COMMIT-ONE: DECOUPLING GENERATION AND COMMITMENT FOR INTERACTIVE VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Interactive video generation requires new conditions to influence ongoing playback promptly. A natural route is fine-grained autoregressive generation, but reducing chunk size can restrict joint temporal modeling and parallel computation. *Must fine-grained interaction therefore come at the cost of a large chunk size?* We introduce Commit-One, an algorithm-system co-design built on a different principle: **low-latency interaction requires fine-grained output commitment, not necessarily fine-grained generation.** By decoupling chunk size from output commitment, Commit-One preserves large-chunk generation while allowing future predictions to remain revocable. This enables chunk-wise autoregressive models to deliver fine-grained video streams without reducing their chunk size. The optimization target therefore shifts from making chunks smaller to making joint denoising faster: chunk size becomes a design choice balancing modeling capacity and computational efficiency, rather than a hard constraint on interaction granularity. The same principle opens the possibility of using even bidirectional teacher models directly for interactive streaming. Commit-One thus offers an alternative route to interactive video generation, centered on accelerating high-quality joint generation rather than shrinking chunk size.

1 INTRODUCTION

Interactive video generation requires newly arriving conditions to influence ongoing playback promptly. A viewer may change a camera command or action instruction at any time, while existing video continues to play. Generation throughput alone does not capture this requirement: a fast model can still leave the viewer waiting for a generation or playback boundary.

A natural route is fine-grained autoregressive generation. Advances in rollout training and few-step distillation have made this route increasingly practical (Huang et al., 2025; Zhao et al., 2026; Yang et al., 2025). Smaller generation chunks provide more frequent opportunities to incorporate new conditions, enabling fine-grained interaction, but may limit joint temporal modeling and parallel efficiency. This raises our central question: *must fine-grained interaction require a fine-grained generation model?*

We introduce Commit-One, an **algorithm-system co-design** based on a different principle: **fine-grained interaction requires fine-grained output commitment, not necessarily fine-grained generation.** The generator retains its chunk size, while the serving system distinguishes completed predictions from output that can no longer be replaced. This separation allows chunk-wise autoregressive models to deliver at a finer granularity without reducing their chunk size (Figure 1).

We first formulate a serving model that separates condition arrivals, generation, output commitment, and playback. Generation is the computation needed for deliverable video output; for the latent diffusion models studied here, it includes DiT sampling and required VAE decoding. The model distinguishes generation time from phase-dependent *random-arrival waiting* imposed by scheduling and playback boundaries, and identifies replaceable outputs. When new output can enter playback only at chunk boundaries, the chunk duration imposes a waiting floor even if generation is fast.

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.

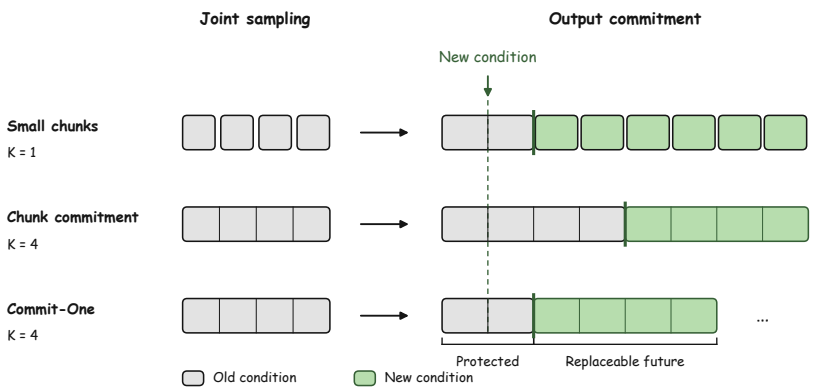

Figure 1: Fine-grained interaction without shrinking generation chunks. Commit-One preserves large-chunk joint sampling while allowing updates beyond the protected prefix, rather than only at chunk boundaries. Gray/green denote old/new conditions; the dashed line marks playback position at condition arrival, and thick green edges mark updated output. Positions represent video content, not computation time.

We next examine chunk size K, the number of latent frames jointly sampled in one DiT sampling call. Changing K affects joint temporal modeling and parallel efficiency, as well as the time needed to generate each chunk. Our chunk-size ablation shows higher aggregate video quality for larger-chunk configurations than their frame-wise counterparts in the evaluated settings (Section 4). Together with the system model, this motivates retaining the benefits of joint generation while reducing the waiting imposed by coarse output commitment.

Guided by the system model and chunk-size ablation, we propose speculative sampling to combine large-chunk joint generation with fine-grained delivery. We formulate the controller as a two-state finite-state machine that coordinates joint generation and incremental consumption of cached predictions, while playback continues independently. This decouples DiT sampling calls from individual deliveries: completed predictions are reused between condition updates, avoiding repeated sampling merely to support finer-grained delivery.

The resulting design points to an alternative route to interactive video generation: shrinking chunks need not be a prerequisite for low interaction latency. Instead, the optimization focus can shift to accelerating high-quality joint generation. DiT sampling acceleration and parallel acceleration can support this route. This perspective also motivates exploring even bidirectional teacher models for interactive streaming.

Our contributions are fourfold:

1. **A simple serving-system model for interactive video generation.** We formalize condition arrivals, generation, output commitment, and playback, distinguishing generation time from the additional waiting imposed by scheduling and playback boundaries (Section 3.1).

2. **A systematic analysis of chunk size.** We study how chunk size affects video quality, generation time, and redundant computation, clarifying the trade-offs between joint modeling and responsiveness (Section 4; Appendices A–B).

3. **Recomputation-efficient speculative sampling.** Guided by this model and analysis, we propose a speculative sampling scheme formulated as a two-state finite-state machine. It combines large-chunk joint generation with fine-grained output commitment, reusing predictions and replanning only a feasible suffix when conditions change (Section 3).

4. **An alternative route to interactive video generation.** By separating chunk size from output commitment, we shift the optimization focus from shrinking chunks to accelerating high-quality joint generation, making per-chunk generation time a central target for improving responsiveness.

2 Related work

Autoregressive video generation. Autoregressive video generation exploits temporal structure by predicting new content conditioned on the past (Yan et al., 2021). This provides a natural interface for extending videos, including segment-wise continuation under changing prompts (Villegas et al., 2022). Bringing sequential prediction to diffusion models (Chen et al., 2024) leaves two practical challenges: the cost of repeated sampling and errors accumulated through generated history. Recent progress therefore targets efficient few-step causal generation (Yin et al., 2024; Zhao et al., 2026) and training aligned with self-generated, long-horizon rollouts (Huang et al., 2025; Yang et al., 2025). Related rolling and bidirectional-window approaches retain joint temporal processing during sequential output (Kim et al., 2024; Kodaira et al., 2025; Liu et al., 2025; Rui et al., 2026). NFD additionally describes speculative multi-frame sampling on a frame-causal backbone to improve throughput (Cheng et al., 2025). Related control mechanisms include reuse or replanning of fully denoised futures with ground-truth observation feedback (Lillemark et al., 2026) and management of generation–playback lead (Bai et al., 2026). Commit-One studies a complementary interface: retaining temporally bidirectional joint generation within each chunk while committing outputs at a finer granularity under newly arriving conditions.

