AsymHP: Load-Balanced Sparse Attention for Video Diffusion Transformers

Xinwei Qiang[1]* **Yue Guan**[1]* **Ruihan Zhu**[1] **Mihir Jagtap**[1] **Zaifeng Pan**[1]
Zhongkai Yu[1] **Chang Chen**[1] **Zhengding Hu**[1] **Yufei Ding**[1] **Adnan Aziz**[2]
[1]University of California, San Diego [2]Meta
{x2qiang,yueguan,mjagtap,zapan,zhy055,chc278,zhh068,yufeiding}@ucsd.edu
ruihanzhu.tiger@gmail.com adnanaziz@meta.com

Abstract

Video diffusion transformers rely on self-attention over long spatio-temporal token sequences, making inference expensive even when sparse attention reduces the number of computed blocks. This paper studies a systems bottleneck that appears in dynamic sparse video attention: per-head sparsity can vary widely across denoising steps, layers, and prompts, so equal-head parallel execution leaves some GPUs waiting for dense heads assigned to other GPUs. We propose AsymHP, a runtime system that uses the previous denoising step's per-head sparse density to place a non-uniform number of heads on each GPU. AsymHP combines this lightweight online density estimate, a cost model for mask construction and sparse attention computation, and an asymmetric head redistribution primitive that avoids padding variable-size head shards into symmetric collectives. In our evaluation on H100 GPUs, AsymHP improves sparse attention latency by up to $1.54\times$ without changing the underlying sparse-attention algorithm.

1 Introduction

Diffusion Transformers (DiTs) [25] are now widely used for high-quality video generation [28, 16, 27, 22, 12, 23, 3, 42, 33, 26, 43, 41, 13, 10], but their inference cost remains a major obstacle to practical deployment. A key bottleneck is self-attention: video models operate over long spatio-temporal token sequences, making attention expensive at high resolution and long duration. Sparse attention methods have therefore become an attractive way to accelerate video DiTs by computing only a subset of important tokens or blocks. Recent systems such as Sparse VideoGen [30], Sparse VideoGen2 [32], and other sparse attention variants [20, 35, 6, 27, 36, 37, 19, 38, 4, 40, 29, 39] show that sparse attention can reduce latency while preserving visual quality through spatio-temporal sparse patterns, semantic-aware token permutation, threshold-based budget control, and specialized kernels.

However, dynamic sparse attention introduces a new systems problem: sparsity is highly imbalanced across attention heads. Many recent sparse mechanisms select blocks using top-p or other threshold-based rules rather than a fixed mask or fixed top-k budget. Since video DiT heads often specialize in different spatio-temporal patterns [30], thresholding can retain vastly different numbers of blocks across heads in the same layer. Denser heads therefore execute more sparse-attention blocks, creating substantial per-head workload imbalance. Figure 1 shows this effect: per-head sparse density varies widely, and the default contiguous head placement can assign much more sparse-attention work to one GPU than to others. In our measurements, per-head density differs by up to $9.1\times$ within a layer; after contiguous 4-GPU head placement, the summed per-GPU sparse density differs by up to $5.3\times$.

This imbalance exposes a limitation of existing attention parallelization strategies. Conventional head parallelism [14] assigns each GPU an equal, contiguous range of head IDs, implicitly assuming

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

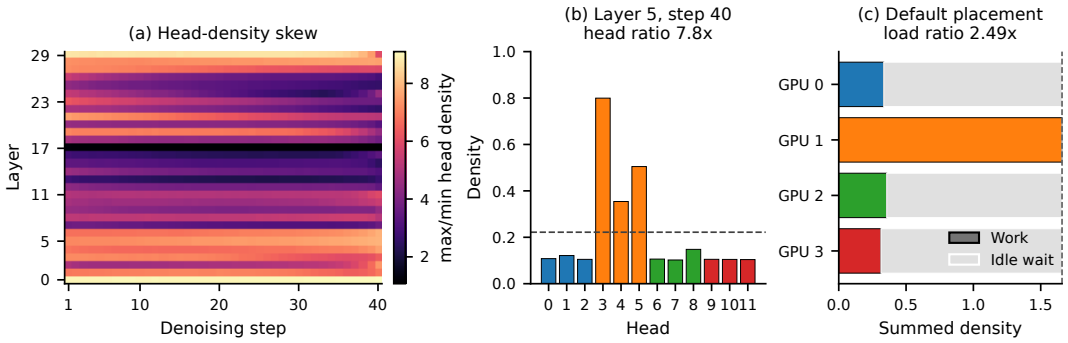


Figure 1: Head-level sparse density is highly imbalanced in dynamic sparse video attention. Equal numbers of heads do not imply equal execution time under the default contiguous head placement.

comparable per-head costs; under dynamic sparsity, this static partitioning may place high-density heads on one GPU and low-density heads on another. Ring attention [21] avoids assigning complete heads to individual GPUs, but many runtime sparse mechanisms cannot be directly mapped to its regular streaming schedule: their sparse masks are generated independently per head from data-dependent query/key statistics before sparse attention is executed. Thus, runtime sparse attention requires a strategy that balances unequal per-head workloads while preserving the per-head mask-generation semantics of existing sparse methods.

The main difficulty is that a head's workload is only known after its sparse mask has been generated, but placement must be decided before execution. Using the current mask for placement is too late, while static profiling ignores request-, layer-, and denoising-step-dependent sparsity variation. We observe that sparse density in video diffusion is dynamic but not arbitrary: adjacent denoising steps exhibit strong temporal locality. Heads that are dense in one step often remain relatively dense in the next, while sparse heads tend to remain light. This allows us to predict each head's next-step workload from its recent density history and use the prediction for placement before the current mask is built.

We present AsymHP, a runtime system for load-balanced sparse attention in video DiTs. AsymHP preserves the standard sequence-parallel interface before and after each sparse attention layer, but internally executes heads with an asymmetrically redistributed head-parallel schedule. At each layer and denoising step, AsymHP estimates per-head density from the previous denoising step, converts the estimate into a predicted runtime cost, and assigns a non-uniform number of heads to GPUs to balance predicted execution time. Since non-uniform head placement creates variable-size communication, AsymHP also introduces an asymmetric all-to-all primitive that redistributes only the required head shards, avoiding the padding and synchronization overheads of symmetric collectives.

Across real video generation workloads, AsymHP achieves up to $1.54\times$ speedup over equal-head sparse attention. An end-to-end generation experiment demonstrates a $1.13\times$ speedup. In summary, this paper makes the following contributions:

- We show that threshold-based sparse attention creates severe head-level workload imbalance in video DiTs, causing substantial GPU idle time under default head placement.
- We propose online asymmetric head placement, using previous-step per-head density to predict and balance sparse attention workloads.
- We build AsymHP with asymmetric all-to-all communication, achieving up to $1.54\times$ speedup while preserving the same sparse masks and local sparse-attention kernels.

2 Background and Motivation

2.1 Dynamic Sparse Attention in Video DiTs

Video DiTs process a video as a sequence of spatio-temporal latent tokens and repeat the same transformer stack over many denoising steps. Self-attention is therefore expensive at high resolution and long duration, motivating sparse attention methods that compute only selected query-key blocks.

