

Name the Traffic Before You Name the Fabric

Sean Fraser

17 October 2026

Abstract

InfiniBand and RoCEv2 both move RDMA. Neither one repairs an inference job that did not need to leave the server. This paper separates scale-up links inside a machine from the 400 Gbit/s hop between machines, then sorts inference traffic into small synchronized collectives and large KV moves. NVIDIA’s July 2025 note on an eight-H100 NVLink node reports that a small all-reduce was about 23 percent of decode latency, and that a fused one-shot kernel cut end-to-end decode latency by about 27 percent. That experiment did not cross InfiniBand or RoCE. Meta’s March 2024 clusters, 24,576 H100s each, one fabric of each kind, 400 Gbit/s endpoints, ran large training jobs on both after the unoptimized large-cluster case had ranged from 10 to 90 percent AllGather utilization. Those are training reports. They still rule out the claim that one logo is the inference answer. The decision starts with the message, the tail, and who will operate the failure.

Introduction

Fabric arguments usually arrive as logos. One quote is for InfiniBand. Another is for RoCE, often with a claim that the team already knows Ethernet. Both quotes can be real. Neither quote knows whether the next token is stalled inside one server.

My current inference machine is one Threadripper PRO 7960X with one RTX PRO 6000. It is not a cluster. I have not run an InfiniBand-versus-RoCE bake-off. I come to this from designing SASE and RoCEv2 networks, with experience around InfiniBand-adjacent infrastructure. That limit is useful. You cannot pick a scale-out fabric for a workload that never crosses a NIC.

The practical claim is narrow. Keep communication inside the fastest scale-up domain that can hold it. Choose a scale-out fabric only after the workload, the latency objective, the contention pattern, the failure model, and the operator capacity are explicit. InfiniBand and RoCEv2 can both carry serious jobs. They do not fix an architecture that crosses nodes because the purchase came first.

The Boundary That Makes the Logo Irrelevant

A GPU does not, by itself, perform a distributed collective. Several GPUs in one server can talk over PCIe or NVLink. Only a message that leaves the host hits the InfiniBand or Ethernet fabric. If tensor parallelism fits in the NVLink domain, the external switch is a spectator.

NVIDIA’s July 2025 note is the cleanest published illustration I have for this, and it is not a fabric comparison. They ran Gemma 2 decode with eight-way tensor parallelism on one node of eight H100s connected by NVLink. The all-reduce in the tensor-parallel layers accounted for about 23 percent of end-to-end decode latency. Dependencies stopped them from overlapping that collective

small all-reduce stays left; a KV move may have to cross

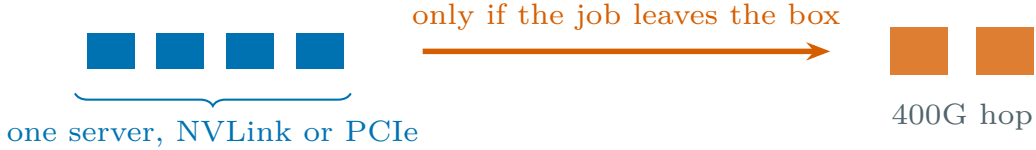


Figure 1: Four GPUs in one server, then a hop that exists only if the job leaves the box. A small all-reduce can finish on the left. A KV migration may have to take the arrow. Schematic of the boundary, not a cabling diagram.

with the nearby compute. A custom one-shot all-reduce, fused with RMSNorm and a pointwise add, gave about a 3 times kernel-time speedup over the stand-alone all-reduce kernel, and about 27 percent better end-to-end decode latency [Shankar et al., 2025].

The post writes the message size as “~30 KB/s” and, in the next paragraph, says the ring algorithm is bandwidth optimal for messages larger than 10 MB. Those two units do not agree. I will not silently correct the post, and I will not treat “KB/s” as a measured rate. The operational fact does not need the unit resolved. A small, synchronous reduction inside one NVLink node was about a quarter of decode time. The repair was the collective, not a leaf switch.

Buying 400 Gbit/s optics for that job would have been the wrong category of purchase. The bytes did not leave the chassis.

Two Fabrics, One Training Result, and the Inference Gap

Meta described two clusters of 24,576 H100 GPUs each. One used RoCE on Arista 7800 switches, with Wedge400 and Minipack2 rack switches. The other used NVIDIA Quantum-2 InfiniBand. Both tied 400 Gbit/s endpoints. Meta reports using both for large generative-AI workloads, including ongoing Llama 3 training on the RoCE cluster, and says careful co-design removed network bottlenecks for those jobs [Lee et al., 2024].