Accelerating video generation. Acceleration targets several components of generation cost. Few-step distillation (Yin et al., 2024; Zhao et al., 2026) and adversarial post-training (Lin et al., 2025) reduce DiT sampling steps. Pyramid Flow performs much of its DiT sampling below full spatial resolution and compresses history through a temporal pyramid (Jin et al., 2024); FramePack reduces the historical-context token budget (Zhang et al., 2025a). FasterCache and TeaCache reuse intermediate computations across diffusion steps (Lv et al., 2024; Liu et al., 2024), whereas Sliding Tile Attention and Sparse VideoGen reduce attention work through local or spatial–temporal sparsity (Zhang et al., 2025b; Xi et al., 2025). USP addresses distributed attention through sequence parallelism (Fang & Zhao, 2024). These methods have distinct training and inference requirements. Our perspective shifts the emphasis from shrinking generation chunks to accelerating their generation. By decoupling generation from output commitment, Commit-One highlights per-chunk generation time, rather than chunk size alone, as a key determinant of interaction latency. This places greater importance on accelerating joint generation instead of requiring increasingly fine-grained autoregressive models.

3 Method

3.1 Serving-System Model

An interactive video serving system generates video into a buffer while the player consumes previously generated content. New conditions arrive online and must influence subsequent playback promptly. For the latent diffusion models considered here, the *chunk size* K counts the latents jointly sampled with **bidirectional** temporal modeling.[1] Let f denote the video's temporal sampling frame rate, used for generation and playback at its intended speed, and r the pixel frames represented by each latent. A latent spans $\delta = r/f$ seconds, so a chunk covers $T = K\delta = Kr/f$ seconds.

Figure 2 illustrates why response latency is not just DiT sampling time. A condition arrives at t_{arrival}, waits until generation starts at t_{start}, and yields a first output group ready at t_{ready} and displayed at t_{playback}. For a displayed update, these stages give

$$\begin{aligned} R &= t_{\text{start}} - t_{\text{arrival}}, \qquad C = t_{\text{ready}} - t_{\text{start}}, \\ Q &= t_{\text{playback}} - t_{\text{ready}}, \quad L = R + C + Q. \end{aligned} \tag{1}$$

Here R and Q are the waits before generation and playback, respectively. Generation time C covers the critical path to the first deliverable output, including the required DiT sampling and VAE decoding.

For a conventional interface that admits newly conditioned output only at fixed chunk playback boundaries, the expected combined waiting satisfies $\mathbb{E}[R + Q] \geq T/2$ under fixed generation time

[1] Throughout this paper, our chunk-based generation setting uses bidirectional temporal modeling within each chunk and autoregressive continuation across chunks, conditioned on retained past content.

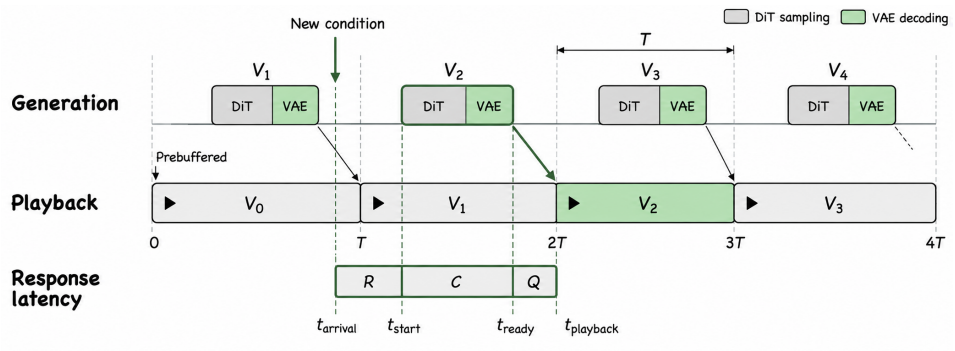

Figure 2: Generation speed alone does not determine response latency. In this periodic reference schedule, the highlighted update waits R for generation, takes C to produce deliverable output, and waits Q for playback. Each chunk is prepared for the next playback period; V_0 is prebuffered. Fixed-phase scheduling and serial computation are illustrative, not measured results.

and isolated, uniformly phased arrivals (Section 3.4). Since $T = Kr/f$, two direct ways to reduce this waiting floor at fixed r are **smaller chunks** and **higher temporal frame rates**. Both approaches reduce latency by shortening the chunk period T. We instead ask: **can we eliminate the chunk-period waiting floor by decoupling joint generation from output commitment?** This would remove the need to shrink chunks or increase the model's temporal frame rate merely to obtain finer-grained interaction, shifting the optimization focus to generation time rather than chunk duration.

3.2 Speculative Sampling via a Finite-State Machine

Our design separates how much the model jointly predicts from how much the system commits at each delivery. *Delivery granularity* q is the number of latents whose pixel frames are transferred together; Commit-One uses $q = 1$, with each regular group spanning δ seconds. *Output commitment* makes content non-replaceable and may precede actual playback. A large chunk supplies a reusable future; the controller keeps that future replaceable until needed. This is the role of *speculative sampling*, distinct from the individual DiT sampling calls it schedules.

Joint predictions and retained state. At a branch position j, let $\mathcal{H}_j$ be the conditioning state derived from the retained prefix, and let c denote the current condition. One DiT sampling call produces a K-latent prediction $\widetilde{Y}_j$:

$$\widetilde{Y}_j \sim p_\theta(\cdot \mid \mathcal{H}_j, c), \qquad |\widetilde{Y}_j| = K. \tag{2}$$

The active chunk retains bidirectional temporal modeling. Its predictions are cached with source/playback mappings and condition versions, independently of the state used to condition subsequent calls. For a chunk-wise AR model, $\mathcal{H}_j$ includes prefix KV; Section 3.3 defines an alternative for bidirectional clip models. Completed latents require the appropriate VAE decoding before pixel delivery.

Control states and events. The controller is an *extended finite-state machine* with two control states, $\mathcal{S} = \{\mathsf{Produce}, \mathsf{Consume}\}$. Produce means a generation job is active; Consume means no job is active and predictions can be reused. Playback continues in both states. Registers x store the prediction cache, conditioning and VAE state, condition versions, playback and commitment positions, and the active job J.

The events are condition arrival, output/state readiness, job completion, playback tick, and refill/retry wake-up. An event e at time t first updates the registers to $\bar{x} = U_e(x)$. The dispatch guard $g(\bar{x}, t)$ requires an idle worker, pending-update or refill demand, and a feasible branch boundary. The transition is

$$(z^+, x^+) = \begin{cases} (\mathsf{Produce}, \mathrm{Launch}(\bar{x})), & g(\bar{x}, t), \\ (\mathsf{Produce}, \bar{x}), & \bar{J} \neq \varnothing, \\ (\mathsf{Consume}, \bar{x}), & \text{otherwise.} \end{cases} \tag{3}$$

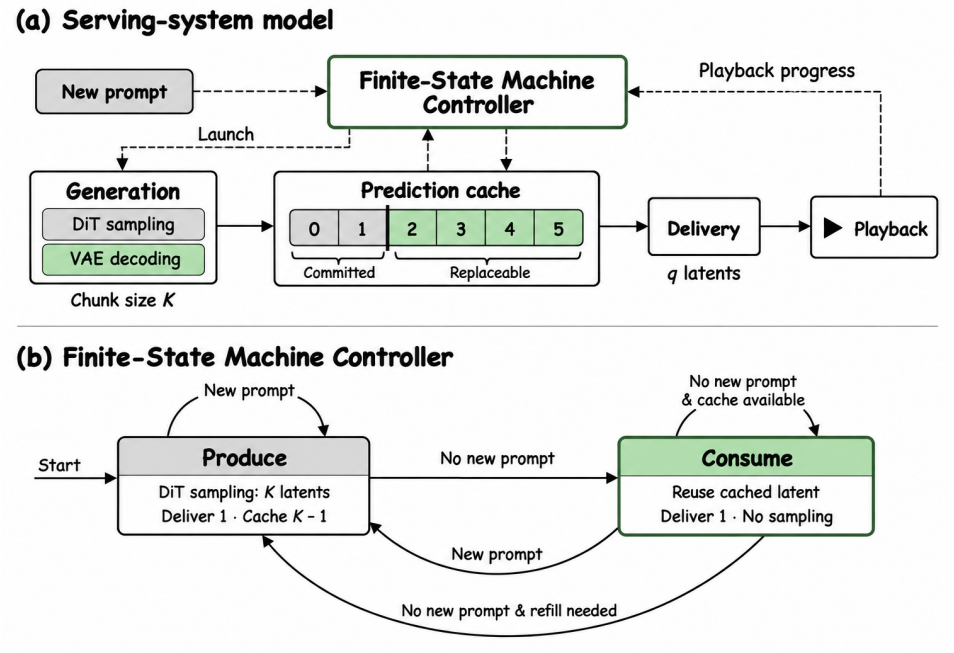

Figure 3: Commit-One serving system and control. (a) Joint generation populates a prediction cache for fine-grained delivery and playback; playback progress guides refill. (b) The finite-state machine coordinates generation in Produce and prediction reuse in Consume, illustrated for $q = 1$. Condition updates trigger feasible replanning without interrupting active generation; playback continues in both states.

