GPU SMITH
COMPUTE ENGINEERING

GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

Multimodal Inference Vision Encoder Disaggregation Guide

> Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-11-21 · multimodal inference · vision encoder disaggregation · encode prefill decode disaggregation · epd disaggregation · nvidia dynamo · vllm disaggregated encoder · vlm serving · gpu capacity planning · nixl

Original article: https://gpusmith.com/articles/en/multimodal-inference-split-vision-encoding-prefill-decode



In this article

1. Executive Summary
2. Introduction and Background
3. Key Changes: Placement Topologies and Where Latency Moves
4. Capacity Model: Encoder Load, Embedding Traffic and Same-SLO Break-Even
5. Implementation Considerations and Process Changes
6. Data Analysis and Evidence
7. Cases Where Aggregated Serving Wins
8. Implications and Future Directions
9. Frequently Asked Questions (FAQs)
10. Conclusion

Executive Summary

Encode-prefill-decode (EPD) disaggregation moves the vision encoder of a multimodal model out of the worker that runs LLM prefill and decode. **NVIDIA's Dynamo study**, published on September 9, 2026 (developer.nvidia.com), describes three placements: aggregated, a colocated encoder that shares each GPU with a prefill-decode (PD) worker, and a disaggregated encoder on a separate lower-cost GPU tier. Earlier academic work on the idea separates the encoding, prefill, and decode stages onto dedicated resources (arxiv.org). The study is vendor-run and used one model family and one **goodput** SLO, defined as inter-token latency (ITL) under 100 ms (developer.nvidia.com).

In the ten-image, 1024-output-token test, time to first token (TTFT) **fell 58%** with the colocated encoder and 50% with the heterogeneous tier, and the heterogeneous tier served **70% more traffic** at the same SLO (developer.nvidia.com). The same post shows the limits: end-to-end gains narrowed from **20.3% to 5.2%** as output length grew, and colocation turned into a **2.5% regression** (developer.nvidia.com). NVIDIA left the homogeneous-GPU disaggregated setup out of its analysis and states, without presenting data, that it always underperforms colocation (developer.nvidia.com).

For a buyer, the useful question is not "does EPD help" but "how many extra same-SLO requests per hour pay for the encoder GPUs and the embedding transfer". This report supplies a break-even worksheet. With the study's 4 PD GPUs and 2 encoder GPUs, the tiered design breaks even when the encoder GPU costs up to **1.4 times a PD GPU**, assuming the 70% uplift reproduces on the buyer's model (analyst arithmetic). At illustrative on-demand prices from a public price list (an **HGX B200** instance at $7.69 and an **A6000** at $1.09 per GPU-hour, as of November 2026) (lambda.ai), the tier needs only **about 7% more goodput** to break even. That is a pricing illustration, not the study's hardware.

Embedding traffic is modest for small image counts. For a model with a **3072**-wide embedding in bfloat16, ten images at 256 tokens each produce about **15.7 MB** per request (analyst arithmetic from the model configuration) (huggingface.co), which is about **6.3 ms** of wire time at the study's measured 20 Gbps peak. The worksheet below treats that as a queueing input, not a rounding error.

Aggregated serving stays the right default when images are few, outputs are long, or the LLM is a large dense model. In the study, colocated goodput relative to aggregated was **2.62x, 1.50x and 0.65x** for 4B, 9B and **27B** models (developer.nvidia.com). Vendor results should not be copied across models. The recommended path is a pilot with mixed text and image traffic, at least two output-length bins, explicit **rollback** gates and the same SLO on both arms.

Introduction and Background

Private AI serving teams that move from text-only models to vision-language models (VLMs) meet a new cost center: every image must be decoded, resized, split into patches and passed through a vision transformer (ViT) before the language model can begin prefill. Per NVIDIA, the stack must preprocess media and run the ViT before prefill can start, and in aggregated serving that work shares one worker and one scheduling domain with prefill and decode (developer.nvidia.com).

This report answers a narrow buyer question: should a multimodal fleet keep requests aggregated, run a colocated vision-encoder worker beside each PD worker, or buy a cheaper encoder GPU tier? It is written for serving architects and GPU fleet buyers. It assumes familiarity with text prefill/decode (PD) disaggregation, which several research systems established. **DistServe** assigns prefill and decoding computation to different GPUs, hence eliminating prefill-decoding interferences (arxiv.org), and **Splitwise** reports 1.4x higher throughput at 20% lower cost than current designs (arxiv.org). Those results concern text. The EPD question is different because the encoder is a third workload with its own compute profile.

