WAVE FORCING: TOWARDS SPEED-OF-LIGHT STREAMING VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-time long-video generation requires fast, continuous output with consistent appearance and natural motion. Chunk-wise autoregressive methods serialize denoising and history-cache refresh, requiring two forwards even with one-step denoising. Their long rollouts also often suffer from quality degradation and camera drift. We propose **Wave Forcing (WF)**, a co-design of generation, training, and execution for continuous output with full multi-step denoising. WF pipelines chunks along a causal wavefront at staggered denoising stages, using chunk-wise causal attention to access earlier, cleaner chunks before they finish. Distributing denoising and cache-refresh stages across full-model replicas yields **1F1C**: one chunk per concurrent forward round after pipeline fill. Three-stage training develops joint mixed-noise denoising, strengthens motion and framing, and adapts the model to causal inference with self-generated history. **Wave Runtime (WaveRT)** uses **Wave Parallelism (WP)** for early K/V delivery and communication–computation overlap, complemented by kernel optimization and load-balanced VAE decoding. On Wan2.1-T2V-1.3B, 5-step WF maintains overall quality comparable to Rolling Forcing with higher Dynamic Degree scores on short and long videos. With SageAttention, our 4-step model reaches 117.7 E2E FPS on eight H200 GPUs, delivering a 7.9× speedup over the 4-step single-GPU baseline.

1 INTRODUCTION

Real-time streaming video generation must sustain playable output with consistent appearance and natural motion. Chunk-wise autoregressive methods such as Self Forcing (SF) and Causal Forcing (CF) denoise each chunk conditioned on preceding content (Huang et al., 2025a; Zhu et al., 2026). Under our clean-history cache convention, an additional forward called *store* refreshes key/value (K/V) history from the clean result. Sequential execution places denoising and store between consecutive outputs: **NF1C (N Forward Rounds, 1 Chunk Output)**. Even one-step denoising needs two forwards, including store.

Instead, we target **1F1C**: one completed chunk per concurrent forward round after fill, retaining every denoising update. Inspired by wavefront scheduling (Jin et al., 2016), earlier chunks execute later denoising stages while newer chunks execute earlier ones. These staggered tasks form a moving *wavefront* (Fig. 1a). Assigning a full-model replica to each denoising stage and store creates a bounded-width pipeline (Fig. 1c). This is a scheduling target, not a latency guarantee: communication and synchronization can lengthen each round and offset the benefit.

To advance chunks before their predecessors finish, we use *chunk-wise causal context*: full attention within each chunk, cached history, and live context from earlier, cleaner chunks, with no dependence on later chunks. This directional conditioning allows stage-wise progress differences and overlapped execution (Fig. 1b). Our staged training builds on mixed-noise joint denoising (Liu et al., 2026), supervises motion and framing with teacher videos, and adapts to causal inference under self-generated history. This generation structure and training recipe form **Wave Forcing (WF)**.

To turn this scheduling potential into shorter output intervals, **Wave Runtime (WaveRT)** implements **Wave Parallelism (WP)**. Following GPU software-pipelining principles (NVIDIA, 2026a;b), WP exploits stage-wise progress differences for early layer-wise K/V delivery and communication–computation overlap, reducing exposed dependency waits. Data placement, kernel optimization, and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

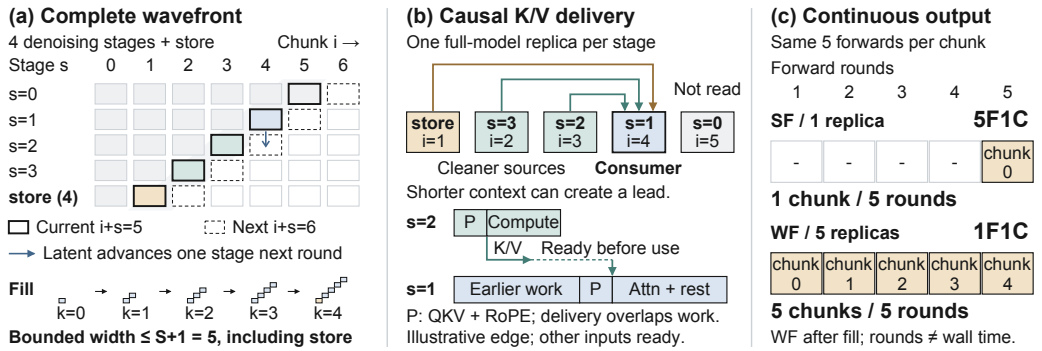

Figure 1: Causal wavefront and continuous output ($S = 4$). (a) Denoising and store advance on anti-diagonals, with dashed cells marking the next round. (b) One later-layer K/V edge arrives before use. Other edges and initial waits are omitted. (c) Both schedules retain four denoising forwards plus store per chunk. Over five rounds, one SF replica emits one chunk, whereas five WF replicas emit five after fill. Wall-clock gains depend on stage costs and overlap.

load-balanced VAE pipeline parallelism complete the path from concurrent denoising to sustained video output.

On Wan2.1-T2V-1.3B (Wan et al., 2025), 5-step WF achieves overall quality comparable to 5-step RF, with VBench Total scores of 81.64 versus 81.15 and VBench-Long Quality scores of 81.26 versus 81.18 (Huang et al., 2024; 2026). Dynamic Degree increases from 51.39 to 94.44 on short videos and from 36.63 to 91.45 on approximately 100-second videos. These gains accompany better Subject Preservation (97.10 versus RF's 92.75) while maintaining low Camera Drift (0.37 versus RF's 0.17), as shown in Tab. 1.

For fast generation, our 4-step model running on WaveRT with SageAttention (Zhang et al., 2025) reaches 117.7 E2E FPS on eight H200 GPUs, a 7.9× speedup over the 15.0-FPS 4-step single-GPU baseline (Tab. 4). Across six tested model scales from 1.3B to 14B, WP exceeds both Ulysses and Ring in DiT throughput under BF16 and Sage+W8A8, reaching 1.04–1.28× and 1.02–2.02× the faster CP baseline, respectively. Detailed system ablations and profiling examine communication overlap, K/V placement, compute backends, and VAE stage balance.

2 PRELIMINARIES

Chunk denoising and store. A VAE decodes latent-frame chunks into video (Wan et al., 2025). Following SF (Huang et al., 2025a), chunk i undergoes S history-conditioned denoising updates, from noise $Z_i^{(0)}$ to clean output $B_i = Z_i^{(S)}$. At noise timesteps $\nu_0 > \cdots > \nu_S = 0$,

$$Z_i^{(s+1)} = \Psi_{\nu_{s+1}}\Big(G_\theta(Z_i^{(s)}, \nu_s \mid \mathcal{C}_{i,s}), \epsilon_{i,s}\Big), \qquad 0 \le s < S, \tag{1}$$

G_θ predicts clean latents, which Ψ_ν re-noises using draw $\epsilon_{i,s}$, with $\Psi_0(x, \epsilon) = x$. $\mathcal{C}_{i,s}$ denotes layer-wise K/V context, with text conditioning omitted for brevity. Denoising K/V represents the noisy input, so our clean-history convention adds *store*: a full-model forward on B_i that retains K/V and discards the prediction. Sequential generation thus uses $N = S + 1$ forwards per chunk (NF1C).

Wavefront states and execution rounds. At filled runtime tick k, the denoising inputs are

$$\mathcal{F}_k = \{Z_k^{(0)}, Z_{k-1}^{(1)}, \ldots, Z_{k-S+1}^{(S-1)}\}. \tag{2}$$

Adding store at $s = S$ gives $\mathcal{W}_k = \mathcal{F}_k \cup \{B_{k-S}\}$, of width at most $S + 1$ excluding anchor/history caches. Each round advances active stages and admits fresh noise. With stage-owned replicas, this schedule permits 1F1C after fill without changing per-chunk work.

3 WAVE FORCING

We introduce **Wave Forcing (WF)** to advance successive chunks before their predecessors finish denoising. WF couples a causal wavefront—newer, noisier chunks read earlier, cleaner states, but not vice versa—with training for this mixed-noise context and self-generated history.

3.1 A BOUNDED-WIDTH CAUSAL WAVEFRONT

Cell (i, s), $s < S$, in Fig. 1a updates $Z_i^{(s)}$ to $Z_i^{(s+1)}$ in round

$$k = i + s, \qquad i = k - s. \tag{3}$$

Store at $(k - S, S)$ refreshes B_{k-S}'s K/V. For N_{chunk} requested chunks, active denoising stages are $\mathcal{A}_k = \{s : 0 \le s < S,\ 0 \le k - s < N_{\text{chunk}}\}$. Store is active when $0 \le k - S < N_{\text{chunk}}$.

WF retains full attention within each chunk and uses *chunk-wise causal attention* across video time. Query slot s sees K/V slot j iff

$$M_{\text{WF}}(s, j) = \mathbf{1}[j \ge s], \qquad s, j \in \mathcal{A}_k. \tag{4}$$

Since $i' = k - j \le i = k - s$, the mask is lower triangular in chunk order and retains self and *all* earlier, cleaner chunks.

At Transformer layer $\ell \in \{1, \dots, L\}$, the ordered context is

$$\mathcal{C}_{k,s}^{\text{WF},\ell} = A_k^\ell \oplus H_k^\ell \oplus U_k^\ell \oplus \Big[\bigoplus_{\substack{j \in \mathcal{A}_k \\ j \ge s}} (K_{k,j}^\ell, V_{k,j}^\ell) \Big], \tag{5}$$

Here $\oplus$ concatenates initial-chunk anchor A_k^ℓ, bounded clean history H_k^ℓ, live store U_k^ℓ, and denoising K/V in increasing j (self first). Store appears once, is empty when inactive, and enters history after the round. Appendix B details fill/drain and caching.

This removes Rolling Forcing's (RF) reverse refinement (Liu et al., 2026). Without post-training, short-video Quality and Dynamic Degree fall in our evaluation (Sec. 5.1), motivating adaptation.

3.2 TRAINING FOR THE DEPLOYMENT GRAPH

The causal mask removes access to later, noisier active chunks, changing RF's learned conditioning. We adapt Wan2.1-T2V-1.3B (Wan et al., 2025) through mixed-noise initialization, teacher guidance, and on-policy training. The 4-step student starts from SF's ODE initialization, whereas the 5-step student reuses released RF weights and skips Stage 1.

