A NVIDIA H100 SXM Roofline Parameters

We derive the roofline parameters for the NVIDIA H100 SXM GPU. The dense FP16 throughput is 990 TFLOP/s and GPU memory bandwidth is 1.5 TB/s, which are derived from empirical measurements and hardware specifications [5]. The ridge arithmetic intensity is

$$\mathrm{AI}_{\mathrm{ridge}} = \frac{990\ \mathrm{TFLOP/s}}{1.5\ \mathrm{TB/s}} = 660\ \mathrm{FLOP/Byte} \quad (5)$$

B Hardware Intensity Analysis and Configuration

Consider a DiT backbone with hidden dimension C, query length L_q, KV-cache length L_{kv} (assuming full KV read/write each step), batch size b, and N layers. The total compute cost F is defined as:

$$F = N\left(24bL_qC^2 + 4bL_qL_{kv}C\right) \quad (6)$$

For memory movement B, we account for: (i) activations, (ii) model weights, and (iii) KV-cache traffic. Furthermore, we include additional system-level memory operations, such as sliding-window buffering and feature concatenation. This yields:

$$B = N\left(24C^2 + 4bL_{kv}C + 4bL_qC + 36bL_qC\right) + b\gamma \quad (7)$$

where γ captures the extra memory movement beyond the standard DiT read/write pattern. The normalized hardware intensity is then defined as:

$$\text{Intensity} = \frac{F/\text{PeakCompute}}{B/\text{PeakBandwidth}} \quad (8)$$

Using a practical configuration on an NVIDIA H100 with 480p video input, we set $C = 2,048$, $L_q = 1,536$, $L_{kv} = 1,536 \times 6$, and $N = 30$. Importantly, using NVIDIA Nsight Systems, we directly measure the total memory traffic during DiT inference—corresponding to γ—to be 5.2 GiB, while the realized device bandwidth is approximately 1.5 TB/s.

The practical intensity is calculated as follows (using FP16 dense peak $\approx$ 990 TFLOPS as an upper bound, noting that the effective compute peak is lower in practice):

$$\text{Intensity} \approx \frac{F/990\,\text{TFLOPS}}{B/1.5\,\text{TB/s}} \approx 0.84 < 1 \quad (9)$$

This result indicates a **memory-bound regime**. This conclusion is consistent with our empirical observations in Fig. 15 (batch denoising), where performance scales with memory bandwidth rather than peak computation.

[5] https://resources.nvidia.com/en-us-gpu-resources/h100-datasheet-24306

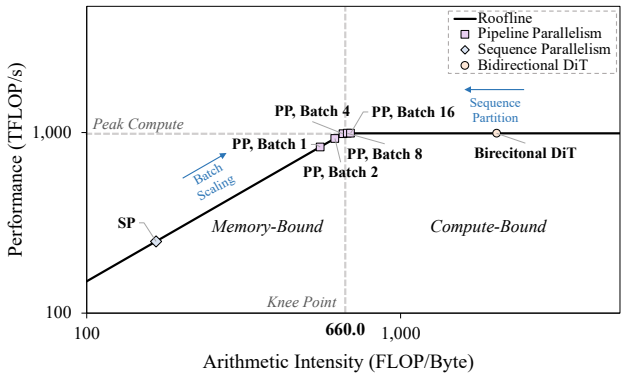

Figure 19. **Roofline analysis of sequence parallelism and our pipeline orchestration.** We compare the Sequence Parallelism and Pipeline Parallelism under varying batch sizes in the causal DiT, compared with the bidirectional DiT. The results demonstrate that our approach operates near the knee point of the roofline, effectively avoiding compute under-utilization as seen in the bidirectional DiT and memory bandwidth limitations in Sequence Parallelism. The model is profiled on an NVIDIA H100 SXM GPU with a peak performance of 990 TFLOP/s and a knee point at an arithmetic intensity (AI) of 660.0 FLOP/Byte. The token length is 1,536.

C REPA finetuning for quality enhancement

To enhance the quality of generated videos, we employ the REPA (Yu et al., 2025) training strategy during Causal-DiT distillation (Yin et al., 2025). The alignment loss is denoted by

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

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

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

where λ is a scaling factor.

D Figure illustration

Figure overview. Fig. 20 shows how we minimize pipeline bubbles by separating compute and communication into two concurrent streams, overlapping kernels with P2P transfers.

Fig. 21 illustrates the rolling KV cache with sink tokens for consistent, frame-by-frame updates.

Fig. 22 reports a comparison of parallelization strategies (e.g., sequence parallelism vs. our pipeline scheduler).

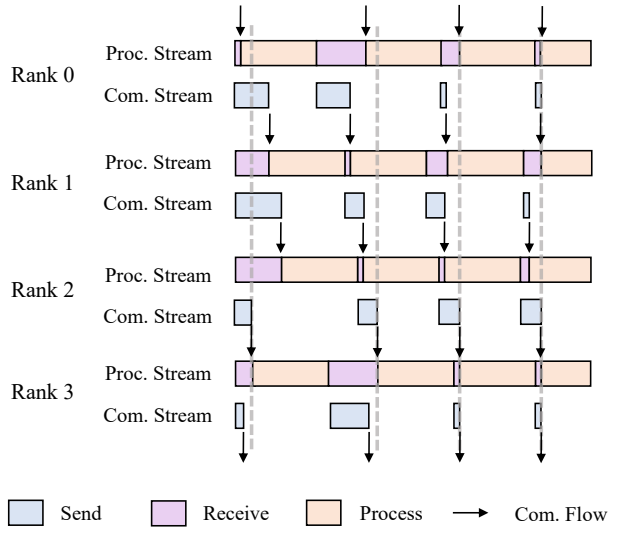

Figure 20. **Execution timeline of the Pipeline-orchestration architecture.**

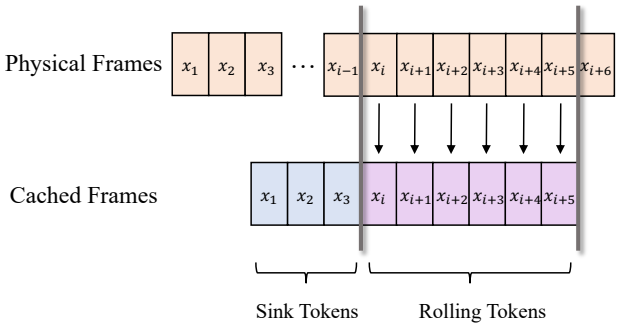

Figure 21. **The detailed illustration of the Rolling KV Cache and Sink Token designs.**

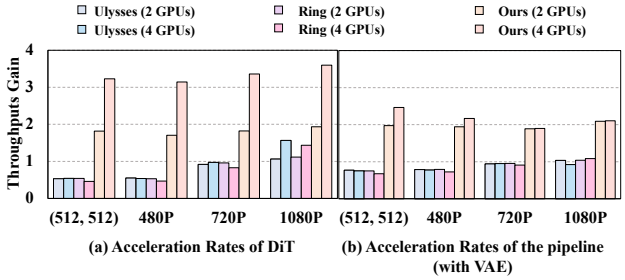

Figure 22. **Acceleration rate of different approaches on various resolutions.** *Left*: Testing the acceleration rate of DiT only. *Right*: Testing the acceleration of the whole pipeline (with VAE).