For layer ℓ, denoising step t, and head h, let $M_{\ell,t,h}$ be the retained sparse blocks and define the per-head density as $\rho_{\ell,t,h} = |M_{\ell,t,h}|/|M_{\text{full}}|$. Unlike fixed sparse patterns, recent video sparse attention methods construct $M_{\ell,t,h}$ at runtime using score estimates, token similarity, semantic grouping, permutation, or threshold-based importance rules [32, 35, 38, 29]. Thus, the retained block count can vary across heads, layers, prompts, and denoising steps. Since denser heads execute more sparse-attention blocks, this density variation creates per-head execution-time imbalance.

2.2 Default Parallel Execution

Video DiT inference is commonly parallelized around transformer layers using sequence parallelism. Before attention, each device holds a shard of the token sequence with all attention heads for that shard. To execute multi-head attention, systems such as DeepSpeed-Ulysses [14] use an all-to-all collective to redistribute the tensor layout: the sequence dimension is gathered across devices while the head dimension is partitioned. After this exchange, each device owns a subset of attention heads over the full sequence and computes those heads locally. A second all-to-all restores the sequence-parallel layout before the following layer operations.

The default head placement after this layout transformation is head-count balanced. Heads are ordered by head ID and divided into equal-sized contiguous partitions across devices. With H heads and G devices, device g receives a consecutive range of approximately H/G heads. This policy is simple, deterministic, and independent of the input or the sparse masks selected at runtime.

Ring attention follows a different sequence-parallel execution style. Instead of converting the layout into head-parallel partitions, devices stream key-value blocks through a ring and accumulate partial attention results according to a regular communication schedule. This works naturally when attention over sequence blocks follows a predictable streaming pattern. However, many dynamic sparse mechanisms generate masks independently per head from input-dependent scores, producing irregular sparse patterns that are not known as a fixed ring schedule over sequence blocks. Mapping such methods to ring attention can therefore require additional mask communication, redundant mask construction, or substantial redesign of the sparse mechanism.

2.3 Head-Level Sparsity Imbalance

Attention heads in video DiTs tend to specialize. Some heads focus on local spatial neighborhoods, while others capture temporal dependencies across frames. Dynamic sparse mask selection exposes this specialization as workload skew. Under a threshold or top-p rule, each head retains the blocks needed to satisfy its own score distribution. A head with a diffuse importance distribution may retain many blocks, while a head with concentrated high-confidence scores may retain far fewer. Thus, each head has an input-dependent workload that varies across layers and denoising steps.

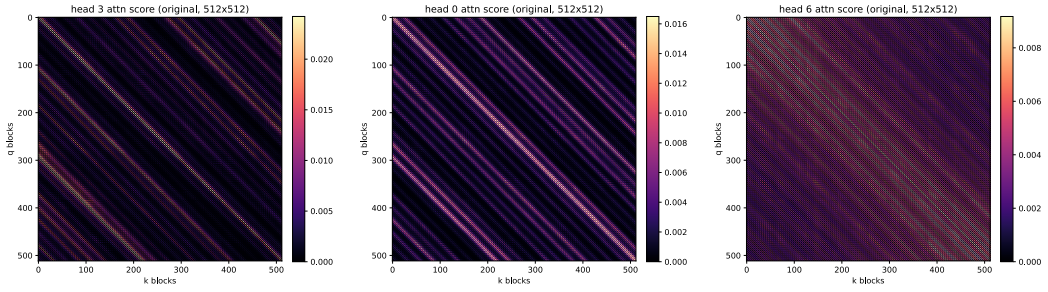


Figure 2: Original block-level attention scores differ substantially across attention heads. These head-specific score patterns cause threshold-based sparse attention to retain different numbers of query-key blocks per head.

Figure 2 shows examples of original block-level attention scores from three heads. The score patterns are visibly different, which means a common thresholding rule can produce different retained-block counts across heads.

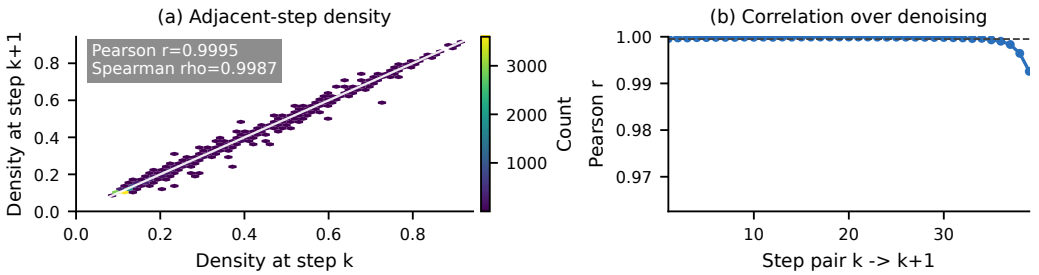


Figure 3: Per-head sparse density is strongly correlated across adjacent denoising steps. Each point compares the same layer and head between consecutive steps, supporting the use of the previous step's realized density as an online predictor for current-step placement.

Figure 1 plots the per-head density $\rho_{\ell,t,h}$ across layers and denoising steps and shows the resulting load skew under the default contiguous head placement. In our measurements, the densest and sparsest heads within the same layer differ by up to $9.1\times$. The default head placement therefore gives each GPU the same number of heads but not the same amount of sparse-attention work. A GPU assigned several dense heads can dominate the layer latency, while other GPUs finish early and wait at the next synchronization point.

This effect matters because distributed attention latency is gated by the slowest GPU. Let $\mathcal{H}_g$ denote the set of heads assigned to GPU g. Under dynamic sparsity, the load on GPU g is modeled as

$$L_{\ell,t,g} = \sum_{h \in \mathcal{H}_g} \left(C^{\text{mask}}_{\ell,t,h} + C^{\text{attn}}_{\ell,t,h} \right), \tag{1}$$

rather than by the head count $|\mathcal{H}_g|$. Since $C^{\text{attn}}_{\ell,t,h}$ grows with the retained sparse blocks $|M_{\ell,t,h}|$, clustering dense heads on one GPU creates a straggler even when every GPU receives the same number of heads.

2.4 Opportunity: Temporal Locality Across Denoising Steps

A workload-aware placement must be chosen before sparse attention executes. However, the exact current-step workload of a head is only known after its sparse mask has been generated. Recomputing placement after mask generation would put additional synchronization and data movement on the critical path, while a static placement cannot capture prompt-, layer-, and step-dependent sparsity variation.

The key opportunity is that sparse density in video diffusion is dynamic but temporally correlated. Adjacent denoising steps often preserve the relative importance of heads: heads that are dense in one step tend to remain dense in the next, while sparse heads tend to remain light. Figure 3 shows this temporal locality by comparing per-head densities between consecutive denoising steps. This correlation allows the previous step's density $\rho_{\ell,t-1,h}$ to serve as an online predictor for the current step's workload.

This observation turns sparse-attention load balancing into an online placement problem. Instead of waiting for the current sparse mask, AsymHP predicts each head's cost from recent density history and assigns heads to GPUs before the current attention computation begins. The next section describes this placement algorithm and the asymmetric runtime that implements it.