Stage 1: mixed-noise initialization. Following RF (Liu et al., 2026), distribution matching distillation (DMD) (Yin et al., 2024) uses an equal mixture of RF and SF rollouts to learn joint mixed-noise denoising with bidirectional attention, with SF rollouts helping stabilize motion during training.

Stage 2: off-policy DMD. Following LPM 1.0 (Zeng et al., 2026), we re-noise consecutive teacher chunks Y_{win} at staggered timesteps, with preceding teacher chunks as clean K/V history. Retaining Stage 1's bidirectional attention, this supplies stronger motion and subject guidance (Sec. 5.2). DMD and LPIPS (Zhang et al., 2018) supervise the same text-conditioned prediction $\widehat{Y}_\theta$:

$$\mathcal{L}_2 = \mathbb{E}\Big[\mathcal{D}_{\text{DMD}}(\widehat{Y}_\theta; \mathbf{c}) + \lambda_{\text{p}} \,\text{LPIPS}\Big(\mathcal{T}_{\text{perc}}(\widehat{Y}_\theta), \mathcal{T}_{\text{perc}}(Y_{\text{win}})\Big)\Big], \qquad \lambda_{\text{p}} = 0.25, \tag{6}$$

Here $\mathcal{D}_{\text{DMD}}$ is the DMD objective, and $\mathcal{T}_{\text{perc}}$ decodes downsampled latents to sampled RGB frames.

Stage 3: causal context and self-generated history. Each iteration uniformly samples a self-generated RF, WF, or SF rollout for on-policy DMD. WF uses Eq. (5), including store-refreshed history from its own outputs. RF and SF provide complementary training distributions, but only WF is deployed. For branch objective $\mathcal{L}_b$,

$$\mathcal{L}_3 = \mathbb{E}_{b \sim \text{Unif}\{\text{RF},\text{WF},\text{SF}\}}[\mathcal{L}_b] = \tfrac{1}{3}\mathcal{L}_{\text{RF}} + \tfrac{1}{3}\mathcal{L}_{\text{WF}} + \tfrac{1}{3}\mathcal{L}_{\text{SF}}. \tag{7}$$

Fig. 7 and Appendix A give the curriculum, stage transitions, and implementation.

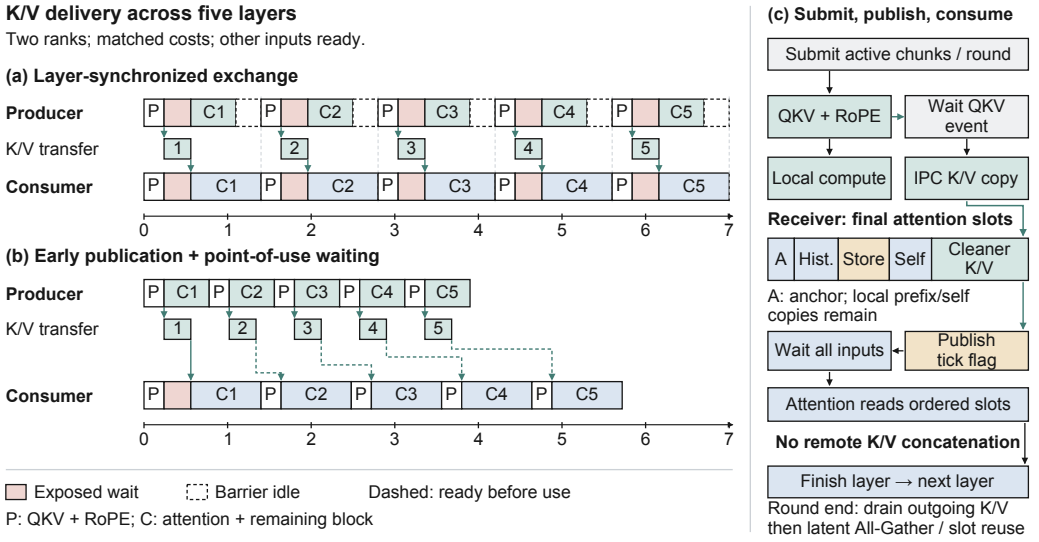

Figure 2: Hiding K/V delivery with identical schematic operation costs. (a) Layer barriers expose communication. (b) Early publication hides later waits, not dependencies or the initial wait. (c) Direct writes to receiver slots precede readiness signals and consumption. Local copies and round-boundary synchronization remain. Panels (a,b) omit other ranks.

4 WAVE RUNTIME

WF exposes stage concurrency, but K/V dependencies can still stall attention. **Wave Runtime (WaveRT)** realizes **Wave Parallelism (WP)** by publishing K/V early from stage-owned replicas and overlapping delivery with computation when cleaner producers run ahead.

4.1 STAGE PLACEMENT AND DEPENDENCIES

Each denoising stage and store owns a full-model GPU replica. Denoising rank s updates $Z_{k-s}^{(s)}$ and passes it to rank $s+1$ for the next round (Fig. 1). Within a round, let $P_{s,\ell}$ produce Q/K/V and RoPE, and $C_{s,\ell}$ perform attention and the remaining block:

$$P_{s,\ell} \to C_{s,\ell}, \qquad C_{s,\ell} \to P_{s,\ell+1} \quad (\ell < L), \tag{8}$$

$$P_{j,\ell} \to C_{s,\ell}, \qquad j > s, \quad s, j \in \mathcal{A}_k. \tag{9}$$

Store and all legal cleaner ranks (not only neighbors) must supply K/V before attention, but local projection need not wait. RF additionally includes reverse edges from noisier ranks, whereas WF removes these dependencies through its causal mask. The synchronized reference still gathers all ranks' K/V at every layer.

4.2 PRODUCER LEAD AND ASYNCHRONOUS DELIVERY

In a filled round, rank s reads N_C query tokens and N_s keys, where

$$N_s = N_{\text{prefix}} + (S - s)N_C. \tag{10}$$

Here N_C is tokens per chunk and N_{prefix} covers anchor, history, and store. Cleaner ranks read fewer keys, permitting later-layer producer lead. WaveRT publishes K/V after SGLang's (Zheng et al., 2024) QKV projection and RoPE, then checks readiness immediately before attention (Fig. 2).

On a common time base, edge e has payload-ready time t_e^{ready}, consumer local-ready time t_e^{need} before remote waits, and delivery cost τ_e including queueing, transfer, and signaling:

$$\Delta_e = t_e^{\text{need}} - t_e^{\text{ready}}, \qquad w_e = \max(0, \tau_e - \Delta_e). \tag{11}$$

A consumer's exposed wait is the maximum w_e over required inputs. Thus, delivery is fully hidden only when every $\tau_e \le \Delta_e$ (Appendix B.7).

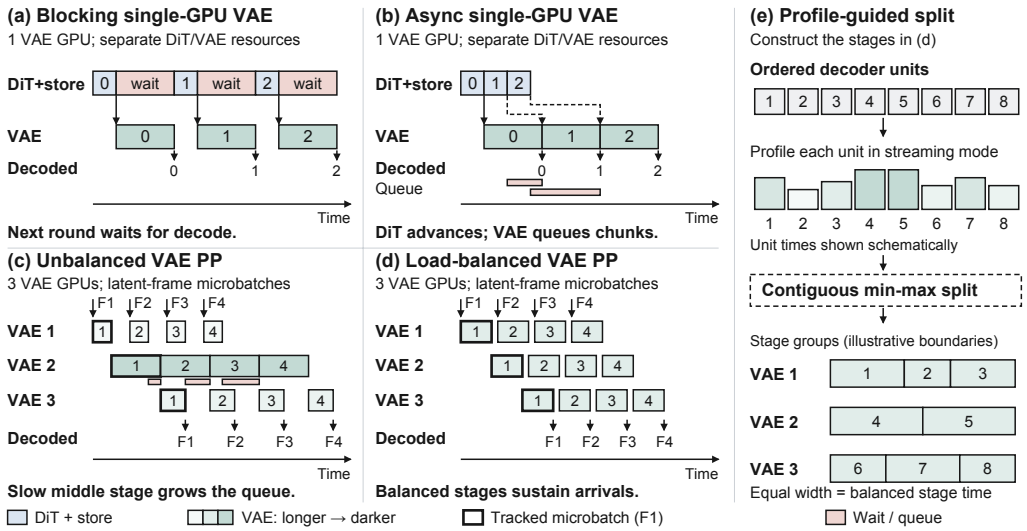

Figure 3: Streaming VAE schedules. (a,b) Blocking versus asynchronous decoding with matched costs. (c,d) Unbalanced versus balanced PP with identical work and arrivals, where darker teal indicates longer service. (e) Profile-guided contiguous partitions. Each pair uses its own schematic time scale. Transfer costs are omitted and boundaries are illustrative. The first output incurs pipeline fill latency (Appendix B.5).

Following software-pipeline principles (NVIDIA, 2026a), *paged* delivery writes K/V directly into ordered attention slots and signals readiness before use, avoiding remote concatenation. When weights, noise draws, positions, and local operators are held fixed, attention consumes the same K/V values as in synchronized execution (Appendix B.6). However, this delivery scheme does not eliminate initial dependency waits, store computation, resource contention, local copies, or the latent All-Gather at round boundaries. Appendix B details buffer lifetimes and readiness signaling.

4.3 Local computation and streaming decode

Local computation and K/V delivery require joint optimization. SageAttention and W8A8 linear layers target faster local operators but also change the relative progress of producers and consumers. As a result, faster computation need not reduce communication waits. K/V quantization reduces payload bytes, while paged direct writes reduce receiver-side context assembly. These optimizations aim to make each layer's K/V ready before the consumer executes attention, reducing exposed communication and synchronization waits on the critical path.

Asynchronous decoding overlaps VAE and DiT, but the slower pipeline limits E2E throughput. We use profiled streaming times to balance contiguous VAE partitions across dedicated GPUs while preserving request-local causal caches (Fig. 3). Decoding and inter-stage transfers must sustain DiT's output rate. The first decoded output still incurs pipeline fill latency (Appendix B.5).

WP uses full-model replicas, with execution width determined by the denoising-step count. Logical K/V fan-out remains quadratic in the number of stages, while attention workloads vary across stages. Appendices B.8 and B.4 quantify communication traffic and pipeline fill/drain costs, respectively.