The academic precedent for a third stage is the EPD paper from Singh and co-authors, which separates the encoding, prefill, and decode stages onto dedicated resources (arxiv.org) and appeared in the Proceedings of the 42nd International Conference on Machine Learning (proceedings.mlr.press). Its abstract reports TTFT reductions of up to 71% reduction (arxiv.org) against systems that do not disaggregate, on different models and hardware from NVIDIA's study. The two sources should be read as independent evidence that the idea works under some conditions, not as a combined forecast.

GPU Smith is an independent private AI infrastructure consultancy that advises on GPU procurement, workload sizing, networking and inference optimization. It does not sell GPUs, cloud capacity or a serving engine, and this report does not rely on any private measurements. Every benchmark figure below is attributed to its public originator, and every worked number labeled as an example is analyst arithmetic on stated assumptions.

The primary benchmark source is one vendor post, backed by **Dynamo** and vLLM documentation, systems papers and public price lists. Prices are stated as of November 2026, and no result here should be read as a promised production uplift.

Key Changes: Placement Topologies and Where Latency Moves

The encoder as a separately placed worker

Dynamo's documentation defines the change in one sentence: Encoder disaggregation separates the vision encoder from the prefill/decode pipeline into its own worker (docs.nvidia.com). Its listed benefits include independent scaling of encode workers and reduced GPU memory pressure on the prefill/decode workers. The vLLM documentation adds that vision encoders are lightweight, while language models are orders of magnitude larger (docs.vllm.ai).

Table 1 below maps the three topologies the NVIDIA study tested, plus one untested variant (homogeneous disaggregation). The study's test bed used one tensor-parallel-1 worker per GB200 for aggregated serving, two encoder workers plus one PD worker per GB200 for colocated serving, and RTX nodes as the encoder tier with GB200 GPUs as PD for the disaggregated case.

Topology	What runs where	Extra GPUs	Transfer path	Documented fit
Aggregated	Vision encoding, prefill and decode in one worker per GPU, the monolithic design the EPD paper describes (arxiv.org)	None	None	Light media, long decode, simple operations
Colocated encoder	One or more encoder workers beside one PD worker on each GPU, separate queues and batching	None (shares PD GPUs)	Embeddings move between processes on the same GPU	Homogeneous clusters, image-heavy and mixed traffic
Disaggregated encoder (heterogeneous tier)	Encoder workers on a separate, lower-cost GPU tier; PD on the primary tier	Yes: the encoder tier	Network, over **NIXL**; the study's measured peak was 20 Gbps (see Step 3)	Clusters that already own a cheaper GPU class; image-heavy workloads
Disaggregated encoder (homogeneous)	Dedicated same-class GPUs for encoding	Yes	Network	Vendor advises against it and did not test it; NVIDIA says it always underperforms colocation (developer.nvidia.com)

Table 1. Placement matrix drawn from NVIDIA's EPD study and Dynamo documentation. Gains are vendor-measured on one model.

Splitwise already explored per-phase hardware choice for text, designing clusters using the same or different types of machines for the prompt computation and token generation phases (arxiv.org). The matrix shows why the hardware inventory comes before the benchmark. If every GPU in the fleet is the same class, the study's own recommendation is colocation, because the vision encoder is light relative to the LLM and a reserved same-class GPU would sit underused. The dedicated tier only enters the choice when a cheaper GPU class is already available or purchasable.

Backend support levels

Support is not uniform across engines. Dynamo's support matrix (docs.nvidia.com) lists:

- **vLLM:** both E/PD (a separate encoder with combined prefill and decode) and E/P/D (encoder, prefill and decode all separate). Buyers should check one limit: the separate encode worker handles images, while video inputs stay on the prefill/PD path (docs.nvidia.com).
- **TensorRT-LLM:** E/P/D only.
- **SGLang:** both patterns.

In Dynamo's E/P/D flow, the encode worker transfers embeddings over NIXL to the prefill worker (docs.nvidia.com).

NIXL, the transfer library, is described in its repository as targeted for accelerating point to point communications in AI inference frameworks such as NVIDIA Dynamo (github.com). The vLLM documentation names a different embedding path for its own connector layer: transfers encoder outputs using the Mooncake TransferEngine (docs.vllm.ai), and describes the abstraction as a connector transfers encoder-cache (EC) embeddings from the encoder instance to the PD instance (docs.vllm.ai). Buyers choosing between Dynamo and a native vLLM deployment therefore choose a transport as well as a scheduler.

The vLLM EPD tracking issue shows the area is moving quickly, including a sizing item to choose an E:PD GPU split using measured worker capacity (github.com). The practical reading is that transfer shapes are model-specific and should be measured, not assumed.

