TriInfer: Hybrid Disaggregated Scheduling for Multimodal LLM Serving

MLSys 2026 – Oral Presentation

Xianzhe Dong, Tongxuan Liu, Yuting Zeng, Weizhe Huang, Xiaoyang Zhao, Siyu Wu, Liangyu Liu, Yang Liu, Yu Wu, Hailong Yang, Ke Zhang, Jing Li

*Presented by **Hao Ren** (on behalf of the authors)*

University of Science and Technology of China Beihang University JD.com

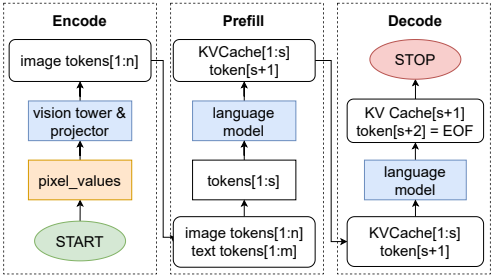
Multimodal LLMs are Everywhere

- MLLMs (LLaVA, Qwen-VL, MiniCPM, Gemini, ...) power image understanding, VQA, and visual reasoning.
- A single image is converted into hundreds to thousands of visual tokens.
- Inference cost is much higher than text-only LLMs $\Rightarrow$ **serving efficiency is critical**.

Three-stage MLLM Inference

Encode – vision tower turns image into visual tokens
Prefill – LLM processes visual + text tokens, emits 1st token
Decode – LLM iteratively generates following tokens

Three Stages of MLLM Inference



- **Encode**: ViT-style, full attention, batchable.
- **Prefill**: long input, compute-bound.
- **Decode**: token-by-token, memory-bound.

Each stage has very different compute/memory characteristics and SLO targets.

Performance Metrics

- **TTFT** (Time-To-First-Token): user-perceived responsiveness.
- **TBT** (Time-Between-Tokens): smoothness of streaming output.
- **SLO attainment**: % of requests meeting both TTFT and TBT SLOs.
- **Goodput** (our objective):

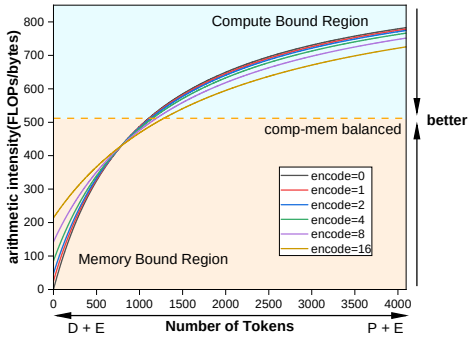
$$\textit{the maximum request rate at which} \geq 90\% \textit{ requests meet their SLO.}$$

Goal of `TriInfer`
Maximize **per-GPU goodput** for MLLM serving.

Problem 1: Insufficient Parallelism

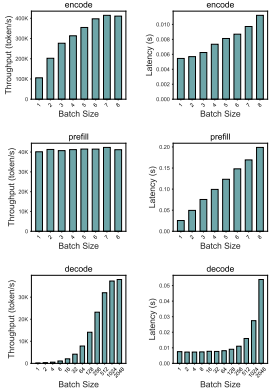
- Existing systems (vLLM, SGLang, TGI) run **vision and language sequentially**.
- Decode is memory-bound; encode/prefill are compute-bound.
- Stages have **complementary** resource demands – but no system exploits this.

Takeaway 1: Parallel execution of vision and language models can dramatically improve hardware utilization.



Arithmetic intensity vs. #tokens at different image batch sizes.

Problem 2: Coarse Scheduling Granularity

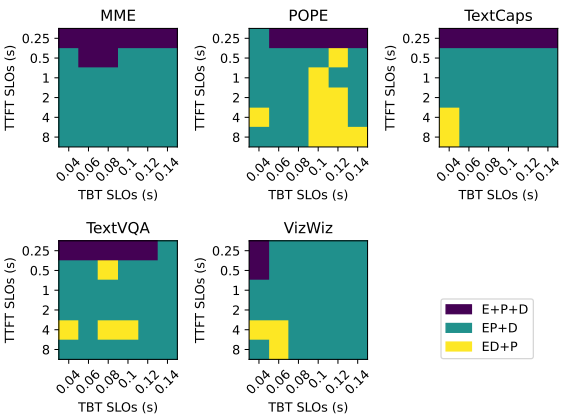


Throughput saturates at very different batch sizes per stage.

- Encode saturates at batch ≈ 6.
- Prefill saturates at *batch* $= 1$.
- Decode keeps scaling up to ≈ 512.
- LLM-style schedulers (chunked prefill, stall-free) limit only **token count** – ignoring image-encoding time $\Rightarrow$ **TBT SLO violations**.

Takeaway 2: Need *stage-aware* batching that bounds execution time, not just token count.

Problem 3: Which Disaggregation is Best?



Best disaggregation depends on workload & SLO.

Three plausible disaggregations:

- *E+P+D*: full disagg., best under tight TTFT.
- *EP+D*: only one KV-cache transfer, vision/language parallel.
- *ED+P*: prefill isolated, low TTFT under bursts.