5 Experiments

Models and training. We evaluate 4-step and 5-step Wave Forcing built on Wan2.1-T2V-1.3B (Wan et al., 2025) against Rolling Forcing (Liu et al., 2026) and Causal Forcing (Zhu et al., 2026). All methods retain their native denoising schedules and three latent frames per chunk. Training and checkpoint details appear in Sec. 3.2 and Appendices A and C.

Table 1: Short- and long-video comparison. Standard VBench reports Total, Quality, Semantic, and Dynamic Degree (Dynamic), while VBench-Long reports Quality and Dynamic Degree. Subj. and Drift denote our long-video Subject Preservation and Camera Drift diagnostics. Quality and diagnostic scores are scaled to 0–100, with the best available value in each evaluation column in bold. Deployment E2E FPS includes VAE decoding, and WaveRT rates use 8 GPUs (Tab. 4). Unconfirmed measurements are marked –.

		Standard VBench ↑				VBench-Long ↑				
Method	Steps	Total	Quality	Semantic	Dynamic	Quality	Dynamic	Subj. ↑	Drift ↓	E2E FPS ↑
Rolling Forcing	5	81.15	83.77	70.66	51.39	81.18	36.63	92.75	**0.17**	11.90
+ Causal Attention	5	80.89	83.35	**71.05**	37.50	80.50	46.60	91.26	0.40	11.90
Causal Forcing (Full Attn.)	4	79.61	82.83	66.72	**100.00**	**82.15**	**96.61**	71.11	11.14	15.00
Wave Forcing (Ours)	4	80.74	83.93	67.95	84.72	80.89	74.97	91.88	0.36	84.90
+ SageAttention		81.03	84.02	69.07	65.28	81.16	77.24	90.79	0.36	**117.70**
Wave Forcing (Ours)	5	**81.64**	**84.40**	70.59	94.44	81.26	91.45	**97.10**	0.37	83.70
+ SageAttention		81.02	83.92	69.41	80.56	81.14	91.14	94.43	0.37	84.98

Evaluation. For short videos, we report VBench (Huang et al., 2024) Total, Quality, Semantic, and Dynamic Degree, generating one 129-frame video per original prompt. For long videos, we report VBench-Long Quality and Dynamic Degree on 200 MovieGen (Polyak et al., 2024) prompts, generating 100-second videos. All videos use 832×480 resolution, 16 FPS, and seed 42, and benchmark scores follow official normalization and weighting. Tab. 1 summarizes the results, with the full protocol in Appendix C.

Long-video diagnostics. High motion scores can coexist with subject exit or unidirectional camera drift. To assess these failures, we evaluate a fixed 100-prompt subject-focused subset of MovieGen, shared across methods and training ablations. Subject Preservation measures the visibility of the SAM3-tracked (Carion et al., 2026) principal entity, with boundary weighting for partial disappearance. Camera Drift penalizes a subject's approach to an image boundary only when accompanied by persistent outward background motion in the same direction. Appendix C gives the prompt-selection procedure and both metric definitions.

Compute configurations. BF16 is the reference. Sage uses SageAttention (Zhang et al., 2025) with BF16 linear layers, while Sage+W8A8 additionally uses 8-bit linear weights and activations. These labels describe local computation, not K/V transport precision.

5.1 Generation quality and motion

Tab. 1 compares short- and long-video generation.

We compare Wave Forcing with open-source autoregressive video generation models of similar scale. Specifically, we compare against Rolling Forcing (Liu et al., 2026), which adopts a rolling-window design, and additionally evaluate it with a causal attention mask to measure its performance under WaveRT inference. We further compare against Causal Forcing (Zhu et al., 2026), which uses causal ODE initialization from an autoregressive teacher, using its long-video variant with full attention.

As shown in Tab. 1, Wave Forcing maintains VBench quality comparable to the baselines while achieving a high Dynamic Degree, and it keeps the principal subject in the frame with a stable camera. Moreover, Wave Forcing can be deployed on WaveRT with SageAttention for high-throughput generation, with acceptable dynamic degradation on short videos, and still generate high-quality long videos without drifting.

A high Dynamic Degree alone does not indicate good long-video motion. Causal Forcing attains the highest Dynamic Degree, but its subjects frequently leave the frame and its camera drifts, yielding a Subject Preservation of 71.11 and a Camera Drift of 11.14. Rolling Forcing keeps the subject visible (92.75) with the lowest drift (0.17), but its Dynamic Degree on VBench-Long is only 36.63. 5-step Wave Forcing raises this score to 91.45 while achieving the highest Subject Preservation (97.10) and a low Camera Drift (0.37).



Figure 4: Long-video comparison on a reading prompt. Rows show Causal Forcing, Rolling Forcing, and 4-step and 5-step Wave Forcing, sampled from 0 to 95 seconds.

5.2 Training Ablations

We conduct ablation studies on the 4-step model to assess the contribution of several training design options, as summarized in Tab. 2. The evaluation settings are identical to the main experiments.

Table 2: 4-step training ablations. All rows evaluate final models.

Model	Quality ↑	Dynamic ↑	Subj. ↑	Drift ↓
w/o off-policy DMD	76.68	15.73	76.53	0.26
w/o RF	80.19	88.82	77.48	6.24
w/o WF	80.34	48.12	95.64	0.10
w/o SF	75.35	23.66	84.04	1.89
Ours full	80.89	74.97	91.88	0.36

Off-policy DMD. In w/o off-policy DMD, we skip Stage 2 and directly adapt the model with on-policy training in Stage 3. As reported in Tab. 2, this leads to substantial degradation in visual quality and Dynamic Degree, with disappearance of the principal subject.

Mixed Training Strategy. We evaluate the importance of each branch in stage 3 mixed training. Removing the SF branch produces a degradation in quality and motion, and the subject again disappears. Removing the RF branch raises Dynamic Degree, but the increase is caused by severe subject disappearance and camera shift. Removing the WF branch still lowers Dynamic Degree. The full model preserves visual quality while raising Dynamic Degree, stabilizing the subject, and suppressing camera drift.

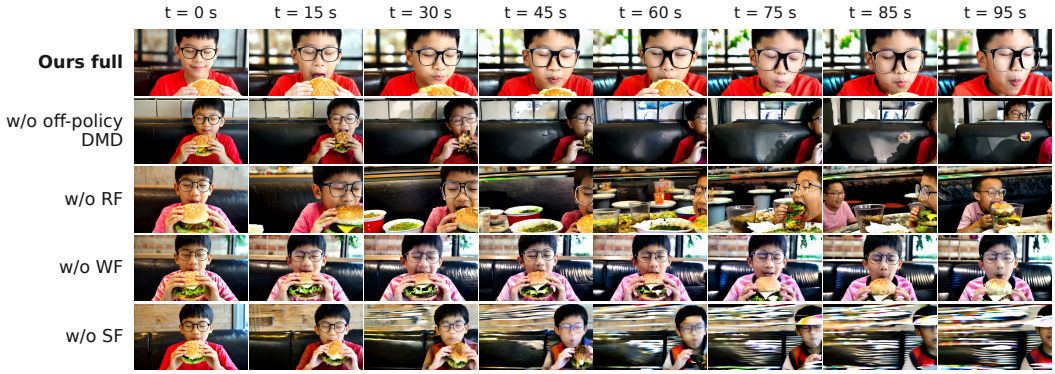

Figure 5: Qualitative training ablations on a burger-eating prompt. Rows compare the full 4-step model with each training ablation, and columns sample the trajectory from 0 to 95 seconds.

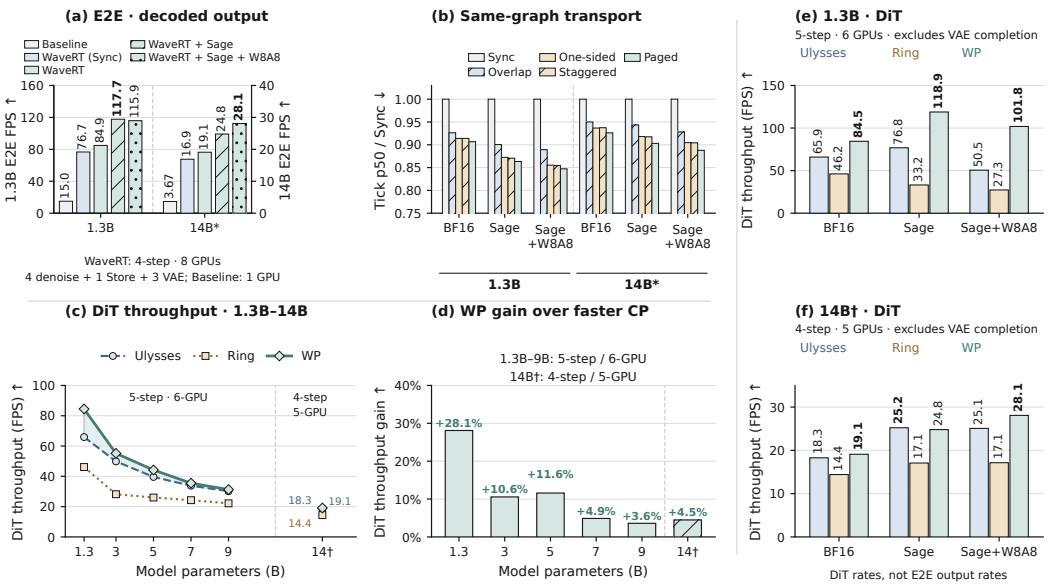

Figure 6: System performance. (a) compares 4-step E2E FPS of 8-GPU WaveRT and single-GPU Baseline, with left/right axes for 1.3B/14B. (b) normalizes same-graph tick p50 to Sync (axis starts at 0.75), using random weights for 14B (*). (c–f) report DiT FPS excluding VAE completion, using 5-step, 6-GPU setups for 1.3B–9B and 4-step, 5-GPU for 14B (†). GPU counts satisfy Ulysses' head-divisibility constraint. (c,d) show BF16 throughput and gains over the faster CP, with shading in (c). The 14B points are unconnected to mark the configuration change. (e,f) compare backends. Tab. 4 and Appendix C.2 provide matched DiT/E2E pairs and full measurements, respectively.