Dispatch requires $\bar{J} = \varnothing$, so the first two guards are disjoint. Only Launch starts a DiT sampling call.

Guarded replacement and refill. A feasible boundary lies beyond the protected prefix, has the required conditioning inputs, preserves a valid VAE continuation, and lets the old prefix cover computation until the new content can meet its display deadlines. On an update, Launch selects the earliest such boundary, locks the prefix, and replaces only the eligible suffix with a new complete chunk. It does not shorten the new sample to the old chunk's remaining length. Without an update, it appends a chunk when remaining playable coverage approaches the estimated compute lead, before exhaustion. Infeasible requests remain pending without discarding useful predictions.

Arrivals during Produce update the pending condition but not the active job. Completion validates its results and either returns to Consume or immediately launches another eligible job. Multiple pending updates coalesce to the latest version, with superseded events recorded. Playback ticks consume ready groups in either state; they do not themselves require DiT sampling. Output readiness may precede job completion, so time C is not necessarily the full Produce residence time. Figure 3 illustrates these actions; Appendix C specifies the guards and event updates.

3.3 Adapting Bidirectional Models for Streaming

Bidirectional teacher models often define the quality target for causal streaming students, yet their full-sequence generation interfaces are not designed for interactive serving. Directly converting the same backbone to causal generation can also introduce a quality gap without careful adaptation (Yin et al., 2024). This motivates a further extrapolation of chunk size: *can a large-chunk bidirectional model itself serve an interactive video stream?* We preserve bidirectional joint sampling within each call and adapt its conditioning and output interfaces, allowing successive calls to continue from retained video while outputs are committed incrementally.

Continuation interface. Bidirectional video generators commonly provide native image- or video-conditioned continuation interfaces, allowing successive jointly sampled chunks to be connected without modifying within-chunk attention. A practical complication arises with temporally compressive 3D VAEs: the continuation interface often expects a temporally uncompressed first-frame latent, whereas an interior cached frame belongs to a temporally compressed latent group. We therefore decode the retained boundary to pixels and re-encode its final frame through the VAE's first-frame path before the next DiT sampling call. This produces the latent format expected by the pretrained continuation interface. Alternatively, the bidirectional model can be fine-tuned to accept compressed latent continuation directly.

3.4 Latency and Recomputation Analysis

Removing coarse playback alignment. Consider uniform latent groups, fixed generation time C, and an isolated update uniformly distributed over a chunk period $T = K\delta$. If new output can enter playback only at chunk boundaries, then

$$t^{\min}_{\text{playback}} = T \left\lceil \frac{t_{\text{arrival}} + C}{T} \right\rceil, \qquad \mathbb{E}[L] \geq C + \frac{T}{2}. \tag{4}$$

Since $L = R + C + Q$ and C is fixed, this also gives $\mathbb{E}[R + Q] \geq T/2$. It bounds the mean combined waiting, not each request or each component separately; Appendix A gives the derivation. This boundary-imposed waiting persists whether it occurs before or after generation. With a ready prefix, no additional buffering constraint, and fine-grained commitment, the earliest latent-aligned display is

$$\begin{aligned} t_{\text{playback}} &= \delta \left\lceil \frac{t_{\text{start}} + C}{\delta} \right\rceil, \\ Q &= t_{\text{playback}} - (t_{\text{start}} + C), \qquad 0 \leq Q < \delta. \end{aligned} \tag{5}$$

Thus alignment is tied to delivery granularity rather than chunk size. The implementation uses measured compute-lead estimates and safety margins; these ideal formulas do not guarantee that every deadline is met.

Generation time and prefix waiting. Fine commitment does not remove computation or its data dependencies. For one indivisible C-second job every T seconds, with $C \leq T$, an isolated uniform arrival is blocked with probability C/T and waits $C/2$ on average when blocked, provided that job supplies the required prefix. Hence

$$\mathbb{E}[R] = \frac{C^2}{2T}, \qquad \mathbb{E}[L] = C + \frac{C^2}{2T} + \mathbb{E}[Q]. \tag{6}$$

Reducing C lowers both generation time and this conditional residual wait (Figure 4). Completion phases need not be uniform, so $\mathbb{E}[Q]$ is not automatically $\delta/2$. The model equates blocking-job duration with time to first deliverability; stage-overlapped pipelines, adapter-specific output mappings, and frequent updates require their actual timing models (Appendix A).

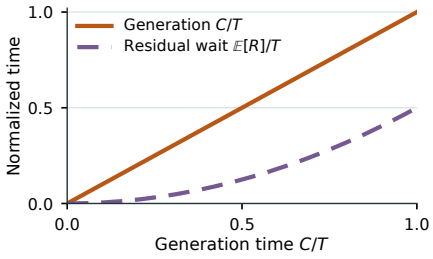

Figure 4: At fixed T, direct generation scales as C/T, while residual wait scales as $\mathbb{E}[R]/T = (C/T)^2/2$ under the stated periodic model.

Cost of revocable predictions. Without condition changes, prediction reuse preserves the matched chunk-wise baseline's complete DiT sampling-call count over steady-state output. Normal refill remains necessary; finer delivery does not cause resampling. Updates introduce extra work by replacing unused predictions. For equal-cost calls producing K usable latent groups each, let w be the fraction ultimately discarded rather than played. Relative DiT sampling work satisfies

$$A \simeq \frac{1}{1 - w}. \tag{7}$$

Infrequent updates can keep this overhead small, but update frequency alone does not determine w: chunk size and the selected branch also matter. VAE decoding, adapter preparation, overlap trimming, aborted jobs, and cache reconstruction require separate cost accounting. Appendix B gives the assumptions and a discrete update model. Together, these results motivate accelerating high-quality joint generation rather than shrinking chunks solely for responsiveness. DiT sampling acceleration and parallel acceleration target the generation path; their benefit is measured through C and total resource cost.

4 Experiments

We first characterize how chunk size affects generation quality and computational efficiency within and across GPUs. We then compare conventional chunk commitment and Commit-One in end-to-end interactive serving.

4.1 Experimental Setup

Models and quality evaluation. For Wan1.3B, we follow CF++ (Zhao et al., 2026) and train on Toy6K, $K \in \{1, 3, 7, 21\}$ models with 1/2/4-step sampling. VBench (Huang et al., 2023) evaluates 81-frame, 16-fps videos at 832×480 and 1280×720 over 946 prompts and five seeds. For LTX-2.3 (HaCohen et al., 2026), we train $K \in \{1, 3, 9, 15, 45\}$ models with three DiT forwards under schedule $[999, 757, 522, 0]$, where 0 is the terminal boundary, on the internal Internal8K training set. Internal32 contains 32 matched cases/seeds of 361 frames at 832×448 and 24 fps. We report its six VBench visual scores separately; each configuration has one training run.

