Episode Book

EPISODE #6349

Splitting Model Layers Across the Internet

PUBLISHED	29 October 2026 at 17:36 UTC
DURATION	28:31
LISTEN	myweirdprompts.com/episode/distributed-llm-inference-latency-73369/
HOSTS	Corn, Herman, Dorothy, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.25
SOURCES	14 cited by the research (listed at the end)
BOOK GENERATED	30 October 2026 at 07:19 UTC

SYNOPSIS

Running large language models across distributed consumer and rented GPUs turns network round-trips into the primary performance bottleneck. While frameworks like Shard, ShardFlow, and Mesh-LLM prove that scattered hardware can execute massive models without sacrificing deterministic accuracy, naive layer splitting across the public internet slows generation to a crawl. Overcoming those delays demands asynchronous pipelining, speculative decoding, and optimized routing paths, making the scheduler far more critical to real-world throughput than raw compute speed.

TRANSCRIPT



Corn

The cable gets rather more expensive once it leaves the building. Daniel wants to stretch shared inference across the internet, with his GPU doing part of one request and a rented cloud GPU doing the rest. Not sending easy prompts one way and difficult prompts another. Actually sharing the computation, and doing it to save money.



Herman

The distinction matters because a gateway can choose a machine once and then get out of the way. Here the network can stay inside the generation loop for the entire answer.



Corn

He means layers on either side, or a small local model drafting for a remote model to check, or anything that makes the paid machine do less. Even if both machines have the same card. I can see you already mentally renting six.



Herman

Only because somebody has made six work.



Corn

Your restraint remains theoretical.



Herman

Six scattered GPUs generating one answer is real. Whether renting them is cheaper than renting a suitable complete machine is a different claim, and the published demonstrations don't settle it.



Corn

That's the distinction I want kept on the invoice. Working software, useful performance, actual savings. Three very different things to sell me.



Herman

Shard, ShardFlow, Mesh-LLM, and coder nine zero three's split-inference all give us working implementations to discuss. They're not interchangeable products. Some pool geographically scattered hardware, some target free notebook GPUs, and one is explicitly about privacy-aware local and cloud inference.



Corn

None sounds like a button marked reduce bill.



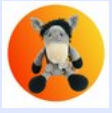
Herman

The closest explicitly cost-motivated experiment is splitcompute, using browser WebGPU. It calls itself a work in progress and doesn't publish performance benchmarks. The mature claim here is that computation can be shared. The economic claim still needs a controlled comparison.



Corn

Before you get carried away, shared computation can mean wildly different traffic. Sending candidate words sounds cheap. Sending everything the model remembers about a long document sounds less cheap.



Herman

There are three useful architectural families. Layer splitting passes intermediate activations between blocks of the same model. Speculative decoding lets a small drafter propose a sequence that a larger model verifies. Prefill and decode disaggregation moves the prompt-processing phase to one device and the generation phase to another.



Corn

Those can overlap, presumably. The big verifier could itself be split across several machines.



Herman

Shard does precisely that. Its README puts the constraint bluntly. Over WAN the round-trip is the scarce resource, not compute, so speculative decoding, marginal in a datacenter, becomes the whole game.



Corn

Marginal sounds a bit sweeping for datacenters.



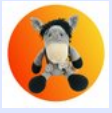
Herman

It is their wording. Speculation can matter there too. Their point is that over the public internet, avoiding repeated waits can outweigh making an individual GPU operation faster. What would you expect to cross the wire if I divided a transformer halfway through?



Corn

The partially processed representation, not the original words. Although I'd expect you to accidentally send diagnostic logs larger than the representation.



Herman

The logs would be beautifully timestamped. Your first answer is right. Each machine holds a contiguous range of layers. It runs those layers and sends their output to the next stage. The token's computation travels through the stages before the next token can depend on its result.



Corn

Beautiful timestamps don't make the packet arrive earlier.