Symptoms that signal an encoder bottleneck

The NVIDIA post reports that vision encoding can take hundreds of milliseconds or longer, that a media-heavy request can delay its own prefill because encoder and LLM work share the same GPU, and that text-only requests can queue behind multimodal ones (developer.nvidia.com). These produce four observable symptoms:

- **TTFT grows with image count.** In the study's 5-to-50-image sweep, aggregated TTFT rose steeply with **visual tokens** while both EPD topologies stayed nearly flat (Table 2).
- **Text TTFT degrades when image traffic arrives.** This is **head-of-line blocking**, the same mechanism that chunked prefill addresses for long text prompts: **Sarathi-Serve** splits a prefill request into near equal sized chunks (arxiv.org) to keep decodes from stalling.
- **ITL jitter may appear on decode (analyst inference, not a reported result).** The post reports no ITL jitter measurements; because encoder and decode work share GPU compute when colocated, jitter is plausible, which is consistent with the study's choice of an ITL target as its goodput SLO.
- **Encoder queue time dominates image-request TTFT.** Instrument it separately from prefill time; the pilot sheet in a later section shows how.

Not all TTFT is encoder time. The post found that 43% of TTFT was spent before ViT even started (developer.nvidia.com), and EPD does not touch those stages, which include media download and decode.

Preprocessing placement as a separate lever

Dynamo's **frontend media decoding** moves decode out of the model worker: the frontend decodes supported images and videos on a CPU worker pool and transfers the decoded pixel buffers (docs.nvidia.com) to the backend. The same page notes that it does not skip vision encoding. In an encoder-only benchmark at 30

images of 256 tokens on one GB200 with two encode workers, the NVIDIA post reports that parallel decoding cut mean encoder request latency 26%, from 281.3 ms to 207.0 ms (developer.nvidia.com).

Image-token budgets and encoder load

Encoder work scales with visual tokens, and token budgets differ by model. Qwen2-VL introduced a mechanism that lets the model dynamically process images of varying resolutions into different numbers of visual tokens (arxiv.org). Claude's public documentation states that each patch is a 28×28-pixel block of the image, referred to as a visual token (platform.claude.com) and that High-resolution images can use up to roughly three times more visual tokens than the same image on a standard-tier model (platform.claude.com). Google's Gemma documentation states that each image typically uses 256 tokens (ai.google.dev). The NVIDIA study capped images at 256 tokens in the ten-image test and used 128 or 256 token budgets in the sweep. The test is a tidy benchmark; real document or screenshot traffic can be several times heavier per image, so the token budget is the first input to measure in the worksheet.

Capacity Model: Encoder Load, Embedding Traffic and Same-SLO Break-Even

The capacity model has four parts: encoder fleet size, embedding bytes on the wire, transfer queueing, and cost at a fixed goodput. Goodput here follows the definition DistServe popularized: the maximum rate that can be served within both TTFT and TPOT constraints on each GPU (www.usenix.org). DistServe also chooses placement to minimize the communication caused by disaggregation (www.usenix.org). NVIDIA's study uses a simpler single constraint: inter-token latency (ITL) under 100 ms (developer.nvidia.com). A buyer should record every latency target that the product requires, then find the highest request rate at which all hold, per arm, using the same load generator.

Step 1: encoder capacity per request

Define four inputs:

- `lambda_r`: arrival rate of image-bearing requests.
- `n`: images per request.
- `mu_img`: measured encoder throughput in images per second at an SLO-safe batch size on one encoder GPU.
- `rho_t`: target encoder utilization.

The encoder fleet is then `N_E = ceil(lambda_r * n / (mu_img * rho_t))`. The throughput `mu_img` must be measured on the buyer's model and GPU, because the NVIDIA post gives no per-GPU encoder throughput, only latency in an encoder-only test.

Utilization matters because encoder queues follow ordinary queueing behavior. Under a simple **M/M/1 approximation** (a single server with random arrivals, analyst arithmetic, not a vendor result), mean waiting time in service-time units is `rho / (1 - rho)`: 1.5 at 60% utilization, 2.33 at 70%, 5.67 at 85% and 9.0 at

90%. Real encoders batch, so the true curve differs, but the shape explains why the worksheet sizes for 60 to 70% rather than for peak throughput. Sizing for 60% also protects p95 TTFT against bursts of image-heavy requests.

Step 2: embedding bytes per request

The bytes transferred per image request are `n * t * h * b`, where `t` is visual tokens per image, `h` the embedding width and `b` bytes per element. For the Qwen3.5 122B A10B NVFP4 checkpoint used in the study, the published configuration lists a text hidden size of 3072 and a vision output hidden size of 3072 in bfloat16 (huggingface.co). With ten images at 256 tokens, the total is 10 x 256 x 3072 x 2 bytes = 15,728,640 bytes, or about 15.7 MB per request.