Forward profiling. We profile Wan with USP over 1/2/4/8 H200 GPUs at batch one and BF16, using shared frame-wise/4-step EMA weights and synthetic inputs with a 21-latent prefix plus K current latents. We report the slowest-rank median over 30 synchronized forwards after 10 warm-ups. Timings include forward communication and output gathering but exclude text encoding, VAE decoding, and the sampler loop; these are shape-level profiles rather than generation times C.

End-to-end serving protocol. For Wan1.3B, LTX-2.3, and Sol-H3 (NVIDIA, 2026; Li et al., 2026), we compare model-specific small/large chunks under conventional commitment or Commit-One (S-Base, L-Base, S-CO, and L-CO). Sol-H3 runs four DiT steps on $8\times$B300 with $K = 37/72$ sampled video latents; both policies share continuation, decoding, and 24-fps playback.

Each resolution/arm receives 30 matched 60-second traces at low (1–2 updates) and high (3–11 updates) frequency. Table 3 reports the 480p block. Jittered arrivals are independent of playback and vary actions within a scene. Base applies updates at chunk refills; CO may replace a cached suffix.

Serving metrics. L_{50}/L_{95} measure arrival-to-display response after minimum-RTT clock alignment. We report time-weighted NVML GPU kernel activity and the fraction of the 60-second window spent in playback underflow. Startup and unshown updates are excluded from latency statistics.

4.2 Chunk-Size Analysis Across Quality and Efficiency

Chunk size is a quality–efficiency design choice. Across Wan configurations, $K > 1$ generally improves VBench over $K = 1$ within the same step budget and resolution; for example, the four-step 480p VBench total rises from 79.00 to 81.66 at $K = 3$. The gains are not monotonic, so the best K depends on the setting. Meanwhile, increasing K from 1 to 3 raises forward latency by only $2.05\times$ at 480p and $2.74\times$ at 720p, despite tripling the current-chunk latents. With USP8, 480p $K = 7$ takes 78.6 ms, close to 77.5 ms for $K = 1$ on USP1, while 720p $K = 21$ falls from 6432.8 to 899.7 ms. Suitable parallelism can therefore keep larger chunks efficient, although more GPUs do not always help.

These results suggest that chunk size should not be minimized by default: it is a design parameter that affects both quality and interactive granularity, and the best operating point need not use the smallest chunk. We next test whether a model-specific large chunk can remain responsive under interactive serving.

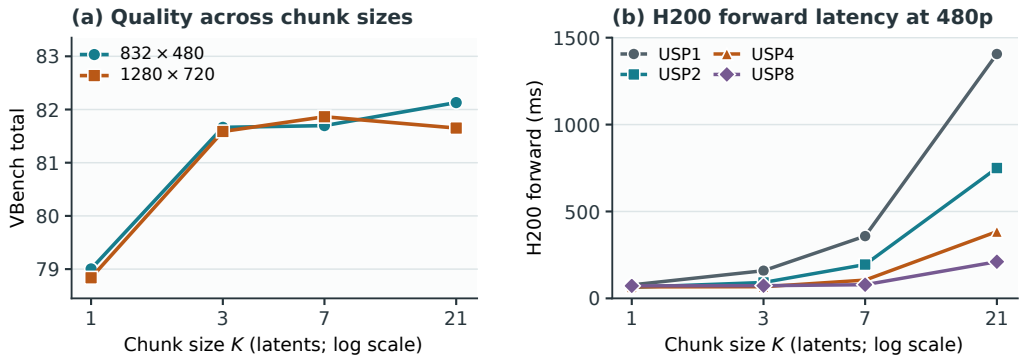

Figure 5: Chunk size exposes a quality–efficiency tradeoff across models. (a) Wan four-step VBench totals at two resolutions. (b) Wan H200 forward latency across USP sizes at 480p. Quality uses separately trained configurations; timings use a shared reference checkpoint and synthetic fixed-length history. Lines connect tested settings, not an inferred quality–latency frontier. LTX quality and cached-forward profiles are reported in Table 2 but are not included in this figure.

Table 1: Wan1.3B quality measurements. Total is the reported VBench aggregate, with its quality and semantic components; dynamic degree, VisionReward, and instruction following are retained from the same verified exports. Bold marks the best value within each resolution–step group.

Resolution	Steps	K	Total	Quality	Semantic	Dynamic	VisionReward	Instruction
480p	**1**	1	79.11	81.80	68.34	4.00	6.50	90.00
	1	3	80.38	83.40	68.29	58.00	4.81	94.00
	1	7	81.35	83.97	**70.86**	48.00	5.79	**100.00**
	1	21	**81.68**	**84.83**	69.08	**85.00**	**9.93**	94.00
	2	1	80.07	82.76	69.34	14.00	9.31	98.00
	2	3	81.62	**84.63**	69.57	57.00	9.82	**100.00**
	2	7	80.84	84.06	67.94	**93.00**	9.29	**100.00**
	2	21	**81.84**	84.62	**70.74**	71.00	**10.17**	**100.00**
	4	1	79.00	81.38	69.50	13.00	7.65	96.00
	4	3	81.66	84.70	69.51	80.00	9.64	**100.00**
	4	7	81.70	84.73	69.56	78.00	**10.19**	98.00
	4	21	**82.13**	**85.03**	**70.54**	**85.00**	9.66	98.00
720p	**1**	1	78.94	81.37	69.19	4.00	5.82	92.00
	1	3	79.99	82.70	69.18	47.00	3.93	98.00
	1	7	**80.78**	83.05	**71.67**	41.00	5.23	96.00
	1	21	80.49	**83.10**	70.04	**67.00**	**9.56**	**100.00**
	2	1	79.86	82.40	69.70	9.00	9.43	98.00
	2	3	**81.52**	**84.11**	**71.18**	48.00	**10.10**	**100.00**
	2	7	80.97	83.79	69.67	**84.00**	9.25	94.00
	2	21	80.26	82.61	70.84	39.00	9.91	98.00
	4	1	78.84	81.19	69.44	22.00	7.18	92.00
	4	3	81.59	84.36	70.49	**74.00**	9.59	**100.00**
	4	7	**81.86**	**84.62**	**70.83**	73.00	**9.77**	96.00
	4	21	81.65	84.43	70.52	66.00	9.36	96.00

4.3 Real Serving-System Evaluation

Table 3 reports completed four-arm comparisons by model. The matched comparison tests whether the S-CO/L-CO gap is smaller than the S-Base/L-Base gap, and whether L-CO approaches S-Base at comparable stalls and GPU activity.

Table 2: H200 single-forward profiles and LTX quality. Wan timings use one shared reference checkpoint across nominal step configurations; LTX timings are cached-forward measurements for each (K, USP). LTX uses three steps. SC/BC denote subject/background consistency; visual scores are multiplied by 100 without a combined total. For quality, bold marks the column best; for timings, bold, single underline, and double underline mark the first, second, and third lowest values within each timing group.