5.3 End-to-end inference efficiency

Setup and accounting. On one 8×H200 141-GB node at 832×480, 4-step, 8-GPU WaveRT uses four denoising, one Store, and three VAE GPUs. We run three repetitions and report steady-state tick p50. DiT timing covers diffusion, communication, Store, and fill/drain, while E2E extends through VAE completion. Since decoding overlaps diffusion, their time gap is the exposed output tail, not total VAE compute. Both exclude loading and file encoding (Appendix C.3).

Fig. 6a separates runtime and kernel gains. At 1.3B, WaveRT improves BF16 E2E throughput from 76.7 FPS with Sync to 84.9 FPS (10.7%) on the same WF graph. WaveRT + Sage reaches 117.7 E2E FPS versus 124.1 DiT FPS in the same runs (Tab. 4). Adding W8A8 lowers tick p50 further but reduces E2E throughput to 115.9 FPS. Thus, faster ticks need not improve E2E throughput.

At 14B, the corresponding E2E rates are 16.9, 19.1, 24.8, and 28.1 FPS, versus a 3.67-FPS single-GPU reference. The best single-GPU-relative gains (7.9× at 1.3B and 7.7× at 14B) combine changes to parallelism, the graph, kernels, and decoding.

5.4 System ablations

Fig. 6b fixes the model, WF graph, schedule, topology, and local compute per group, normalizing tick p50 to Sync. Sync gathers layer K/V before attention, whereas Overlap publishes it early. One-sided uses CUDA IPC P2P, Staggered prefetches readiness checks, and Paged writes directly into final attention-buffer slots.

Early publication gives the largest adjacent reduction per group, and Paged has the lowest p50 in all six. Sync-to-Paged lowers BF16 tick p50 by 9.3% at 1.3B and 7.4% at 14B. Gains on this already distributed graph are not monotonic, as Staggered slightly slows 14B BF16. Appendix C.3 gives repeat ranges and transport diagnostics. VAE profiling gives approximately 95/99/96-ms stage times per chunk, close to the 95.4-ms Sage DiT tick (Tab. 8).

5.5 Comparison with traditional context parallelism

Fig. 6c,d compares BF16 DiT throughput for attention-level CP (Ulysses and Ring) (Jacobs et al., 2023; Liu et al., 2024) and WP's chunk–stage parallelism across six scales. Timing includes communication and synchronization but excludes VAE decoding. All methods use matched step counts and GPU budgets: 5-step, 6-GPU for 1.3B–9B and 4-step, 5-GPU for 14B. WP uses one GPU per denoising stage and one for Store, and these budgets satisfy Ulysses' head-divisibility requirement. Tab. 11 lists the standard 1.3B/14B and intermediate synthetic architectures. Only 1.3B uses trained weights, while larger models use random weights for throughput evaluation.

Under BF16, WP achieves the highest DiT throughput at every evaluated scale, with 1.04–$1.28\times$ speedups over the faster CP baseline. At 1.3B, WP reaches 84.47 FPS versus 65.94 FPS for Ulysses and 46.15 FPS for Ring. Panel (d) makes the relative gains explicit (3.6–28.1%). At 14B, WP reaches 19.10 FPS versus 18.28 FPS for the faster CP baseline. The gap narrows at larger scales. These configurations differ from the 4-step, 8-GPU E2E pipeline in panel (a), so cross-panel FPS differences cannot be attributed solely to decoding.

Fig. 6e,f shows the completed backend sweep at 1.3B and 14B, with results for all six scales in Tab. 6. With Sage+W8A8, WP is fastest at every scale (1.02–$2.02\times$ the faster CP). With Sage, Ulysses slightly leads at 7B, 9B, and 14B. Backend choice therefore changes the observed ranking.

6 Related Work

Streaming video generation. CausVid and Causal Forcing (Yin et al., 2025; Zhu et al., 2026) distill few-step causal generators. Self Forcing (Huang et al., 2025a) reduces the train–test gap by training on self-generated histories. Rolling Forcing (Liu et al., 2026) jointly denoises mixed-noise chunks, while EditStream and LPM (Zhou et al., 2026; Zeng et al., 2026) target interactive editing and character performance. Live Avatar and HiAR (Huang et al., 2025b; Zou et al., 2026) use fixed-timestep or matched-noise stage pipelines. WF trains with clean history and live K/V from earlier, cleaner chunks to make dependencies directional (Appendix D).

Context-parallel acceleration. Ulysses (Jacobs et al., 2023) redistributes tokens and attention heads through all-to-all communication, whereas Ring (Liu et al., 2024) circulates K/V blocks with overlapped attention. Both partition a forward pass. WP distributes chunk–stage tasks across full-model replicas, overlapping directional K/V delivery with computation (Sec. 5.5). It adapts wavefront scheduling (Hou et al., 2018; Jin et al., 2016) to WF's causal context and complements optimized attention and runtimes (Dao et al., 2022; Dao, 2024; Zhang et al., 2025; Zheng et al., 2024).

7 Limitations

Training combines mixed-noise initialization, teacher-guided distillation, and multi-branch on-policy adaptation. Simplifying its objectives and stage transitions while preserving motion and long-video stability remains open. WaveRT still exposes initial dependency waits and round-boundary synchronization. With Sage+W8A8, WP leads Ulysses across tested scales, but the 14B margin remains modest (Sec. 5.5). Further runtime optimization should jointly address local computation, K/V delivery, and synchronization. We aim to make communication–computation overlap driven by stage-wise workload differences a reusable runtime primitive for streaming video generation, and evaluate its broader applicability.

8 Conclusion

Wave Forcing co-designs causal context, training, and multi-GPU execution to advance chunks concurrently with multi-step refinement. Its bounded-width wavefront targets one output chunk per forward round after pipeline fill. WP realizes shorter output intervals by overlapping directional K/V delivery with computation to reduce exposed waits. Our evaluations show competitive aggregate quality, stronger motion, and accelerated output. Streaming efficiency thus depends on both chunk conditioning and dependency execution.

AI USE STATEMENT

AI assistants were used to support conceptual framing and manuscript organization, discuss methodology and systems analogies, interpret available implementation and experimental evidence, translate and edit text, and develop and revise figures. This assistance does not constitute independent verification of the reported results. The authors remain responsible for the paper's claims, citations, and disclosures.

REFERENCES

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollar, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. In *International Conference on Learning Representations*, pp. 138846–138923, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/e0982cbc81401df3430ee1ff780dc7a2-Abstract-Conference.html.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *IEEE/CVF International Conference on Computer Vision*, pp. 9630–9640. IEEE, 2021.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, pp. 35549–35562, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/98ed250b203d1ac6b24bbcf263e3d4a7-Abstract-Conference.html.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-Awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022. doi: 10.52202/068431-1189. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract.html.

Kaixi Hou, Hao Wang, Wu-chun Feng, Jeffrey S. Vetter, and Seyong Lee. Highly efficient compensation-based parallelism for wavefront loops on GPUs. In *IEEE International Parallel and Distributed Processing Symposium*, pp. 276–285. IEEE, 2018.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, volume 38, pp. 167283–167308, 2025a. doi: 10.52202/085713-5576. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f4823f831af67a3ef15e41a85434422a-Abstract-Conference.html.

Yubo Huang, Hailong Guo, Fangtai Wu, Weiqiang Wang, Shifeng Zhang, Shijie Huang, Qijun Gan, Lin Liu, Sirui Zhao, Enhong Chen, Jiaming Liu, and Steven Hoi. Live avatar: Streaming real-time audio-driven avatar generation with infinite length. *arXiv preprint arXiv:2512.04677*, 2025b. URL https://arxiv.org/abs/2512.04677.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818. IEEE, 2024. URL https://github.com/Vchitect/VBench.

Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, Yaohui Wang, Xinyuan Chen, Ying-Cong Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench++: Comprehensive and versatile benchmark suite for video generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(3):3268–3285, 2026. doi: 10.1109/TPAMI.2025.3633890.

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. URL https://arxiv.org/abs/2309.14509.

Tian Jin, Nirmal Prajapati, Waruna Ranasinghe, Guillaume Iooss, Yun Zou, Sanjay Rajopadhye, and David Wonnacott. Hybrid static/dynamic schedules for tiled polyhedral programs. *arXiv preprint arXiv:1610.07236*, 2016. URL https://arxiv.org/abs/1610.07236.

Hao Liu, Matei Zaharia, and Pieter Abbeel. RingAttention with blockwise transformers for near-infinite context. In *International Conference on Learning Representations*, pp. 3992–4008, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/1119587863e78451f080da2a768c4935-Abstract-Conference.html.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. In *International Conference on Learning Representations*, pp. 91177–91196, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/935151cc6cb5d8b6816133b75233775a-Abstract-Conference.html.

NVIDIA. *NVIDIA Ampere GPU Architecture Tuning Guide*, 2026a. URL https://docs.nvidia.com/cuda/ampere-tuning-guide/index.html. CUDA 13.4, Secs. 1.4.1.2–1.4.1.3. Accessed September 21, 2026.

NVIDIA. *CUTLASS: Efficient GEMM in CUDA*, 2026b. URL https://docs.nvidia.com/cutlass/latest/media/docs/cpp/efficient_gemm.html. Software Pipelining section. Accessed September 21, 2026.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024. URL https://arxiv.org/abs/2410.13720.

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

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo 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*, pp. 6613–6623. IEEE, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Yin_One-step_Diffusion_with_Distribution_Matching_Distillation_CVPR_2024_paper.html.

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 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22963–22974. IEEE, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Yin_From_Slow_Bidirectional_to_Fast_Autoregressive_Video_Diffusion_Models_CVPR_2025_paper.html.

Ailing Zeng, Casper Yang, Chauncey Ge, Eddie Zhang, Garvey Xu, Gavin Lin, Gilbert Gu, Jeremy Pi, Leo Li, Mingyi Shi, Shawn Wang, Sheng Bi, Steven Tang, Thorn Hang, Tobey Guo, Vincent Li, Xin Tong, Yikang Li, Yuchen Sun, Yue Zhao, Yuhan Lu, Yuwei Li, Zane Zhang, Zeshi Yang, and Zi Ye. LPM 1.0: Video-based character performance model. *arXiv preprint arXiv:2604.07823*, 2026. URL https://arxiv.org/abs/2604.07823.

