VISPI: EFFICIENT 2D PARALLELISM FOR SPARSE LONG-SEQUENCE VIDEO DIT INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion Transformers (DiTs) have become a dominant architecture for video generation. However, scaling DiTs to longer videos is challenging as rapidly growing sequence lengths exacerbate the quadratic cost of full self-attention. While sparse attention reduces computational overhead by selectively enabling token interactions, it fundamentally changes the execution characteristics of DiTs, introducing irregular workloads and communication patterns. Existing distributed parallel strategies designed for dense computation do not adapt well to these sparse workloads. To address this challenge, we present ViSPI, a distributed inference framework tailored for sparse video DiTs. Specifically, it models hardware-level execution costs to balance heterogeneous attention workloads across GPUs. Meanwhile, it overlaps irregular Key-Value (KV) communication with computation via asynchronous execution. Experiments on representative video DiT models demonstrate that ViSPI significantly improves inference efficiency over state-of-the-art distributed inference frameworks while preserving generation quality and scaling efficiently across multiple GPUs.

1 INTRODUCTION

Diffusion Transformers (DiTs) Peebles & Xie (2023) have become a cornerstone of modern video generation. They model spatio-temporal latent tokens through global self-attention, producing highly realistic and temporally coherent outputs Brooks et al. (2024). However, as video resolution and duration continue to scale, the number of latent tokens grows rapidly, and the quadratic complexity of full self-attention becomes a dominant bottleneck in both computation and memory Vaswani et al. (2017). Under these conditions, distributed inference across multiple GPUs is essential for making long-sequence video generation practical.

To make long-context execution tractable, current distributed systems rely on two complementary sequence-level parallelism strategies. Head Parallelism (HP) Jacobs et al. (2023) partitions computation along the attention head dimension, while Context Parallelism (CP) Liu et al. (2024) divides the sequence dimension and exchanges Key-Value (KV) blocks across devices. Modern systems frequently combine both approaches into a hybrid HP$\times$CP framework to scale distributed attention Fang & Zhao (2024); Gu et al. (2024). These traditional hybrid systems operate under two foundational assumptions. First, balanced workload partitioning assumes that equal-sized shards along either the head or sequence dimension yield uniform computational loads across GPUs. Second, bulk communication assumes that GPUs must exchange complete KV partitions via monolithic collective primitives before beginning the attention calculation.

In addition to scaling full attention through hybrid HP-CP execution, recent video DiTs independently address the long-sequence bottleneck by introducing sparse attention, where only a significant subset of KV interactions is computed. Yang et al. (2026b); Tan et al. (2026) However, integrating dynamic sparsity directly into traditional HP$\times$CP systems creates a fundamental execution mismatch, as sparse attention invalidates both underlying assumptions.

On the one hand, dynamic sparsity destroys balanced workload partitioning across both parallel axes. At the HP level, active-block density varies significantly across attention heads, leaving fixed-head allocations vulnerable to severe stragglers Tan et al. (2026). Along the CP axis, active blocks sim-

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.

ilarly cluster around specific spatio-temporal tokens, which causes equal-sized sequence partitions to carry vastly unequal compute burdens.

On the other hand, dynamic sparsity violates the bulk communication assumption by introducing demand-driven KV access. Since each query block requires only a sparse fraction of KV blocks, standard bulk collective primitives waste precious cross-GPU bandwidth by transmitting unreferenced KV data. Furthermore, these monolithic transfers enforce rigid synchronization barriers, which completely serializes communication and computation.

Several recent efforts have explored distributed inference for video DiTs. xDiT Fang et al. (2024) and SwiftFusion Yang et al. (2026a) adapt long-context Transformer parallelism to DiT serving, while PipeFusion Fang et al. (2026), DistriFusion Li et al. (2024), and PAB Zhao et al. (2025) exploit temporal, spatial, or cross-step redundancy to reduce generation latency. However, these methods target dense DiT execution pipelines and do not address the irregular workloads and data accesses introduced by sparse attention.

More recently, DSV Tan et al. (2026) pioneered sparsity-aware parallelism for DiTs. It rebalances attention heads using algorithm-level sparsity statistics to mitigate HP load imbalance and selectively gathers required KV blocks to reduce unnecessary communication. However, its workload balancing remains limited to HP and does not account for execution-level cost differences induced by sparse GPU execution. Its KV gathering also remains synchronous, resulting in communication-computation serialization during sparse attention execution.

In this paper, we introduce VISPI, a distributed inference framework that adapts 2D parallelism to sparse video DiTs. VISPI jointly addresses the two remaining challenges through execution-aware workload balancing across HP and CP and asynchronous communication-computation execution. Our contributions are summarized as follows.