Table 2 below summarizes the NVIDIA results next to the per-request embedding bytes they imply, so the vendor's reported gains and the network cost sit side by side. The byte figures are analyst arithmetic for a 3072-wide bfloat16 embedding.

Workload (study setting)	Reported effect (vendor-measured, one model)	Embedding bytes per request (analyst arithmetic)
10 images, 256 tokens, OSL 1024	TTFT down 58% (colocated) and 50% (heterogeneous); heterogeneous tier served 70% more traffic at ITL under 100 ms (docs.nvidia.com)	15.7 MB
5 images, OSL 128 to 2048	Heterogeneous end-to-end gain narrowed from 20.3% to 5.2%; colocated moved from an 11.8% gain to a 2.5% regression (developer.nvidia.com)	7.9 MB at 256 tokens per image
5 to 50 images, 128 or 256 token budgets	TTFT improvement of 25 to 93% for both EPD topologies across the grid (developer.nvidia.com)	39.3 MB for 50 images at 128 tokens
50:50 text and image mix, OSL 128, colocated	Text TTFT 92.3 to 53.3 ms (**42.2%** lower); image TTFT 289.9 to 200.6 ms (**30.8%** lower) (developer.nvidia.com)	1.6 MB per single 256-token image
4B, 9B and 27B models, colocated	Goodput versus aggregated 2.62x, 1.50x and 0.65x (docs.nvidia.com)	Depends on model width
BF16 versus NVFP4 LLM weights, colocated	Goodput versus aggregated **1.78x** versus **2.64x** (developer.nvidia.com)	Unchanged (encoder stays BF16)

Table 2. NVIDIA's reported results (Qwen3.5 family, four GB200 GPUs, optional RTX 6000D encoder tier, ITL under 100 ms goodput SLO) with the embedding payload each setting implies.

Three conclusions follow from the table. First, the largest per-request payloads (30 to 50 images) are also the settings where EPD helped most in TTFT, so the network cost and the benefit rise together. Second, the model-size and precision rows show the reported gains track the ViT-to-LLM compute ratio rather than EPD alone. Third, the 27B row is below 1.0, which is the clearest evidence that vendor uplift cannot be copied across models.

Step 3: wire time and link utilization

At the study's measured 20 Gbps peak (NIXL over UCX RC/TCP Ethernet for vision embedding transfer, measured peak 20 Gbps (developer.nvidia.com)), a 15.7 MB payload takes about 6.3 ms of wire time (15.7 MB x 8 / 20 Gbps). A link running at 100% would carry about **159 such requests per second**, 318 requests per second for 5-image payloads and 53 per second for 30-image payloads. Treat 50% link utilization as the planning ceiling, and apply the same `rho / (1 - rho)` shape: at 50% utilization the mean queueing delay equals one transfer time, and at 80% it is four. Both figures are analyst arithmetic on a single-queue assumption.

Wire time adds to TTFT only for requests that cross the network. For a colocated encoder the transfer stays on the node. For the heterogeneous tier, the encoder's embedding transfer adds roughly the wire time plus queueing plus any serialization cost the engine imposes. The NVIDIA post itself warns that the encode work being isolated must be large enough to offset worker coordination and embedding transfer overhead (developer.nvidia.com). On buyer hardware, measure transfer p50 and p95 with the real fabric, because the study's 20 Gbps TCP path is not a statement about InfiniBand or RoCE deployments, and Dynamo's documentation describes the transfer as to downstream workers via NIXL (RDMA) (docs.nvidia.com).

Step 4: cost at a fixed goodput

Define total GPU-hours as PD GPU-hours plus encoder GPU-hours. Let `N_PD` and `N_E` be the GPU counts, `c_PD` and `c_E` the hourly prices, and `g` the same-SLO goodput multiplier versus aggregated serving on the same PD budget. Cost per request versus the aggregated baseline is `(N_PD c_PD + N_E c_E) / (g N_PD c_PD)`. The tiered design breaks even when:

- **Break-even multiplier:** `g* = 1 + (N_E c_E) / (N_PD c_PD)`.
- **Maximum encoder GPU price:** `c_E,max = (N_PD / N_E) (g - 1) c_PD`.
- **Colocated case:** no extra GPUs, so `g* = 1`, before accounting for any slower decode on shared GPUs.

With the study's 4 PD GPUs and 2 encoder GPUs and its 70% uplift, `c_E,max = 2 x 0.7 x c_PD = 1.4 c_PD`. The encoder GPU can cost up to 1.4 times a PD GPU before the tier loses money, if the uplift holds. That is a generous bound, so a buyer should check it against an uplift discounted for their own model, for example `g = 1.2`, which gives `c_E,max = 0.4 c_PD`.