Jintao Zhang, Jia Wei, Pengle Zhang, Jun Zhu, and Jianfei Chen. SageAttention: Accurate 8-bit attention for plug-and-play inference acceleration. In *International Conference on Learning Representations*, pp. 71566–71585, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/b286c344d38e10d2466c0514b78e2f36-Abstract-Conference.html.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595. IEEE, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583, 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Yuqian Zhou, Zhenghong Zhou, Zongze Wu, Cameron Smith, Richard Zhang, Jiebo Luo, Eli Shechtman, and Zhe Lin. EditStream: A unified autoregressive framework for interactive video generation and editing. *arXiv preprint arXiv:2608.21424*, 2026. URL https://arxiv.org/abs/2608.21424.

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. *arXiv preprint arXiv:2602.02214*, 2026. URL https://arxiv.org/abs/2602.02214.

Kai Zou, Dian Zheng, Hongbo Liu, Tiankai Hang, Bin Liu, and Nenghai Yu. HiAR: Efficient autoregressive long video generation via hierarchical denoising. In *European Conference on Computer Vision*, pp. 225–242. Springer, 2026. URL https://link.springer.com/chapter/10.1007/978-3-032-37026-6_13.

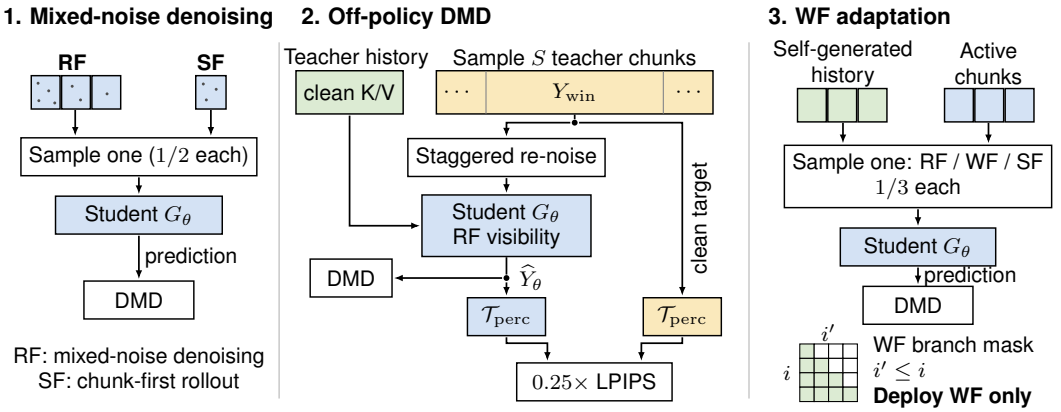

Figure 7: Training flow. Stage 1 samples one RF or SF rollout and applies DMD. Stage 2 re-noises sampled teacher chunks and applies DMD and LPIPS to the same prediction $\widehat{Y}_\theta$. Stage 3 samples one self-generated RF, WF, or SF rollout. The WF branch uses the causal mask in chronological chunk order and is the only branch deployed. Arrows carry states and predictions, and panels follow the curriculum order.

A Training Implementation Details

Noise schedules and score networks. The 4-step schedule is $[1000, 750, 500, 250]$ and the 5-step schedule is $[1000, 800, 600, 400, 200]$, before timestep warping with shift factor 5. Both use three latent frames per chunk and preserve their respective schedules through all stages and inference. The frozen DMD teacher is bidirectional Wan2.1-T2V-14B. The trainable fake-score model starts from bidirectional Wan2.1-T2V-1.3B and does not share weights with the causal student. It learns to denoise noisy, detached student predictions. Stage transitions transfer raw student weights and restart the fake-score model, optimizer, and EMA states. Evaluation uses the resulting EMA student.

Off-policy targets and perceptual loss. Stage 2 uses 2048 training and 128 validation teacher examples generated by Wan2.1-T2V-14B with 50 denoising steps, guidance scale 3, and timestep shift 5. Each example contains 21 latent frames. The off-policy implementation samples S consecutive chunks ($3S$ latent frames), constructs student K/V from preceding clean teacher chunks, and re-noises the sampled chunks to form a set of chunks at staggered noise levels. DMD and LPIPS are applied to the same clean prediction, with LPIPS weight 0.25 for both model versions. The fake-score update uses student predictions from these teacher-derived states.

For LPIPS, latent spatial resolution is bilinearly reduced from 60×104 to 30×52, then decoded by a frozen Wan VAE to 240×416 RGB frames. Every third decoded frame is evaluated. The 4-step training input contains 12 latent frames and decodes to 45 frames, giving 15 perceptual comparisons. The 5-step training input contains 15 latent frames and decodes to 57 frames, giving 19 comparisons. Decoded values are clamped to $[-1, 1]$, and VGG-based LPIPS is averaged over the sampled frames.

Optimization and updates. The student and fake-score learning rates are 1.5×10^{-6} and 4×10^{-7}, with Adam moments $(0, 0.999)$. EMA uses decay 0.99 and starts after 200 training iterations. The native 4-step runs used eight workers with one sample each and no gradient accumulation, giving an effective batch size of eight. Over the three stages, the 3000/1000/2000 training-loop iterations correspond to 600/1000/400 student updates and 3000/1000/2000 fake-score updates. Stage 1 and Stage 3 update the student once per five fake-score updates, whereas Stage 2 updates both models each iteration.

Rollout visibility. The Stage 3 sampler selects RF, WF, or SF for the same student with equal probability. The WF path reuses the RF rollout implementation with chunk-wise causal attention to advance the wavefront: clean history and the same or earlier active chunks are visible, while later active chunks are masked. This matches Eq. (4). The RF and SF paths remain complementary training distributions.

B Runtime Implementation and Analysis

B.1 Process ownership and reference execution

WaveRT uses S denoising processes and one store process, each owning a GPU and a full DiT replica. A dedicated group of P_{VAE} GPUs owns contiguous decoder partitions, giving $S + 1 + P_{\mathrm{VAE}}$ GPUs. GPU allocations list denoising + Store + VAE: $4 + 1 + 3$ for 4-step and $5 + 1 + 2$ for 5-step WaveRT, both totaling eight GPUs. The store process performs a full clean-state cache refresh, rather than an additional denoising update.

The local forward uses SGLang Diffusion for loading, conditioning, scheduling, dense operators, and cross-attention. WaveRT interposes after self-attention QKV projection and RoPE to publish K/V and construct the legal context. It owns the temporal pages and replaces the base runtime's unused large self-attention cache with a placeholder. BF16/SDPA serves as the reference, with SageAttention and W8A8 linear layers providing approximate compute modes.

The synchronized reference gathers every rank's layer K/V, including store:

$$\big((K^{\ell}_{k,r}, V^{\ell}_{k,r})\big)_{r=0}^{S} = \mathrm{AllGather}^{S}_{r=0}(K^{\ell}_{k,r}, V^{\ell}_{k,r}). \tag{12}$$

Idle ranks provide dummy payloads during fill/drain, but their slots never enter attention. Each active rank keeps all query heads. The RF path reads every active denoising slot, whereas WF selects the context in Eq. (5). Synchronized and asynchronous WF share the same context rules. Native RF comparison additionally changes the visibility graph and requires matching history and positions.

B.2 K/V publication and buffer lifetime

At initialization, consumers allocate remote K/V slots and device-side generation flags indexed by producer, round buffer, and layer, then exchange IPC memory handles. After rank j produces layer ℓ K/V, its communication stream waits on the producer event, copies the payload to each active consumer $s < j$, and publishes the generation flag. A consumer's compute stream waits for the matching flag immediately before attention. All legal cleaner ranks, including nonadjacent chunks and active store, must be ready.

This produce–publish–wait-before-use order follows buffered software-pipeline principles (NVIDIA, 2026a;b), applied here to newly computed inter-GPU K/V rather than prefetched static tiles. Causal visibility permits producer lead but does not guarantee it. WaveRT enforces the order explicitly because ordinary CTA scheduling supplies no such guarantee. The CUDA IPC implementation uses `cuMemcpyDtoDAsync`, whose peer copies may execute through an SM kernel. An isolated copy-engine benchmark therefore does not establish the deployed transport path.

The paged path writes remote segments directly into their final attention locations:

$$\mathrm{KPage}, \mathrm{VPage} \in \mathbb{R}^{2 \times L \times N_{\max} \times n_{\mathrm{head}} \times d_{\mathrm{head}}}. \tag{13}$$

The leading dimension double-buffers rounds, and $N_{\max}$ bounds context tokens. Consumers fill anchor, retained history, and self slots, while producers fill legal remote slots. Attention reads the resulting ordered prefix without per-layer remote `torch.cat`. Here, "paged" means fixed-slot placement and does not eliminate local/history copies. Outgoing copies complete before the round boundary permits reuse, and generation flags distinguish successive uses of a slot.

B.3 Round schedule and cache semantics

Algorithm 1 implements the coordinates and active set defined in Sec. 3. It runs $N_{\mathrm{chunk}} + S$ rounds, including fill/drain. The latent All-Gather at each round boundary remains in the asynchronous implementation and is distinct from the removed per-layer K/V collective.

Context order is exactly Eq. (5): anchor, bounded clean history, live store, self, then cleaner denoising slots in increasing stage order. The live store page becomes retained history after the round, avoiding a second steady-state broadcast. The first clean chunk serves as a persistent anchor, with its unrotated key retained so RoPE can be reapplied when the bounded window shifts. Other pages reuse their stored RoPE-applied keys.