Herman

Plain autoregressive decoding exposes that dependency brutally. You can't simply generate the final answer on both machines in parallel, because the next token depends on what the whole model decided about the previous one. Pipeline throughput across many requests is another matter. For one request, the network waits are right in your path.



Corn

Buying a second GPU doesn't buy a second next word.



Herman

Shard's headline demonstration uses GLM five point two, a seven hundred forty-four billion parameter model, across six RTX PRO six thousand cards in separate American states. It reaches about thirty tokens a second.



Corn

That is a model most of us aren't slipping onto the spare desktop. Capacity is already doing quite a lot of the justification.



Herman

It uses NVFP four quantization and divides seventy-eight layers into equal blocks. Each node gets thirteen layers. The measured network edge round-trips range from twenty-two to seventy-five milliseconds.



Corn

Perfectly ordinary internet delays, then. Not six machines secretly in the same rack wearing different state flags.



Herman

It also reports DeepSeek V four Flash across six RTX fifty-ninety cards spread through Poland, Czechia, Denmark, and Estonia. About thirty tokens a second again, with output bit-identical to the single-machine result in that test.



Corn

Useful distinction. The network didn't make the model worse. It made getting the same answer harder.



Herman

Deterministic greedy output lets them check that directly. It doesn't establish every possible sampling configuration, but it does show that distributed execution needn't mean approximate answers.



Corn

I would have guessed the first working split would already look respectable, and the cleverness would polish it. These numbers apparently conceal a much less flattering first attempt.



Herman

The plain WAN decoding baseline is one point eight seven tokens a second. Adding deep-draft speculative decoding moves it to one point nine nine.



Corn

An expensive way to remain under two.



Herman

That tiny gain is important. Speculation by itself wasn't sufficient. They then changed the return path, using ring direct-return, and reached two point nine four tokens a second. Less unnecessary routing, but still far below the headline.



Corn

You sound personally offended by the return path.



Herman

I admire a removed detour. You'd admire it too if every sentence you spoke had to visit a relay before you could choose the next word.



Corn

I'd put fewer words through it.



Herman

Async pipelining is the big jump, to sixteen point six tokens a second. The system overlaps work and communication rather than exposing all the waits serially. Their accounting puts WAN time down around five percent of the loop.



Corn

Hm.



Corn

That changes what I'd buy. If plain splitting is dreadful and speculation alone barely moves it, the product I'm evaluating is the whole scheduler. Not the promise that somebody supports distributed layers.



Herman

Yes, and then the local drafting implementation becomes worth optimizing. CUDA-graphing the draft takes the system toward thirty tokens a second. From the original baseline, that's roughly a sixteenfold improvement, assembled from separate fixes.



Corn

The headline has several engineering departments hidden inside it.



Herman

Would you tolerate the baseline if it let you run a model you otherwise couldn't fit?



Corn

For a batch job, maybe. Waiting isn't automatically a defect, Herman. Paying by the hour while the expensive part waits is. I could leave a task overnight and still dislike discovering most of the rental bought empty pauses.



Herman

That's where your preference for patience stops being an economic answer.



Corn

Patience is free. The GPU isn't.



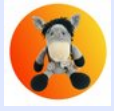
Herman

ShardFlow brings this down to hobbyist scale. It targets resources like Kaggle and Colab GPUs. Its Qwen two point five seven billion setup splits the model into two blocks of fourteen layers, hosted on Kaggle T fours.



Corn

Free-tier pooling is a very different purchase decision. If I already have access to the cards, cumbersome coordination can beat having no machine at all.



Herman

The path in the example runs between Iowa and Oregon through an Ohio relay. Eighty-six milliseconds of round-trip latency. Without speculation it reports four point nine two tokens a second.



Corn

Ohio is doing quite a lot of travelling on behalf of that sentence.



Herman

Adding speculation gets it to twenty-eight point one tokens a second, with about four accepted tokens per round. There is also a small local drafter, so the result combines layer splitting with speculative generation rather than proving that a simple two-stage pipeline is fast.