3 AsymHP Design

3.1 Overview

AsymHP is a host-side runtime scheduling layer for dynamic sparse attention in video DiTs. It preserves the distributed layout expected by the surrounding transformer blocks and changes only how sparse attention heads are assigned to GPUs. Before a sparse attention layer, AsymHP converts

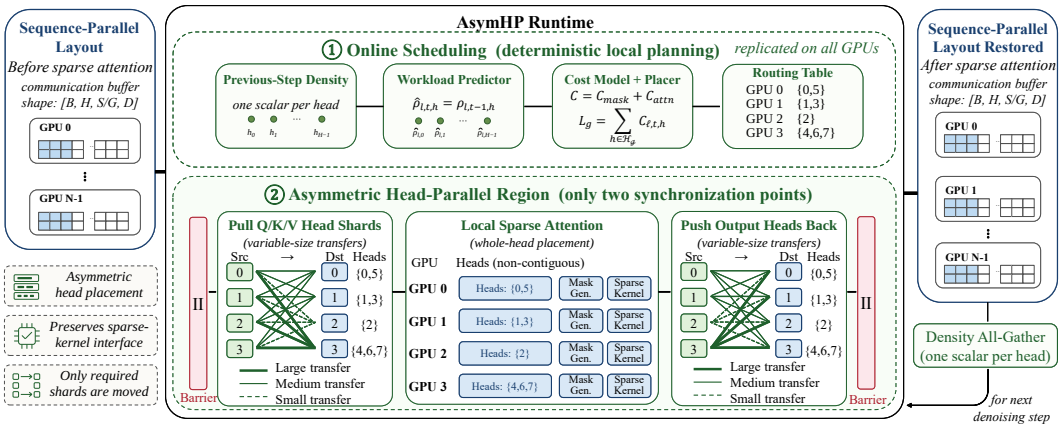


Figure 4: AsymHP overview. The system preserves the sequence-parallel interface around the attention layer while using an asymmetric head-parallel schedule inside dynamic sparse attention.

the sequence-parallel layout into an asymmetric head-parallel layout; after local sparse attention finishes, it restores the original layout.

AsymHP has three components. The predictor estimates each head's current workload from recent sparse density. The placer assigns heads to GPUs to balance the predicted workload, allowing different GPUs to receive different and non-contiguous head sets. The redistribution runtime moves only the Q/K/V and output head shards required by this placement, avoiding padding to the maximum head count.

Figure 4 summarizes this data path. AsymHP first uses the previous step's density metadata to predict per-head work, then computes an asymmetric placement, redistributes the required head shards, executes sparse attention locally, and restores the original sequence-parallel layout.

AsymHP treats each attention head as the atomic placement unit. This choice keeps the runtime simple and preserves the existing sparse-kernel interface. We study the remaining granularity limit of whole-head placement with a trace-driven single-head splitting simulation in Section 5.4.

3.2 Online Workload Prediction

The exact density of head h in layer ℓ at denoising step t is known only after its sparse mask is generated. AsymHP therefore uses a one-step lag predictor:

$$\hat{\rho}_{\ell,t,h} = \rho_{\ell,t-1,h}, \quad t > 0. \tag{2}$$

For the first denoising step, AsymHP uses a deterministic static placement before density history becomes available. Specifically, the first $H \bmod G$ ranks receive $\lceil H/G \rceil$ contiguous heads each and the remaining ranks receive $\lfloor H/G \rfloor$. Each subsequent sparse invocation updates the placement using the preceding-step density for the same layer.

This predictor is deliberately lightweight. It requires only the retained-block count already produced by the sparse attention method, and it avoids adding an extra profiling or mask-construction pass on the critical path.

After each sparse attention execution, AsymHP records the realized $\rho_{\ell,t,h}$ for every head and uses it to update the per-layer density history.

3.3 Asymmetric Head Placement

AsymHP converts predicted density into a per-head cost estimate. The cost of a head has two components: mask generation and sparse attention computation. For layer ℓ at denoising step t, let $N_{\ell,t}$ be the sequence length and $\hat{\rho}_{\ell,t,h}$ be the predicted density of head h. AsymHP models the head cost as

$$C_{\ell,t,h} = C^{\text{mask}}_{\ell,t,h} + C^{\text{attn}}_{\ell,t,h}, \tag{3}$$

where

$$C^{\text{mask}}_{\ell,t,h} = \alpha^{\text{mask}}_{\ell} N_{\ell,t}, \tag{4}$$

and

$$C^{\text{attn}}_{\ell,t,h} = \alpha^{\text{attn}}_{\ell} \hat{\rho}_{\ell,t,h} N^2_{\ell,t}. \tag{5}$$

The mask-generation term is linear in sequence length, while the sparse attention term scales with the number of retained query-key interactions, approximated by $\hat{\rho}_{\ell,t,h} N^2_{\ell,t}$. The coefficients and intercepts are fitted once per model–kernel–hardware configuration from short profiling runs and reused across prompts, steps, and measured runs. For a placement $\{\mathcal{H}_1, \ldots, \mathcal{H}_G\}$, the resulting predicted load of GPU g is

$$L_g = \beta^{\text{mask}}_{\ell} + \beta^{\text{attn}}_{\ell} + \sum_{h \in \mathcal{H}_g} C_{\ell,t,h}, \tag{6}$$

where $\beta^{\text{mask}}_{\ell}$ and $\beta^{\text{attn}}_{\ell}$ are fixed intercepts paid once per rank's batched mask-generation and sparse-attention invocation, respectively. Their common sum does not affect placement. Placement is insensitive to moderate coefficient error: a $2\times$ error in $\alpha^{\text{mask}}_{\ell}$ raises the mean predicted makespan by only 0.6%, and planning HunyuanVideo with the Wan2.1-1.3B fit costs 0.03% (Appendix C.2).

This model captures two effects that matter for placement. Dense heads are more expensive because their sparse attention term is larger. At the same time, even low-density heads still incur mask-generation and fixed per-head overheads, so placing too many sparse heads on one GPU can also create imbalance. Thus, AsymHP balances predicted total head cost rather than density alone.

Given the predicted head costs, AsymHP aims to minimize the straggler load $\max_g L_g$. AsymHP approximates this objective with a greedy longest-processing-time heuristic: it sorts heads by predicted cost and repeatedly assigns the next head to the GPU with the smallest current predicted load.

3.4 Asymmetric Head Redistribution

Asymmetric placement requires variable-size data movement. The planner produces a routing table that specifies, for each source GPU and destination GPU, which Q/K/V head shards must be moved before sparse attention. All GPUs construct the same routing table deterministically from the replicated density history and cost model. Unlike symmetric all-to-all, which would pad every destination to the maximum number of assigned heads, this routing table encodes variable-size transfers directly.

AsymHP uses the routing table to scatter only the required head shards, executes mask generation and sparse attention locally on each destination GPU, and then applies the inverse routing table to gather output heads back to the sequence-parallel layout. This design keeps the sparse attention kernel interface unchanged: each GPU still receives a local set of complete heads and runs the original per-head mask generation and sparse kernel. The only difference is that different GPUs may receive different numbers of heads.

4 Implementation

4.1 Density Collection and Plan Construction

After each sparse attention execution, AsymHP records the realized density of the heads executed on each GPU. Since asymmetric placement gives each GPU only a subset of heads, AsymHP performs a lightweight all-gather of the per-head density records so that every GPU has the same density table for the next placement decision. The communicated state is small: each sparse-attention layer produces only one scalar per head, independent of the sequence length.