(a) Wan H200 forward (ms)

Resolution	K	USP1	USP2	USP4	USP8
480p	1	77.5	**63.7**	65.9	72.1
	3	159.2	91.5	67.0	72.9
	7	358.3	194.7	105.2	78.6
	21	1406.2	750.0	384.1	211.0
720p	1	218.7	143.4	95.8	**72.7**
	3	599.6	318.4	166.5	119.4
	7	1521.4	809.2	443.8	276.4
	21	6432.8	3335.5	1737.1	899.7

(b) LTX-2.3 / Internal32, 832×448, 3-step

	Internal32 visual scores $\times 100$						H200 forward (ms)			
K	SC	BC	Aesth.	Imaging	Smooth.	Dynamic	USP1	USP2	USP4	USP8
1	87.97	**94.68**	**52.82**	**73.09**	99.62	31.25	274.0	440.5	479.6	526.1
3	88.81	94.31	52.21	65.98	99.53	46.88	**255.6**	423.5	457.0	511.2
9	**89.42**	93.27	50.72	68.96	99.56	34.38	426.7	417.8	456.6	502.0
15	84.96	92.21	51.01	65.98	99.50	**78.12**	660.3	420.4	461.0	509.4
45	88.72	93.68	52.40	68.75	**99.64**	40.62	2016.2	1158.4	634.0	507.9

Table 3: Online serving result layout at 480p for Wan1.3B, LTX-2.3, and Sol-H3. S/L denote model-specific small/large chunk pairs; Base denotes chunk commitment and CO denotes Commit-One. L_{50}/L_{95} are seconds; GPU kernel activity and stall ratio are percentages. Smoke, partial, and invalidated measurements are excluded. Bold marks a unique minimum within each model–frequency group; tied minima are unmarked.

Model	Resolution	Policy	Low-frequency updates				High-frequency updates			
			L_{50}	L_{95}	GPU	Stall	L_{50}	L_{95}	GPU	Stall
Wan1.3B	480p	S-Base	1.01	1.68	30.03	0.00	0.95	1.57	30.00	0.15
		L-Base	3.75	6.34	**10.09**	0.00	3.20	5.70	**10.01**	0.00
		S-CO	**0.82**	**1.12**	38.06	0.00	**0.75**	**1.05**	42.04	0.00
		L-CO	1.28	1.73	11.02	0.00	1.15	1.65	14.50	0.00
LTX-2.3	480p	S-Base	3.00	4.70	21.00	0.00	2.91	4.50	22.03	0.20
		L-Base	8.80	15.50	**4.50**	0.00	8.02	14.55	**4.57**	0.00
		S-CO	1.55	2.10	24.00	0.00	1.54	2.05	27.06	0.00
		L-CO	**1.45**	**2.00**	5.00	0.00	**1.40**	**1.95**	6.52	0.00
Sol-H3	480p	S-Base	4.44	6.26	16.84	0.00	4.35	6.73	16.69	0.13
		L-Base	7.72	12.50	**16.39**	0.00	6.34	11.65	**16.40**	0.00
		S-CO	**1.89**	**2.40**	17.90	0.00	**1.88**	**2.46**	21.66	0.00
		L-CO	2.67	3.49	18.04	0.31	2.65	3.49	24.82	0.00

5 Limitations and discussion

The main limitations are twofold. First, Commit-One benefits from predictions only when they are reused before a new condition invalidates them; frequent or continuous updates can therefore discard much of each sampled chunk and waste substantial DiT computation. Second, with bidirectional within-chunk attention, the retained KV may encode information from a subsequently discarded suffix, so deleting its KV entries may not fully remove its influence on a regenerated continuation.

6 Conclusion

We have formulated interactive video generation as a joint problem of prediction and output commitment. Commit-One preserves multi-latent joint generation, reuses completed predictions under unchanged conditions, and replans only the future that can still be replaced while maintaining a feasible playback prefix. This shifts part of the design effort from shrinking the generation unit toward accelerating joint generation, managing revocable predictions, and scheduling timely delivery.

AI USE STATEMENT

This submission was prepared with an AI coding and writing assistant. The assistance included translating and developing author-provided Chinese research notes into English prose, organizing the manuscript, refining methodological definitions and experimental protocols, checking mathematical arguments, locating primary-source references, and preparing LaTeX and document-validation code, drawing vector schematics, and plotting existing chunk-size measurements. No new generator experiments were run during figure preparation, and no empirical values or experimental video frames were synthesized to fill missing results. The authors reviewed the AI-assisted material, verified the reported claims and citations, and take responsibility for the final submission and for any remaining errors.

REPRODUCIBILITY STATEMENT

Appendices A–B state the assumptions underlying the timing and cost arguments. Appendix C specifies the scheduler interface and proposed invariants, and Appendix D defines the event logging needed to reproduce response measurements. The figure source package includes the 12 measured chunk-size study rows, plotting code, and a provenance manifest linking configuration summaries to their recorded hashes. We will provide the source files, plotting code, configuration summaries, and other materials that can be shared under the applicable licenses and access constraints. The scheduler specification and event schema define the information needed for independent reproduction; results not reported in this submission should not be inferred from them.

ETHICS STATEMENT

Responsive synthetic video can support interactive creation, but it can also enable misleading or impersonating media. Demonstrations should identify generated content and use appropriately licensed inputs and consented likenesses.

REFERENCES

Lichen Bai, Tianhao Zhang, Shitong Shao, Dingwei Tan, Qiyu Zhong, Zhengpeng Xie, Haopeng Li, Qinghao Huang, Dandan Shen, Tengjiao Ji, Wei Wang, Peicheng Wu, Yuxuan Zhao, Xiangyu Zhu, Welly Luo, Shurui Yang, and Zeke Xie. MaineCoon: Pursuing A Real-Time Audio-Visual Social World Model. *arXiv preprint arXiv:2606.17800*, 2026. URL https://arxiv.org/abs/2606.17800.

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. *arXiv preprint arXiv:2407.01392*, 2024. URL https://arxiv.org/abs/2407.01392.

Xinle Cheng, Tianyu He, Jiayi Xu, Junliang Guo, Di He, and Jiang Bian. Playing with Transformer at 30+ FPS via Next-Frame Diffusion. *arXiv preprint arXiv:2506.01380*, 2025. URL https://arxiv.org/abs/2506.01380.

Jiarui Fang and Shangchun Zhao. USP: A Unified Sequence Parallelism Approach for Long Context Generative AI. *arXiv preprint arXiv:2405.07719*, 2024. URL https://arxiv.org/abs/2405.07719.

Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, Eitan Richardson, Guy Shiran, Itay Chachy, Jonathan Chetboun, Michael Finkelson, Michael Kupchick, Nir Zabari, Nitzan Guetta, Noa Kotler, Ofir Bibi, Ori Gordon, Poriya Panet, Roi Benita, Shahar Armon, Victor Kulikov, Yaron Inger, Yonatan Shiftan, Zeev Melumian, and Zeev Farbman. LTX-2: Efficient Joint Audio-Visual Foundation Model. *arXiv preprint arXiv:2601.03233*, 2026. URL https://arxiv.org/abs/2601.03233.

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. URL https://arxiv.org/abs/2506.08009.

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. *arXiv preprint arXiv:2311.17982*, 2023. URL https://arxiv.org/abs/2311.17982.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal Flow Matching for Efficient Video Generative Modeling. *arXiv preprint arXiv:2410.05954*, 2024. URL https://arxiv.org/abs/2410.05954.

Jihwan Kim, Junoh Kang, Jinyoung Choi, and Bohyung Han. FIFO-Diffusion: Generating Infinite Videos from Text without Training. *arXiv preprint arXiv:2405.11473*, 2024. URL https://arxiv.org/abs/2405.11473.

Akio Kodaira, Tingbo Hou, Ji Hou, Markos Georgopoulos, Felix Juefei-Xu, Masayoshi Tomizuka, and Yue Zhao. StreamDiT: Real-Time Streaming Text-to-Video Generation. *arXiv preprint arXiv:2507.03745*, 2025. URL https://arxiv.org/abs/2507.03745.

Yitong Li, Junsong Chen, Haopeng Li, Haozhe Liu, Jincheng Yu, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Sol Video Inference Engine: Agent-Native Full-Stack Acceleration Framework for Efficient Video Generation. *arXiv preprint arXiv:2606.23743*, 2026. URL https://arxiv.org/abs/2606.23743.

