

Technical Report

Multi-Node Disaggregated Inference: DeepSeek R1 671B on AMD Instinct MI300X GPUs

Moreh, Inc.

March 2026

Contents

Introduction	1
Background: Why Disaggregate Prefill and Decode?	2
The Interference Problem	2
How Disaggregated Serving Works	2
Experimental Setup.....	3
Test Environment.....	3
Configurations	3
Benchmark Scenarios	4
Results: Disaggregated vs. Colocated Serving.....	4
End-to-End Latency	4
Total Throughput	5
Latency vs. Throughput.....	6
Cost Efficiency (Token/Dollar)	6
Results: Impact of the Prefill-to-Decode Ratio	7
End-to-End Latency: 2P3D vs. 1P4D (1K/8K)	7
Total Throughput: 2P3D vs. 1P4D (1K/8K)	7
Cost Efficiency Comparison.....	7
Summary: Which Ratio to Choose?	8
Latency Stability: P99 Inter-Token Latency	8
Conclusion	9

Multi-Node Disaggregated Inference: DeepSeek R1 671B on AMD Instinct MI300X GPUs

Introduction

Autoregressive LLM inference consists of two distinct phases: prefill, which processes the entire input prompt in parallel to build the KV cache, and decode, which generates output tokens one at a time using the cached keys and values. These two phases have very different execution characteristics — prefill processes many tokens at once in a single, long-running step, while decode runs many short iterations at high frequency — yet in conventional serving systems they share the same GPU resources, causing mutual interference that degrades both throughput and latency.

Prefill-decode disaggregation (also known as disaggregated serving) addresses this by assigning each phase to a dedicated pool of GPU nodes, so that prefill and decode no longer compete for the same resources on the same GPUs. First formalized in *DistServe* (Zhong et al., OSDI 2024), the concept has since gained broad adoption, but realizing its full potential in production requires a highly optimized full-stack software solution — from kernel-level compute efficiency to cluster-wide scheduling and KV cache transfer.

In this report, we use the MoAI Inference Framework — Moreh's production-grade inference framework optimized for high-performance disaggregated serving on AMD GPUs — to measure the impact of prefill-decode disaggregation on a 5-node AMD Instinct MI300X cluster running DeepSeek R1 671B. We first demonstrate the advantage of disaggregated serving over a colocated baseline, then examine how the optimal prefill-to-decode node ratio varies with request patterns by comparing two configurations: 2P3D (2 Prefill + 3 Decode nodes) and 1P4D (1 Prefill + 4 Decode nodes).

Background: Why Disaggregate Prefill and Decode?

The Interference Problem

When prefill and decode are colocated on the same GPUs, they interfere with each other in several ways:

- **Latency spikes.** Prefill processes many tokens at once in a single, potentially long-running GPU step. When a long-context prefill arrives, it occupies the GPU for an extended period, stalling all in-flight decode iterations that share the same device. This inflates P99 inter-token latency (ITL) — sometimes by orders of magnitude — and breaks the smooth streaming experience users expect.
- **Scheduling conflicts.** Prefill and decode have opposing scheduling preferences: prefill wants to process new requests quickly to minimize time-to-first-token (TTFT), while decode needs frequent, uninterrupted iterations to maintain low inter-token latency. A single serving instance must constantly arbitrate between the two, and any compromise degrades at least one metric.
- **Coupled scaling.** Operators cannot independently add prefill or decode capacity; every new node must serve both phases, leading to over-provisioning of whichever phase is less bottlenecked.

How Disaggregated Serving Works

Disaggregated serving splits the cluster into separate prefill nodes and decode nodes. Each pool holds the full model weights (there is no model sharding across pools), but handles only its designated phase. After a prefill node finishes processing the input, it transfers the generated KV cache to a decode node via high-bandwidth interconnect (e.g., RDMA), which then continues token generation without interference.

This separation delivers several key advantages. Even on a homogeneous cluster where every node has the same GPU, disaggregation provides:

- **Elimination of prefill-decode interference.** Decode nodes produce tokens at a stable rate, uninterrupted by incoming prefill work, resulting in dramatically lower and more predictable inter-token latency.
- **Independent scaling.** Operators can tune the prefill-to-decode node ratio to match the workload's input/output length distribution, avoiding over-provisioning.
- **Per-phase optimization.** Each pool can apply different parallelization strategies, scheduling policies, and batching configurations independently tuned for its phase, without compromising the other.