This all-gather is not on the sparse-attention critical path. The density $\rho_{\ell,t,h}$ is used only when the same layer is executed at the next denoising step, so the runtime overlaps the all-gather with subsequent layer execution on a side stream. At the beginning of the next invocation of layer ℓ, every GPU uses the replicated density table, the fitted cost model, and deterministic tie-breaking rules to construct the same placement and routing table locally. Thus, AsymHP does not require a centralized planner or a separate plan broadcast. Appendix A.1 reports the exposed planning overhead and GPU critical-path breakdown.

4.2 Pull–Push Asymmetric Head Exchange

AsymHP implements asymmetric head redistribution with a pull–push protocol over symmetric GPU buffers. Before sparse attention, tensors are in the sequence-parallel layout: each GPU owns a sequence shard for all heads, with communication-visible buffers of shape $[B, H, S/G, D]$. The asymmetric placement changes which heads each GPU will execute, but it does not change the shape of these symmetric communication buffers. As a result, AsymHP allocates communication buffers once and reuses them across placements. This avoids repeated buffer allocation, a costly operation in multi-GPU runtimes, while keeping the reserved buffer size fixed to the standard sequence-parallel activation shape for large video diffusion activations.

For the sequence-parallel to head-parallel transition, AsymHP uses a pull phase. Each destination GPU reads the Q/K/V shards corresponding to its assigned heads from the fixed symmetric buffers on all source GPUs, and writes them into its local head-parallel workspace. Since the destination drives the transfer, each GPU knows exactly when its local input workspace has been filled. A single synchronization before the pull kernel is sufficient to ensure that all Q/K/V shards are visible to their consumers.

For the head-parallel to sequence-parallel transition, AsymHP uses the inverse direction. After local sparse attention, each GPU pushes its output head shards directly into the destination sequence-parallel output buffers. A second synchronization after this push phase ensures that the restored sequence-parallel tensor is ready for the following transformer operations. Thus, the asymmetric exchange requires synchronization only at the two boundaries of the head-parallel region: before sparse attention begins and after the output layout has been restored.

5 Evaluation

We evaluate whether AsymHP improves distributed sparse attention in video DiT inference. AsymHP retains the underlying sparse-attention algorithm and changes only head placement and communication. The evaluation answers four questions: (1) how much AsymHP speeds up the sparse-attention region across models, GPU counts, and video lengths; (2) whether the improvement comes from reducing head-level stragglers; (3) how the region speedup translates into end-to-end generation latency; and (4) whether the benefit carries over to another sparse-attention method and interconnect.

5.1 Experimental Setup

Workloads. We evaluate AsymHP on three video DiT workloads with SVG2 [32] sparse attention: Wan2.1-1.3B, Wan2.1-14B [28], and HunyuanVideo [16]. For each workload, AsymHP and the baseline use the same sparse-attention configuration. Unless otherwise stated, experiments use 720p videos with 120 requested frames. We report sparse-attention-region latency, including layout exchanges, mask construction, sparse kernels, output redistribution, and required synchronization. Each region measurement averages 50 runs after 20 warmup runs; Appendix A describes the generation and replay protocols.

Hardware. Unless otherwise stated, experiments run on a single H100 NVLink node, and the environment uses CUDA 12.9 and PyTorch v2.9.1 [1], NCCL v2.27.5 [24].

5.2 Sparse-Attention Speedup

Figure 5 shows the main result. For each workload, we evaluate on 2, 3, 4, 5, 6, and 8 GPUs whenever supported. Default SVG2 head-parallel execution requires the number of heads to be divisible by the GPU count, so its baseline is unavailable for non-divisible configurations. AsymHP does not require equal headcounts and can run these non-divisible configurations.

AsymHP reduces sparse-attention latency in every measured configuration, by up to $1.48\times$. The gain generally increases with GPU count because equal-head placement becomes more sensitive to head-level workload skew: as heads are divided among more GPUs, assigning several dense heads to one GPU creates a larger straggler effect. AsymHP reduces this effect by assigning a non-uniform number of heads to each GPU according to the previous denoising step's sparse density.

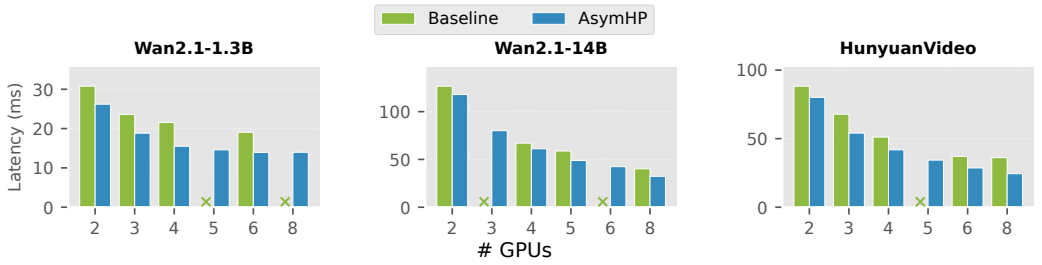
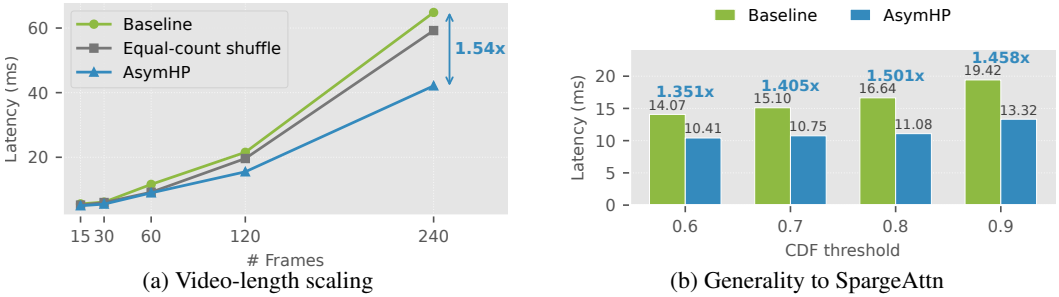


Figure 5: Sparse-attention-region latency for default SVG2 head-parallel execution and AsymHP. Crosses indicate configurations where the default baseline is unsupported.



(a) Video-length scaling

(b) Generality to SpargeAttn

Figure 6: Measured sparse-attention results. (a) Video-length scaling for Wan2.1-1.3B at 720p on four GPUs. (b) SpargeAttn replay on a fixed 720p/125-frame Wan2.1-1.3B snapshot using four H100s; blue annotations report speedup over equal-head execution.

Figure 6(a) scales video length for Wan2.1-1.3B at 720p on 4 GPUs. The benefit grows with video length, because longer sequences make sparse attention a larger share of the region, and reaches $1.54\times$ at 240 frames.

5.3 Ablation Study

We ablate the placement policy while keeping the same SVG2 sparse-attention kernels. To make the placement-policy differences visible, this ablation uses Wan2.1-1.3B at 720p with 240 frames on 4 GPUs, where sparse-attention work is large enough that placement overhead and timing noise do not dominate the comparison.