Hansen Jin Lillemark, Alex Rojas, Zachary Novack, Runqian Wang, Yilun Du, Yian Ma, Taylor Berg-Kirkpatrick, and Rose Yu. Equilibrium Forcing: Adaptive Video Generation Without Noise Conditioning. *arXiv preprint arXiv:2608.14706*, 2026. URL https://arxiv.org/abs/2608.14706.

Shanchuan Lin, Xin Xia, Yuxi Ren, Ceyuan Yang, Xuefeng Xiao, and Lu Jiang. Diffusion Adversarial Post-Training for One-Step Video Generation. *arXiv preprint arXiv:2501.08316*, 2025. URL https://arxiv.org/abs/2501.08316.

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. URL https://arxiv.org/abs/2411.19108.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling Forcing: Autoregressive Long Video Diffusion in Real Time. *arXiv preprint arXiv:2509.25161*, 2025. URL https://arxiv.org/abs/2509.25161.

Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. FasterCache: Training-Free Video Diffusion Model Acceleration with High Quality. *arXiv preprint arXiv:2410.19355*, 2024. URL https://arxiv.org/abs/2410.19355.

NVIDIA. Sol-H3: Speed-of-Light MiniMax-H3 on an 8x NVIDIA B300 Blackwell System. Project page, 2026. URL https://nvlabs.github.io/Sana/Sol-Engine/Sol-H3/.

Shaohao Rui, Xiaofeng Mao, Zhanyu Zhang, Peijia Lin, Yansong Zhu, Yibo Zhang, Haibin Wan, Zhangrui Zhao, and Weijie Ma. BiWM: Advancing Open-Source Interactive Video World Models with Bidirectional Autoregression. *arXiv preprint arXiv:2606.10135*, 2026. URL https://arxiv.org/abs/2606.10135.

Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable Length Video Generation From Open Domain Textual Description. *arXiv preprint arXiv:2210.02399*, 2022. URL https://arxiv.org/abs/2210.02399.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse VideoGen: Accelerating Video Diffusion Transformers with Spatial-Temporal Sparsity. *arXiv preprint arXiv:2502.01776*, 2025. URL https://arxiv.org/abs/2502.01776.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video Generation using VQ-VAE and Transformers. *arXiv preprint arXiv:2104.10157*, 2021. URL https://arxiv.org/abs/2104.10157.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, Song Han, and Yukang Chen. LongLive: Real-time Interactive Long Video Generation. *arXiv preprint arXiv:2509.22622*, 2025. URL https://arxiv.org/abs/2509.22622.

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. *arXiv preprint arXiv:2412.07772*, 2024. URL https://arxiv.org/abs/2412.07772.

Lvmin Zhang, Shengqu Cai, Muyang Li, Gordon Wetzstein, and Maneesh Agrawala. Frame Context Packing and Drift Prevention in Next-Frame-Prediction Video Diffusion Models. *arXiv preprint arXiv:2504.12626*, 2025a. URL https://arxiv.org/abs/2504.12626.

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast Video Generation with Sliding Tile Attention. *arXiv preprint arXiv:2502.04507*, 2025b. URL https://arxiv.org/abs/2502.04507.

Min Zhao, Hongzhou Zhu, Kaiwen Zheng, Zihan Zhou, Bokai Yan, Xinyuan Li, Xiao Yang, Chongxuan Li, and Jun Zhu. Causal Forcing++: Scalable Few-Step Autoregressive Diffusion Distillation for Real-Time Interactive Video Generation. *arXiv preprint arXiv:2605.15141*, 2026. URL https://arxiv.org/abs/2605.15141.

A TIMING ASSUMPTIONS AND DERIVATIONS

A.1 FIXED CHUNK BOUNDARIES

The reference model has playback boundaries at integer multiples of T, fixed generation time C, and a condition available at time t_{arrival}. The new condition is unknown before t_{arrival}, and a generation job fixes its condition at launch. We exclude in-flight condition injection or partial DiT sampling reuse that could shorten the new condition's C-second generation. We consider an isolated update that is not subsequently superseded, with no added transport or client-buffer delay. For attainability of a steady single-worker schedule, take $C \leq T$ and assume the required prefix is available when that schedule starts a job.

Proposition 1. *For uniform condition-arrival phase modulo T, the fixed-boundary model has $\mathbb{E}[L] \geq C + T/2$. This lower bound is attainable under the stated ideal scheduling assumptions.*

Proof. The new result cannot finish earlier than $t_{\text{arrival}}+C$. Its earliest allowed playback boundary is therefore $t^{\min}_{\text{playback}} = T\lceil (t_{\text{arrival}}+C)/T \rceil$. The phase $U = (t_{\text{arrival}}+C) \bmod T$ is uniform on $[0, T)$ because C is fixed. Ignoring the zero-measure case of exact alignment, $t^{\min}_{\text{playback}} - (t_{\text{arrival}} + C) = T - U$, whose expectation is $T/2$. Thus $\mathbb{E}[t_{\text{playback}} - t_{\text{arrival}}] \geq C + T/2$. For attainability, start generation at $nT - C$ and finish at nT. An update is used by the next generation start after its arrival. Those starts are separated by T, so their mean forward waiting time under a uniform arrival phase is $T/2$. Generation takes C and its completion is aligned with playback. □

For any realized start t_{start} and display t_{playback}, the two waiting terms obey

$$\begin{aligned}(t_{\text{start}} - t_{\text{arrival}}) + [t_{\text{playback}} - (t_{\text{start}} + C)] \\ = t_{\text{playback}} - t_{\text{arrival}} - C.\end{aligned} \tag{8}$$

Changing t_{start} alone redistributes waiting without changing the total for a fixed t_{playback}. The $T/2$ in the proposition already accounts for the combined waiting in the ideal baseline; adding a second copy of residual generation waiting would double-count it. Additional dependencies can make an actual baseline slower, but they must be modeled explicitly.

A.2 LATENT-BOUNDARY ALIGNMENT

If the prefix is immediately available, there are no extra buffer constraints, and generation takes fixed time C, the earliest latent-aligned display is $t_{\text{playback}} = \delta\lceil (t_{\text{arrival}} + C)/\delta \rceil$. Uniform phase modulo δ gives $\mathbb{E}[L] = C + \delta/2$ in this ideal case. With nonzero R, however, the relevant phase is $(t_{\text{arrival}} + R + C) \bmod \delta$. Dependence between t_{arrival} and R can concentrate completion times at a few phases. Consequently, uniform arrivals do not by themselves imply a uniform completion phase.

A.3 RESIDUAL GENERATION UNDER AN ISOLATED UPDATE

Assume the no-update system runs one indivisible job of length C in each period T, with $C \leq T$. A uniformly arriving isolated update lands inside generation with probability C/T. Suppose that job must complete to supply the prefix required for the new branch. Conditional on arriving during it, the residual duration is uniform on $[0, C]$. When generation is idle, the prefix and resources needed for the new branch are assumed ready. Then

$$\mathbb{E}[R] = \Pr(\text{busy})\mathbb{E}[R \mid \text{busy}] = \frac{C}{T}\frac{C}{2} = \frac{C^2}{2T}. \tag{9}$$

Adding the new generation time and actual alignment gives $\mathbb{E}[L] = C + C^2/(2T) + \mathbb{E}[Q]$. For the illustrative values $T = 5\,\text{s}$ and $C = 1.6\,\text{s}$, the residual mean is $0.256\,\text{s}$. Excluding Q, an idle arrival responds in $1.6\,\text{s}$, a busy arrival averages $2.4\,\text{s}$, and the unconditional average is $1.856\,\text{s}$. These numbers are arithmetic examples, not video-system measurements.

The derivation intentionally uses the same C for the blocking job and the new branch's time to deliverability. A real pipeline may separate DiT sampling, KV completion, VAE decoding, and

transmission, or overlap some of these stages. In that case each critical path must be recorded and the blocking model rederived. Repeated updates can also change the busy fraction from C/T, so this expression is not a high-frequency law.