In heterogeneous clusters, disaggregation unlocks an additional dimension of optimization: since prefill is compute-bound while decode is memory-bandwidth-bound, operators can assign compute-dense GPUs to the prefill pool and bandwidth-optimized GPUs to the decode pool, matching hardware characteristics to each phase's bottleneck. This report focuses on a homogeneous AMD MI300X cluster, isolating the interference-elimination, scaling, and per-phase optimization benefits from any hardware-mix effects.

Experimental Setup

Test Environment

All experiments were conducted on a 5-node GPU cluster. Each node is a Gigabyte G593-ZX1-AAX1 server equipped with dual AMD EPYC 9474F CPUs (48 cores, 3.6 GHz), 2,304 GB of main memory, and 8 AMD Instinct MI300X GPUs (192 GB HBM3e each). Nodes are interconnected via InfiniBand HDR.

Category	Specification
Server	Gigabyte G593-ZX1-AAX1 × 5 nodes
CPU	2× AMD EPYC 9474F (48-core, 3.6 GHz) per node
Memory	2,304 GB per node
GPU	8× AMD Instinct MI300X (192 GB HBM3e) per node, 40 GPUs total
Interconnect	InfiniBand HDR
OS	Ubuntu 22.04.4 LTS
Model	DeepSeek R1 671B (MoE)
Precision	FP8
Parallelism	Expert Parallelism (EP8) + Data Parallelism (DP8) per node
Inference Engine	Moreh vLLM (within MoAI Inference Framework)

Configurations

We evaluated three configurations, all using the same 5-node cluster:

- **Colocated (Baseline):** All 5 nodes handle both prefill and decode. Load balancing distributes requests across nodes.
- **Disaggregated 2P3D:** 2 nodes dedicated to prefill, 3 nodes dedicated to decode. Allocates more capacity to prefill relative to 1P4D.
- **Disaggregated 1P4D:** 1 node dedicated to prefill, 4 nodes dedicated to decode. Allocates more capacity to decode relative to 2P3D.

Benchmark Scenarios

All benchmarks were run using **vllm bench serve**. We tested four ISL/OSL (Input Sequence Length / Output Sequence Length) combinations at varying concurrency levels. These combinations are designed to stress different parts of the serving pipeline rather than to model specific applications:

- **1K/1K**: Balanced input and output length
- **1K/8K**: Short input, long output (decode-heavy)
- **8K/1K**: Long input, short output (prefill-heavy)
- **8K/8K**: Long input and output

For each scenario, traffic load was controlled by two parameters: `N_REQs` (total number of requests) and `REQ_RATE` (request arrival rate in req/s). `N_REQs` was set to $2\times$ the concurrency level across all scenarios. `REQ_RATE` was set higher for short-output scenarios (3–6 req/s for 1K/1K, 2–4 req/s for 8K/1K) and lower for long-output scenarios (2 req/s for 1K/8K and 8K/8K), reflecting the fact that long-generation requests occupy GPU resources for longer durations. Once `REQ_RATE` is specified, it — together with `N_REQs` — determines the actual traffic pattern more precisely than the nominal concurrency level alone.

Results: Disaggregated vs. Colocated Serving

We first compare the disaggregated 2P3D configuration (2 prefill + 3 decode nodes) against the colocated baseline where all 5 nodes handle both phases. With 40% of the cluster's compute reserved for prefill, 2P3D serves as our primary disaggregation configuration across all four ISL/OSL scenarios.

End-to-End Latency

Disaggregated 2P3D achieved a geometric mean improvement of 1.35x in median end-to-end latency across all tested configurations.