All variants use AsymHP's asymmetric exchange, so the comparison isolates head placement; the baseline uses contiguous equal-head placement as in default SVG2. Equal-count shuffle reorders heads using previous-step density but still assigns the same number of heads to each GPU. The density-only variant allows non-uniform head counts but uses density directly as the cost. AsymHP further uses the fitted cost model. Oracle AsymHP runs the same placer with current-step density.

Table 1: Ablation study. All variants use the same exchange; speedup is normalized to contiguous equal-head placement.

Method	Equal-count shuffle	Density-only	AsymHP	Oracle AsymHP
Attention speedup	$1.119\times$	$1.482\times$	$1.500\times$	$1.500\times$

The results show that reordering helps, but non-uniform head counts provide most of the benefit. The fitted cost model improves over density-only placement and guards against short-sequence regressions, where fixed per-head costs such as mask construction can dominate and an extra low-density head may still create a straggler. The negligible gap between AsymHP and oracle indicates that previous-step density is sufficient for online scheduling, while a trace analysis shows that keeping the initial placement fixed is markedly worse than re-planning at every step (Appendix C.3).

5.4 Whole-Head Granularity Limit

AsymHP schedules complete attention heads to preserve the existing mask construction and sparse-kernel interface. This granularity leaves residual imbalance when the target per-GPU load falls below the cost of one heavy head. A trace-driven simulation splits bottleneck heads across lower-loaded ranks, while keeping fixed mask-generation costs on the original owner. Once a single heavy head dominates, whole-head placement plateaus, whereas the simulated splitting keeps reducing the straggler. Our FlashInfer-based prototype [34] does not realize this gain because metadata, scatter/gather, and synchronization costs offset the benefit. We therefore report modeled headroom rather than measured acceleration; Appendix D gives the simulation protocol and curve.

5.5 End-to-End Performance

Table 2 reports end-to-end (E2E) generation latency for Wan2.1-1.3B on four H100s over five runs, with both methods enabling sparse attention from the first step and layer. The E2E speedup closely matches the Amdahl estimate from the baseline's sparse-region share and the region speedup: the region gain carries over to generation latency in proportion to the time spent in sparse attention.

Table 2: End-to-end generation on four H100s, Wan2.1-1.3B, 720p/121 frames, 50 steps; 95% Student-t intervals over five runs. Sparse attention takes 67.8% of baseline time, so Amdahl's law predicts $1.133\times$.

Metric	Equal-head	AsymHP	Speedup
Generation (s)	103.91 ± 0.20	92.10 ± 0.15	$1.128\times$
Sparse region (ms/call)	23.49	19.42	$1.209\times$

5.6 Generality and Sensitivity

Figure 6(b) evaluates AsymHP with SpargeAttn [38] on four H100s. On a fixed Wan2.1-1.3B 720p/125-frame snapshot, with per-threshold calibration and an unmodified mask selector and kernel, AsymHP speeds up the region at every tested CDF threshold, extending the scheduling benefit to a second sparse method. The benefit also persists without NVLink (Table 3): on four L40S GPUs connected only through PCIe, unequal placement alone already helps, and the asymmetric exchange adds a further gain over padded NCCL. Varying only SVG2's top-p on the same H100 snapshot, the speedup grows steadily as attention becomes denser, because the density-dependent attention cost increasingly outweighs the fixed mask-construction cost (Appendix C.1).

Table 3: Sparse-attention-region latency on four PCIe-only L40S GPUs (Wan2.1-1.3B, 720p/121 frames, layer 21, step 20; maximum per-rank mean over 50 iterations).

Placement and transport	Region (ms)	Speedup
Contiguous equal-head + NCCL	63.42	$1.000\times$
Cost-aware unequal + padded NCCL	56.84	$1.116\times$
Cost-aware unequal + asymmetric (AsymHP)	53.86	$1.177\times$

5.7 Numerical Agreement

AsymHP runs each head with the same mask construction and sparse kernel as the baseline and only routes tensors between GPUs. On a Wan2.1-1.3B snapshot, the asymmetric transport is bitwise exact, and operator outputs match the equal-head baseline bitwise whenever every GPU holds at least two heads. A GPU holding a single head runs the local kernel with a head batch of one, which takes a different code path and causes floating-point-level differences. For decoded videos, AsymHP differs from the baseline no more than two baseline runs differ from each other (Appendix B).

6 Related Works

6.1 Sparse Attention for Video Diffusion

Sparse attention has become a common way to reduce the attention cost of video diffusion transformers. Some methods use structured or mostly static spatio-temporal masks: Sparse VideoGen [30] exploits head-dependent spatial and temporal sparse patterns, Sparse-vDiT [4] searches layer- and head-specific sparse strategies, Sliding Tile Attention [40] uses local 3D sliding tiles, and Radial Attention [19] derives radial sparse masks from spatio-temporal energy decay. These methods show that video attention has a strong spatio-temporal structure, but their sparse layouts are largely determined by predefined patterns.

Other methods make sparsity more adaptive or integrate it into training. Sparse VideoGen2 [32] uses semantic-aware permutation, while VMoBA [29] routes attention through 1D/2D/3D mixture-of-block patterns. VSA [39] replaces full attention with trainable sparse attention during both training and inference, BSA [35] sparsifies both query and key-value dimensions, and SLA [36, 37] combines exact sparse attention with linear attention after fine-tuning. General sparse-attention and hardware co-design systems such as SpargeAttn [38], FG-Attn [6], and FPSAttention [20] further improve mask prediction, fine-grained sparse execution, or quantization-sparsity co-design. Large-scale systems such as LongCat-Video also adopt block-sparse attention in production-scale video DiTs [27].

6.2 Distributed Attention and DiT Parallelism

DeepSpeed-Ulysses [14], Ring Attention [21], LoongTrain [11], and USP [9] parallelize long-sequence attention by partitioning heads, sequence blocks, or both. Diffusion-specific systems such as DistriFusion [18], PipeFusion [7], and xDiT [8] further exploit the denoising process with patch, pipeline, sequence, and CFG parallelism for high-resolution diffusion or DiT inference. These systems address how to scale regular dense attention or whole DiT inference across devices.

6.3 Load Balancing for Distributed Attention

Load imbalance has been studied in distributed attention, but prior work mostly targets more structured imbalance sources. DistFlashAttn [17] and Striped Attention [2] balance the triangular workload induced by causal masks in Ring-style attention, where the imbalance is fixed by token position. More recent systems such as UltraAttn [31] and DCP [15] target more general sparse or irregular static masks, using offline partitioning over sequence or block dimensions to balance fixed attention masks. Concurrent db-SP targets sparsity-aware USP [9] for visual generation; unlike AsymHP's unequal head placement, its Ulysses path uses equal-count head shuffling and mainly focuses on block-level balancing and strategy selection [5].

7 Conclusion

This paper identifies head-level sparsity imbalance as a practical bottleneck in dynamic sparse attention for video DiTs. AsymHP addresses this bottleneck with a simple online scheduler: it uses the previous denoising step's per-head density to estimate current head cost, assigns heads asymmetrically across GPUs, and redistributes only the required head shards. This design preserves the sparse attention algorithm while reducing straggler time caused by equal-head placement.