The performance paragraph is the one to keep. It is about work, not about a winner. Small clusters reached more than 90 percent of expected AllGather bandwidth out of the box. Large clusters, before the full-system work, ranged from 10 percent to 90 percent. After scheduler changes that respected topology, routing changes, and NCCL changes, large-cluster utilization returned to the 90 percent band [Lee et al., 2024]. That is a training collective at sizes where a bandwidth roofline is supposed to be visible. It is evidence that an untuned large RoCE or InfiniBand fabric can look broken, and that the repair was scheduling, routing, and the collective library together. It is not an inference P99.

I will not upgrade that training result into “RoCE equals InfiniBand for token latency.” I also will not upgrade it into “Ethernet cannot do this.” The post already closed the second claim for the workloads they ran. The first claim is the one that gets smuggled into a later meeting, usually without a decode trace.

Inference Does Not Have One Packet

Training collectives are often large and synchronized. Inference is a mix, and the mix depends on the parallelism you chose.

Tensor-parallel decode wants a small reduction, often on the critical path, often too dependent to overlap. The NVIDIA note is that case, inside one node [Shankar et al., 2025]. If that pattern is forced across a 400 Gbit/s hop, you have added switch latency, congestion, and a failure domain to a latency-sensitive reduction. Sometimes the model does not fit otherwise. Then the hop is a capacity decision. Judge the fabric on the tail of that small message, not on a bandwidth test of a large flow. A 1 GB flow is an example of the wrong test. It is not a measurement from this paper.

Expert parallelism wants all-to-all traffic with uneven destinations. A bandwidth chart with one flow will not show the incast.

Disaggregated prefill wants a third shape. The prefill pool builds KV state. The decode pool has to receive it. DistServe assigns prefill and decode to different GPUs. On the applications and latency limits in that paper, it serves up to 7.4 times more requests or a 12.6 times tighter SLO than the state-of-the-art systems it compares, while staying inside the limits for more than 90 percent of requests [Zhong et al., 2024]. The abstract says those existing systems colocate the two phases. vLLM’s stable documentation of the same idea says disaggregated prefill does not improve throughput. It exists so time to first token and inter-token latency can be tuned apart, and so a long prefill does not inflate the tail gap between tokens. The page also says chunked prefill can chase the same tail, and that picking the chunk size in practice is hard [vLLM Project, 2026]. Mooncake separates the pools and treats KV storage as its own tier across CPU, DRAM, SSD, and NIC [Qin et al., 2025]. All three are reasons to expect KV-sized transfers. None of them is a reason to skip measuring the transfer.

A lower bound on an unhidden transfer of M bytes at effective rate R is M/R . For a hypothetical 1 GiB prefix at 25 GB/s, that is 43 ms of serialization before setup, copies, or contention. If prefill was 50 ms, the transfer is the request. If prefill was 400 ms and the send overlapped, the same 43 ms can disappear from time to first token and still appear if the decode side is not ready to receive. Overlap is a schedule. It is not a property of the cable.

I used 25 GB/s as an assumption, not as a measured goodput on either fabric. A 400 Gbit/s link is 50 GB/s of raw bit rate before encoding and protocol overhead. Half of that as an achieved rate is a planning guess until a counter says otherwise. Do not put the guess in a contract.

A Serialization Table

Assumptions, not measurements. The rate is whatever the counters say after overhead, not the rate printed on the transceiver. Times are M/R only. They omit queueing. GiB is 2^{30} bytes. GB/s in the header is 10^9 bytes per second, so the two units are not the same convention. The table uses that mix on purpose, because link rates are quoted in decimal gigabytes and caches are quoted in binary gibibytes. Do not “correct” one into the other without saying so.