- We propose a workload-aware 2D partitioning strategy for sparse attention. It establishes a unified execution-cost model based on the number of triggered matrix multiply-accumulate (MAC) operations and applies it to both parallelism dimensions. Based on this model, the strategy performs non-uniform partitioning of attention heads for HP and query blocks for CP, balancing the actual sparse computation across GPUs and mitigating stragglers caused by uniform partitioning.
- We propose an asynchronous KV streaming mechanism for sparse CP. It fetches only the KV blocks required by local sparse attention through asynchronous one-sided NVSHMEM access Ma et al. (2026), eliminating the global synchronization barrier of bulk KV exchange. Meanwhile, KV requests are partitioned into micro-batches and executed with double buffering to overlap KV transfer with attention computation.
- We implement VISPI as an end-to-end distributed inference engine for sparse video DiTs. Extensive experiments across diverse video DiT architectures and multi-GPU configurations demonstrate substantial speedups over state-of-the-art distributed baselines, while preserving generation quality and parallel scalability.

2 Related Work

2.1 Distributed Parallelism for Long Contexts

Sequence parallelism and context parallelism partition attention computation and memory across multiple GPUs. Megatron-SP Korthikanti et al. (2023) partitions activations along the sequence dimension within non-tensor-parallel regions. DeepSpeed-Ulysses Jacobs et al. (2023) employs All-to-All communication to enable scalable head-wise partitioning. In parallel, ColAI Li et al. (2023) and Ring Attention Liu et al. (2024) introduce context parallelism by circulating key-value blocks over a ring topology to overlap communication with computation. Hybrid architectures combine these techniques to improve scalability. USP Fang & Zhao (2024) integrates Ulysses and Ring Attention into a two-dimensional framework. LoongTrain Gu et al. (2024) employs Double-Ring-Attention to optimize large-scale distributed training.

These strategies assume dense and uniform attention workloads. This assumption holds for causal large language models but fails for video DiTs due to sparse and heterogeneous attention patterns.

Irregular workloads invalidate steady communication and computation schedules. Therefore, video DiTs require workload-aware scheduling and communication mechanisms.

2.2 Sparse Attention Execution

Sparse attention reduces quadratic computational cost through structured or dynamic execution patterns. Early methods rely on predefined sparsity structures. Longformer Beltagy et al. (2020) and BigBird Zaheer et al. (2020) introduce local and global sparse patterns to improve execution efficiency. Recent works identify important attention regions dynamically based on input features. StreamingLLM Xiao et al. (2024) leverages attention sinks to accelerate long-context inference. MInference Jiang et al. (2024) identifies structured sparse patterns via offline profiling.

For video DiTs, spatio-temporal dynamics create richer attention sparsity. DSV Tan et al. (2026) retains essential key-value interactions to reduce computation without degrading generation quality. However, existing methods focus primarily on identifying skippable computation. They leave distributed execution challenges largely unexplored. Sparse attention creates load imbalance among attention heads and irregular key-value accesses across devices. Standard dense parallelism strategies cannot handle these irregular patterns efficiently. Our work complements existing sparse attention methods by solving these distributed execution challenges.

2.3 Distributed Inference for Video Generation

Recent efforts explore distributed inference for video DiTs through parallelism and pipeline scheduling. xDiT Fang et al. (2024) adapts hybrid sequence parallelism to DiT serving. SwiftFusion Yang et al. (2026a) optimizes communication schedules for DiT execution. PipeFusion Fang et al. (2026) and DistriFusion Li et al. (2024) pipeline spatial and temporal computation across devices. PAB Zhao et al. (2025) reuses attention outputs across timesteps to lower latency.

DSV Tan et al. (2026) addresses training sparsity via head-wise workload redistribution and selective key-value transfer. However, its workload estimation relies solely on sparsity ratios. It lacks fine-grained hardware execution mapping and fine-grained communication-computation overlap. VISPI advances beyond these works by jointly optimizing computation scheduling and communication for irregular sparse video DiT inference.

3 Method

3.1 Overview

Figure 1 illustrates the overall architecture of VISPI. For long video generation with sparse attention, VISPI organizes GPUs into a 2D hybrid HP×CP grid to address execution challenges from two aspects: First, VISPI performs workload-aware 2D partitioning to balance sparse attention across GPUs. A unified execution-cost model estimates the computational load of active sparse blocks. VISPI applies this model to both HP (heads) and CP (query blocks) dimensions, replacing uniform partitioning with workload-balanced non-uniform assignments. Second, VISPI introduces asynchronous KV streaming for CP execution. Since each query block requires only a subset of KV blocks, VISPI retrieves remote KV data on demand via asynchronous one-sided access. Retrieved blocks are transferred in micro-batches, overlapping communication with attention computation using double-buffered pipelining. Together, workload-aware partitioning determines where sparse computation runs, while asynchronous KV streaming manages how required data is delivered.