Algorithm 1 WaveRT round schedule (one request, N_{chunk} chunks)

Require: S denoising ranks, store rank S, identical weights and fixed scheduler
Initialize empty anchor/history and generation-indexed K/V buffers
for $k = 0, \ldots, N_{\text{chunk}} + S - 1$ **do**
  Run the following on each rank r concurrently; let $i = k - r$
  if $r < S$ and $0 \le i < N_{\text{chunk}}$ **then**
    Rank 0 admits $Z_i^{(0)}$; other ranks use the previous tick's latent
    for $\ell = 1, \ldots, L$ **do**
      Produce local Q/K/V and publish K/V to active ranks $s < r$
      Wait for legal cleaner-rank and store pages for (k, ℓ)
      Apply attention with Eq. (5), then finish the layer
    end for
    Apply the stage transition with noise keyed by chunk and stage
  else if $r = S$ and $0 \le i < N_{\text{chunk}}$ **then**
    Refresh K/V from clean B_i; publish each layer to active denoising ranks
    Enqueue B_i for the streaming VAE
  end if
  Drain outgoing copies and promote the current store page into history
  All ranks join latent All-Gather; receivers take their predecessor's output
end for

Store processes B_{k-S} at the configured `context_noise` timestep, discards its prediction, and keeps each layer's K/V. This timestep is specified independently of S. Store attends to its anchor, retained history, and self. Its history budget reserves one query chunk, whereas denoising ranks reserve S chunks, so their prefix lengths can differ despite sharing the same clean-page list. Refresh timestep, prefix budgets, positions, and promotion order are part of the fixed semantics.

Conditioning, attention pages, and VAE convolution caches are request-local. The resident launcher handles one queued request at a time. Teardown orders outstanding copies, closes imported IPC handles before releasing backing tensors, and avoids a failure-path collective that could wait on a failed peer.

B.4 Output cadence, work, and fill/drain

Let T_s be stage s's local service time and T_{store} the clean-refresh time, measured for a specified graph, context, and backend. Sequential generation has period

$$T_{\text{serial}}^{(S)} = \sum_{s=0}^{S-1} T_s + T_{\text{store}}, \qquad T_{\text{serial}}^{(1)} = T_0 + T_{\text{store}}. \tag{14}$$

With $S + 1$ stage-owned replicas, the filled wave period satisfies

$$T_{\text{wave}}^{(S)} \ge \max\{T_0, \ldots, T_{S-1}, T_{\text{store}}\}, \tag{15}$$

with equality only if communication and synchronization are fully covered. This permits an S-update pipeline to emit faster than serial one-step inference, but uses more replicas and does not reduce work or first-output latency. Across N_{chunk} chunks it still executes $N_{\text{chunk}}(S+1)$ full forwards. Structural slot occupancy is

$$\eta_{\text{slot}} = \frac{N_{\text{chunk}}}{N_{\text{chunk}} + S}. \tag{16}$$

All replicas are active only for $S \le k < N_{\text{chunk}}$, so requests with $N_{\text{chunk}} \le S$ never fill the pipeline. The 399-latent-frame workload has 133 chunks, giving $133/137 = 97.1\%$ slot occupancy at $S = 4$. This counts active forwards, not measured GPU utilization: unequal contexts, communication, and waits still leave idle time. First-output latency includes all denoising stages, store computation, VAE pipeline fill, and delivery.

B.5 Streaming VAE partitioning

Completed latents enter the decoder queue. Each latent-frame microbatch traverses contiguous decoder stages in order, preserving per-request causal-convolution caches. Profiling M_{VAE} ordered

units at representative streaming shapes gives service times τ_m^{VAE}. Dynamic programming chooses P_{VAE} partitions by

$$\min_{0=b_0<\cdots<b_{P_{\mathrm{VAE}}}=M_{\mathrm{VAE}}} \max_{1\le p\le P_{\mathrm{VAE}}} \sum_{m=b_{p-1}+1}^{b_p} \tau_m^{\mathrm{VAE}}. \tag{17}$$

This balances estimated local service, including costs missed by unit counts or FLOPs. Inter-stage delivery must also fit the arrival interval. In per-latent-chunk units, the steady decoded-output interval is approximately

$$T_{\mathrm{emit}}^{\mathrm{E2E}} \approx \max\left(T_{\mathrm{round}}^{\mathrm{WP}}, \max_p T_{\mathrm{VAE},p}^{\mathrm{chunk}}\right) + T_{\mathrm{delivery}}^{\mathrm{exposed}}, \tag{18}$$

where decoder service amortizes all frame microbatches in one chunk, and the last term counts residual delivery not already covered by overlap or the service terms. Slower decoding grows queues or triggers bounded-buffer backpressure. The model excludes fill/drain and file encoding. Tab. 8 reports profiling, while Fig. 3 illustrates the schedules.

B.6 Same-graph semantic preservation

Wave Forcing changes the model graph relative to RF. Once the WF graph is fixed, synchronized and asynchronous execution can be compared at the operator level.

Proposition 1 (Same-graph scheduling)**.** *Fix the model parameters, denoising schedule, initial noise, per-update re-noising draws, text conditioning, positional embeddings, and history/store rules. Assume each local operator evaluates the same mathematical function and every attention operation receives the ordered K/V context in Eq.* (5)*, after the corresponding K/V production and delivery have completed. Then dependency-respecting asynchronous execution and synchronized execution produce the same mathematical latent sequence.*

Proof. At the first tick, both executions start from identical inputs and empty history. At each layer, local Q/K/V projections depend only on that rank's current hidden states and fixed conditioning. The synchronized execution gathers all projections before consuming them. The asynchronous execution may publish and consume them at different times, but the readiness condition guarantees the same legal K/V values and sequence order at attention. Consequently each attention result and the remaining local layer outputs agree. Induction over layers gives identical clean-latent predictions, from which the fixed scheduler and noise draws produce identical stage outputs. Store receives the same clean latent, runs the same cache-refresh computation, and promotes the same pages after the tick. Induction over ticks establishes identical histories and emitted latents through fill, steady state, and drain. □

This is mathematical equivalence under fixed operators and inputs, not a bitwise guarantee across kernels, reduction orders, quantization, or random-number assignments. RF versus WF also changes visibility.

B.7 Timing contract and multiple producers

On a common time base, let t_v^{local} be consumer $v = C_{k,s,\ell}$'s local-input/compute-stream ready time before remote waits. For each required edge e, set $t_e^{\mathrm{need}} = t_v^{\mathrm{local}}$ and $t_e^{\mathrm{arrive}} = t_e^{\mathrm{ready}} + \tau_e$. Eq. (11) then gives

$$t_v^{\mathrm{start}} = \max\left(t_v^{\mathrm{local}}, \max_{e\in \mathrm{In}_{\mathrm{KV}}(v)} t_e^{\mathrm{arrive}}\right), \qquad w_v = t_v^{\mathrm{start}} - t_v^{\mathrm{local}} = \max_{e\in \mathrm{In}_{\mathrm{KV}}(v)} w_e. \tag{19}$$

The incoming set includes all legal cleaner ranks and active store. When it is empty, $t_v^{\mathrm{start}} = t_v^{\mathrm{local}}$ and $w_v = 0$. Earlier waits already affect these timestamps. Round critical paths additionally include transfer serialization, contention, and the boundary collective. Summing concurrent durations would double-count overlap. Measuring this contract requires correlated timestamps across calibrated device clocks, not just aggregate rank times.

B.8 BOUNDED ATTENTION STORAGE AND COMMUNICATION

The two K/V arrays in Eq. (13) have fixed capacity for anchor, bounded history, live store, and at most S denoising chunks, so attention-page storage is independent of generated video length. This bound excludes model weights and other request state. Explicitly retained video context is limited to the anchor and recent history.

For direct publication at steady state, denoising rank j sends its K/V chunk to j noisier ranks, for $S(S-1)/2$ payload destinations among denoising stages per layer. Store adds S destinations within the full wavefront. This quadratic count in S covers logical payload copies, while physical link traffic depends on transport and topology. The practical gain is reduced exposed communication under the timing condition in Eq. (11).

Index conventions. Chunk, denoising-stage, and rank indices are zero-based, whereas Transformer layers and VAE partitions are one-based. Diagram cells are tasks, whereas $Z_i^{(s)}$ is an input state. A round is a logical unit with variable wall time. Tab. 3 summarizes the graph-specific symbols, with local timing and shape variables defined alongside their equations.

Table 3: Graph and runtime notation. The complete wavefront includes store, whereas the denoising subset does not.

Symbol	Meaning
N_{chunk}, i	number of requested chunks; chunk index $0 \le i < N_{\text{chunk}}$
S, s, ν_s	denoising-update count; denoising stage $0 \le s < S$; diffusion timestep (store occupies execution stage $s = S$)
k, r	forward round; diffusion rank $0 \le r \le S$ (rank S is store)
$Z_i^{(s)}, B_i$	state after s updates; clean endpoint $B_i = Z_i^{(S)}$
$\mathcal{A}_k, \mathcal{F}_k$	active denoising-stage indices and their input states
$\mathcal{W}_k$	complete wavefront state, including the clean input to store when active
$\mathcal{C}_{k,s}^{\text{WF},\ell}$	ordered K/V context; $k = i + s$ links it to $\mathcal{C}_{i,s}$
$A_k^\ell, H_k^\ell, U_k^\ell$	anchor, bounded clean-history, and live store K/V sequences
$P_{k,s,\ell}, C_{k,s,\ell}$	Q/K/V production; attention and remaining block tasks

C EVALUATION DETAILS

VBench protocol. We follow the standard VBench benchmark (Huang et al., 2024) with the original prompts, normalization, and weights, and scale scores to 0–100. For detailed metric definitions, we refer readers to the VBench paper. All metrics are computed with the official standardized evaluation scripts.

VBench-Long protocol. We report the VBench-Long Quality score and Dynamic Degree, scaled to 0–100. Each method uses the same 200 MovieGen prompts and seed 42, generating 399 latent frames and saving 1593 frames (about 100 seconds) at 16 FPS per video. Causal Forcing (Full Attn.) uses the EMA generator of the released `chunkwise/longvideo.pt` checkpoint with its native 4-step schedule and full attention, for both standard VBench and VBench-Long.

Generation setup. Rolling Forcing and Causal Forcing are evaluated in their own code repositories. RF with the causal mask and Wave Forcing use WaveRT with causal KV context, paged exchange, and a time-partitioned VAE. The 4-step and 5-step models use 4+1+3 and 5+1+2 GPUs, respectively, ordered as denoising + Store + VAE. Standard inference uses SDPA, and fast inference uses SageAttention.

C.1 LONG-VIDEO DIAGNOSTICS

Prompt subset. Both diagnostics require a principal subject named in the prompt. A fixed text rule first removes, for every method, group scenes, scenery, first-person views without a stable visible

subject, ambiguous principal entities, and prompts that ask the subject to disappear or transform. Of the 200 MovieGen prompts, 131 pass this rule. We then sample 100 of these prompts once at random and hold that subset fixed across methods, ablation studies, and both diagnostics. This subject-focused subset is distinct from the full 200-prompt VBench-Long evaluation.