Payload	Case	10 GB/s	25 GB/s	45 GB/s
1 MiB	small collective	0.10 ms	0.04 ms	0.02 ms
64 MiB	short KV	6.7 ms	2.7 ms	1.5 ms

Payload	Case	10 GB/s	25 GB/s	45 GB/s
1 GiB	KV prefix	107 ms	43 ms	24 ms
8 GiB	long context	859 ms	344 ms	191 ms

The first row is why a bandwidth test can look fine and still be the wrong test. A 1 MiB message at 45 GB/s is a rounding error next to a 50 ms token budget, if you set that budget. I did not measure 50 ms. The comparison is an assumption. The transfer is still the wrong test if the message is synchronized, repeated every layer, and unable to overlap. Eight layers of a 0.02 ms transfer are nothing. Eight layers of a launch, a barrier, and a congested queue are the NVIDIA story, moved onto a worse link [Shankar et al., 2025]. The last row is why a KV migration of a long context is a different test from a collective benchmark. If prefill is 200 ms and the transfer is 344 ms and you do not overlap them, the fabric is the time to first token. Those two times are the table’s assumptions, not a trace. No logo changes that arithmetic. Overlap might. Measure the overlap.

Failure Is a Different Surface on Each Side

RoCEv2 gives you routed Ethernet and the operational habits that come with it. It also gives you pause frames, ECN marks, and congested queues that can leave the link up while tokens stall. I am not reciting a switch manual I did not re-read for this draft. The point that survives without a page number is simpler. A bandwidth test can pass while the tail you care about is pause duration. If the runbook does not put pause, ECN, retransmission, and the request’s inter-token gap on one timeline, you cannot tell a bad collective from a bad optic.

InfiniBand removes a class of Ethernet tuning. It does not remove the subnet manager, a dirty cable, a downshifted link, or a collective that still uses a ring on a small payload. The NVIDIA result was on NVLink, which is a third path [Shankar et al., 2025]. Treating “RDMA” as one technology is how all three get blamed for each other’s faults.

The comparison I will make is not “complex versus automatic.” It is which failure the on-call already knows how to see. A team that lives in Ethernet telemetry may spend fewer hours making RoCE decent than staffing an InfiniBand island. A team that wants the small-message tail, and does not want to own pause frames, should not accept a RoCE proof that only showed a large AllReduce. Meta’s published graph is a large AllGather story [Lee et al., 2024]. Ask for the small one.

A Serving Test, If the Job Must Cross Hosts

Do not start with the hop. Run the fit test first. I did not run it on a cluster for this paper. The steps are the decision rule.

1. Write down the GPU count in one NVLink or PCIe domain, and the free device memory after the weights are loaded.
2. Compute the KV bytes for the batch and context you claim to serve. For one request, $M(S) = 2LS h_{kv} d_h b_{kv}$. A batch of B similar requests multiplies by B if nothing is shared. Use the engine’s reported KV if it disagrees with that formula, and record the disagreement.
3. Add a reserve. Measure resident device memory on one short request. Subtract the weights and that request’s KV. What remains is the reserve. Do not invent it.

4. Stop if weights, $B \times M(S)$, and the reserve fit in that domain, and if a placement that keeps every tensor-parallel rank inside the domain leaves the NIC idle during decode. The collective did not leave the server. Do not buy a scale-out fabric for that placement.
5. Treat the hop as required only if the model does not fit unless ranks cross hosts, or a KV move must cross hosts. Then run the serving test below.

If the job must cross, hold the rest still. Goodput in this test means completed logical requests per second that meet limits you chose before the run. The limits are time to first token, and the maximum gap between output events, not the mean gap. A mean can hide a hiccup. Report the median and a high percentile of those gaps as well.

1. Match the GPU, the NIC placement, the model, the quantization, the parallelism, the request trace, and the background load.
2. Change the fabric stack. Record every other deviation. There will be one, and it will try to become the result.
3. Report time to first token, the event-gap distribution, and goodput at the concurrency you claim to serve.
4. Segment by prompt length, so a change in the trace does not look like a change in the switch.
5. Add the counters that can explain a stall: collective time by message size, NIC throughput, queue depth, pause, ECN, drops, retries, and GPU idle.
6. Break a link on purpose. Count requests lost, time to restore, and whether the tail recovered or only the average did.

A fabric that wins an empty-network throughput run and then misses the high percentile under the mix you actually serve has lost. A fabric decision that skips the failure drill is a purchase. It is not an evaluation.

What This Paper Will Not Conclude

I will not rank InfiniBand and RoCEv2 for a cluster I have not traced. Meta showed that both can carry large training jobs once the system around them is tuned, and that the untuned large case can be startlingly bad [Lee et al., 2024]. NVIDIA showed that a small decode reduction on NVLink can dominate latency, and that the collective implementation is the lever [Shankar et al., 2025]. DistServe, vLLM’s disaggregation note, and Mooncake show why a KV move is a different flow from that reduction, and why splitting phases is not free throughput [Zhong et al., 2024, vLLM Project, 2026, Qin et al., 2025].