No single disaggregation wins everywhere. Existing systems pick **one** statically.

Takeaway 3: Disaggregation must be *chosen per workload & SLO*.

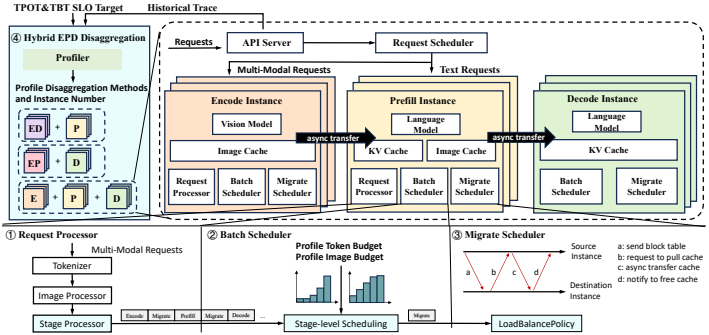
Three contributions, one system:

1. **Dual-stream parallelism** – run vision and language models concurrently on two CUDA streams.
2. **Stage-level scheduling** – a unified, stage-granularity batching algorithm that bounds each batch's execution time.
3. **Hybrid EPD Disaggregation** – a profiler that automatically picks the best of $E{+}P{+}D$ / $EP{+}D$ / $ED{+}P$ for the current workload.

Result

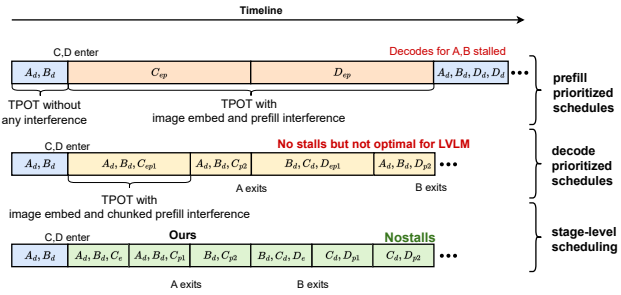
Up to **2.4**$\times$ **higher goodput** than vLLM / SGLang at 90% SLO attainment.

System Architecture



- **Request Processor**: tokenize, image-process, slice into stage-level tasks.
- **Batch Scheduler**: stage-level batching inside each instance.
- **Migrate Scheduler**: pull-based KV / image cache transfer across instances.
- **Hybrid EPD Disaggregation**: chooses instance roles (E / P / D / EP / ED).

Stage-level Scheduling

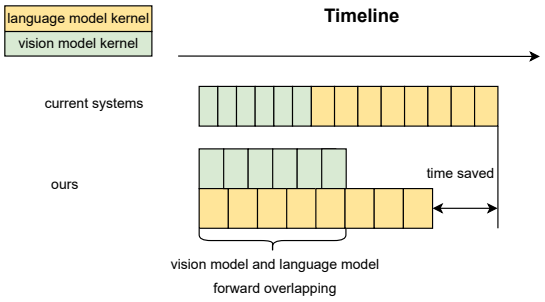


Key idea: bound batch latency, not token count.

- For *E/EP/P* instances: cap latency at $\alpha \cdot \mathrm{TTFT}_{\max}$.
- For *D/PD/ED/EPD*: cap latency at $\mathrm{TBT}_{\max}$.
- Binary-search both an **image budget** τ_e and a **token budget** τ_t at startup.
- In each iteration: include all decoders first $\rightarrow$ chunked prefill / encode $\rightarrow$ fill new requests.

No more generation stalls.

Dual-stream Parallelism



- One **vision stream** for encode kernels.
- One **language stream** for prefill / decode kernels.
- Decode is memory-bound, encode is compute-bound
  $\Rightarrow$ overlap reclaims idle units.
- Inspired by NanoFlow but applied across modalities.

Both throughputs go up simultaneously.

Question: given N instances, an SLO, and a recent workload trace, how do we deploy?

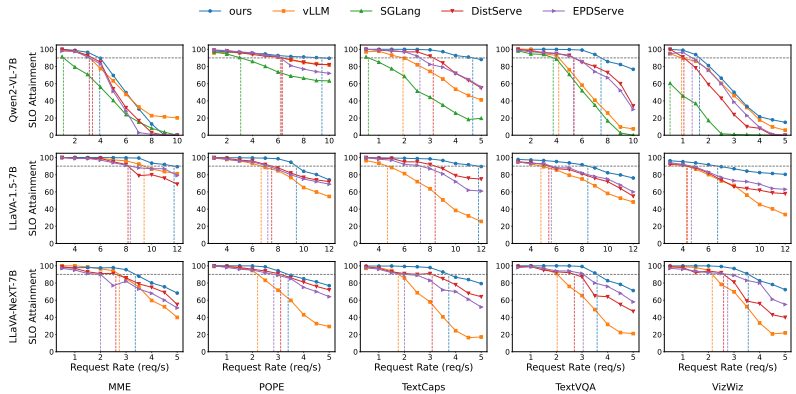
1. Sum stage workloads W_e, W_p, W_d from trace.
2. Find image / token budgets τ_e, τ_p, τ_d via binary search under SLO.
3. Profile per-stage throughput tp_e, tp_p, tp_d at full load.
4. Compute stage execution time t_e, t_p, t_d, then partition $N \to (N_e, N_p, N_d)$.
5. For each candidate $\in \{E{+}P{+}D,\ EP{+}D,\ ED{+}P\}$: replay trace, measure goodput.
6. Pick the disaggregation with the highest goodput.