3.2 Workload-Aware 2D Parallelism

3.2.1 Execution-Cost Workload Estimation.

Multi-GPU execution requires partitioning sparse attention to balance loads across GPUs. While block counts reflect execution cost in dense attention, they fail in sparse attention due to varying active KV blocks across heads and query regions.

Existing sparsity-aware estimators rely on algorithmic metrics (e.g., sparsity ratio or non-zero counts) that ignore hardware execution dynamics. To illustrate this limitation, we evaluate these

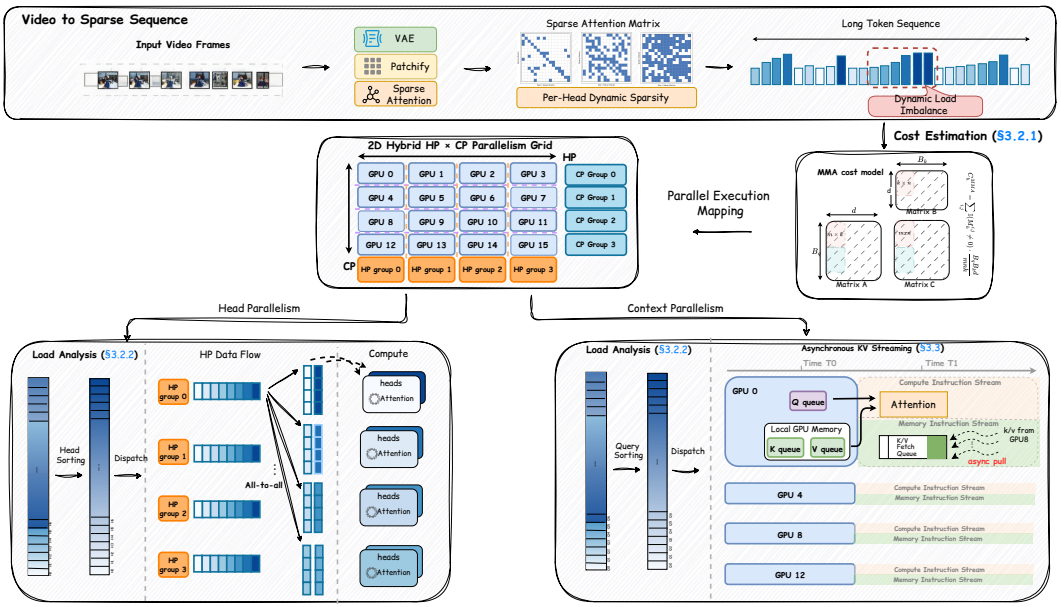

Figure 1: Overview of VISPI. The framework organizes sparse attention execution under 2D hybrid HP×CP parallelism with three design components: workload balancing on the HP axis, workload balancing on the CP axis, and asynchronous KV streaming that overlaps communication with sparse attention computation.

estimators against measured GPU latency using Spearman's rank correlation coefficient ρ (Fig. 2). Global Density shows poor correlation ($\rho = 0.243$) by ignoring spatial sparsity distributions. Max Row Density improves correlation to $\rho = 0.468$ by accounting for row-level stragglers, but ignores hardware block padding and Tensor Core instruction mapping. In contrast, our hardware-aware *MMA Count* achieves a much stronger correlation ($\rho = 0.652$), confirming that instruction-level computation aligns significantly better with actual runtime.

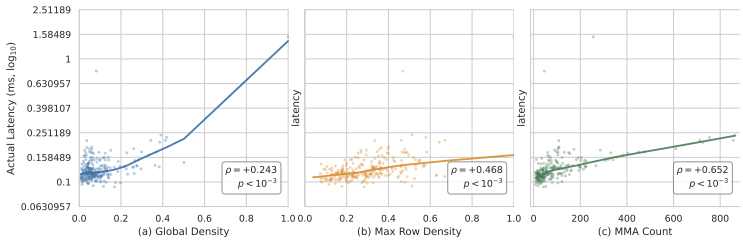

Figure 2: Spearman's rank correlation ρ between different workload estimators and measured GPU execution latency. Higher ρ indicates better alignment between the estimated and actual execution cost. All reported correlation coefficients are statistically significant with $p < 10^{-3}$.