Yoon and Yu’s preprint on an optical KV-cache network says a mechanical optical circuit switch needs tens of milliseconds to reconfigure, and that their simulation tolerates the 50 ms delay of a 384-port switch by hiding it inside prefill [Yoon and Yu, 2026]. That is a simulation, not a fabric ranking. If your constraint is that switch delay, read the preprint on its own terms. Do not let it pick a 400 Gbit/s vendor.

Ultra Ethernet specifications are outside the evidence I re-fetched for this draft. A specification is not a NIC. I am leaving it out rather than decorating the conclusion with a future product.

A Fit Example You Have to Replace

The fit test is a procedure. This section is one substitution into it, so the arithmetic is visible. Every capacity number here is an assumption. Replace it with the card and the engine you have. I did not run the placement.

Assume eight GPUs in one domain, 80×10^9 bytes of device memory each. Assume a 70-billion-parameter model stored as one FP16 value per parameter and nothing else, so the weights are 140×10^9 bytes. One GPU cannot hold those weights. A second identical GPU, used as a replica, cannot hold them either. An even split across the eight puts 17.5×10^9 bytes of weights on each GPU. That split fits. It fits only if the engine actually divides the weights evenly, and only if the ranks stay in the domain.

KV is a separate question. Use the grouped-query toy of 131072 bytes per token only as a placeholder. It is not this model’s configuration unless you measure that it is. One request at 8192 tokens is then 1 GiB. Sixteen such requests are 16 GiB if the KV is replicated on every rank, and 2 GiB per rank if the engine shards that KV by eight. Record which of those the engine does. Do not choose the smaller one because it fits.

The case that fails inside the box is the long concurrent one, with KV replicated. Sixteen requests at 32768 tokens are 16×4 GiB of KV, 68.72×10^9 bytes, plus 17.5×10^9 bytes of weights, which is 86.22×10^9 bytes. That is over the assumed 80×10^9 byte GPU. The same KV sharded by eight is 8.59×10^9 bytes per GPU plus the weights, 26.09×10^9 bytes, which fits. The last token count that fits in the replicated case, with no reserve, is 29802 tokens. Token 29803 does not. A reserve makes the limit lower. Measure the reserve. Do not skip it because 29802 was tidy.

If this placement fits and the NIC is idle during decode, stop. The scale-out fabric is not the constraint. If it fits only when ranks cross hosts, the hop is required, and the serving test below is the next step. A 400 Gbit/s quote does not change the 86.22 against 80 comparison.

Three Flows, Three Tests

Inference in this note is three flows. A test that suits one of them can miss the other two. The sizes below are the classes already used in this paper, not a new trace.

A tensor-parallel decode reduction is small and synchronized. The NVIDIA post’s all-reduce, about 23 percent of Gemma 2 decode on eight H100s in one NVLink node, is the published case [Shankar et al., 2025]. The right test is collective time at that message size, on the critical path, with the overlap you actually get. A bandwidth run at 1 GB, or at any large flow that is not this message, can pass while this reduction still dominates the step. The 1 GB figure is an example of the wrong test. It is not a measurement.

A KV move has the size of the cache you just built. The serialization table is the lower bound once you have M and an effective R . The right test is time to first token with the transfer included, and the same time with the transfer overlapped, on the prompt lengths you serve. An empty-network throughput run does not produce that pair.

Expert all-to-all is uneven. Some destinations receive more than others. A single flow between two ranks will not show the incast. The right test is the slowest destination at the concurrency you claim, with the background traffic left on. I do not have a published expert-parallel inference trace

in the sources for this draft. I am not going to invent one. The absence is a reason to measure your own, not a reason to reuse the AllGather graph from the training cluster [Lee et al., 2024].

Write the class down before you book the bake-off. If the job is the first class and it fits in the NVLink domain, cancel the bake-off.

Overlap Is a Schedule You Can Miss

M/R is a lower bound on an unhidden transfer. It becomes invisible in time to first token only if three conditions hold at once. The send starts while prefill is still running. The prefill lasts at least as long as the part of the transfer you hoped to hide. The decode side is ready to receive when the bytes arrive.