(Hypothetical Example) Use public on-demand prices as placeholders. Lambda's price list shows an NVIDIA HGX B200 instance at $7.69 per GPU-hour and an NVIDIA A6000 at $1.09 per GPU-hour (lambda.ai). These are not the study's hardware (GB200 PD and RTX 6000D encoders), and the A6000 is a stand-in for "a cheaper tier". Baseline: 4 x $7.69 = **$30.76** per hour. Tiered: $30.76 + 2 x $1.09 = **$32.94** per hour, so `g* = 1.071`. At a 1.7x uplift, cost per same-SLO request falls to 0.63 of baseline; at a discounted 1.2x uplift it falls to about **0.89 of baseline**; at 1.0x (no uplift) it rises 7.1%. For the colocated design at the 27B result of 0.65x, cost per request rises to **1.54x of baseline** (1 / 0.65), which is why the model-size row in Table 2 matters.

(Hypothetical Example) A fleet serves 40 requests per second, 25% with ten 256-token images, so 100 images per second arrive. If a pilot measures 60 images per second per encoder GPU and the plan targets 60% utilization, effective capacity is **36 images per second**, so `N_E = ceil(100 / 36) = 3`. Three encoder

GPUs at $1.09 cost $3.27 per hour, so `g* = (30.76 + 3.27) / 30.76 = 1.106`. Embedding traffic is 10 x 15.73 MB = 157 MB per second, about **1.26 Gbps**, or 6.3% of a 20 Gbps link: the encoder fleet, not the network, is the constraint.

Two limits apply. Cloud list prices change, and a private buyer's real costs include power, rack space and networking. Also, the **RTX PRO 6000 Blackwell Server Edition** is a plausible encoder-tier candidate, with 96 GB of GDDR7 memory (www.nvidia.com) and a passively cooled server design (www.nvidia.com), but the study used RTX 6000D parts and no public benchmark ties either card to this workload. The Lambda page retrieved for this report did not list an RTX PRO 6000 price.

Implementation Considerations and Process Changes

Pilot design

Dynamo's own guidance on when to adopt is conditional: use encoder disaggregation when vision encoding is a bottleneck and you need to scale encoders independently of LLM workers (docs.nvidia.com), or when you want the encoder on different hardware. A pilot should confirm that condition before spending on hardware. A defensible pilot has six steps:

- **Capture a traffic sample.** Record the distribution of images per request, pixel dimensions, output length, text-only share and arrival burstiness from production logs, not from a synthetic default.
- **Convert to visual tokens.** Run the model's own processor to get tokens per image, since budgets vary by model family (patch-based in some vendors' documentation, a fixed 256 tokens in others').
- **Run three arms on the same SLO.** Aggregated, colocated and (if a cheaper GPU class exists) heterogeneous, with identical engines, versions, prompts and load generator.
- **Sweep to find goodput.** Increase arrival rate until the SLO fails, using at least two output-length bins (for example about 128 and about 1024 tokens) and at least two image-count bins.
- **Include mixed traffic.** Send text-only and image requests together; report text TTFT separately, because the study's text-request gain (42.2%) is easy to miss in image-centric averages.
- **Price the result.** Apply the break-even formulas with the buyer's real hourly cost for each GPU class.

The reference to reproduce is the Dynamo repository's EPD experiment, whose README says the experiment compares aggregated and colocated EPD topologies for Dynamo with vLLM and **SGLang** (github.com). Two caveats follow from that sentence. The released sweep covers aggregated and colocated topologies, so the heterogeneous encoder-tier result in the blog post is not directly reproducible from that directory. The sweep also pins vLLM 0.26.0 and SGLang 0.5.16 (github.com) and its downloader fetches 50 images by default (github.com), which suggests small, fixed image sets. For capacity planning, replace them with images that match production dimensions.

The reproduction command in the repository names the model nvidia/Qwen3.5-122B-A10B-NVFP4 (github.com), which matches the post's model. A pilot on a different model should expect different ratios.

The measurement sheet

Table 3 below is a measurement sheet and rollback-gate template. Each row names a metric, how to compute it and a suggested gate. Gate values are analyst suggestions for a pilot, not vendor or standards thresholds.