Driven by these insights, we design an execution-cost model based on Tensor Core MMA operations. For each active sparse block, query-key matrix multiplications are decomposed into hardware tile-level MMA operations. The total MMA workload for head h is formulated in Equation 1.

$$C_h^{MMA} = \sum_{i,j} \mathbb{I}(M_h^{i,j} \neq 0) \cdot \frac{B_q B_k d}{mnk} \tag{1}$$

where $M_h^{i,j}$ indicates the i-th query block and j-th key block in head h; B_q, B_k, and d denote query block size, key block size, and head dimension, respectively. m, n, k are the Tensor Core tile dimensions for a single MMA instruction.

Algorithm 1 Unified Workload Partitioning

Input : Scheduling units $\mathcal{U}$ with sparse attention masks $\mathcal{M}$; Number of GPUs G; MMA tile size (m, n, k)
Output : GPU-wise partition $\mathcal{P}$

Function `EstimateCost`(M_u)**:**
  `// Count active KV blocks`
  $N_u^{blk} \leftarrow countActiveBlocks(M_u)$
  `// Estimate MMA workload`
  $C_u^{MMA} \leftarrow N_u^{blk} \cdot \frac{B_q B_k d}{mnk}$
  return C_u^{MMA}

`// Estimate workload of each scheduling unit`
foreach $u \in \mathcal{U}$ **do**
  $unitLoad[u] \leftarrow$`EstimateCost`(M_u)

`// Sort units by decreasing workload`
$sortedUnits \leftarrow sortDescending(\mathcal{U}, unitLoad)$
`// Initialize GPU loads`
for $g \leftarrow 0$ **to** $G-1$ **do**
  $gpuLoad[g] \leftarrow 0$ $\mathcal{P}[g] \leftarrow \emptyset$

`// Greedy LPT assignment`
foreach $u \in sortedUnits$ **do**
  $g^* \leftarrow argmin(gpuLoad)$
  $\mathcal{P}[g^*] \leftarrow \mathcal{P}[g^*] \cup \{u\}$
  $gpuLoad[g^*] \leftarrow gpuLoad[g^*] + unitLoad[u]$

return $\mathcal{P}$

Algorithm 2 Asynchronous Pipelined CP

Input : Local query blocks $\mathbf{Q}_{local}$; Remote KV requests $\mathcal{K}$; Micro-batch size B_μ
Output : Attention output $\mathbf{O}$

`// Partition KV requests and allocate double buffer`
$\mathcal{B} \leftarrow$ partition$(\mathcal{K}, B_\mu)$
buffer[0], buffer[1] $\leftarrow$ allocateKVBuffer()

`// Initiate first KV fetch`
$\text{handle}_0 \leftarrow$ AsyncPull$(\mathcal{B}[0]$, buffer[0]$)$

`// Pipeline: overlap communication with computation`
for $i \leftarrow 0$ **to** $|\mathcal{B}| - 1$ **do**
  $cur \leftarrow i \bmod 2$
  $next \leftarrow (i+1) \bmod 2$
  `// Prefetch next micro-batch`
  if $i + 1 < |\mathcal{B}|$ **then**
    $\text{handle}_{i+1} \leftarrow$ AsyncPull$(\mathcal{B}[i+1]$, buffer$[next])$
  `// Wait for current batch to arrive`
  waitEvent(handle_i)
  `// Compute attention on current buffer`
  $\mathbf{O}_i \leftarrow$ SparseAttention$(\mathbf{Q}_{local}$, buffer$[cur])$

`// Aggregate partial attention outputs`
$\mathbf{O} \leftarrow$ Aggregate$(\{\mathbf{O}_i\})$
return $\mathbf{O}$

3.2.2 Unified Workload Partitioning.

The execution-cost model provides a unified metric to balance sparse attention across both HP and CP dimensions. Along the HP dimension, conventional HP assigns an equal number of attention heads to each GPU, causing imbalance when sparse activation varies across heads. VISPI treats each head h as a scheduling unit and uses its estimated cost C_h^{MMA} to guide partition, assigning heavier heads first to equalize GPU workloads.

Similarly, standard CP splits queries into equal-length shards regardless of sparse KV block counts. VISPI instead treats each query block as a scheduling unit, estimating its cost by aggregating the MMA operations of its active KV blocks and constructing non-uniform, load-balanced query shards.

Both HP and CP partitions adopt the Longest Processing Time (LPT) greedy algorithm Graham (1969). Scheduling units are sorted by descending estimated cost and assigned to the GPU with the minimum accumulated workload, as detailed in Algorithm 1.

3.3 Asynchronous KV Streaming

While workload-aware partitioning determines where sparse computation occurs, CP execution still requires bringing the sparse, demand-driven KV data to the computing GPU. Conventional CP relies on blocking collective operations such as All-to-All to exchange KV data before computation begins.