Scenario	CON	N_REQs	REQ_RATE	Colocated E2EL (s)	2P3D E2EL (s)	Improvement
1K/1K	160	320	3	88.11	68.18	1.29x
1K/1K	320	640	5	103.00	73.62	1.40x
1K/1K	480	960	6	127.45	79.03	1.61x
8K/1K	160	320	2	141.74	77.04	1.84x
8K/1K	320	640	3	171.49	94.48	1.81x
8K/1K	480	960	4	208.43	118.51	1.76x
1K/8K	160	320	2	575.83	549.46	1.05x
1K/8K	320	640	2	665.46	596.69	1.12x
1K/8K	480	960	2	719.67	665.85	1.08x
8K/8K	160	320	2	656.52	582.00	1.13x
8K/8K	320	640	2	760.98	636.52	1.20x
8K/8K	480	960	2	889.59	738.50	1.20x
Geometric Mean						1.35x

The largest gains appear in the 8K/1K (prefill-heavy) scenario, reaching up to 1.84x at CON=160. This is expected: with dedicated prefill nodes, long input sequences no longer block decode operations, and the prefill nodes can batch and process them more efficiently.

Total Throughput

Disaggregated 2P3D achieved a geometric mean improvement of 1.20x in total throughput (tokens per second).

Scenario	CON	N_REQs	REQ_RATE	Colocated TPS	2P3D TPS	Improvement
1K/1K	160	320	3	2,865.69	3,329.94	1.16x
1K/1K	320	640	5	5,162.46	6,270.40	1.21x
1K/1K	480	960	6	6,457.95	8,415.03	1.30x
8K/1K	160	320	2	8,606.33	11,781.94	1.37x
8K/1K	320	640	3	13,832.82	19,795.51	1.43x
8K/1K	480	960	4	17,883.09	24,689.61	1.38x
1K/8K	160	320	2	2,363.14	2,463.28	1.04x
1K/8K	320	640	2	3,906.88	4,348.48	1.11x
1K/8K	480	960	2	5,341.88	5,668.40	1.06x
8K/8K	160	320	2	3,739.94	4,145.35	1.11x
8K/8K	320	640	2	6,091.02	7,322.27	1.20x
8K/8K	480	960	2	8,107.45	9,245.05	1.14x
Geometric Mean						1.20x

Latency vs. Throughput

The following scatter plots visualize the latency-throughput trade-off for each ISL/OSL scenario. Points toward the upper left (lower latency, higher throughput) are better. Each point represents a different concurrency level.

