Serving PyTorch LLMs at Scale: Disaggregated inference with Kubernetes and llm-d

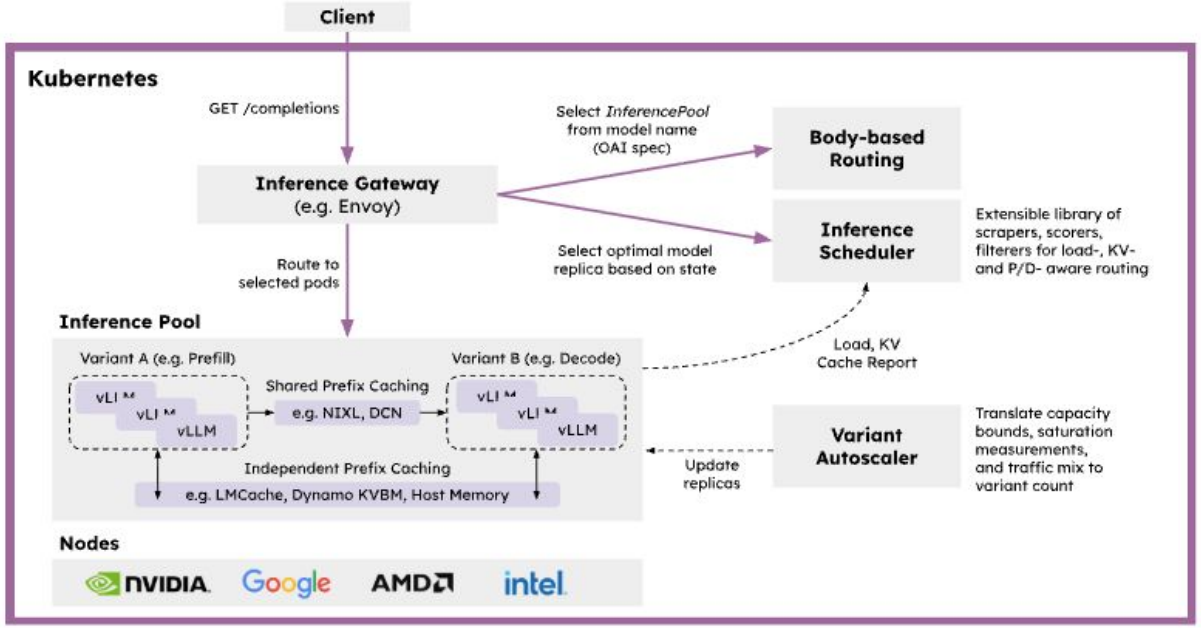
Cong Liu, Google
Maroon Ayoub, IBM Research

Outline

- Background [Cong]
- llm-d implementation [Maroon]
- Well-lit path walkthrough [Cong]
- Benchmark [Maroon]
- Future work [Cong]
- Summary [Maroon]

Background: llm-d

- Open-source, k8s-native, pluggable inference stack
- Accelerator agnostic
- Streamlined well-lit paths
- Reproducible benchmarks
- LLM aware inference gateway + LLM aware autoscaler



Background: LLM Inferencing Phases

Phase	Prefill	Decode
Description	Processes prompt and generate the *first* token in *one step*	Generates output tokens autoregressively in *multiple steps*
Characteristic	Compute bound	Memory bound
Metric	TTFT (time to first token)	TPOT (time per output token)
Dependency	Decode relies on the KV caches of previous tokens generated by prefill	

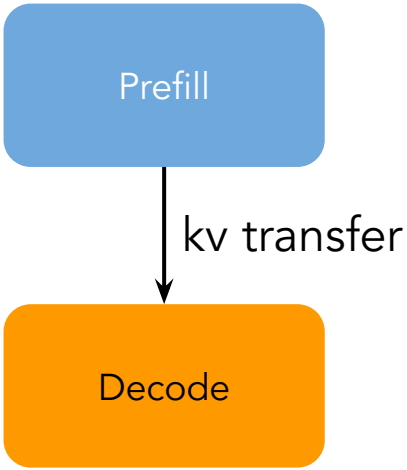
Background: Disaggregated Inference

What

- Prefill and decode worker on different instances
- Relies on high-performance kv transfer between P and D workers