As shown in Figure 3(a), this blocking approach enforces global synchronization across GPUs and creates a strict communication barrier. Attention computation remains stalled until the entire KV transfer completes, incurring substantial GPU idle time. This bottleneck becomes especially severe in sparse attention where GPUs only require a fraction of remote KV blocks.

To eliminate this barrier, we propose an asynchronous, one-sided pipelined streaming mechanism for sparse CP. It combines asynchronous one-sided KV retrieval to bypass global synchronization with a micro-batch streaming scheduler to achieve fine-grained communication-computation overlap.

Asynchronous KV Retrieval. We introduce an on-demand, one-sided direct-pull mechanism based on NVSHMEM for asynchronous KV exchange. Each GPU independently retrieves required remote KV blocks via direct memory access operations without global synchronization barriers.

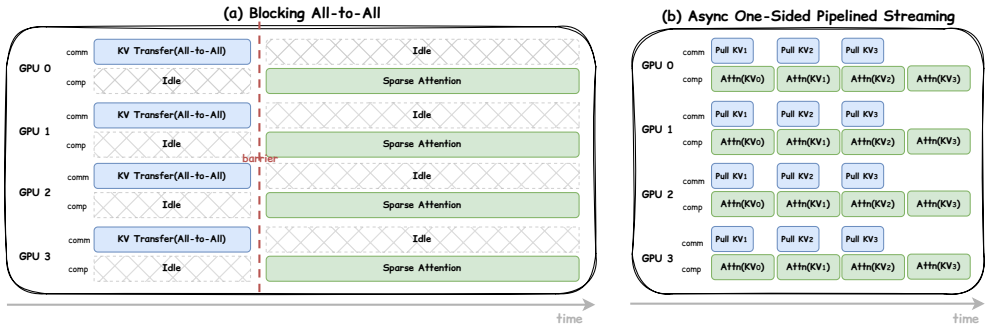

Figure 3: Execution timeline comparison. (a) Blocking All-to-All communication forces GPUs to remain idle during the entire KV exchange phase, creating a strict sequential dependency between communication and computation. (b) Our asynchronous one-sided pipelined streaming partitions KV transfer into micro-batches and overlaps communication with attention computation, hiding data transfer latency behind useful work.

Algorithm 2 summarizes this retrieval procedure. Given the sparse attention requests, the scheduler partitions them into micro-batches and initializes double-buffered KV storage. Each GPU then issues asynchronous one-sided fetches into local buffers, allowing remote data acquisition to proceed concurrently with other operations.

Equation 2 formalizes the communication latency of this one-sided pull mechanism:

$$T^d_{\text{pull}} = \alpha + \frac{\sum_{j \in \mathcal{N}_d} |\mathcal{K}_{d,j}| \cdot B_{\text{kv}} \cdot d_{\text{head}}}{\beta} \tag{2}$$

where $\mathcal{N}_d$ is the set of remote GPUs providing KV blocks to GPU d, $\mathcal{K}_{d,j}$ is the number of KV blocks requested from GPU j, α is the aggregated startup latency, β is effective bandwidth, B_{kv} is KV block size, and d_{head} is the head dimension. Compared to dense CP, total data transfer is reduced proportionally to attention sparsity by fetching only the required KV blocks instead of complete sequence partitions.

Pipelined Communication-Computation Overlap. Although one-sided pull eliminates global synchronization barriers, waiting for all remote KV blocks before starting computation still introduces latency. To hide this delay, we design a micro-batch streaming scheduler using double buffering, as illustrated in Figure 3(b). While the sparse attention kernel computes over the current micro-batch in one buffer, the next micro-batch is concurrently retrieved into the alternate buffer via asynchronous one-sided access.

Algorithm 2 details this pipelined execution. The scheduler continuously prefetches upcoming KV micro-batches into the idle buffer during kernel computation, and subsequently aggregates partial outputs across micro-batches to compute the final attention result.

Equation 3 formalizes the total pipeline execution latency:

$$T^d_{\text{pipeline}} = \max\left(T^d_{\text{pull}}, T^d_{\text{comp}}\right) + \epsilon \tag{3}$$

where T^d_{comp} is the sparse attention computation time on GPU d, and ϵ represents pipeline startup and drain overheads. By hiding communication behind computation, this approach reduces total execution latency from the sequential sum $T^d_{\text{pull}} + T^d_{\text{comp}}$ required by blocking communication down to the dominant component.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Hardware. We conduct experiments on two GPU clusters: 8× NVIDIA A800-SXM4 with 400 GB/s NVLink bandwidth, and 8× NVIDIA H20-3e with over 900 GB/s NVLink bandwidth. These two platforms represent contrasting compute-to-communication characteristics, allowing us to evaluate performance under both communication-constrained and communication-rich settings.