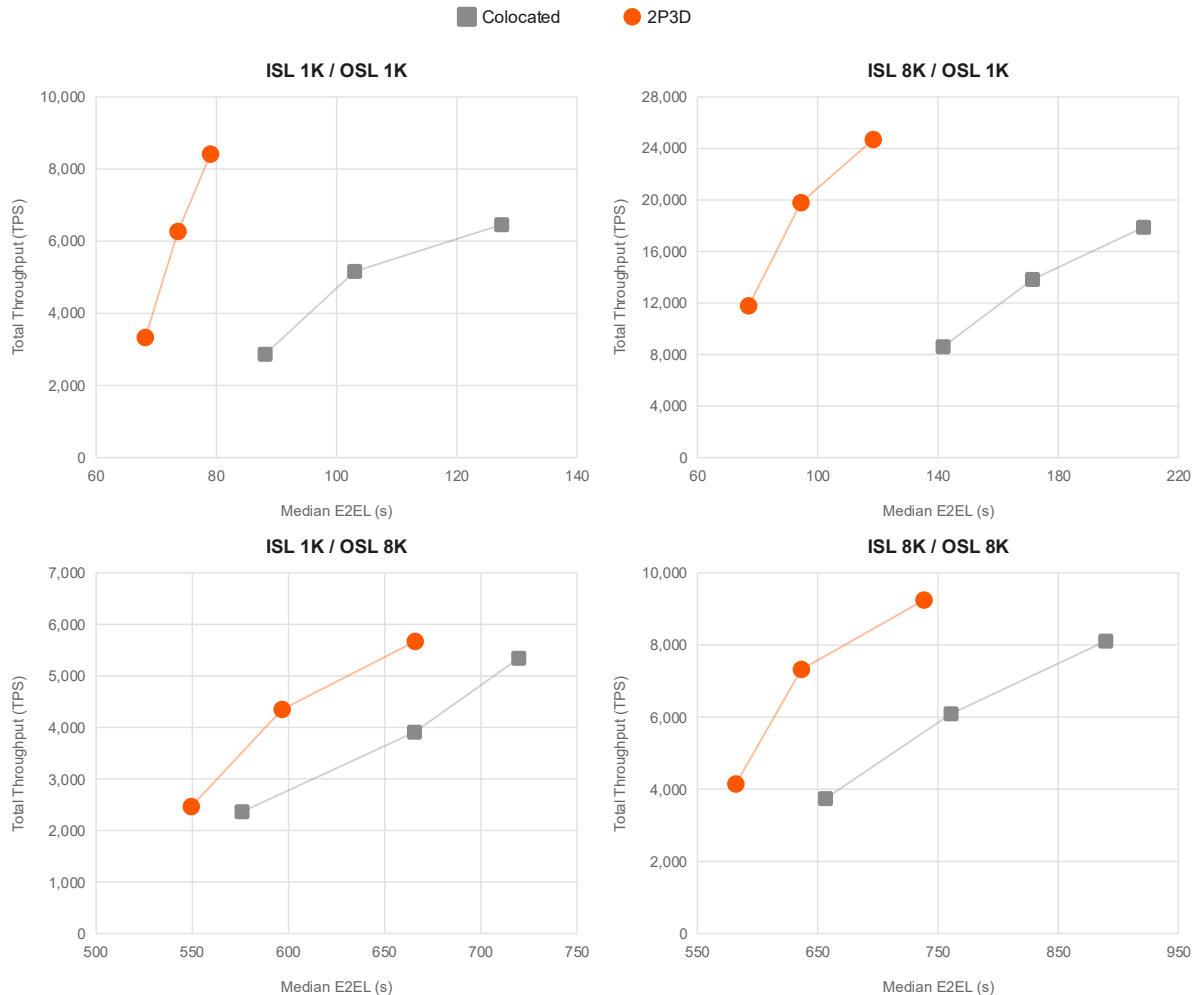


Figure 1. Latency vs. throughput for colocated and disaggregated 2P3D configurations. Upper left is better. Disaggregated 2P3D consistently achieves lower latency and higher throughput across all scenarios.

Cost Efficiency (Token/Dollar)

Since disaggregated serving uses the same total number of GPUs as the colocated baseline, throughput improvements translate directly into cost-per-token savings. The 2P3D configuration achieved a geometric mean of 107.57% token/dollar efficiency compared to the colocated baseline, meaning operators can serve 7.57% more tokens for the same infrastructure cost.

Results: Impact of the Prefill-to-Decode Ratio

Having established that disaggregated serving outperforms colocated serving, we now ask a follow-up question: how should the cluster's nodes be divided between prefill and decode? To answer this, we compare the 2P3D configuration (2 prefill + 3 decode) with 1P4D (1 prefill + 4 decode) on the decode-heavy 1K/8K scenario, where shifting a node from prefill to decode has the most visible impact.

End-to-End Latency: 2P3D vs. 1P4D (1K/8K)

CON	N_REQs	REQ_RATE	Colocated E2EL (s)	2P3D E2EL (s)	1P4D E2EL (s)
160	320	2	575.83	549.46	544.86
240	480	2	633.48	586.02	553.47
320	640	2	665.46	596.69	584.97
480	960	2	719.67	665.85	605.61
Geomean Improvement				1.08x	1.13x

Total Throughput: 2P3D vs. 1P4D (1K/8K)

CON	N_REQs	REQ_RATE	Colocated TPS	2P3D TPS	1P4D TPS
160	320	2	2,363.14	2,463.28	2,462.89
240	480	2	3,129.68	3,404.86	3,518.22
320	640	2	3,906.88	4,348.48	4,456.96
480	960	2	5,341.88	5,668.40	6,082.84
Geomean Improvement				1.08x	1.11x

On this decode-heavy workload, 1P4D consistently outperforms 2P3D. The additional decode node provides more aggregate decode capacity, yielding lower end-to-end latency and higher throughput — particularly at high concurrency where decode becomes the bottleneck. At CON=480, 1P4D achieves 6,082.84 TPS vs. 5,668.40 TPS for 2P3D.

Cost Efficiency Comparison

In token/dollar efficiency (geomean), 1P4D reached 111.07% while 2P3D achieved 107.57% relative to the colocated baseline. On decode-heavy workloads, the additional decode node translates directly into better cost efficiency.

Summary: Which Ratio to Choose?