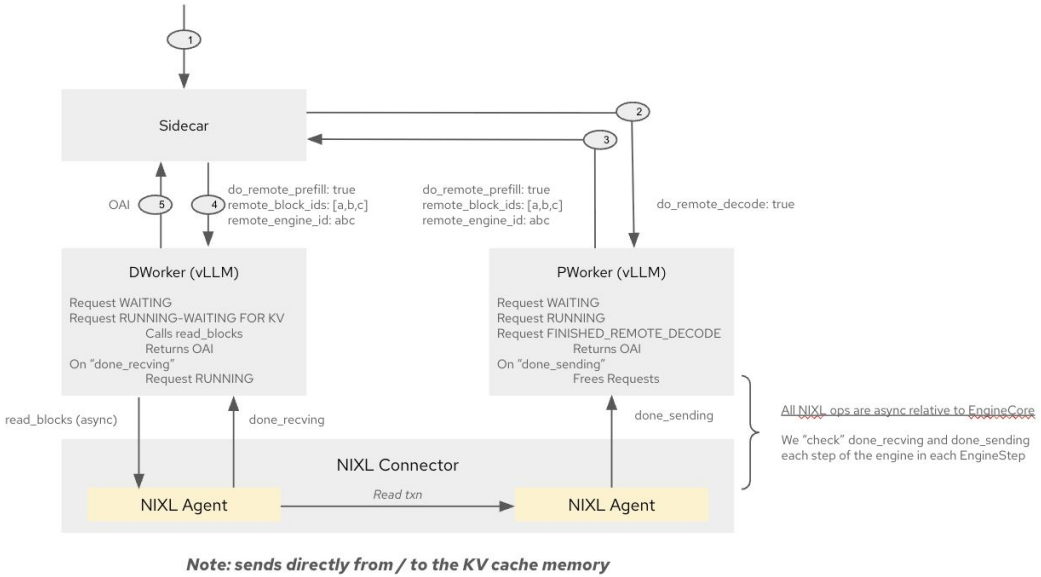
Why

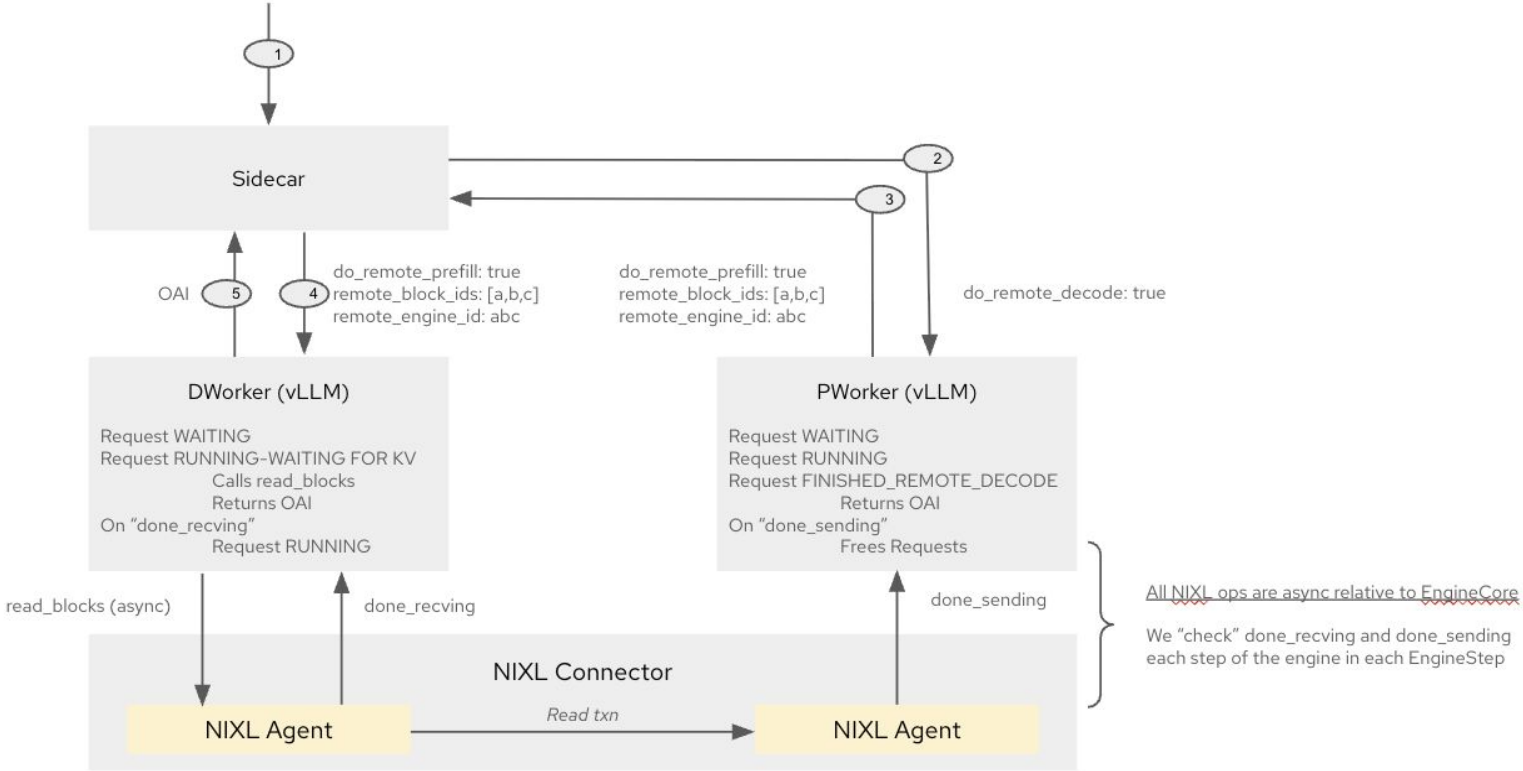
- Prefill-decode interference
- Tuning TTFT and TPOT separately
- Flexibility in scaling and optimizing prefill and decode workers independently
    - E.g., heterogeneous accelerator, heterogenous TP sizing