Models and Workloads. We evaluate our method on two official video DiT models with 3D full self-attention: Wan 2.1 1.3B (30 layers, 12 heads, $d_{\text{model}} = 1536$) and Wan 2.1 14B (40 layers, 40 heads, $d_{\text{model}} = 5120$) Wan et al. (2025). Both operate at 720P resolution using a 3D VAE ($4 \times 8 \times 8$ compression) and a patch embedding size of $(1, 2, 2)$, yielding $3{,}600$ spatial tokens per latent frame. We evaluate video generation across 81, 129, 257, and 401 frames, corresponding to context lengths of 76K, 119K, 234K, and 364K latent tokens. All 946 text prompts from the VBench suite Huang et al. (2024), spanning 16 evaluation dimensions, are used for inference workloads. All experiments use official pretrained checkpoints without fine-tuning.

Sparsity Source. All distributed baselines share the k-means-based semantic clustering of SVG2 Yang et al. (2026b) as their upstream sparsity identification module, which produces per-head binary sparsity masks at each denoising step. This ensures that any performance differences reflect the effectiveness of the distributed execution strategy rather than the quality of sparsity identification.

Baselines. We compare VISPI against three representative baselines: 1) **xDiT** Fang et al. (2024), a production-grade framework based on hybrid HP×CP parallelism and dense Ring Attention representing conventional dense-oriented inference; 2) **DSV-CP** Tan et al. (2026), an inference adaptation of DSV's sparsity-aware CP strategy featuring sparsity-based head rebalancing and selective KV gathering; 3) **SVG2** Yang et al. (2026b), the single-GPU sparse inference pipeline used to quantify the speedup that distributed execution provides over optimized single-device sparse attention. For distributed methods, we evaluate different HP/CP configurations and report the best-performing configuration for each method.

4.2 INFERENCE PERFORMANCE

Figure 4 reports the per-step denoising latency of all methods across representative Video DiT workloads on 8 GPUs. Several observations emerge.

First, distributed execution provides substantial benefits over single-GPU sparse inference. Although SVG2 significantly reduces the attention computation by exploiting sparse patterns, its single-GPU execution remains constrained by the massive computation of long video sequences. VISPI reduces latency by up to 7.1× over SVG2 on Wan 2.1 14B and 6.7× on Wan 2.1 1.3B, confirming that multi-GPU execution is essential for practical long-context video DiT inference.

Second, dense-oriented parallelism is fundamentally misaligned with sparse workloads. xDiT, despite providing an effective hybrid HP×CP strategy for dense DiT inference, suffers from two mismatches under sparse attention: uniform head partitioning causes severe workload imbalance due to heterogeneous per-head computation, while dense Ring Attention incurs unnecessary KV exchange for interactions that sparse attention has already pruned. VISPI resolves these issues through MMA-aware head scheduling and asynchronous sparse KV streaming, achieving 1.15–1.96× speedup over xDiT across all configurations.

Third, sparsity-aware optimization alone is insufficient without hardware-aware execution modeling. DSV-CP improves over xDiT through sparsity-based head rebalancing and selective KV gathering, but its workload estimation remains at the sparsity-ratio level and its communication still imposes synchronization between KV exchange and attention computation. VISPI further reduces latency by 1.06–1.12× over DSV-CP by replacing sparsity-statistic workload estimation with MMA-count-based modeling and by overlapping communication with computation through pipelined streaming. This demonstrates that effective sparse execution requires not only recognizing that workloads are sparse, but also understanding how sparse operations map onto GPU hardware.

