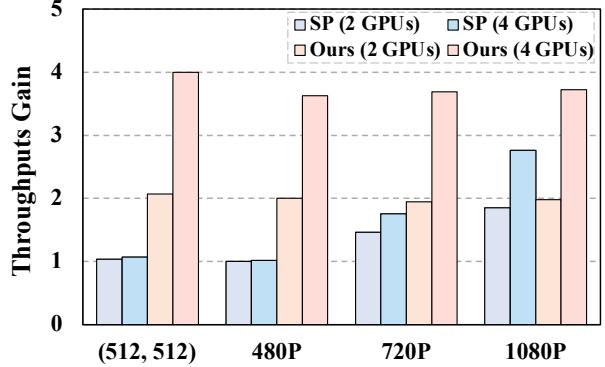


Figure 14 Theoretical computation consumption of various parallelism acceleration methods on H100. We test the time consumption of sequence partition and block partition to simulate the corresponding parallel inference approach.

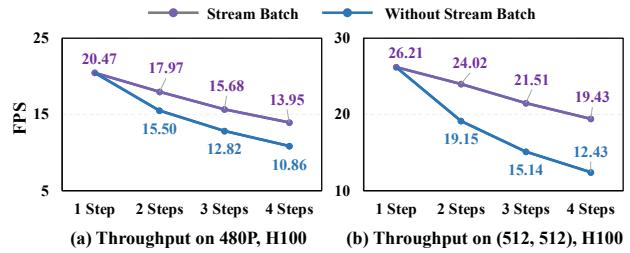


Figure 15 The throughput comparison between with and without Stream Batch.

Effectiveness of Stream Batch in the dual-pipeline scheduler. Figure 15 demonstrates that *Stream Batch* substantially improves throughput, especially as the number of denoising steps increases. More denoising steps create deeper in-flight pipelines, amplifying the benefit and delivering progressively larger throughput gains.

6 Conclusion

We propose StreamDiffusionV2, which closes the gap between offline video diffusion and live streaming constrained in real-time with SLO constraints. Our training-free system couples an SLO-aware batching/block scheduler with a sink-token-guided rolling KV cache, a motion-aware noise controller, and a pipeline orchestration that parallelizes across denoising steps and model layers—delivering near-linear FPS scaling without violating latency. It runs on heterogeneous GPUs and flexible step counts, achieving 0.5 s TTFF and up to 58.28 FPS (14B) / 64.52 FPS (1.3B) on 4×H100, and maintaining high FPS even as steps increase. These results make state-of-the-art generative live streaming practical for both individual creators and enterprise platforms.

References

- Max Bain, Arsha Nagrani, Güл Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1728–1738, 2021.
- Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Junchen Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengcheng Ma, et al. Skyreels-v2: Infinite-length film generative model. *arXiv preprint arXiv:2504.13074*, 2025a.
- Junsong Chen, Yuyang Zhao, Jincheng Yu, Ruihang Chu, Junyu Chen, Shuai Yang, Xianbang Wang, Yicheng Pan, Daquan Zhou, Huan Ling, et al. Sana-video: Efficient video generation with block linear diffusion transformer. *arXiv preprint arXiv:2509.24695*, 2025b.
- Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. *arXiv preprint arXiv:2410.10733*, 2024a.
- Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. δ -dit: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024b.
- Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang-wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, et al. Panda-70m: Captioning 70m videos with multiple cross-modality teachers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13320–13331, 2024c.
- Karan Dalal, Daniel Koceja, Jiarui Xu, Yue Zhao, Shihao Han, Ka Chun Cheung, Jan Kautz, Yejin Choi, Yu Sun, and Xiaolong Wang. One-minute video generation with test-time training. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17702–17711, 2025.
- Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.
- Jiarui Fang, Jinzhe Pan, Xibo Sun, Aoyu Li, and Jiannan Wang. xdit: an inference engine for diffusion transformers (dits) with massive parallelism. *arXiv preprint arXiv:2411.01738*, 2024a.
- Jiarui Fang, Jinzhe Pan, Jiannan Wang, Aoyu Li, and Xibo Sun. Pipefusion: Patch-level pipeline parallelism for diffusion transformers inference. *arXiv preprint arXiv:2405.14430*, 2024b.
- Yu Gao, Jiancheng Huang, Xiaopeng Sun, Zequn Jie, Yujie Zhong, and Lin Ma. Matten: Video generation with mamba-attention. *arXiv preprint arXiv:2405.03025*, 2024.
- Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, Baixin Xu, Hao-Xiang Guo, Kaixiong Gong, Cyrus Wu, Wei Li, Xuchen Song, Yang Liu, Eric Li, and Yahui Zhou. Matrix-game 2.0: An open-source, real-time, and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025.
- Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025.
- Yan Huang, Tom ZJ Fu, Dah-Ming Chiu, John CS Lui, and Cheng Huang. Challenges, design and analysis of a large-scale p2p-vod system. *ACM SIGCOMM computer communication review*, 38(4):375–388, 2008.
- Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023.
- Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*, 2023.
- Akio Kodaira, Chenfeng Xu, Toshiki Hazama, Takanori Yoshimoto, Kohei Ohno, Shogo Mitsuhori, Soichi Sugano, Hanying Cho, Zhijian Liu, and Kurt Keutzer. Streamdiffusion: A pipeline-level solution for real-time interactive generation. *arXiv preprint arXiv:2312.12491*, 2023.
- Akio Kodaira, Tingbo Hou, Ji Hou, Masayoshi Tomizuka, and Yue Zhao. Streamdit: Real-time streaming text-to-video generation. *arXiv preprint arXiv:2507.03745*, 2025.

- Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Kai Li, and Song Han. Distrifusion: Distributed parallel inference for high-resolution diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7183–7193, 2024.
- Feng Liang, Akio Kodaira, Chenfeng Xu, Masayoshi Tomizuka, Kurt Keutzer, and Diana Marculescu. Looking backward: Streaming video-to-video translation with feature banks. *arXiv preprint arXiv:2405.15757*, 2024.
- Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It’s time to cache for video diffusion model. *arXiv preprint arXiv:2411.19108*, 2024.
- Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.
- Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024.
- Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14297–14306, 2023.
- Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, pages 1–31, 2024.
- Ryan Po, Yotam Nitzan, Richard Zhang, Berlin Chen, Tri Dao, Eli Shechtman, Gordon Wetzstein, and Xun Huang. Long-context state-space video world models. *arXiv preprint arXiv:2505.20171*, 2025.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.
- Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024.
- Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:4263–4276, 2023.
- Kunwadee Sripanidkulchai, Aditya Ganjam, Bruce Maggs, and Hui Zhang. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. *ACM SIGCOMM computer communication review*, 34(4):107–120, 2004.
- Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.
- Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, WQ Zhang, Weifeng Luo, et al. Magi-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.
- Yeung Man Tsung, Penghui Qi, Min Lin, and Xinyi Wan. Balancing pipeline parallelism with vocabulary parallelism. In *Eighth Conference on Machine Learning and Systems*.
- Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *The Thirteenth International Conference on Learning Representations*.
- Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.
- Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, et al. Cache me if you can: Accelerating diffusion models through block caching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6211–6220, 2024.
- Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

- Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024.
- Dongjie Yang, Suyuan Huang, Chengqiang Lu, Xiaodong Han, Haoxin Zhang, Yan Gao, Yao Hu, and Hai Zhao. Vript: A video is worth thousands of words. *Advances in Neural Information Processing Systems*, 37:57240–57261, 2024.
- Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, et al. Longlive: Real-time interactive long video generation. *arXiv preprint arXiv:2509.22622*, 2025a.
- Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, et al. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *arXiv preprint arXiv:2505.18875*, 2025b.
- Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and Bill Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024a.
- Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6613–6623, 2024b.
- Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22963–22974, 2025.
- Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025.
- Jintao Zhang, Haofeng Huang, Pngle Zhang, Jia Wei, Jun Zhu, and Jianfei Chen. Sageattention2: Efficient attention with thorough outlier smoothing and per-thread int4 quantization. In *International Conference on Machine Learning (ICML)*, 2025a.
- Jintao Zhang, Jia Wei, Pngle Zhang, Jun Zhu, and Jianfei Chen. Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration. In *International Conference on Learning Representations (ICLR)*, 2025b.
- Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattn: Accurate sparse attention accelerating any model inference. In *International Conference on Machine Learning (ICML)*, 2025c.
- Xinyan Zhang, Jiangchuan Liu, Bo Li, and Y-SP Yun. Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 2102–2111. IEEE, 2005.
- Xin Zhou, Dingkang Liang, Kaijin Chen, Tianrui Feng, Xiwu Chen, Hongkai Lin, Yikang Ding, Feiyang Tan, Hengshuang Zhao, and Xiang Bai. Less is enough: Training-free video diffusion acceleration via runtime-adaptive caching. *arXiv preprint arXiv:2507.02860*, 2025.

Appendix

A NVIDIA H100 SXM Roofline Parameters

We derive the roofline parameters for the NVIDIA H100 SXM GPU. The dense FP16 throughput is 1,979 TFLOP/s and GPU memory bandwidth is 3.35 TB/s, which are taken from the H100 datasheet⁴. The ridge arithmetic intensity is

$$AI_{\text{ridge}} = \frac{1979 \text{ TFLOP/s}}{3.35 \text{ TB/s}} = 590.75 \text{ FLOP/Byte} \quad (3)$$