Disaggregated Inference in llm-d

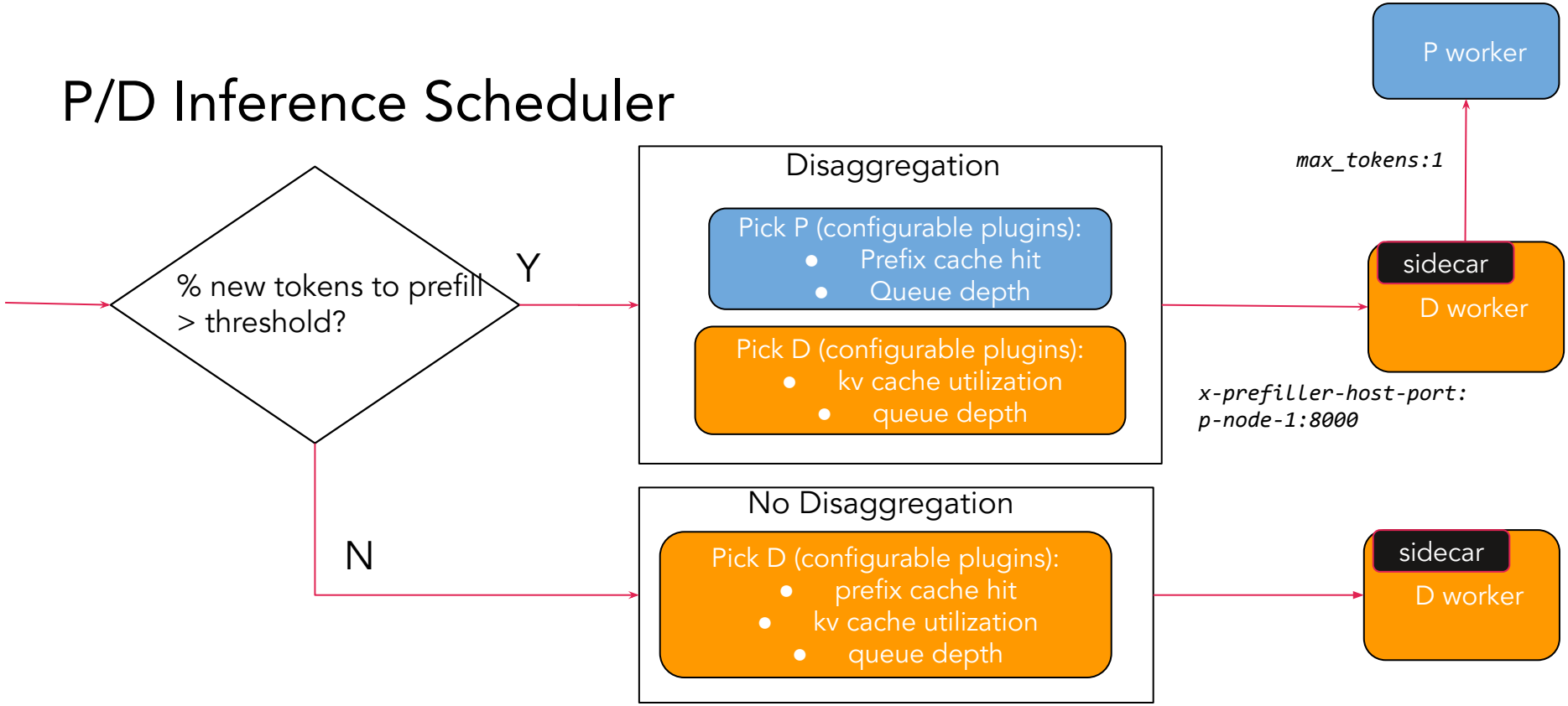
- P/D aware inference scheduler
- Leverage vLLM NIXL connector
- Implements a routing-sidecar proxy deployed with the decode workers