Metric	How to measure	Suggested gate (analyst proposal)
TTFT p50 and p95, text-only	Client-side time to first streamed token for requests with no media	No worse than aggregated baseline at equal load; the study saw 42.2% lower mean text TTFT in mixed traffic
TTFT p50 and p95, image requests, by image-count bin	Same, stratified by images per request	At least the same-SLO result of aggregated at every bin; investigate any bin where p95 worsens
ITL p95 and p99	Gap between streamed tokens during decode	Within the product SLO (the study used under 100 ms) at the chosen load
E2E latency by output-length bin	Request start to final token	Roll back an arm that regresses at the long-output bin (the study's colocated arm regressed 2.5% there)
Goodput at SLO	Highest arrival rate where all latency targets hold for the required share of requests	At least g^* from the break-even formula, with margin
Encoder utilization and queue depth	Per-GPU busy time, encoder queue length, batch size	Target 60 to 70% mean; alert above 80%
Embedding transfer time p50 and p95	Timestamps at encoder send and PD receive, per request	p95 under a fixed share of TTFT budget (for example 10%)
Bytes per request and link utilization	Sum of embedding tensor sizes, link counters	Below 50% of the measured link peak
GPU-hours and cost per 1,000 SLO-meeting requests	PD GPU-hours plus encoder GPU-hours, divided by goodput	Below aggregated baseline by a stated margin

Table 3. Measurement sheet and rollback gates for an EPD pilot. Gates are proposals to tune per product.

The sheet separates TTFT by request type because aggregate averages hide the structure of the problem. It also puts cost per 1,000 SLO-meeting requests last, since the cost number is only valid once the latency gates hold on every arm.

Rollback criteria

Define rollback before the pilot starts. Roll back if, at planned production load, text-only p95 TTFT or any image-bin p95 TTFT is worse than baseline, goodput is below g^*, encoder utilization stays above 80%, or transfer p95 exceeds its budget. A feature-flag route that keeps part of the image traffic on the aggregated pool makes rollback a configuration change.

GPU sharing and MPS constraints

The colocated topology relies on two processes sharing a GPU. The NVIDIA post's topology diagram describes colocated serving as a shared GPU "through CUDA MPS" (Multi-Process Service) (developer.nvidia.com). NVIDIA's MPS documentation says that MPS is useful when each application process does not generate enough work to saturate the GPU (docs.nvidia.com), which is the situation of a small vision encoder beside a large LLM. The same documentation lists constraints that matter operationally. Address spaces are isolated, but error containment is limited: a fatal GPU fault generated by a Volta MPS client process is reported to all clients on the affected GPUs (docs.nvidia.com). Monitoring also changes, since system monitoring tools attribute all MPS client behavior to the MPS server process (docs.nvidia.com). For a buyer, that means per-worker GPU utilization must come from engine metrics, not from `nvidia-smi` alone, and a colocated encoder fault domain includes the PD worker beside it.

Data Analysis and Evidence

What the independent systems literature establishes

Three text-serving papers set the baseline for any disaggregation claim. DistServe reports that it can serve 7.4x more requests or 12.6x tighter SLO (arxiv.org) than prior systems. Splitwise reports 2.35x more throughput with the same cost and power budgets (arxiv.org) and splits phases onto separate machines. Sarathi-Serve reports 2.6x higher serving capacity (arxiv.org) for Mistral-7B on single A100 GPUs by chunking prefill instead of moving it to other machines. None of these is a multimodal result, and each used its own baseline. They support one structural point: the gain from separating phases depends on how much interference exists in the baseline and how much transfer the separation adds.

The EPD paper extends this to multimodal serving and reports 10x more images per request (arxiv.org) and SLO attainment improvements on MiniCPM-V and InternVL class models. Its authors evaluate encoders as a third independently scaled stage. This is peer-reviewed work, but its models and hardware differ from NVIDIA's, so its percentages cannot be added to the Dynamo numbers.

Mechanisms the literature documents

The EPD paper explains why aggregation hurts. Sequential execution of encoding and prefilling introduces interference (arxiv.org), and co-hosting both models limits memory: removing the LLM and its associated KV cache significantly increases the maximum batch size and the number of images per request (arxiv.org). Its transfer design uses direct, asynchronous transfers via high-bandwidth channels (NVLink, and InfiniBand) (arxiv.org), a different fabric assumption from the Ethernet path in NVIDIA's test. It also proposes a dynamic role-switching capability that monitors the system for bottlenecks (arxiv.org), so instances can move between encode, prefill and decode roles.

On transport, the vLLM **Mooncake** guide includes a TCP path labeled TCP (no RDMA device required; CI default) (docs.vllm.ai) beside an RDMA path, so a pilot can start on commodity Ethernet. The vLLM documentation adds that a remote, shared cache lets any worker retrieve existing embeddings (docs.vllm.ai) and encoder nodes can be added or removed independently (docs.vllm.ai).