Limitations. AsymHP targets sparse methods with data-dependent per-head budgets, such as top-p or threshold rules; methods with a fixed per-head budget, such as a static mask or fixed top-k selection (e.g., Radial Attention [19]), expose little head-level imbalance to exploit. Whole-head placement leaves residual imbalance when a single head exceeds the per-GPU target (Section 5.4). Finally, since the evaluated models fit within one node, our evaluation is limited to single-node execution; cross-node asymmetric head redistribution, where communication becomes a larger fraction of latency, remains future work.

Broader Impacts. AsymHP improves the efficiency of sparse-attention video generation, reducing inference cost for long-video DiT serving, but cheaper generation may also lower the cost of misuse, such as large-scale synthetic media generation.

Acknowledgments and Disclosure of Funding

The authors received no specific funding for this work and declare no competing interests.

References

[1] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, page 929–947, New York, NY, USA, 2024. Association for Computing Machinery.

[2] William Brandon, Aniruddha Nrusimha, Kevin Qian, Zachary Ankner, Tian Jin, Zhiye Song, and Jonathan Ragan-Kelley. Striped attention: Faster ring attention for causal transformers, 2023.

[3] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024.

[4] Pengtao Chen, Xianfang Zeng, Maosen Zhao, Peng Ye, Mingzhu Shen, Wei Cheng, Gang Yu, and Tao Chen. Sparse-vdit: Unleashing the power of sparse attention to accelerate video diffusion transformers, 2025.

[5] Siqi Chen, Ke Hong, Tianchen Zhao, Ruiqi Xie, Zhenhua Zhu, Xudong Zhang, and Yu Wang. db-sp: Accelerating sparse attention for visual generative models with dual-balanced sequence parallelism, 2025.

[6] Sankeerth Durvasula, Kavya Sreedhar, Zain Moustafa, Suraj Kothawade, Ashish Gondimalla, Suvinay Subramanian, Narges Shahidi, and Nandita Vijaykumar. Fg-attn: Leveraging fine-grained sparsity in diffusion transformers, 2025.

[7] Jiarui Fang, Jinzhe Pan, Aoyu Li, Xibo Sun, and Jiannan Wang. Pipefusion: Patch-level pipeline parallelism for diffusion transformers inference, 2025.

[8] Jiarui Fang, Jinzhe Pan, Xibo Sun, Aoyu Li, and Jiannan Wang. xdit: an inference engine for diffusion transformers (dits) with massive parallelism, 2024.

[9] Jiarui Fang and Shangchun Zhao. Usp: A unified sequence parallelism approach for long context generative ai, 2024.

[10] Yu Gao, Haoyuan Guo, Tuyen Hoang, Weilin Huang, Lu Jiang, Fangyuan Kong, Huixia Li, Jiashi Li, Liang Li, Xiaojie Li, Xunsong Li, Yifu Li, Shanchuan Lin, Zhijie Lin, Jiawei Liu, Shu Liu, Xiaonan Nie, Zhiwu Qing, Yuxi Ren, Li Sun, Zhi Tian, Rui Wang, Sen Wang, Guoqiang Wei, Guohong Wu, Jie Wu, Ruiqi Xia, Fei Xiao, Xuefeng Xiao, Jiangqiao Yan, Ceyuan Yang, Jianchao Yang, Runkai Yang, Tao Yang, Yihang Yang, Zilyu Ye, Xuejiao Zeng, Yan Zeng, Heng Zhang, Yang Zhao, Xiaozheng Zheng, Peihao Zhu, Jiaxin Zou, and Feilong Zuo. Seedance 1.0: Exploring the boundaries of video generation models, 2025.

[11] Diandian Gu, Peng Sun, Qinghao Hu, Ting Huang, Xun Chen, Yingtong Xiong, Guoteng Wang, Qiaoling Chen, Shangchun Zhao, Jiarui Fang, Yonggang Wen, Tianwei Zhang, Xin Jin, and Xuanzhe Liu. Loongtrain: Efficient training of long-sequence llms with head-context parallelism, 2024.

[12] Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Li Fei-Fei, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models, 2023.

[13] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. Ltx-video: Realtime video latent diffusion, 2024.

[14] 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, 2023.

[15] Chenyu Jiang, Zhenkun Cai, Ye Tian, Zhen Jia, Yida Wang, and Chuan Wu. Dcp: Addressing input dynamism in long-context training via dynamic context parallelism. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*, SOSP '25, page 221–236. ACM, October 2025.

[16] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. Hunyuanvideo: A systematic framework for large video generative models, 2025.

[17] Dacheng Li, Rulin Shao, Anze Xie, Eric P. Xing, Xuezhe Ma, Ion Stoica, Joseph E. Gonzalez, and Hao Zhang. Distflashattn: Distributed memory-efficient attention for long-context llms training, 2024.

[18] Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Ming-Yu Liu, Kai Li, and Song Han. Distrifusion: Distributed parallel inference for high-resolution diffusion models, 2024.

[19] Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, Maneesh Agrawala, Ion Stoica, Kurt Keutzer, and Song Han. Radial attention: $o(n \log n)$ sparse attention with energy decay for long video generation, 2025.

[20] Akide Liu, Zeyu Zhang, Zhexin Li, Xuehai Bai, Yizeng Han, Jiasheng Tang, Yuanjie Xing, Jichao Wu, Mingyang Yang, Weihua Chen, Jiahao He, Yuanyu He, Fan Wang, Gholamreza Haffari, and Bohan Zhuang. Fpsattention: Training-aware fp8 and sparsity co-design for fast video diffusion, 2025.

[21] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context, 2023.

[22] Haoyu Lu, Guoxing Yang, Nanyi Fei, Yuqi Huo, Zhiwu Lu, Ping Luo, and Mingyu Ding. Vdt: General-purpose video diffusion transformers via mask modeling, 2023.

[23] Xin Ma, Yaohui Wang, Xinyuan Chen, Gengyun Jia, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation, 2025.

[24] NVIDIA Corporation. *NVIDIA Collective Communications Library (NCCL)*, 2026. Version 2.27.5.

[25] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023.

[26] 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, 2025.

[27] Meituan LongCat Team, Xunliang Cai, Qilong Huang, Zhuoliang Kang, Hongyu Li, Shijun Liang, Liya Ma, Siyu Ren, Xiaoming Wei, Rixu Xie, and Tong Zhang. Longcat-video technical report, 2025.

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

[29] Jianzong Wu, Liang Hou, Haotian Yang, Xin Tao, Ye Tian, Pengfei Wan, Di Zhang, and Yunhai Tong. Vmoba: Mixture-of-block attention for video diffusion models, 2025.

[30] Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity, 2025.

[31] Haoyu Yang, Zan Zong, Yuyang Jin, Kinman Lei, Jiaao He, Qigang Yang, and Jidong Zhai. Ultraattn: Efficiently parallelizing attention through hierarchical context-tiling. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '25, page 1769–1784, New York, NY, USA, 2025. Association for Computing Machinery.

[32] Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, Jianfei Chen, Song Han, Kurt Keutzer, and Ion Stoica. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation, 2025.

[33] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer, 2025.

[34] Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. Flashinfer: Efficient and customizable attention engine for llm inference serving, 2025.