B REPA finetuning for quality enhancement

StreamDiffusionV2 is a training-free system solution to transform efficient video models for live streaming applications. Orthogonal to the training-free solution, we employ the REPA (Yu et al., 2025) training strategy during Causal-DiT distillation (Yin et al., 2025) to enhance the video quality. The alignment loss is denoted by

$$\mathcal{L}_{\text{REPA}}(\theta, \phi) := -\mathbb{E}_{\mathbf{x}_*, \epsilon, t} \left[\frac{1}{N} \sum_{n=1}^N \text{sim}\left(\mathbf{y}_*^{[n]}, h_\phi(\mathbf{h}_t^{[n]})\right) \right],$$

where $\mathbf{y}_*^{[n]}$ indicates the output of DINOv2 (Oquab et al., 2024), $\mathbf{h}_t^{[n]}$ represents the hidden state of DiT in timestep t , h_ϕ is the projection network parameterized by ϕ , and $\text{sim}(\cdot, \cdot)$ denotes cosine similarity. When combined with the original DMD (Yin et al., 2024b) distillation objective \mathcal{L}_{DMD} , the overall training objective is defined as

$$\mathcal{L} = \mathcal{L}_{\text{DMD}} + \lambda \mathcal{L}_{\text{REPA}},$$

where λ is a scaling factor.

C Figure illustration

Figure overview. Fig. 16 shows how we minimize pipeline bubbles by separating compute and communication into two concurrent streams, overlapping kernels with P2P transfers.

Fig. 17 illustrates the rolling KV cache with sink tokens for consistent, frame-by-frame updates.

Fig. 18 reports a comparison of parallelization strategies (e.g., sequence parallelism vs. our pipeline scheduler).

⁴<https://resources.nvidia.com/en-us-gpu-resources/h100-datasheet-24306>

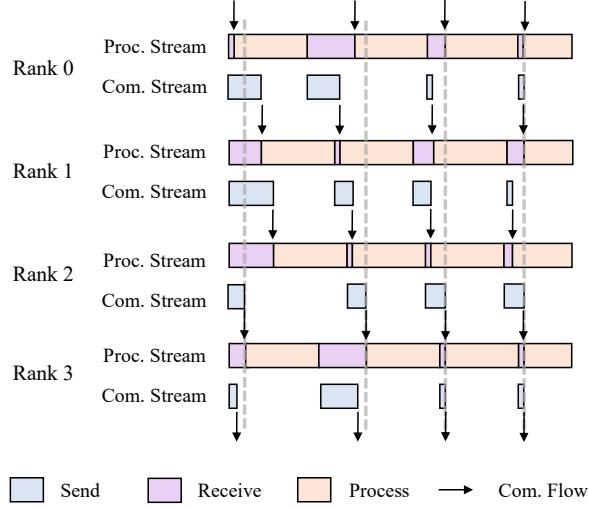


Figure 16 Execution timeline of the Pipeline-orchestration architecture.

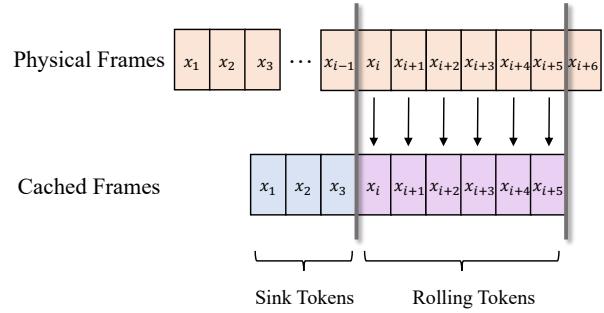


Figure 17 The detailed illustration of the Rolling KV Cache and Sink Token designs.

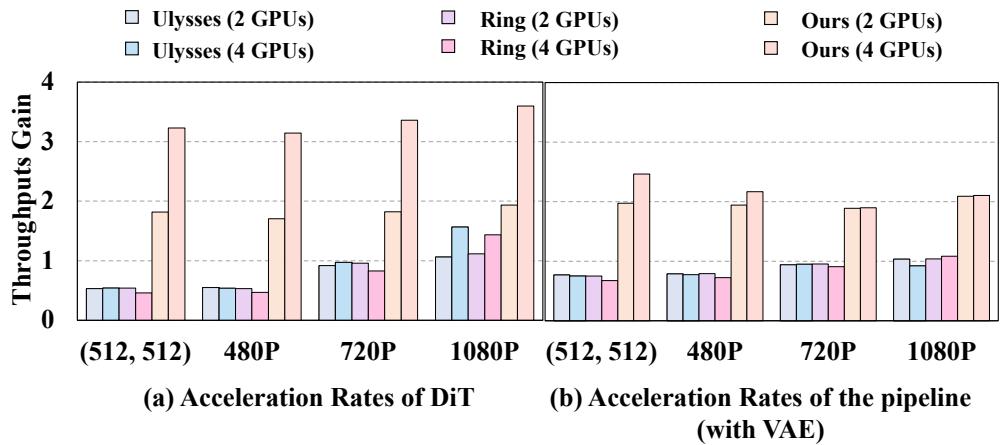


Figure 18 Acceleration rate of different approaches on various resolutions. Left: Testing the acceleration rate of DiT only. Right: Testing the acceleration of the whole pipeline (with VAE).