An instructive limiting case is $C = T = K\delta$. The generation worker is always busy in the reference schedule, so $\mathbb{E}[R] = T/2$, and its new completion is latent-aligned with $Q = 0$. The resulting mean is $3T/2$, equal to the ideal fixed-chunk baseline. Finer delivery alone therefore need not improve the mean when generation consumes the entire period; the available compute headroom and prefix dependence matter.

B Prediction-use accounting

B.1 DiT sampling-call accounting

Let N equal-cost complete DiT sampling calls each produce K latents, and let U of those latents ultimately be played. If $w = 1 - U/(NK)$, the number of DiT sampling calls per played latent, relative to full-use chunk generation, is

$$A = \frac{N/U}{1/K} = \frac{NK}{U} = \frac{1}{1-w}. \tag{10}$$

This equality is exact for the counted DiT sampling work under the stated equal-cost, complete-call accounting. Interpreting A as an end-to-end cost ratio is only approximate and requires separate accounting of VAE decoding, conditioning preparation, cache rebuilding, aborted jobs, and other overheads. Finite-run initialization and unused final buffers must be treated consistently between methods.

For an overlap-trimmed adapter, K remains the number of sampled latents, not the number of newly playable groups. Use the matched adapter's full-use output yield as the baseline, and report mandatory overlap work separately from predictions discarded because of updates. Nonuniform group durations additionally require accounting in played frames or seconds. The formula above cannot be applied by counting re-generated conditioning frames as newly played output.

B.2 A separate independent-update model

For intuition, consider a discrete model in which every δ-period decision independently triggers an update with probability p. Each DiT sampling call provides K completed predictions, its head is always used, and generation is assumed ready at the necessary decision without cross-period prefix blocking. Let M be the number used before the next update or exhaustion. For $m = 1, \dots, K$, $\Pr(M \geq m) = (1-p)^{m-1}$. The tail-sum identity yields

$$\mathbb{E}[M] = \sum_{m=1}^{K} (1-p)^{m-1} = \frac{1-(1-p)^K}{p}, \qquad A = \frac{K}{\mathbb{E}[M]} = \frac{Kp}{1-(1-p)^K}. \tag{11}$$

The continuous extension at $p = 0$ is $A = 1$, and $p = 1$ gives $A = K$. If Poisson arrivals of rate λ are coalesced within each interval, $p = 1 - \exp(-\lambda\delta)$. For $\lambda T \ll 1$,

$$A - 1 \simeq \frac{\lambda\delta(K-1)}{2} = \frac{\lambda(T-\delta)}{2}. \tag{12}$$

This model is not the wall-clock scheduler of Section 3. With finite generation time, a locked prefix can contain several old predictions, updates during a job can be coalesced, and even a new call's head need not have the same usage distribution as in the independent model. The measured M distribution must therefore replace the geometric model for actual high-frequency workloads. If a DiT sampling call occupies its worker for C_{DiT}, $\rho = AC_{\mathrm{DiT}}/T < 1$ provides average DiT worker capacity under the same accounting assumptions; it does not guarantee that all deadlines are met.

C Extended finite-state controller specification

This specification formalizes the two-state control logic of Section 3.2; it is not a claim that every existing experimental pipeline implements these exact semantics. The finite control set is $\mathcal{S} =$

$\{\mathsf{Produce}, \mathsf{Consume}\}$, with invariant

$$z = \mathsf{Produce} \iff J \neq \varnothing. \tag{13}$$

The complete system is an extended FSM, not a two-state representation of all possible tensors, timestamps, or condition histories.

Registers and events. The register tuple is $x = (B, H, V, v, u, J, p, \ell_i, \ell_l, n, \beta)$. B stores versioned predictions and their latent/pixel readiness; H stores conditioning inputs and model-specific state, including prefix KV when used; V stores VAE state and the source-to-playback mapping. v is the latest accepted condition version and u a pending version or $\varnothing$. An active J records a unique job identifier, fixed condition version, branch position, and readiness/deadline estimates. p is the playback cursor; ℓ_i and ℓ_l are exclusive ends of the irrevocable and compute-locked prefixes; n is the end of valid predictions. The playback plan β assigns deadline b_j to position j. A monotone event sequence records the processing order, including events sharing a timestamp.

The normal-operation event alphabet is

$$\mathcal{E} = \{\mathsf{update}, \mathsf{ready}, \mathsf{finish}, \mathsf{tick}, \mathsf{wake}\}. \tag{14}$$

Events carry condition versions, job identifiers, positions, and timestamps as appropriate. Only valid callbacks from the corresponding job may publish results; stale callbacks cannot reinstall an invalidated suffix.

Feasible boundaries and dispatch guard. Let $\ell = \max(\ell_i, \ell_l)$. For a candidate branch j, define

$$\begin{aligned} F_j(x,t) = [j \geq \ell] \ &\wedge \mathrm{ContextReady}(x,j) \wedge \mathrm{Cover}(x,t,b_j) \\ &\wedge \mathrm{ValidVAE}(x,j) \wedge \bigwedge_{m=0}^{M_j-1} [t + \widehat{C}_{j,m} + \epsilon \leq b_{j+m}]. \end{aligned} \tag{15}$$

$\mathrm{ContextReady}$ requires the inputs needed by the selected adapter Φ_θ. For retained-prefix KV continuation, these include the prefix latents and their KV; playable pixels alone are insufficient. For boundary-image continuation, the retained boundary RGB frame suffices as the native image encoder's input; full-prefix KV is not required. Cover requires that the old prefix can meet all remaining playback deadlines before b_j; $\mathrm{ValidVAE}$ requires a continuation that preserves already committed pixels. M_j is the number of newly playable groups after the adapter's overlap trimming and must be positive for dispatch; $M_j = K$ for the untrimmed uniform-latent model. $\widehat{C}_{j,m}$ estimates time from launch until new group m becomes deliverable, and ϵ is a configured safety margin. Deadlines b_{j+m} come from the actual output mapping, including nonuniform native group durations where needed. These estimates do not turn the guard into a runtime guarantee. The feasible set $\mathcal{F}(x,t)$ contains positions with $F_j(x,t)$ true. For a pending update, it includes an append boundary when that boundary is feasible.

Let $h(x,t)$ be remaining continuous playable coverage and $\eta(x,t) > 0$ the configured refill lead, including alignment and margin. Define $\mathrm{Refill}(x,t) = [h(x,t) \leq \eta(x,t)]$ and

$$\begin{aligned} g(x,t) = [J = \varnothing] \ \wedge \big(&[u \neq \varnothing \wedge \mathcal{F}(x,t) \neq \varnothing] \\ &\vee [u = \varnothing \wedge \mathrm{Refill}(x,t) \wedge F_n(x,t)] \big). \end{aligned} \tag{16}$$

With a pending update, choose the earliest feasible j by playback deadline; otherwise append at n. Hence regular refill and update-triggered replacement are distinct reasons to launch, neither tied to every delivery. A nonempty pending version alone does not authorize a launch.

Event actions and transition semantics. For each event, apply the corresponding register action U_e below, then evaluate g and Equation (3) once. This order ensures that a completion's outputs are validated before a dependent branch is launched.

1. update sets v and u to the newly accepted version, recording any replaced pending version as superseded. It leaves J and the protected prefix unchanged. An active job therefore yields a $\mathsf{Produce}$ self-loop.