What the vendor study adds

NVIDIA's study contributes four quantified regularities, all on one family of models:

- **Image load.** Both EPD topologies improved TTFT by 25 to 93% across the grid, with gains concentrated at high image counts (Table 2); the heaviest cell, 50 images at 128 tokens, reached 92 to 93%.
- **Output length.** End-to-end gains erode as decode dominates, and the study reports losses at high output length with low image load (see Cases Where Aggregated Serving Wins).
- **Model size.** The ViT parameter share falls from 7.2% at 4B to 4.7% at 9B and 1.7% at 27B (docs.nvidia.com), and colocated goodput relative to aggregated falls from 2.62x to 0.65x.
- **Precision.** Quantizing LLM weights to NVFP4 while the encoder stays in BF16 raised the colocated ratio from 1.78x to 2.64x (docs.nvidia.com).

The vendor's own caveat is explicit: Aggregate serving remains a good fit when media processing is light or long decode sequences dominate the latency (docs.nvidia.com).

Break-even sensitivity (analyst arithmetic)

The break-even model has two uncertain inputs: the goodput multiplier g and the encoder GPU price. For the study's 4 PD plus 2 encoder GPU layout, with c_{PD} set to the $7.69 B200 placeholder price (lambda.ai):

- **At g = 1.7 (the vendor figure):** the encoder GPU can cost up to 1.4 x $7.69, about **$10.77** per hour, before the tier loses money.
- **At g = 1.2:** the ceiling falls to 0.4 x $7.69, about **$3.08** per hour. A $1.09 placeholder encoder GPU passes, with cost per request at 0.89 of baseline.

The reading is that the tiered design is forgiving on price when the encoder tier is much cheaper than the PD tier and the vendor uplift holds, and unforgiving when the uplift disappears. The colocated design has no additional hardware to fund, so its risk is a latency regression rather than a cost one, as at the study's 27B point.

Cases Where Aggregated Serving Wins

Aggregated serving remains the better choice under identifiable conditions, several of which the vendor's own data support.

Light media, long outputs

The study reports that in scenarios with high OSL and low image load, disaggregating the encoder compromises performance (developer.nvidia.com). **(Hypothetical Example)** A support assistant that attaches one or two screenshots and then writes a 1,500-token answer sits in that corner. In the five-image test, the heterogeneous end-to-end gain narrowed from 20.3% to 5.2% at long outputs, and the colocated arm moved to a regression (Table 2).

Large dense models

The 27B result shows the LLM dominating the compute budget: the post states in our 27B configuration, the LM costs outweighed the benefit of separating the encoder under a colocated setup (docs.nvidia.com). A buyer serving a large dense VLM should expect aggregated serving, or a different optimization such as quantization, to beat EPD.

(Hypothetical Example) A private assistant for field engineers serves a 27B dense VLM, with 1.3 images per request on average and a median answer of 900 tokens, on four GPUs of one class. The expected benefit of EPD is inside the noise of the load generator, the fleet is homogeneous, and the colocated design adds an MPS fault domain. The rational decision is to stay aggregated, apply the frontend-decode and embedding-cache levers if images repeat, and re-run the pilot when image counts or model size change.

Implications and Future Directions

- **For architects.** Placement is becoming a first-class design decision. Dynamo's documentation already separates the roles (see the encoder-placement subsection above), and vLLM is building native connector, proxy and router pieces on the same pattern. The vLLM tracking issue lists work such as support metadata-only images in Rust HTTP frontend (github.com), which reduces payload duplication between stages. The architecture question is shifting from "which engine" to "where does each stage run".
- **For fleet buyers.** The break-even model turns a qualitative design debate into a price ceiling for an encoder GPU. A buyer who already owns lower-tier GPUs has a low-cost option to test; a buyer who must purchase a new tier should require the pilot to show g above g^* with margin across several image-count bins. Power and cooling also matter: the RTX PRO 6000 Blackwell Server Edition is a passively cooled server card (www.nvidia.com), which constrains chassis choice. NVIDIA's family page used for this report gives no price, and the Lambda list retrieved for this report (lambda.ai) shows none for this card.
- **For private and air-gapped deployments.** Dedicated encoder tiers add network hops and processes that a regulated site must inventory and patch, and image fetch, decode and cache choices become part of the data-flow documentation. GPU Smith's advisory view is that these operational costs belong in the same spreadsheet as the GPU price; readers should consult their own security and compliance staff for site-specific requirements.
- **Open questions.** As of November 2026 this research found no public independent benchmark of heterogeneous encoder tiers besides NVIDIA's. Bursty arrivals, video workloads, speculative decoding, prefix caching and cross-vendor GPU tiers remain open, so re-run the worksheet at each major engine upgrade.

Frequently Asked Questions (FAQs)

What is encode prefill decode disaggregation, and how is it different from prefill/decode disaggregation?

EPD disaggregation for multimodal serving splits a multimodal request into three stages (vision encoding, LLM prefill, LLM decode) that can be scheduled and scaled independently. NVIDIA defines it as an inference optimization technique for multimodal models that separates the vision encoder stage from the prefill and decode stages (developer.nvidia.com). Text-only PD disaggregation, as in DistServe, separates only the last two stages.

