CoEncode: Reclaiming Idle Encoder GPUs in Multi-Tenant Multimodal LLM Serving

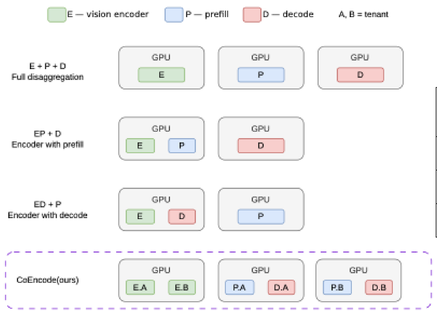
Mona Moghadampanah[1], Xiangchen Li[1], Cheol-Ho Hong[2], Dimitrios S. Nikolopoulos[1]

[1]Virginia Tech, USA [2]Chung-Ang University, Seoul, Korea

Introduction

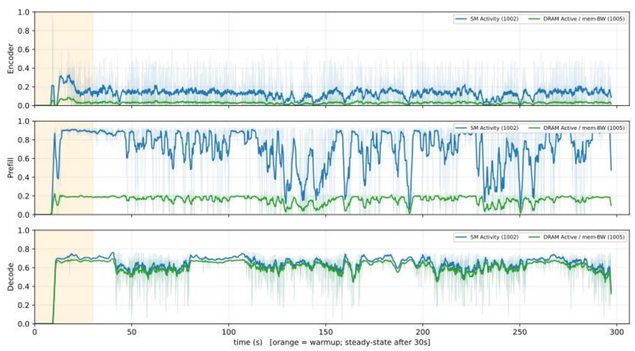
- Dedicated encoder GPUs in disaggregated multimodal LLM (MLLM) serving are substantially underutilized.
- CoEncode pools encoders across tenants using model-specific capacity profiles.
- Retains 99.4% SLO goodput while improving goodput/GPU by 19.3% and reducing encoder energy by 45.9%.

Motivation

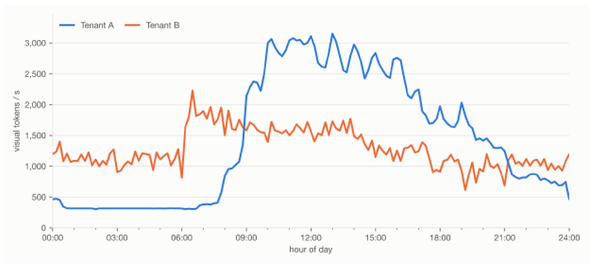


Architecture	Throughput (req/s)	TTFT p95 (s)	TPOT p95 (ms)
E+P+D	4.01	0.68	29.3
EP+D	4.00	1.32	41.1
ED+P	4.01	0.82	54.3

- HydraInfer [1] explores E+P+D, EP+D, and ED+P placement strategies.
- Cross-stage colocation saves a GPU, but hurts latency: EP+D increases TTFT by 1.94×, while ED+P increases TPOT by 85%.



- **Dedicated encoder is mostly idle:** 10.8% SM activity, versus 60.7% for prefill and 63.3% for decode.
- This exposes substantial reclaimable encoder capacity.

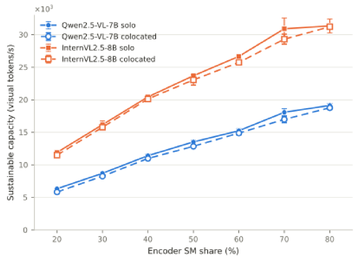


- ServeGen [3] workloads show heterogeneous, time-varying visual-token demand across services.
- Static provisioning leaves encoder capacity idle as demand fluctuates.

Experimental Setup

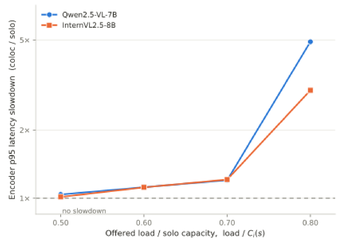
- **GPU:** NVIDIA A100 80 GB
- **Models:** Qwen2.5-VL-7B, InternVL2.5-8B
- **Serving:** vLLM with disaggregated E+P+D execution
- **Sharing:** MPS + disjoint TPC masks via libsmctrl
- **Workload:** ServeGen-derived open-loop requests, 4 req/s after warm-up
- **TTFT SLOs:** 2.3 s for Qwen, 3.2 s for InternVL (1.1× dedicated p95)
- **Evaluation:** 3 trials; error bars show min–max; energy measured with Zeus/NVML

When Can Encoders Safely Share a GPU?