Note: sends directly from / to the KV cache memory

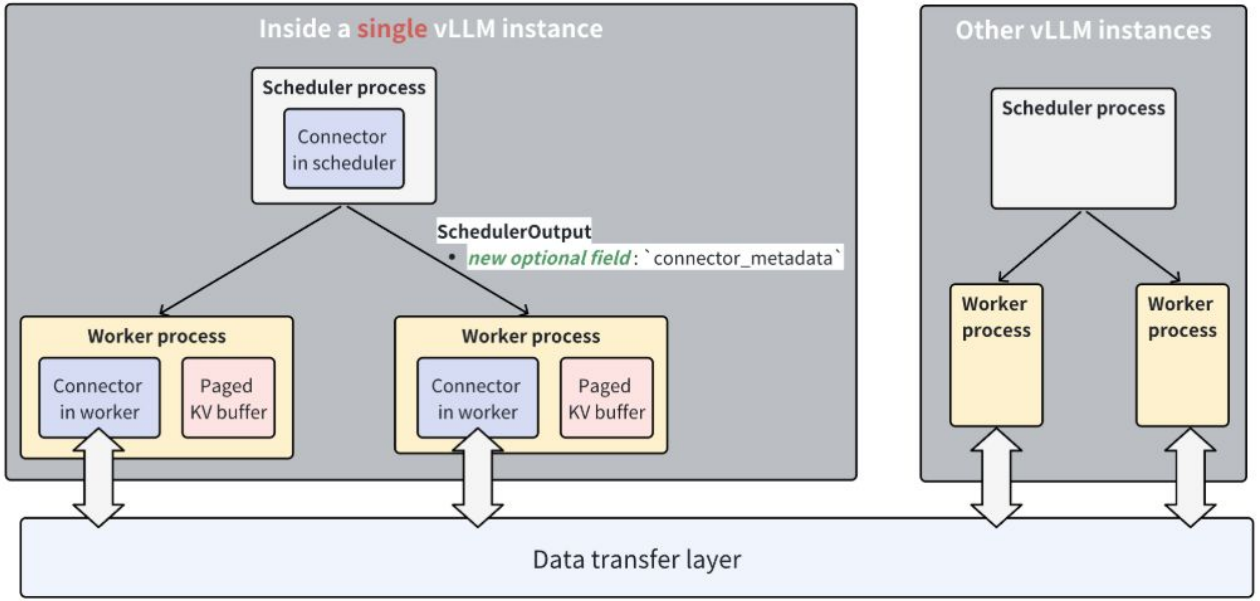
P/D Inference Scheduler



Routing Sidecar

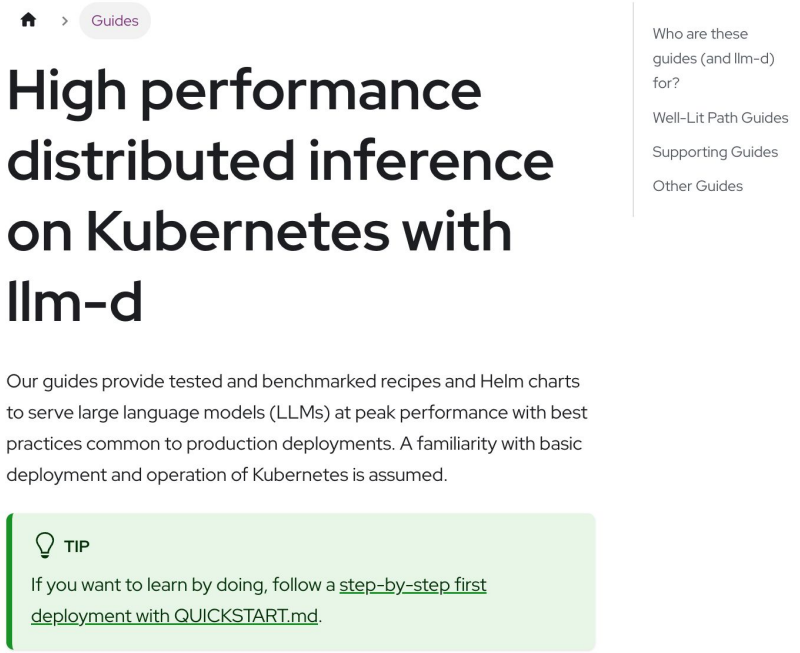
- A proxy deployed with decode nodes
- In non disaggregation mode,
  - No `x-prefiller-host-port` header is present
  - Simply forwards request to the decode node (`localhost`)
- In disaggregation mode,
  - When `x-prefiller-host-port` header is present
  - First sends request to the prefill node with `max_tokens=1`
  - Then sends request to the decode node (`localhost`)
  - Decode transfers kv cache from prefill via NIXL connector

KV Transfer (via NIXL Connector)



llm-d Well-lit Paths

- llm-d.ai/docs/guide/Installation
- Supports various gateway providers and accelerators
- Best practices out of the box
- Regularly tested via CI/CD and benchmark
- Guide structure
  - Infrastructure/tool requirements
  - General tuning guidance for optimal results
  - Streamlined installation recipes