How do you separate the vision encoder from LLM serving in vLLM?

For vision encoder disaggregation in vLLM, the documentation describes a disaggregated encoder that runs the vision-encoder stage of a multimodal LLM in a process that is separate from the pre-fill / decoder stage (docs.vllm.ai), with a connector that moves embeddings to the PD instance and example scripts for one encoder plus one PD instance, or one encoder, one prefill and one decode instance. Check the documentation version that matches the release in use, because connectors and scripts have changed across versions.

What are the main multimodal LLM inference bottlenecks?

Per the NVIDIA post they are media download and decode, ViT forward and projection, LLM prefill and decode. The post reports that a large share of TTFT elapses before the ViT starts (see the symptoms subsection above). EPD helps the ViT stage; frontend decoding and caching help the stages before it.

What drives vision language model inference latency, and how much GPU does vision encoding use?

Image count and token budget, output length, model size and precision, and traffic mix. Vision encoding GPU utilization in a VLM can look low on a dashboard while still dominating image-request TTFT, so measure encoder time per request and visual tokens per request.

When should prefill decode disaggregation be used?

For text, use it when prefill and decode interfere enough that separate resources improve goodput at the SLO by more than the transfer and extra-hardware cost, which is the framing in DistServe and Splitwise. For multimodal requests, add the encoder-share test from this report: separate the encoder only when the same-SLO multiplier exceeds g^*. The vendor's 70% figure is one test on one model family and hardware pairing, so it should not be reused for another model.

Conclusion

The decision to split vision encoding from LLM serving is a capacity-planning problem with a checkable answer, and the answer varies by workload. NVIDIA's study shows that, for an image-heavy, quantized MoE workload with short or medium outputs, EPD can cut TTFT and, with a cheaper encoder tier, serve more traffic

at the same ITL SLO. The same study shows the benefit shrinking as outputs lengthen, as the LLM grows, and as image load falls, and NVIDIA states, without presenting data, that a homogeneous dedicated encoder tier underperforms colocation; that setup was excluded from the analysis.

For buyers, the worksheet reduces to a few measurable quantities: visual tokens per request, encoder throughput at a safe utilization, embedding bytes and transfer p95, the same-SLO goodput multiplier g on the buyer's model, and the break-even multiplier g^* from GPU prices. Where g clears g^* with margin across image-count and output-length bins, and text-only latency does not regress, a colocated worker or a cheaper encoder tier is justified. Where it does not, aggregated serving is the lower-risk and often lower-cost choice. Because public independent evidence for heterogeneous encoder tiers is thin as of November 2026, a pilot with explicit rollback gates should come before any hardware purchase. The worksheet in this report is meant to be re-run with the buyer's own measurements, not to substitute for them.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://developer.nvidia.com/blog/when-to-use-encode-prefill-decode-disaggregation-to-accelerate-multimodal-model-serving
2. https://arxiv.org/abs/2501.05460
3. https://lambda.ai/pricing
4. https://huggingface.co/nvidia/Qwen3.5-122B-A10B-NVFP4/blob/main/config.json
5. https://arxiv.org/abs/2401.09670
6. https://arxiv.org/abs/2311.18677
7. https://proceedings.mlr.press/v267/singh25d.html
8. https://docs.nvidia.com/dynamo/multimodal/encoder-disaggregation.md
9. https://docs.vllm.ai/en/stable/features/disagg_encoder
10. https://arxiv.org/html/2501.05460v4
11. https://github.com/ai-dynamo/nixl
12. https://github.com/vllm-project/vllm/issues/52409
13. https://arxiv.org/abs/2403.02310
14. https://docs.nvidia.com/dynamo/dev/multimodal/parallel-media-decoding
15. https://arxiv.org/abs/2409.12191
16. https://platform.claude.com/docs/en/build-with-claude/vision
17. https://ai.google.dev/gemma/docs/capabilities/vision
18. https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin
19. https://docs.nvidia.com/dynamo/dev/digest/epd-disaggregation-multimodal
20. https://www.nvidia.com/en-us/products/workstations/professional-desktop-gpus/rtx-pro-6000-family
21. https://github.com/ai-dynamo/dynamo/tree/release/1.5.0/benchmarks/multimodal/sweep/experiments/epd
22. https://docs.nvidia.com/deploy/mps/when-to-use-mps.html
23. https://docs.vllm.ai/en/stable/features/mooncake_ec_connector_usage

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the hardware library, GPU references, networking references and vendor research. Contact GPU Smith with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: https://gpusmith.com

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.