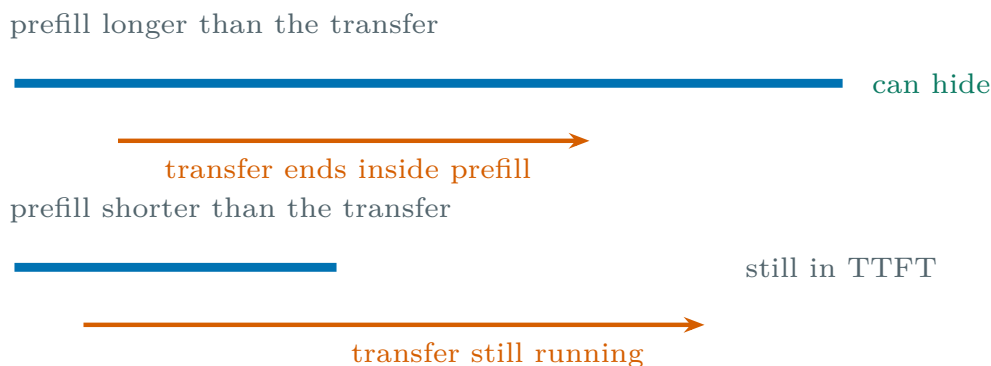


Figure 2: Two schedules for the same transfer. The upper prefill outlasts the send, so the transfer can leave time to first token. The lower prefill ends first, so the transfer is still running. Schematic, not a trace.

The 43 ms figure in the serialization section is 1 GiB at an assumed 25 GB/s. It is not a measured goodput. Set it next to a prefill you measured, not next to a prefill you hoped for. If that prefill is 400 ms and the send overlaps it, 43 ms can leave the time-to-first-token total. If the prefill is 30 ms, the same 43 ms cannot hide inside it. Thirty milliseconds is an assumption, used only to show the inequality. If the receiver is late, the 43 ms returns after prefill even when the prefill was long.

Yoon and Yu hide a mechanical switch delay of tens of milliseconds inside prefill in simulation, and they report that the simulation tolerates a 50 ms reconfiguration [Yoon and Yu, 2026]. That result depends on the prefill outlasting the delay. A short prompt does not give them that cover. Do not import the 50 ms figure into a packet-fabric purchase. It is a statement about their simulated circuit switch, on their traces.

Overlap is therefore a line in the trace: prefill start, send start, send end, receive ready, first token. If those timestamps are missing, you do not know whether M/R was hidden. A lower average is not a substitute for the line.

The Row the Lab Book Needs

A bake-off fails quietly when one deviation becomes the result. The row is how you stop that. One row per candidate. Fill it before you look at which fabric was faster.

1. Write the date, the GPU, the NIC, the placement, and whether every tensor-parallel rank sat in one domain.
2. Write the model revision, the quantization, the parallel degree, and whether KV was replicated or sharded.
3. Write the request-trace identifier and the prompt-length buckets.
4. Write the fabric stack, and every deviation from the other candidate.
5. Write the median and a high percentile of time to first token, the maximum event gap, and goodput, using the definition in the serving test.
6. Write pause, ECN, drops, retries, and GPU idle next to those latency figures, on one timeline.
7. Write the link-failure result: requests lost, time to restore, and whether the tail recovered.

A row that lacks the deviation line is not a comparison. A row that lacks the failure drill is a purchase memo. Keep the old row. You will need it when the first explanation blames the logo.

A Reserve Moves the Token Limit

The 29802-token figure in the fit example used no reserve. A serving process also holds activations, the kernel workspace, and whatever else the engine allocated. If you measure an 8×10^9 byte reserve, the free bytes after the even weight split are $80 \times 10^9 - 17.5 \times 10^9 - 8 \times 10^9 = 54.5 \times 10^9$. Sixteen replicated requests at 131072 bytes per token then fit through 25987 tokens. Token 25988 does not. The 8 billion byte reserve is an assumption, so you can see the move. Measure the reserve on one short request and substitute it. Do not keep 29802 once a reserve exists.

The same arithmetic with sharded KV still fits at 32768 tokens in the example above. That is why the engine's KV placement belongs in the lab-book row. A replicated cache and a sharded cache are different fit tests. Writing "eight GPUs" without that distinction will send you to a fabric quote for a placement that still fits in the box, or keep you in the box for a placement that does not.

If the measured reserve plus the weights already exceed one GPU, stop. The context length is not the constraint yet. The model does not fit. A replica does not fix that. A shard might. The hop comes only after the shard, if the shard itself does not fit in the domain.