[35] Chenlu Zhan, Wen Li, Chuyu Shen, Jun Zhang, Suhui Wu, and Hao Zhang. Bidirectional sparse attention for faster video diffusion training, 2025.

[36] Jintao Zhang, Haoxu Wang, Kai Jiang, Shuo Yang, Kaiwen Zheng, Haocheng Xi, Ziteng Wang, Hongzhou Zhu, Min Zhao, Ion Stoica, Joseph E. Gonzalez, Jun Zhu, and Jianfei Chen. Sla: Beyond sparsity in diffusion transformers via fine-tunable sparse-linear attention, 2025.

[37] Jintao Zhang, Haoxu Wang, Kai Jiang, Kaiwen Zheng, Youhe Jiang, Ion Stoica, Jianfei Chen, Jun Zhu, and Joseph E. Gonzalez. Sla2: Sparse-linear attention with learnable routing and qat, 2026.

[38] Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference, 2025.

[39] Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. Vsa: Faster video diffusion with trainable sparse attention, 2025.

[40] Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention, 2025.

[41] Zangwei Zheng, Xiangyu Peng, Yuxuan Lou, Chenhui Shen, Tom Young, Xinying Guo, Binluo Wang, Hang Xu, Hongxin Liu, Mingyan Jiang, Wenjun Li, Yuhui Wang, Anbang Ye, Gang Ren, Qianran Ma, Wanying Liang, Xiang Lian, Xiwen Wu, Yuting Zhong, Zhuangyan Li, Chaoyu Gong, Guojun Lei, Leijun Cheng, Limin Zhang, Minghao Li, Ruijie Zhang, Silan Hu, Shijie Huang, Xiaokang Wang, Yuanheng Zhao, Yuqi Wang, Ziang Wei, and Yang You. Open-sora 2.0: Training a commercial-level video generation model in $200k, 2026.

[42] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, 2024.

[43] Yuan Zhou, Qiuyue Wang, Yuxuan Cai, and Huan Yang. Allegro: Open the black box of commercial-level video generation model, 2024.

A Experimental Protocol and Critical-Path Accounting

Configuration. The generation experiments use Wan2.1-1.3B with SVG2 sparse attention at 720×1280, 50 denoising steps, guidance scale 5.0, seed 0, 300 query centroids, 1000 key centroids, $\text{top-}p = 0.9$, minimum key-centroid ratio 0.10, and 50/2 k-means iterations for initialization/subsequent steps. Both baseline and AsymHP enable sparse attention from the first step and layer. The snapshot used by the operator-agreement and PCIe studies comes from a run that keeps the first 20% of steps and 3% of layers dense.

A request for 120 frames produces 121 actual frames in Wan, with 31 latent frames and $N = 111600$ tokens. The SpargeAttn and sparsity-sweep replays use layer 21, step 20 of a 125-frame snapshot (32 latent frames, $N = 115200$).

Timing. Generation runs pair the contiguous/symmetric baseline and AsymHP under the same prompt, seed, and sparse configuration. A two-step warmup precedes timing; RNG state, placement, and centroid history are reset before the measured 50-step generation. Timing includes denoising, decoding, and synchronization, and excludes model loading, prompt encoding, JIT warmup, and file export. Each E2E mean uses five runs, with a two-sided 95% interval $\bar{x} \pm t_{0.975,4} s/\sqrt{5}$, where s is the sample standard deviation and $t_{0.975,4} = 2.776$.

A.1 Critical-Path Accounting

Table 4 averages GPU phase times on the critical rank for each sparse invocation in the five 121-frame AsymHP generation runs. Boundary synchronization includes waiting for the slowest rank and is distinct from data transfer. The pull–push exchange replaces the two all-to-alls of equal-head execution rather than adding a separate migration phase.

The host completion wait plus placement construction is about 0.05 ms per re-plan on four H100s. The per-head density exchange runs on a side stream and overlaps subsequent layers; its exposed effect is included in the end-to-end timings.

B Operator and Decoded-Video Agreement

Table 5 uses a Wan2.1-1.3B 720p snapshot at layer 21, step 20, $N = 111600$. Fixing placement while changing only the transport isolates routing from local computation. Changing the head split additionally exercises the local backend's head batching: both four-GPU placements are non-uniform, but a GPU holding a single head runs the local sparse kernel with a head batch of one, which takes a different code path and produces floating-point-level differences.

Table 4: AsymHP GPU critical-path breakdown, Wan2.1-1.3B, 720p/121 frames, four H100s. Values sum before rounding to 19.425 ms/invocation.

Phase	ms/invocation	Share (%)
Sparse attention	11.050	56.89
Mask construction	4.525	23.29
Boundary synchronization	2.102	10.82
Pull–push transfer	1.025	5.27
Other local work	0.598	3.08
Inverse permutation	0.065	0.34
Per-head density computation	0.052	0.27
Local copy	0.007	0.04

Table 5: Operator agreement with the symmetric baseline on the SVG2 snapshot.

Comparison	$\max \lvert\Delta\rvert$	Relative L_2	PSNR (dB)
Transport only, placement fixed	0	0	∞
Two GPUs, 5/7 heads	0	0	∞
Four GPUs, 2/3/3/4 heads	0	0	∞
Four GPUs, 1/3/4/4 heads	4.132×10^{-2}	2.73×10^{-3}	72.88

Since operator outputs have no natural peak value, operator PSNR uses the maximum absolute value of the reference output as the peak.

The decoded-video experiment uses three prompts, seed 0, 720p/125 frames, 50 steps, sparse attention from the first step and layer, and four H100s, and compares post-VAE floating-point video tensors. The pipeline is not bitwise deterministic across runs, so each prompt has a baseline-versus-baseline control under the same configuration (Table 6); AsymHP's deviation from the baseline matches this run-to-run variation.

Table 6: Decoded-video PSNR (dB) with a baseline repeatability control. AsymHP uses at least two heads per GPU.

Prompt	AsymHP vs baseline	Baseline vs baseline
1	34.78	34.58
2	28.51	28.75
3	25.25	25.87
Mean	29.51	29.73

C Sensitivity and Temporal Placement Analysis

C.1 Measured Sparsity Sweep

We vary only SVG2's top-p on a fixed 720p/125-frame snapshot at layer 21, step 20, on four H100s, planning each setting from its own preceding-step density. Table 7 reports means of two runs, each with 20 warmups and 50 measured iterations. The speedup increases monotonically with realized density, in each run as well as on average: the baseline latency grows quickly as heads become denser, while AsymHP keeps the growth much flatter by spreading the dense heads across GPUs.

C.2 Cost-Model Calibration and Mismatch

Each stack is calibrated once and its coefficients reused across prompts and steps. Table 8 reports fit quality for the model configurations on H100s and for the separate L40S calibration. All three H100 configurations use the same sparse kernels at head dimension 128. For fixed sequence length, Equation 3 has the form $C_{\ell,t,h} = A + B\hat{\rho}_{\ell,t,h}$. A common positive rescaling of A and B leaves greedy placement unchanged, whereas their ratio controls the penalty for assigning additional heads.

Table 7: SVG2 sparsity sensitivity: four H100s, fixed 125-frame snapshot. Times include the complete sparse-attention region.