2. ready validates and publishes the indicated latent, KV, or pixel readiness in B, H, V without releasing J. Pixel delivery may begin at its scheduled boundary during Produce; no unready pixel group is marked deliverable.
3. finish validates remaining results and releases the active scheduling slot by clearing J. The resulting transition is Produce $\rightarrow$ Produce when g authorizes another job, and Produce $\rightarrow$ Consume otherwise. Pending versions survive completion until actually dispatched.
4. tick delivers the next valid, ready group and advances p in either control state. Missing output records an underrun rather than a successful display. Delivery updates commitment positions when downstream transfer is irrevocable. It invokes no DiT sampling itself; a coincident refill or update demand can authorize a separate launch.
5. wake refreshes time-dependent guards, including refill lead and retry conditions. From Consume, a true guard gives Consume $\rightarrow$ Produce; a false guard leaves Consume $\rightarrow$ Consume with pending data retained.

Launch atomically locks the retained prefix before j, invalidates only the replaceable suffix and dependent speculative state, and records a new J for a complete K-latent DiT sampling call with the latest condition. It clears only the pending version it consumes. VAE decoding supplies the required pixels for each group's deadline. In-flight conditioning is fixed; later arrivals populate u, not J. Initialization uses a separate bootstrap launch before the playback clock starts, establishing initial coverage and VAE state. Startup latency is recorded separately from steady-state response.

Required invariants. No delivered or locked prefix is overwritten. A new branch never directly attends to invalid suffix entries. At most one job occupies the active slot, and only a guarded launch starts new DiT sampling. Playback and cache positions remain consistent after a branch. No predicted latent is counted as displayed before a display event. VAE decoder invalidation cannot alter already delivered pixel frames. Missed deadlines, superseded updates, and additional work are observable in logs, not silently removed.

Runtime estimates and failure handling. The ideal formulas use known fixed C; an implementation does not. The chosen estimator, safety margin, and retry policy must be specified in the experimental configuration. Any fallback that freezes playback, drops frames, uses stale content, or alters the target boundary must be logged and reflected in quality and response measurements. Defining a deadline policy without a runtime distribution is insufficient to guarantee real-time behavior.

Job failure publishes no invalid outputs and releases the scheduling slot only when the worker confirms termination. Coverage loss is reported and handled by the declared fallback; it is not hidden by a normal-state label. Both states may receive playback ticks, while their effect on the control state is determined by remaining work and dispatch demand.

D EVENT AND MEASUREMENT SCHEMA

Generation is the computation needed to produce deliverable video; its general definition does not require a VAE. For the latent diffusion models considered here, C includes conditioning preparation, DiT sampling, and required VAE decoding on the critical path to the first output group, not necessarily the entire chunk. Overlapping stages are not simply added. R may include scheduling, resource, and conditioning-input readiness waits; Q may include transport and buffering as well as playback alignment. User-end-to-end latency additionally includes processing before t_{arrival}. Semantic response, when the condition's effect is visibly recognizable, is a separate measure. The relation $T = Kr/f$ assumes a uniform latent-to-frame mapping; models with nonuniform mappings or overlap trimming use their actual output frame counts. The temporal frame rate f is not computational throughput in generated frames per wall-clock second.

Each event log should include a run identifier, event identifier, condition version, condition-ready time, prefix-ready time, generation start, KV-ready time, DiT sampling completion, VAE decoding completion, first deliverable output time, target switch boundary, actual display time, and final outcome. Generation logs additionally record call identifier, absolute positions, used and discarded predictions, runtime, GPU count, and any canceled or repeated work. Model, DiT sampling, resolution, frame-rate, and buffer configurations belong in run metadata. Record the conditioning adapter,

retained-context identity, source-to-playback offsets, and trimmed overlap, so native image continuation is not conflated with retained-prefix KV reuse.

The response distribution is defined over events whose relevant new output is actually displayed. Alongside it, report the fractions of all events that are displayed, superseded, failed, and censored. A superseded event is not a zero-latency success. End-of-run censoring is distinct from a condition that fails to induce a semantic response despite displayed conditioned output. Annotation rules should specify a semantic observation window and how ambiguous responses are handled.

Use identical external event streams for paired policy comparisons. Report the randomization of wall-clock arrival phases and do not restrict events to convenient generation boundaries. Repeated-run uncertainty should be estimated at the run or paired-run level when events within a rollout are dependent. Record the warm-up and final-buffer accounting policy.

E CONDITIONING AND OUTPUT ADAPTERS

The controller operates on retained content and playback deadlines. Its conditioning state $\mathcal{H}_j$ need not have the same representation across generators. This section distinguishes the adapters used to expose continuation, rather than treating all streaming paths as full-prefix KV reuse.

Retained-prefix KV. For a generator with history-conditioned DiT sampling, the adapter exposes the selected prefix's inference state at the correct positions. Direct suffix entries are invalidated while the retained KV is reused. This preserves the original inference representations; it does not reconstruct them from clean latents. An implementation that instead builds KV by clean prefill of each retained latent is a different cache policy. An LTX proposal-reuse implementation may instead use incremental clean prefill; that should be identified separately from direct retained-KV reuse. Neither policy can erase old-future information already present in fixed prefix latent values. Separating the additional effect of KV reuse requires a matched controlled ablation.

Boundary-image continuation. The bidirectional image-conditioned path retrieves the full-resolution RGB frame immediately preceding the selected branch from the retained timeline, then submits it through the checkpoint's native image conditioning interface with the latest available prompt. The H3 image-conditioned path can implement this operation without a new temporal causal mask or wrapper-specific training. The evaluated Sol-H3 configuration is already accelerated; its use is not evidence that an undistilled teacher has achieved the same latency. Only the boundary image is passed as visual history, so this path neither preserves full-prefix KV nor claims equivalent long-range conditioning.

Latent and multi-frame variants. The H3 experimental latent-input path replaces image encoding with a normalized clean video latent while retaining condition packing and condition-noise handling. It is not a hard copy into the target noise tensor, and a temporal video latent is not assumed distribution-equivalent to a single encoded image. A separate multi-frame path supplies a phase-aligned group of retained latents as keyframe conditions and removes the corresponding output overlap. These variants change context, eligible boundaries, and preparation costs; their results should not be silently attributed to the boundary-image adapter.

Output mapping and VAE state. A native clip may include a regenerated anchor or context span. The output adapter omits that span and maps only the remaining pixels to the new playback interval. It retains source indices for subsequent anchors, and never selects history from a discarded or trimmed-away prediction. This mapping can have nonuniform latent-to-frame spans and fewer playable groups than sampled latents; the controller uses their actual deadlines.

Streaming VAE decoding can publish finalized groups before the entire clip has finished decoding, while DiT sampling remains joint over the chunk. This does not imply that a prefix of bidirectional DiT sampling is available early. VAE overlap and any state dependent on a replaced suffix must be handled before pixels are committed; the adapter may instead decode the full clip and deliver its completed frames incrementally. Group-ready and job-finish events remain separate in both cases. History encoding, cache construction, VAE decoding, and mandatory overlap work are included in the corresponding timing and cost measurements.

F FUTURE WORK: CO-OPTIMIZING DIT SAMPLING AND VAE DECODING

Accelerating joint DiT sampling is one route to retaining large chunks, but the serving objective is to reduce generation time C to the first deliverable output group. A natural next step is to co-design DiT sampling and VAE decoding as a single generation pipeline rather than optimize their isolated runtimes. This includes when completed latent groups are handed to the VAE decoder, how decoding is scheduled against playback deadlines, and how compute and memory are allocated between stages.

Any overlap must respect actual data dependencies: temporally bidirectional DiT sampling does not imply that an arbitrary latent prefix is ready for VAE decoding before sampling completes. Candidate schedules should measure time to the first deliverable group, later-group deadlines, total GPU cost, and playback stalls. Parallel acceleration is a broader design opportunity, not a commitment to one parallelization strategy. Its benefit must be evaluated within the complete pipeline and resource budget, rather than assumed from parallel execution alone. These are directions for future optimization, not additional measured speedups claimed by the current analysis.