Subject Preservation. We measure the prompt's principal-subject visibility.

SAM3 tracks the text-specified entity at 8 FPS, giving 797 sampled frames per video. The initial binding is the largest detection in the first eight frames, and masked DINO (Caron et al., 2021) features decide which later detections share that identity. Let A_t be the current mask area and A_0 the median mask area of the bound identity over the eight frames that begin at this binding. An associated box within 1% of the image border has visibility weight $\min(1, A_t/(0.18A_0))$, while any other associated detection has weight 1. A new instance elsewhere keeps its own track while the recognized subject still has weight at least 0.5 and mask area at least $0.18A_0$. Frames without an associated detection have weight 0. The video score is 100 times the mean weight over all sampled frames, including frames before the binding and frames without support. Videos that never receive an initial binding, and videos whose binding was reviewed as incorrect, are left unscored.

Camera Drift. Camera Drift measures persistent background motion accompanied by the subject moving toward an image boundary. We use the union of SAM3 detections for the text-specified entity at 8 FPS and point trajectories seeded outside those masks at 4 FPS. All distances are normalized by the image diagonal. For each image edge e, let g_e and q_e be the 5th and 90th percentiles of mask-pixel distances to that edge. Foreground outward velocity is the decrease in q_e divided by the 0.25-second pair duration. Background velocity is the component-wise median of per-cell median track velocities over supported spatial cells, projected onto the same outward axis. We apply $z(v) = \operatorname{sign}(v)\max(|v| - 0.002, 0)$ to both velocities.

Within the preceding five seconds, we use pairs with observed foreground and background motion. For either corrected velocity sequence, define outward displacement $A = \max(0, 0.25\sum_t v_t)$, path length $P = 0.25\sum_t |v_t|$, and response $H = (A/P)A^2/(A^2{+}0.1^2)$, with $H = 0$ when $P = 0$. These give foreground and background factors F_e and B_e. Edge proximity is $E_e = \max(0, 1 - g_e/0.1)$. The instantaneous risk is $\max_e E_e F_e B_e$ over edges with positive current foreground and background velocities, or zero if no edge qualifies. Thus, background motion with a stationary subject does not incur a drift penalty.

To retain evidence after an exit, we require one second of consecutive outward foreground and background motion, at least two native pixels of foreground displacement, a final g_e within two native pixels of the boundary, and $q_e \leq 0.1$. After one second of consecutive explicitly empty masks, we hold the exit's $E_e F_e$ fixed and update B_e from current background observations, without backfilling the confirmation interval. Missing background tracks can be replenished by points consistent with the existing motion layer, but new points contribute only after tracking through a subsequent pair. This is a geometric association, not an independent semantic background label. A returning detection, uncertain mask, or scene cut ends the extension. Stopped or reversed background motion immediately gives zero risk and ends the extension after one consecutive observed second. Background gaps remain unscored, and later continuation is distinguished from uninterrupted support.

The video score is 100 times the mean risk over valid 0.25-second intervals, so lower is better. Startup intervals before five seconds, unavailable current measurements, and windows crossing a detected or unresolved cut remain unscored. Missing motion within a window is neither interpolated nor extrapolated. We average video scores equally over the prompt subset, requiring at least 20 scored seconds per video and using each method's own valid intervals.

C.2 Complete performance measurements

Tables 4 and 5 provide the WaveRT measurements plotted in Fig. 6a,b, including matched DiT/E2E rates in Tab. 4. Table 6 reports DiT throughput for each model scale, backend, and parallel strategy in Fig. 6c–f.

Experiment groups. Tables 4 and 6 belong to the throughput evaluation, covering runtime performance and parallel-strategy comparisons. Tables 5 and 7 come from a different set of runs for

controlled transport ablations, reporting aggregate tick p50 and per-run ranges, respectively. Matching configuration names across these two groups do not identify the same measurements. In particular, the Paged timings in Tab. 5 need not equal those in Tab. 4. Within the throughput group, comparisons must also respect the step counts, GPU allocations, and timing scopes specified in each table.

Table 4: Runtime throughput. WaveRT uses the WF graph and paged K/V unless marked Sync, while Baseline is single-GPU rolling inference. Both use BF16 unless a kernel suffix is specified. GPU counts include denoising, Store, and VAE: 4+1+3 for 4-step and 5+1+2 for 5-step. Speedup is E2E relative to the same-scale, same-step baseline. The 14B runtime uses random weights.

Configuration	GPUs	Tick p50 (ms)	Steady (FPS)	DiT (FPS)	E2E (FPS)	Speedup
		1.3B, 4-step				
Baseline	1	–	–	–	15.0	1.0×
WaveRT (Sync)	8	153.4	78.3	77.3	76.7	5.1×
WaveRT	8	139.1	86.3	85.6	84.9	5.7×
WaveRT + Sage	8	95.4	125.7	124.1	**117.7**	**7.9**×
WaveRT + Sage + W8A8	8	**94.9**	126.5	121.5	115.9	7.7×
		1.3B, 5-step				
Baseline	1	–	–	–	11.9	1.0×
WaveRT	8	139.6	86.0	84.5	83.7	7.0×
WaveRT + Sage	8	95.1	126.1	118.9	**85.0**	**7.1**×
WaveRT + Sage + W8A8	8	111.3	107.8	101.9	83.8	7.0×
		14B, 4-step				
Baseline	1	–	–	–	3.67	1.0×
WaveRT (Sync)	8	701.6	17.1	16.9	16.9	4.6×
WaveRT	8	619.5	19.4	19.1	19.1	5.2×
WaveRT + Sage	8	476.0	25.2	24.9	24.8	6.8×
WaveRT + Sage + W8A8	8	**420.8**	28.5	28.2	**28.1**	**7.7**×

Table 5: WaveRT same-graph system ablations report steady tick p50 (ms, lower is better) for the 4-step, 8-GPU (4+1+3) setup. Each row fixes the compute backend. The 14B runs use random weights and measure execution cost only.

Backbone	Compute	Sync	Overlap	One-sided	Staggered	Paged
1.3B	BF16	153.4	142.1	140.2	140.2	**139.1**
1.3B	Sage	110.5	99.5	96.4	96.2	**95.4**
1.3B	Sage+W8A8	112.0	99.6	95.8	95.7	**94.9**
14B	BF16	701.6	666.5	657.2	657.7	**649.8**
14B	Sage	531.7	501.6	488.1	487.8	**480.2**
14B	Sage+W8A8	474.3	440.2	429.1	428.9	**421.1**

C.3 Runtime implementation and measurement protocol

The WaveRT runtime sweep uses one NVLink-connected node with eight H200 141-GB GPUs, 832×480 video, and three latent frames per chunk. We implement WaveRT using Python 3.12, PyTorch 2.11.0+cu128, and SGLang Diffusion.

For the WaveRT measurements in Tabs. 4 and 5, each repetition generates a 1593-frame video, matching the long-video evaluation. Tick p50 is aggregated over steady-state ticks across three repetitions. DiT timing includes fill/drain, and E2E timing extends through VAE completion. Loading and file encoding are excluded. We evaluate BF16/SDPA, SageAttention, and SageAttention with W8A8 linear computation.

The 14B measurements use the standard Wan2.1-T2V-14B architecture with randomly initialized weights to evaluate execution cost.

Table 6: Distributed-strategy comparison across three backends. Steps and GPU counts apply to each method in the row. DiT FPS excludes VAE completion, whereas E2E FPS includes it. Tab. 4 reports matched DiT/E2E pairs. Bold marks the highest DiT FPS per row. WP/CP is WP divided by the faster CP, so values below one indicate a CP lead.

				DiT throughput (FPS) ↑			
Scale	Compute	Steps	GPUs	Ulysses CP	Ring CP	WP	WP/CP
1.3B	BF16	5	6	65.94	46.15	**84.47**	1.28×
1.3B	Sage	5	6	76.82	33.17	**118.88**	1.55×
1.3B	Sage+W8A8	5	6	50.53	27.29	**101.85**	2.02×
3B	BF16	5	6	49.89	28.19	**55.16**	1.11×
3B	Sage	5	6	66.66	35.76	**75.38**	1.13×
3B	Sage+W8A8	5	6	56.41	32.84	**78.20**	1.39×
5B	BF16	5	6	39.61	26.02	**44.20**	1.12×
5B	Sage	5	6	50.90	29.39	**57.11**	1.12×
5B	Sage+W8A8	5	6	44.39	27.15	**62.24**	1.40×
7B	BF16	5	6	33.83	24.27	**35.48**	1.05×
7B	Sage	5	6	**47.39**	29.92	46.23	0.98×
7B	Sage+W8A8	5	6	45.62	29.41	**51.84**	1.14×
9B	BF16	5	6	30.19	22.20	**31.28**	1.04×
9B	Sage	5	6	**41.47**	27.15	40.37	0.97×
9B	Sage+W8A8	5	6	44.62	29.08	**45.48**	1.02×
14B	BF16	4	5	18.28	14.42	**19.10**	1.04×
14B	Sage	4	5	**25.23**	17.08	24.80	0.98×
14B	Sage+W8A8	4	5	25.08	17.15	**28.08**	1.12×

C.4 Throughput accounting and repeatability

Tab. 4 reports steady, complete-request DiT, and E2E rates together for each runtime configuration. Fill/drain and decoder service explain why these rates are not interchangeable. In particular, the lowest 1.3B tick p50 (WaveRT + Sage + W8A8) does not yield the highest E2E FPS.

VAE bottleneck in 5-step generation. The small E2E gain from SageAttention in the 5-step results of Tab. 1 is limited by VAE decoding, rather than a lack of DiT acceleration. Under the 8-GPU budget, 5-step WaveRT assigns five GPUs to denoising, one to Store, and two to a two-stage VAE pipeline (PP2). Compared with the three-stage VAE pipeline in the 4-step setup, PP2 has longer per-stage decoding service times. Its slowest stage limits the decoded-output rate even when DiT produces chunks faster. Sage raises 5-step DiT throughput from 84.5 to 118.9 FPS (Tab. 4), but E2E throughput increases only from 83.70 to 84.98 FPS (Tab. 1), about 1.5%. Accelerating DiT alone therefore does not remove this output-path bottleneck.