Top-p	Realized density	Baseline (ms)	AsymHP (ms)	Speedup
0.70	0.13	18.74	16.52	$1.134\times$
0.80	0.15	21.41	17.68	$1.211\times$
0.90	0.19	27.13	19.69	$1.378\times$
0.95	0.28	37.77	23.98	$1.575\times$

This explains why the relative mask/attention weighting matters more to placement than an absolute latency scale.

Table 8: Calibration fit quality. The L40S stack receives its own fit.

Stack	Mask R^2	Attention R^2
Wan2.1-1.3B / H100	0.922	0.976
Wan2.1-14B / H100	0.964	0.949
HunyuanVideo / H100	0.993	0.992
Wan2.1-1.3B / L40S	0.982	0.997

For coefficient sensitivity, we use a recorded Wan2.1-1.3B trace with 360 layer/step points, $N = 99840$, and four GPUs. We scale α^{mask} while keeping α^{attn} fixed, construct a placement, then score it under the native cost model including the per-rank intercept. The penalty is the resulting predicted makespan divided by that of the native-coefficient greedy placement. It grows with the size of the error but stays small within a factor of two, while the fitted mask/attention weighting differs by at most $1.4\times$ across our H100 stacks.

Table 9: Offline coefficient sensitivity under the native scoring model.

Mask-slope multiplier	Mean penalty	p99 penalty	Maximum penalty
$0.5\times$	$1.0020\times$	$1.0280\times$	$1.0524\times$
$2\times$	$1.0063\times$	$1.0720\times$	$1.0968\times$
$4\times$	$1.0349\times$	$1.1511\times$	$1.1593\times$
$8\times$	$1.0576\times$	$1.2702\times$	$1.3105\times$

Under the same protocol, planning a HunyuanVideo trace ($N = 108256$) with the Wan2.1-1.3B fit changes the mean predicted makespan by less than 0.1% at both four and eight GPUs. Placement is therefore robust to coefficient mismatch across stacks that share the same kernels; a new kernel, head dimension, or GPU is recalibrated with one short profiling run, as for the L40S stack.

C.3 Why Update Placement Across Steps?

A 720p trace analysis at $N = 115200$ and four GPUs compares three placements, all scored on current-step costs under the cost model: the deterministic near-even initialization held fixed, the greedy placement built from preceding-step density, and the same greedy placer using current-step density. The best placement changes for most adjacent-step pairs, so a fixed placement drifts away from balance as density evolves, while planning from the preceding step tracks current-step planning closely (Table 10).

D Whole-Head Granularity Analysis

Whole-head placement can leave residual imbalance when the target per-GPU load falls below the cost of a single heavy head. For the trace-driven curve in Figure 7, we start from AsymHP's whole-head placement and split a bottleneck head only when the maximum rank load remains imbalanced. Its divisible sparse-attention work is assigned to lower-loaded ranks by water filling, while fixed

Table 10: Predicted makespan summed over the trajectory, relative to current-step planning (720p trace, four GPUs). The current-step placement changes at 62% of adjacent-step pairs.

Placement	Relative makespan
Current-step density (oracle)	1.000×
Preceding-step density (AsymHP)	1.004×
Initial placement held fixed	1.321×

per-head mask-generation costs remain on the original owner. Split shares are rounded to query-block boundaries.

The simulation models placement costs and excludes the additional metadata, scatter/gather, and synchronization work required to execute a split head.

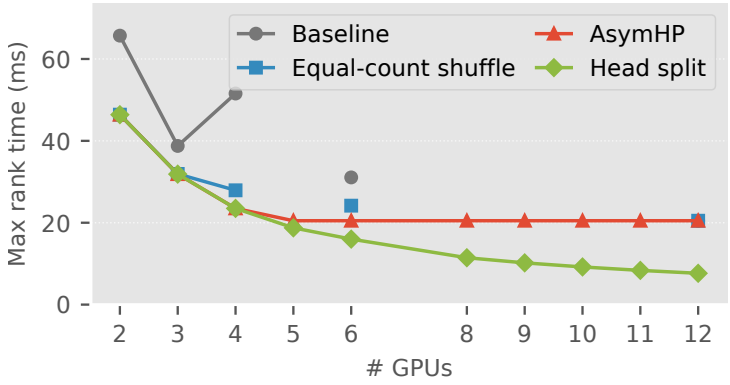


Figure 7: Whole-head granularity limit in a 12-head trace snapshot (layer 10, recorded timestep 241). Curves show modeled maximum rank time for each placement policy; the head-split curve excludes split-execution overhead.

As the GPU count grows, whole-head placement plateaus once a single heavy head dominates the maximum rank load, whereas the simulated split policy keeps reducing it. Our split-head prototype on the FlashInfer backend [34] does not realize this gain because additional execution overhead offsets it, so the curve reports simulated headroom.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the scope of the work as a runtime system for load-balanced sparse attention in video DiTs, and the evaluation reports sparse-attention speedup.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Limitations paragraph in Section 7 discusses the scope to sparse methods with data-dependent per-head budgets, the residual imbalance of whole-head placement, and the single-node evaluation scope. Section 5.4 quantifies the whole-head limit with a trace-driven split-head headroom estimate and explains why the prototype split-head implementation is not used as a main result due to backend overheads.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present theoretical theorems or formal proofs. The cost model is used as a runtime scheduling heuristic and is evaluated empirically.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the evaluated workloads, sparse-attention configuration, baseline, implementation, hardware, and GPU counts needed to reproduce the main experimental conclusions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is available at `https://github.com/tie-pilot-qxw/AsymHP-NeurIPS26`. It contains the AsymHP implementation, the end-to-end integration, replay/profiling scripts, and configurations, with instructions to reproduce the results reported in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental section specifies the video DiT workloads, sparse-attention method, baseline, hardware platform, GPU counts, and the fact that AsymHP uses the same sparse-attention kernels and generation settings as the baseline.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No].

Justification: The original region benchmarks report means without confidence intervals. Table 2 adds 95% Student-t intervals across five generation runs, with assumptions and protocols specified in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The paper reports the GPU platforms (an H100 NVLink node and, for the PCIe study, four L40S GPUs) and the GPU counts used for each experiment. The experiments are inference-only latency measurements rather than training runs, so the compute requirement is limited to repeated video-generation or sparse-attention replay runs on the reported configurations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work conforms to the NeurIPS Code of Ethics. It does not involve human subjects, private user data, deceptive data collection, or safety-critical deployment.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses the positive impact of reducing inference cost, and notes that cheaper video generation may also lower the cost of misuse such as large-scale synthetic media generation.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new pretrained video generation model, dataset, or other high-risk asset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the corresponding papers for the models, sparse-attention method, and runtime systems used in the experiments. The main artifacts are used under their released terms: Wan2.1, SVG2, SpargeAttn, and FlashInfer are released under Apache-2.0, while HunyuanVideo is released under the Tencent Hunyuan Community License Agreement.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We release the code artifact linked in the open-access item above. The artifact includes the standalone AsymHP implementation, replay/profiling scripts, and experiment configurations. The accompanying README documents the environment and commands for reproducing the main sparse-attention-region measurements, and the artifact documents third-party components and licenses. It does not introduce a new dataset or pretrained model.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.