	2P3D (2 Prefill + 3 Decode)	1P4D (1 Prefill + 4 Decode)
Stronger on	Prefill-heavy scenarios (long inputs)	Decode-heavy scenarios (long outputs)
E2EL improvement, 8K/1K (geomean)	1.80x	-
E2EL improvement, 1K/8K (geomean)	1.09x	1.13x
Token/Dollar, 8K/1K (geomean)	139%	-
Token/Dollar, 1K/8K (geomean)	108%	111%
Prefill capacity	Higher (2 nodes)	Limited (1 node)
Decode capacity	Moderate (3 nodes)	Higher (4 nodes)

2P3D delivers a 1.80x geomean latency improvement on the prefill-heavy 8K/1K workload, but on the decode-heavy 1K/8K workload its advantage narrows to 1.08x. In that regime, 1P4D pulls ahead with 1.13x E2EL improvement and 111% token/dollar efficiency, thanks to the additional decode node. In production, real workloads are rarely uniform — a mix of short queries, long-context RAG, and reasoning requests arrives simultaneously, and the optimal ratio may shift throughout the day as traffic patterns change.

This makes manual configuration of the prefill/decode ratio inherently fragile: a ratio tuned for peak-hour traffic may be suboptimal during off-peak, and vice versa. The challenge is not just choosing the right ratio once, but continuously adapting it.

Latency Stability: P99 Inter-Token Latency

One of the most impactful benefits of disaggregated serving is the dramatic improvement in tail latency. In a colocated setup, long prefill requests intermittently block decode steps, causing P99 inter-token latency (ITL) to spike to several seconds. This directly degrades user experience in streaming applications.

With disaggregated serving, prefill and decode never compete for the same GPU resources. As a result, P99 ITL drops dramatically:

Scenario	CON	N_REQs	REQ_RATE	Colocated P99 ITL (ms)	Disaggregated P99 ITL (ms)	Reduction
8K/1K	160	320	2	3,921.21	77.61	50.52x
8K/1K	320	640	3	4,085.65	87.38	46.76x
8K/1K	480	960	4	4,172.20	115.97	35.97x
1K/1K	160	320	3	997.97	72.55	13.76x
1K/1K	320	640	5	1,007.54	78.96	12.76x
1K/1K	480	960	6	1,039.30	84.23	12.34x
Geometric Mean						23.85x

This means that with disaggregated serving, users experience consistent, smooth token streaming even under mixed-workload conditions — a critical requirement for production chat and reasoning applications.

Conclusion

Prefill-decode disaggregation delivers clear, measurable gains over colocated serving for large-scale MoE model inference. On a 5-node AMD MI300X cluster running DeepSeek R1 671B:

- Both disaggregated configurations outperform the colocated baseline across all tested scenarios, with end-to-end latency improvements up to 1.84x and P99 inter-token latency reductions of 12–51x.
- 2P3D (more prefill nodes) excels at prefill-heavy workloads, achieving 1.80x geomean E2EL improvement on 8K/1K.
- 1P4D (more decode nodes) delivers better cost efficiency on decode-heavy workloads, reaching 111% token/dollar on 1K/8K.

However, the optimal prefill-to-decode ratio is not static — it depends on the workload's input/output length distribution and concurrency, both of which shift over time in production. Choosing the wrong ratio can leave either prefill or decode capacity underutilized, eroding the very gains disaggregation provides.

The MoAI Inference Framework addresses this by automating the configuration of disaggregated serving. Rather than requiring operators to manually select and maintain a fixed prefill/decode ratio, MoAI dynamically adjusts the allocation based on observed workload characteristics — along with expert parallelism, routing, and other distributed inference optimizations — so that operators can realize the full benefits of disaggregation on AMD Instinct GPU clusters without manual tuning.



To learn more, please visit our website (<https://moreh.io>) or contact us (contact@moreh.io).

Copyright ©2026 Moreh, Inc. All rights reserved.

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions, and typographical errors, and Moreh, Inc. is under no obligation to update or otherwise correct this information. Moreh, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and assumes no liability of any kind for the consequences or use of such information or for any infringement of patents. Moreh, Inc. reserves the right to make corrections, modifications, enhancements, improvements, and other changes to this information, at any time and/or to discontinue any service without notice.