P/D Well-lit Paths

P/D over RDMA

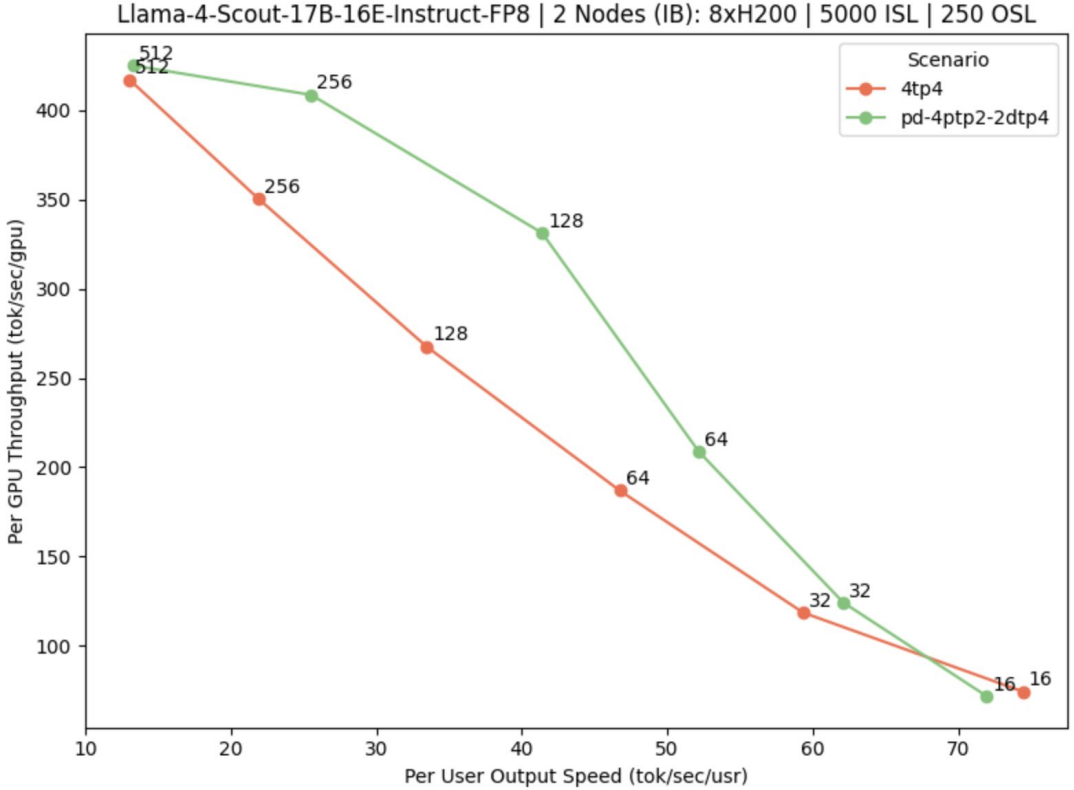
- 8xH200 GPUs
- InfiniBand OR RoCE RDMA
- Llama-3.3-70B-Instruct-FP8
- 4 TP=1 Prefill Workers, 1 TP=4 Decode Worker

Wide-ep (expert parallelism) with P/D

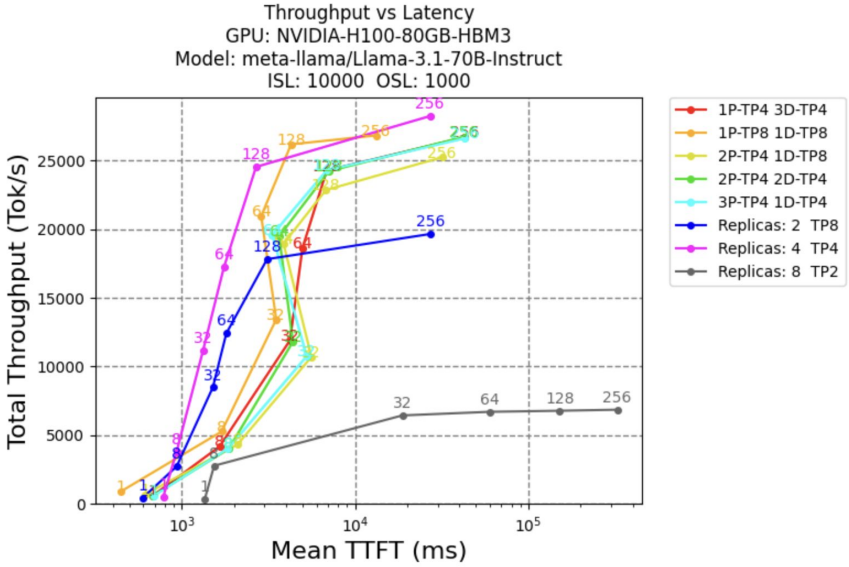
- 32xH200 GPUs
- InfiniBand OR RoCE RDMA
- DeepSeek-R1-0528
- 1 DP=16 Prefill worker, 1 DP=16 decode workers