Read the two limits side by side before you name a fabric. With no reserve, sixteen replicated requests fit through 29802 tokens. With an assumed 8×10^9 byte reserve, they fit through 25987. The difference is 3815 tokens. That is not a small rounding error, and it is still not a measurement. A team that quotes the no-reserve figure in a purchase review has skipped step 3 of the fit test. Put the measured reserve in the lab-book row. Recompute the token limit. Then decide whether the job leaves the box.

Do the same comparison for one request before you do it for sixteen. One request at 131072 bytes per token, KV replicated, with the even weight split and no reserve, fits until the KV alone consumes the 62.5×10^9 free bytes. That is 476837 tokens. Nobody is serving that window on this toy. The failure mode that matters is concurrency, not a single impossible context. Sixteen requests fail near 30,000 tokens. One request does not. A fabric quote written for "long context" without the batch size is answering a different problem from the one in the fit example.

If the engine shards KV by eight, both of those limits move out by about eight, until some other reserve appears. Record the shard. Then recompute. A diagram of eight GPUs does not tell you which limit you are on.

How to Read the Two Figures

The boundary figure is the first question. Four GPUs sit in one server. The arrow exists only if the job leaves that server. A small all-reduce can finish on the left. A KV move may have to take the arrow. The figure does not say which of those your job is. The fit test does. If weights, KV, and the measured reserve fit in the domain, and the NIC is idle during decode, you are still on the left. Do not price the arrow.

The overlap figure is the second question, and only after the first one has failed. The upper row is a prefill that outlasts the transfer. The lower row is a prefill that ends first, so the transfer is still in time to first token. Both rows use the same arrow. The difference is the schedule, not the cable. If you do not have the five timestamps, prefill start, send start, send end, receive ready, and first token, you cannot tell which row you are on. A lower average is not those timestamps.

Read them in that order. Boundary, then fit numbers, then overlap, then the fabric bake-off. Reversing the order is how a 400 Gbit/s quote becomes the answer to a job that never left the chassis. The NVIDIA decode result sits on the left of the boundary figure. It is not evidence about the arrow.

Limits

I did not operate either fabric for this paper. The Meta and NVIDIA documents are the measurements. Mine are the boundaries and the serialization sketch. The 23 percent figure is an eight-GPU NVLink decode experiment on Gemma 2, as reported [Shankar et al., 2025]. It will not survive unchanged on a different model, a different parallelism, or a hop the authors did not have.

No cost model is included. Port prices move. A price typed from memory would be fiction. The operator-hour comparison is a judgment from other networks I have worked on. It is not a table.

AI assistance was used to retrieve the pages, draft the note, and compile the PDF. I would re-open the Meta figure and the NVIDIA post before quoting them in a purchase review. This note is not peer reviewed.

Conclusion

Keep small, synchronous decode traffic inside the fastest domain that can hold it. When a KV move, or a model that does not fit, forces a hop, test that hop with the message you will actually send, under contention, and with the failure counters the on-call can read. InfiniBand and RoCEv2 can both be the right answer after that test. Neither is the right answer before it.

References

Kevin Lee, Adi Gangidi, and Mathew Oldham. Building meta’s genai infrastructure. Engineering at Meta, 12 March 2024. Retrieved 17 October 2026, 2024. URL <https://engineering.fb.com/2024/03/12/data-center-engineering/building-metas-genai-infrastructure/>.

- Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation — a KVCache-centric architecture for serving LLM chatbot. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*, pages 155–170, Santa Clara, CA, February 2025. USENIX Association. ISBN 978-1-939133-45-8. URL <https://www.usenix.org/conference/fast25/presentation/qin>.
- Jaya Shankar, TJ Xu, and Tejash Shah. Optimizing for low-latency communication in inference workloads with jax and xla. NVIDIA Technical Blog, 18 July 2025. Retrieved 17 October 2026, 2025. URL <https://developer.nvidia.com/blog/optimizing-for-low-latency-communication-in-inference-workloads-with-jax-and-xla/>.
- vLLM Project. Disaggregated prefilling (experimental). vLLM documentation, stable. Retrieved 17 October 2026, 2026. URL https://docs.vllm.ai/en/stable/features/disagg_prefill/.
- Seongwon Yoon and Shimeng Yu. An optical kv cache network for disaggregated llm serving, 2026. URL <https://arxiv.org/abs/2610.18444v1>. Version 1, 13 October 2026; user-supplied PDF.
- Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving, 2024. URL <https://arxiv.org/abs/2401.09670>.