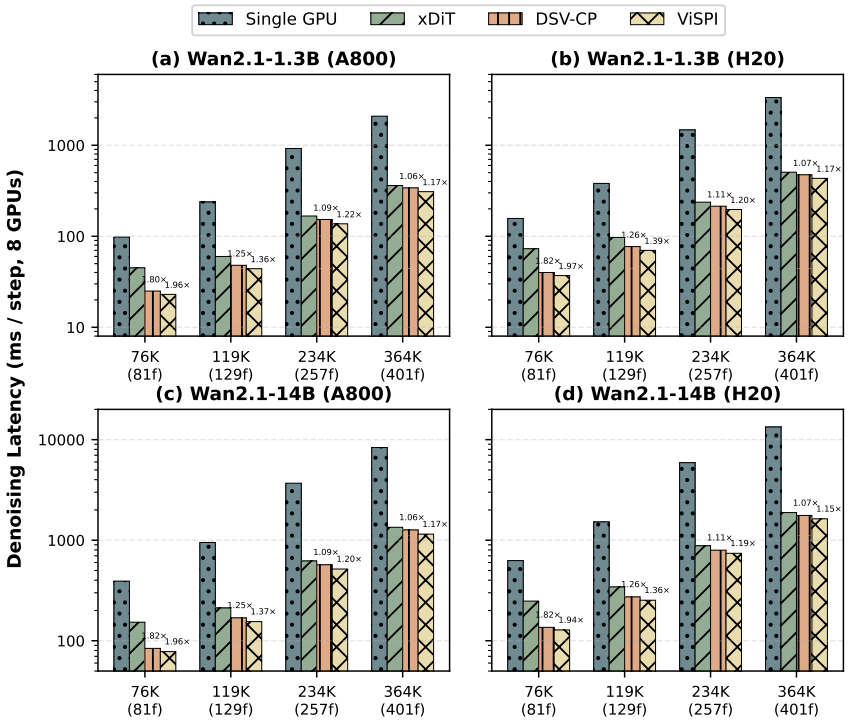

Figure 4: Per-step denoising latency across sequence lengths on 8 GPUs . Subplots (a)–(b) evaluate Wan2.1-1.3B and (c)–(d) evaluate Wan2.1-14B on NVIDIA A800 and H20 platforms, respectively. Numbers above bars indicate speedup over xDiT.

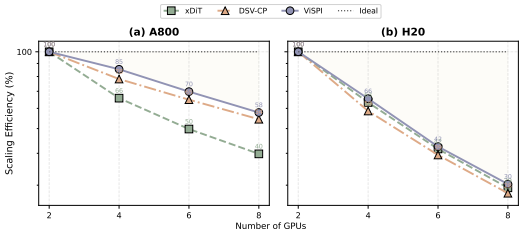

Figure 5: Strong scaling efficiency of ViSPI with Wan 2.1 on A800 and H20 platforms. Efficiency is reported relative to the 2-GPU configuration; 100% represents ideal linear scaling.

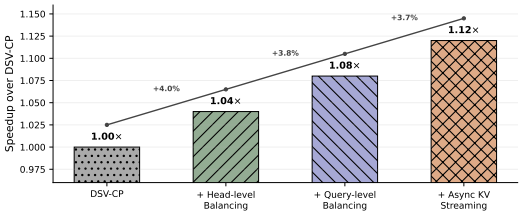

Figure 6: Component contribution analysis. Starting from DSV-CP, each bar incrementally enables one design component of ViSPI. All configurations use identical SVG2 sparsity masks.

Overall, these results confirm that ViSPI effectively bridges the gap between sparse attention computation and distributed execution, consistently achieving the lowest latency across diverse Video DiT workloads.

4.3 Scalability

We evaluate scalability on two GPU platforms with different compute-to-communication characteristics. A800 represents a communication-constrained setting with lower NVLink bandwidth, whereas H20 provides higher interconnect bandwidth to evaluate scalability under different hardware bottlenecks.

Figure 5 evaluates the strong scaling efficiency of ViSPI on Wan 2.1 by increasing the GPU count from 2 to 8 while maintaining a fixed inference workload. We report scaling efficiency with respect to the 2-GPU configuration, where ideal linear scaling corresponds to 100% efficiency.

VISPI consistently achieves the highest scaling efficiency on both platforms. On the A800 cluster, VISPI maintains 58% efficiency at 8 GPUs, compared with 39% for xDiT and 54% for DSV-CP. Similar trends are observed on H20, where higher interconnect bandwidth reduces communication pressure but does not eliminate the benefits of VISPI's optimizations.

These results highlight the complementary nature of VISPI's two design components. Workload-aware head scheduling mitigates the increasing computation imbalance as the HP degree grows, while communication-computation pipelining effectively hides KV transfer overhead as the CP degree increases. Together, these mechanisms enable VISPI to scale more efficiently than existing frameworks across different hardware environments.

4.4 Inference Quality Preservation

All distributed methods use identical per-head sparsity masks generated by SVG2 and perform mathematically equivalent sparse attention computation. VISPI's head scheduling and CP communication only redistribute computation and reorganize KV transfer across GPUs without altering the attention outputs. Consequently, VISPI preserves exactly the same sparse attention results as the underlying SVG2 execution.

Following SVG2 Yang et al. (2026b), we verify that all distributed methods produce identical video outputs. Compared with the single-GPU SVG2 execution, the videos generated by VISPI achieve a PSNR exceeding 45 dB across all evaluated models. This negligible discrepancy arises solely from minor floating-point reordering during parallel execution, confirming that VISPI introduces no quality degradation while providing substantial speedup.

4.5 Ablation Analysis

Figure 6 isolates the contribution of each component by progressively enabling VISPI's three key designs on top of DSV-CP.