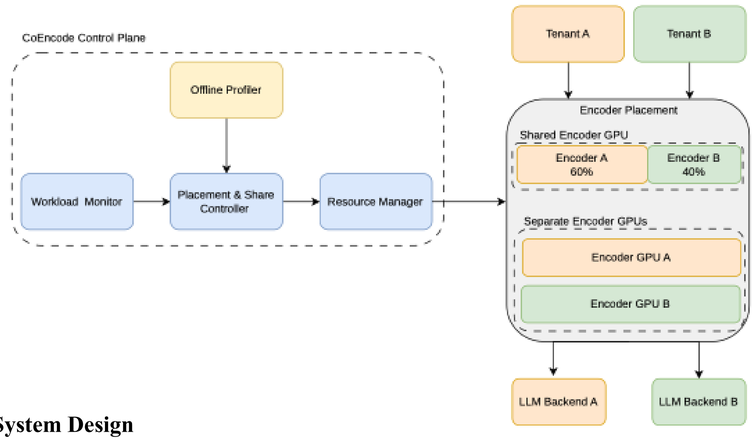


- Encoder capacity scales differently across models and SM shares.
- Colocated capacity retains **≥92% of solo capacity** across tested allocations.
- Solo profiles can therefore guide CoEncode's SM-share selection.

- At 0.5–0.6× capacity, colocation introduces little p95 slowdown.
- Slowdown grows sharply as demand approaches 0.8× capacity.
- Safe colocation therefore requires load-aware placement and adaptation.



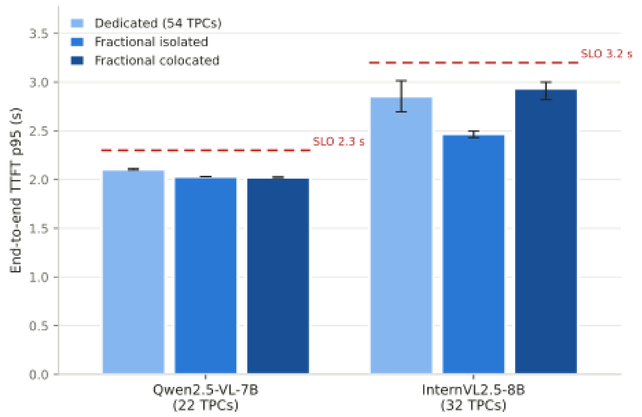
CoEncode Design



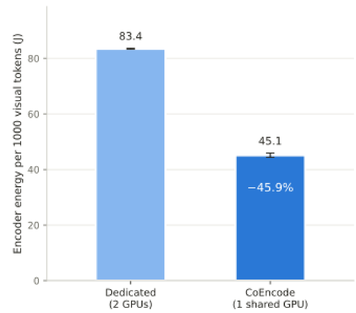
System Design

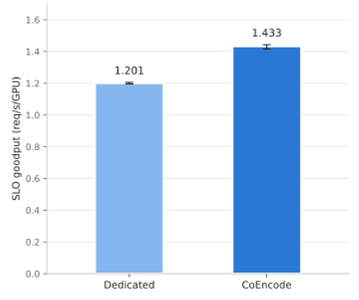
- Build model-specific encoder capacity profiles offline.
- Use tenant demand and SLO targets as inputs.
- Select a feasible placement and SM share.
- Colocate encoders when the selected allocation meets throughput and TTFT targets.

Results



- **SLO preservation:** Colocated encoders remain within the 2.3 s Qwen and 3.2 s InternVL TTFT SLOs.





- **Energy efficiency:** Encoder energy drops from 83.4 to 45.1 J/1K visual tokens — 45.9% lower.
- **Resource efficiency:** Retains **99.4%** of dedicated **SLO goodput** while improving SLO goodput/GPU by **19.3%.**

Future Work

- **Larger tenant pools:** Scale encoder sharing beyond two tenants.
- **Interference-aware placement:** Identify compatible encoder combinations.
- **Dynamic adaptation:** Resize SM shares or de-colocate as demand changes.
- **Energy-aware provisioning:** Incorporate SM-share energy profiles into allocation decisions.

Acknowledgements

This work was supported by the National Science Foundation (Grant No. 2315851), a Sony Faculty Innovation Award, an Amazon Research Award, and the Virginia Tech College of Engineering.

References

- [1] Dong et al., **HydraInfer**, 2025.
- [2] Qiu et al., **ModServe**, 2025.
- [3] Xiang et al., **ServeGen**, NSDI 2026.
- [4] Singh et al., **EPD Disaggregation**, ICML 2025.