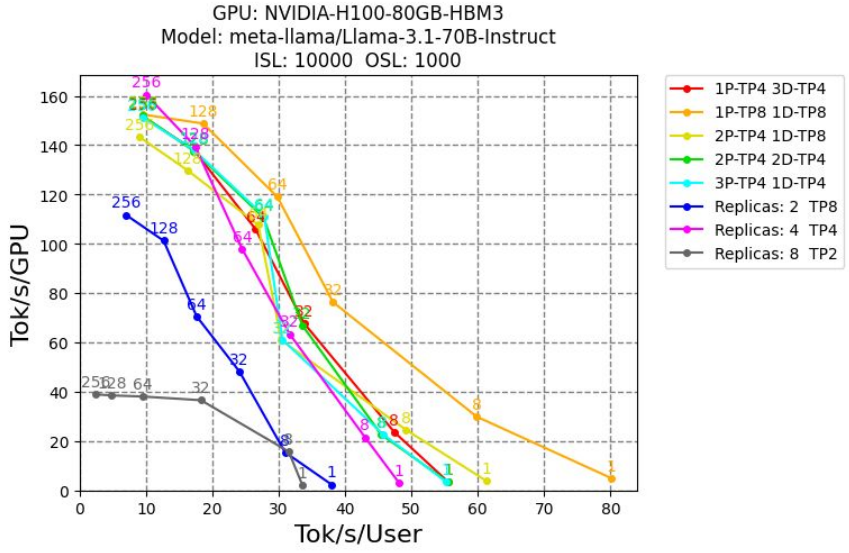
Benchmark Results

Pareto curve for Llama-Scout on dual 8×H200 IB nodes, comparing monolithic (4tp4) and P/D-disaggregated (4ptp2–2dtp4) topologies.



Benchmark Results





Getting value is not so simple

P/D Best Practices

P/D disaggregation is not a target for all workloads. We suggest exploring P/D disaggregation for workloads with:

- Large models (e.g. Llama-70B+, not Llama-8B)
- Longer input sequence lengths (e.g 10k ISL | 1k OSL, not 200 ISL | 200 OSL)
- Sparse MoE architectures with opportunities for wide-EP

As a result, as you tune your P/D deployments, we suggest focusing on the following parameters:

- Heterogeneous Parallelism: deploy **P workers** with **less parallelism** and **more replicas** and **D workers** with **more parallelism** and **fewer replicas**
- xPyD Ratios: tuning the ratio of P workers to D workers to ensure balance for your ISL|OSL ratio

Future explorations

Deferred Decoder Picking

- Currently gateway picks P and D at request arrival time
- However the request lands on D at a later time -> load can be stale
- Idea: Pick D after prefill is done, with freshest load metrics

“Decode-first” PD

- Router doesn’t have info about cache in remote storage -> let vLLM calculate local+remote cache hit (via KVConnector)
- Router opportunistically pick decoder with potentially high cache hit to avoid P/D
- Decoder returns “low cache hit” and router will then pick P/D

Key Takeaways

- Disaggregated inference is crucial for large scale LLM inference
  - Especially for data-parallel and expert-parallel deployments
    - Uninterrupted decode CUDA graphs, DeepEP optimizations and more
- Productionizing disaggregated inference is non-trivial
  - llm-d research teams do it for you!
- llm-d enables K8s-native disaggregated inference with production ready well-lit paths

Come see our P/D & KV disaggregation demos at the Red Hat station!

(IBM Booth - D2)

PyTorch CONFERENCE 2025

llm-d