Starting from DSV-CP (1.00×), enabling head-level workload balancing improves the performance to 1.04×, demonstrating that execution-aware cost estimation further reduces head-level stragglers beyond sparsity-based balancing. Adding query-level workload balancing further improves the performance to 1.08×, showing that sparse attention also introduces computation imbalance across context partitions. Finally, asynchronous KV streaming increases the performance to 1.12× by overlapping irregular KV transfers with attention computation.

The three components are complementary: 2D workload balancing mitigates computation imbalance across parallel dimensions, while asynchronous streaming hides communication stalls that cannot be eliminated by workload balancing alone.

5 Conclusion

In this work, we presented VISPI, a distributed inference framework that redesigns hybrid HP×CP execution for sparse video Diffusion Transformers. By unifying execution-cost modeling across both head and context dimensions and coupling this balanced partitioning with asynchronous, one-sided pipelined KV streaming, VISPI resolves the fundamental mismatches between dynamic sparsity and traditional dense-oriented parallelism strategies. Extensive evaluations demonstrate that VISPI achieves substantial end-to-end speedups over state-of-the-art baselines while fully preserving generation quality.

More fundamentally, our findings reveal that integrating dynamic sparsity into distributed attention is not merely an algorithmic optimization. Instead, sparsity alters the core execution characteristics of distributed inference. By introducing irregular workload density across parallel axes alongside demand-driven data accesses, sparsity invalidates classical assumptions of uniform partitioning and monolithic bulk exchanges. Scalable sparse inference therefore demands a systemic redesign that harmonizes execution-aware scheduling with asynchronous communication primitives.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and grammar correction of the manuscript. We have not used generative AI tools for generating research ideas, experimental results, or figures. We have reviewed all AI-assisted text edits and take full responsibility for the final content of this work.

REFERENCES

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Leo Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. *OpenAI Blog*, 1(8):1, 2024.

Jiarui Fang and Shangchun Zhao. Usp: A unified sequence parallelism approach for long context generative ai. *arXiv preprint arXiv:2405.07719*, 2024.

Jiarui Fang, Jinzhe Pan, Xibo Sun, Aoyu Li, and Jiannan Wang. xdit: an inference engine for diffusion transformers (dits) with massive parallelism. *arXiv preprint arXiv:2411.01738*, 2024.

Jiarui Fang, Jinzhe Pan, Aoyu Li, Xibo Sun, and WANG Jiannan. Pipefusion: Patch-level pipeline parallelism for diffusion transformers inference. *Advances in Neural Information Processing Systems*, 38:98434–98455, 2026.

Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429, 1969.

Diandian Gu, Peng Sun, Qinghao Hu, Ting Huang, Xun Chen, Yingtong Xiong, Guoteng Wang, Qiaoling Chen, Shangchun Zhao, Jiarui Fang, et al. Loongtrain: Efficient training of long-sequence llms with head-context parallelism. *arXiv preprint arXiv:2406.18485*, 2024.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems*, 5:341–353, 2023.

Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Kai Li, and Song Han. Distrifusion: Distributed parallel inference for high-resolution diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7183–7193. IEEE, 2024.

Shenggui Li, Hongxin Liu, Zhengda Bian, Jiarui Fang, Haichen Huang, Yuliang Liu, Boxiang Wang, and Yang You. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, pp. 766–775, 2023.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ringattention with blockwise transformers for near-infinite context. In *International Conference on Learning Representations*, volume 2024, pp. 3992–4008, 2024.

Yijun Ma, Siyuan Shen, Tiancheng Chen, Akhil Langer, Jiri Kraus, Benjamin Glick, Craig Belusar, Jeff Hammond, and Torsten Hoefler. Demystifying nvshmem: A system-level analysis on symmetric memory and device-initiated operations in gpu communication. *arXiv preprint arXiv:2606.05951*, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Xin Tan, Yuetao Chen, Yimin Jiang, Xing Chen, Kun Yan, Nan Duan, Yibo Zhu, Daxin Jiang, and Hong Xu. Dynamic sparsity in large-scale video dit training. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 101–116, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Jiacheng Yang, Jun Wu, Yaoyao Ding, Zhiying Xu, Yida Wang, and Gennady Pekhimenko. Swiftfusion: Scalable sequence parallelism for distributed inference of diffusion transformers on gpus. In *Proceedings of the ACM Conference on AI and Agentic Systems*, pp. 1037–1050, 2026a.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, et al. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *Advances in Neural Information Processing Systems*, 38:96965–96991, 2026b.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. In *International Conference on Learning Representations*, volume 2025, pp. 3296–3319, 2025.