Search space shrinks from $O(N^2)$ brute force to a few candidate methods $\Rightarrow$ **minutes, not hours**, and reruns when workload shifts.

Experimental Setup

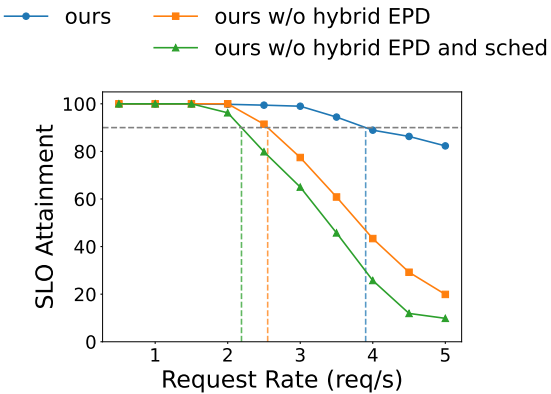
- **Cluster**: 4 servers $\times$ 8 H20 GPUs (141 GB), NVLink + 2$\times$200 Gbps IB.
- **Models**: LLaVA-1.5-7B, LLaVA-NeXT-7B, Qwen2-VL-7B.
- **Workloads**: MME, POPE, TextCaps, TextVQA, VizWiz; arrival pattern from Mooncake production trace.
- **Baselines**: vLLM 0.11.0, SGLang 0.5.3, EPDServe (E+P+D), DistServe (EP+D).
- **Metric**: per-GPU goodput at 90% SLO attainment.

Main Result: SLO Attainment vs. Request Rate



- Goodput = rightmost rate keeping SLO attainment $\geq 90\%$.
- `TriInfer` achieves up to **2.4× higher goodput** than vLLM/SGLang, and clearly exceeds EPDServe / DistServe.

Ablation Study

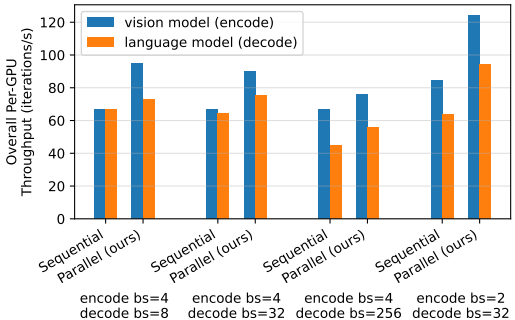


LLaVA-NeXT-7B on TextCaps:

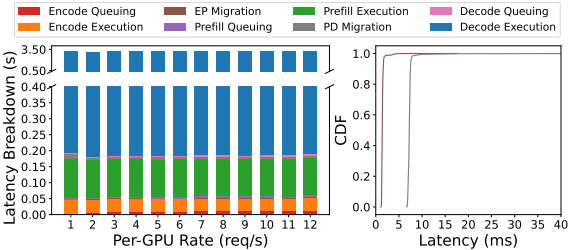
- Full `TriInfer` – best goodput.
- Disable Hybrid EPD (use fixed $E{+}P{+}D$, even split) $\rightarrow$ noticeable goodput drop.
- Further disable stage-level scheduling $\rightarrow$ goodput drops again, TBT SLO violations spike.

Both contributions matter, and they are complementary.

Dual-stream Parallelism & Latency Breakdown



Encode + decode, sequential vs. parallel.



Per-stage latency breakdown on TextCaps.

- Dual-stream: clear throughput gain across all batch sizes.
- Migration overhead is $<1\%$ of end-to-end latency (P95 image-cache migration ≈ 2 ms, KV-cache ≈ 8 ms).

Conclusion

`TriInfer`: an MLLM serving system built on **Hybrid EPD Disaggregation**.

- **Stage-centric abstraction** – encode / prefill / decode are first-class stages with their own scheduling and SLOs.
- **Dual-stream parallelism** reclaims complementary GPU resources between vision and language models.
- **Stage-level scheduling** bounds each batch's execution time and eliminates generation stalls.
- **Hybrid disaggregation profiler** adapts to workload & SLO automatically.
- Up to **2.4**$\times$ **goodput** over state-of-the-art systems.

Thank You!

Questions are welcome via email

`TriInfer`: Hybrid Disaggregated Scheduling for Multimodal LLM Serving

First author: Xianzhe Dong
Email: `dongxianzhe2019@gmail.com`

The presenter is delivering this talk on behalf of the authors.
For technical questions, please contact the first author directly by email.