The same-graph transport ablation (Fig. 6b and Tab. 5) compares Sync with asynchronous delivery. Tab. 7 reports the minimum and maximum per-run tick p50 from the same ablation group, across three repetitions on the same node for causal 4-step, 8-GPU WaveRT. These ranges characterize the ablation measurements in Tab. 5, not the throughput measurements in Tabs. 4 and 6.

Table 7: Minimum–maximum of per-run tick p50 (ms) across three repetitions on the same node for causal 4-step, 8-GPU WaveRT.

Scale	Compute	Sync	Overlap	One-sided	Paged
1.3B	BF16	153.24–153.50	142.04–142.13	140.14–140.47	139.04–139.16
1.3B	Sage	110.15–110.96	99.41–99.72	96.40–96.52	95.21–95.57
14B	BF16	700.06–702.38	666.30–666.66	654.57–658.51	647.38–650.87

Table 8: Component profiles of stage service, K/V transfer, receiver-side assembly, and VAE partitioning. Each row specifies its measurement scope.

Diagnostic	Measurement	Value
14B critical consumer	r0 steady stage	646 ms
14B cleaner producers	r1/r2/r3/store	575–581 ms
Aggregate rank-time difference	Critical minus producers	≈ 70 ms
BF16 K/V communication	Synchronized critical path	≈ 79 ms
FP8 K/V communication	Byte-reduction experiment	≈ 40 ms
Copy-engine P2P	One-sided microbenchmark	343 GB/s
Context assembly	`torch.cat` to paged	18.49 to 4.08 ms
VAE FLOP-based split	Maximum stage service per chunk	146 ms
VAE time-based split	Three stage services per chunk	95 / 99 / 96 ms

C.5 Mechanism diagnostics

Tab. 8 reports component-level profiles. The ≈ 70-ms difference compares aggregate rank service times, while per-layer overlap depends on K/V readiness and consumer timing (Appendix B.7). FP8 K/V reduces communication time from approximately 79 to 40 ms, and paged placement reduces receiver-side context assembly from 18.49 to 4.08 ms. These components can overlap with computation, so their E2E impact depends on the waits exposed along the critical path rather than the sum of their durations. Time-based VAE partitioning gives approximately 95/99/96-ms stage service times per chunk. The slowest stage is close to, but still above, the 95.4-ms Sage DiT tick.

Transport correctness. In the tested 1.3B, 4-step configuration, synchronized, one-sided, and paged execution of the same WF graph produce bit-identical latent outputs (`torch.equal=True`, zero maximum absolute difference). Same-graph scheduling equivalence is established in Appendix B.6.

C.6 Exposed communication, synchronization, and payload size

WaveRT targets time exposed on the consumer's critical path, not only bytes on the link. Early publication overlaps delivery with computation. One-sided writes replace matching collective coordination with point-of-use readiness checks, and paged placement avoids rebuilding remote context. Together, these changes reduce exposed communication, synchronization, and destination-assembly costs without eliminating data dependencies. At fixed BF16 computation, the complete 4-step 14B Sync/Overlap/One-sided/Paged tick p50 decreases from 701.6 to 666.5, 657.2, and 649.8 ms (Tab. 5).

Payload size determines how much delivery must fit under computation. One chunk sent to one consumer at one layer carries $m_e(b) = 2N_C n_{\text{head}} d_{\text{head}} b$ bytes, with b bytes per element. Larger K/V width increases each payload, while more layers and destinations increase round traffic (Appendix B.8). Fixed hardware has unchanged peak link capacity, but effective bandwidth varies with topology and contention. Taking unquantized K/V readiness as the reference, a simple usable-delivery model is

$$\widetilde{\tau}_e(b) \approx t_e^{\text{quant}} + \alpha_e + \frac{m_e(b)}{B_e^{\text{eff}}} + t_e^{\text{dequant}}, \tag{20}$$

where α_e covers queueing and signaling, and B_e^{eff} is achieved payload bandwidth. BF16 requires no conversion, whereas FP8 halves payload bytes, excluding metadata, at the cost of conversion. With unchanged local timing, a previously exposed dependency can become hidden if

$$\widetilde{\tau}_e(1) \le \Delta_e < \widetilde{\tau}_e(2). \tag{21}$$

Every required edge, including store, must meet its deadline: local readiness for *attention*, not the start of projection. Quantization can change execution timing, so actual slack must be measured per configuration.

The component profiles in Tab. 8 illustrate reduced transfer volume with FP8 K/V and reduced receiver-side assembly with paged placement. FP8 conversion can offset transfer savings under SageAttention. In the paged backend sweep (Fig. 6e,f), K/V remains unquantized, while W8A8

changes local DiT linear computation. These configurations vary the local compute backend while retaining the same K/V transport precision.

D Comparison of Execution Designs

D.1 Generation schedules and conditioning

Similar chunk–stage diagrams need not compute the same model. Tab. 9 distinguishes the state supplied to attention from the placement of work. All methods retain full within-chunk attention, but their weights, denoising schedules, and cache rules are method-specific.

Table 9: Conditioning and exchanged state in different generation schedules. Live K/V is produced by active chunks at the current Transformer layer.

Schedule	Context state	Across-stage transfer	Execution structure
Self Forcing	Completed clean-history K/V	None	Sequential chunks
Live Avatar	Fixed-timestep stage-local K/V snapshots	Updated latent	Stage pipeline
HiAR	Refreshed output-timestep K/V within each hierarchy	Updated latent	Causal hierarchy pipeline
Rolling Forcing	Clean history and bidirectional live K/V	All active K/V if distributed	Joint mixed-noise denoising
Wave Forcing	Clean history and causal live K/V	Cleaner-to-noisier K/V and latent	Directed wavefront

Sequential and joint-chunk generation. Self Forcing (Huang et al., 2025a) completes all updates of one chunk before starting the next, reusing clean-history K/V. Its bounded history cache can evict the initial chunk, whereas RF retains it as an anchor. RF (Liu et al., 2026) advances consecutive mixed-noise chunks jointly, emitting one clean chunk per joint call after fill. Distributed RF retains every active chunk's corresponding-layer K/V dependencies in both temporal directions. Communication overlap is still possible, but every consumer must respect those inputs. Both RF and WF are DAGs when unrolled over layers, so WF is distinguished by cleaner-only visibility rather than the mere existence of a topological order.

Stage-local and matched-noise pipelines. Live Avatar (Huang et al., 2025b) assigns a fixed denoising timestep to each process. Earlier stages retain their original K/V snapshots as a chunk progresses, and only latents move across stages. HiAR (Zou et al., 2026) instead builds a causal hierarchy: each level uses freshly produced K/V from earlier chunks at its output timestep ν_{s+1}. The cache write for chunk i can be fused with denoising chunk $i + 1$. Its pipelined levels retain independent matched-noise caches and exchange latents, not layer-wise K/V between processes. This refreshed output-state context distinguishes HiAR from fixed-timestep snapshots.

Wavefront execution. WF preserves current-layer live K/V from all legal cleaner chunks and store, and WP publishes this K/V across full-model replicas. Thus an anti-diagonal schedule alone does not imply WF conditioning. Its causal mask, cache rules, and training establish the model, while publication and readiness checks implement it. Appendix B.4 gives the cadence and resource accounting without equating steady output rate with first-output latency.

D.2 Partitioning a forward versus distributing stages

Tensor and context parallelism partition one forward while retaining its conditioning graph. Ulysses (Jacobs et al., 2023) limits CP degree by attention-head divisibility, whereas Ring Attention (Liu et al., 2024) supports head-independent degrees through ordered K/V circulation. Synchronized chunk ownership keeps local queries and all their heads intact but retains Eq. (12)'s layer collective. WP instead relies on the WF graph and sufficient producer lead, at the cost of full-model replication.

Table 10: Resource-partitioning strategies, communication patterns, and constraints. TP and CP preserve the chosen graph, while WP schedules the fixed WF graph.

Method	Partitioned object	Main communication	Main constraint
Tensor parallelism	Hidden/head dimensions	Per-operator collectives	Small local GEMMs at high degree
Ulysses CP	Tokens to attention heads	Two All-to-All phases	Head divisibility; layer collective
Ring Attention	K/V sequence blocks	Multi-round K/V circulation	Enough tile compute to hide P2P
Chunk-owned sync	Active query chunks	Layer K/V All-Gather	Global layer barrier
Wave Parallelism	Chunk–stage tasks	One-way K/V and latent	Full replica; sufficient slack

Comparison setup. The three-backend comparison in Fig. 6c–f and Tab. 6 matches parameter count, backend tier, GPU count, and denoising-step count across methods within each model scale: a 5-step, 6-GPU setup for 1.3B–9B and a 4-step, 5-GPU setup for 14B. These rates measure DiT throughput, whereas Fig. 6a reports E2E throughput for the 4-step, 8-GPU setup. CP operates on RF and WP on WF, so the comparison evaluates their generation and execution designs together. Tab. 10 summarizes how each strategy partitions work, communicates dependencies, and constrains resource allocation.

Model Configuration. The 1.3B and 14B models retain the standard Wan2.1 architectures. They share the same Wan Transformer block design but differ in the number of blocks, attention heads, hidden dimensions, and FFN dimensions. For the parameter-scale comparison in Tab. 6, we adjust these dimensions and the number of blocks to construct the intermediate 3B, 5B, 7B, and 9B architectures. Their attention-head counts are divisible by six to support six-way Ulysses parallelism. The standard 14B architecture has 40 attention heads and supports the five-way configuration used in our comparison. Tab. 11 lists all six configurations.

Only the 1.3B model uses trained weights in this scaling study. All larger models use randomly initialized weights, including the architecturally unchanged Wan2.1-T2V-14B model. We do not train a 14B Wave Forcing model because the required computation exceeds our available resources.

Table 11: Model configurations for the parameter-scale comparison. The upper group contains the standard Wan2.1 architectures, and the lower group contains the synthetic architectures.

Parameters	Layers	Heads	Hidden dim.	FFN dim.
1.3B	30	12	1536	8960
14B	40	40	5120	13824
3B	32	18	2304	10752
5B	30	24	3072	14336
7B	29	30	3840	15360
9B	26	36	4608	18432