Corn

That's the comparison I'd want labelled prominently. Otherwise someone rents the two cards, copies the layer placement, and wonders where the other twenty tokens went.



Herman

Their earlier gateway-based design is more instructive still. It achieved two point two seven tokens a second, less than half the comparison baseline. Every token was taking multiple intercontinental network round-trips.



Corn

The gateway became a tollbooth that also moved away.



Herman

The architecture description says every token suffers two or more intercontinental round-trips, adding over three hundred seventy milliseconds and rendering GPU compute speed irrelevant. That is their own diagnosis, not an outside takedown.



Corn

A faster card would just reach the queue sooner.



Herman

Placement and topology become part of the inference engine. Even where the coordinator runs can matter. Shard reports that moving its layer-less coordinator closer to the swarm cut latency by around forty percent.



Corn

The cheapest machine had the most expensive address.



Herman

Before we push further, that's a lot of moving parts. We've got Mum calling with a rather different perspective, which I'm delighted to hear.

D

Dorothy

Hello dear, it's me, it's Dorothy's History of Toast.

D

Dorothy

Sliced bread was first sold in July nineteen twenty-eight, in Chillicothe, Missouri, on a machine built by Otto Rohwedder, and his first prototype and plans had burned in a factory fire in nineteen seventeen.

D

Dorothy

Eleven years, sweetheart. You build the thing, it burns, and you wait eleven years. That's the history of toast for you, mostly waiting.

D

Dorothy

Anyway, I've got a load of washing on, so I'll let you go. Bye bye now.



Corn

Bye, Mum. The coordinator placement matters even when the actual model blocks stay exactly where they are.



Herman

Mesh-LLM takes a useful position on that. Its Skippy stage splits distribute contiguous layer ranges, and peers download packages containing only their assigned GGUF fragments. You don't need every participant downloading the whole model.



Corn

Sensible. Distributed execution shouldn't require a ceremonial copy of every weight.



Herman

It's actively released Rust software under Apache two point zero, with an October release. That makes it a much more substantial artifact than an architecture diagram. Still, active releases aren't proof of a managed service's reliability guarantees.



Corn

Its default is the revealing part. If one machine fits the model, use that machine and avoid stage traffic. The people building distributed inference don't seem desperate to distribute every inference.



Herman

Single-machine fit first is the project's stated policy. Splitting expands what the available fleet can host. It doesn't automatically improve a model that already fits comfortably on one participant.



Corn

I'd predict you would ignore that default once, just to see the stages light up.



Herman

Once is an offensively low estimate.



Corn

At least you budget honestly.



Herman

The two-machine split-inference implementation is especially close to Daniel's local and rented GPU scenario. It runs the beginning and end of Mistral seven billion locally on Apple's MPS backend, with the middle layers on a remote CUDA GPU.



Corn

Keeping the ends local means the remote side doesn't directly receive the original token identifiers or produce the final word choice?



Herman

That is the intended boundary. Intermediate representations travel over binary WebSocket connections, with session KV caches and lookahead speculation. The local device holds two layers at either end. Most of the transformer is still remote.



Corn

Then local participation doesn't necessarily mean local majority. Four little slices around a rented middle could be valuable for privacy while barely reducing the paid compute.



Herman

Correct. Well... valuable for a particular privacy threat model. Hidden states can reveal information too. We should return to that before recommending this for confidential documents.



Corn

I wasn't about to mail the cloud my accounts in vector form and declare them invisible.



Herman

The cleaner performance idea is the drafter and verifier. The small model generates several candidate tokens locally. The large model checks that proposed sequence in a pass that can exploit parallel work across its positions.



Corn

It still has to read and compute over those candidates. We're reducing expensive serial waits, not making remote verification free.



Herman

That's the crucial economic distinction. The verification pass can be much more efficient than generating each candidate serially, but it consumes remote compute. Savings depend on acceptance, batching, implementation, and how the provider charges.



Corn

If the first proposed token is wrong, all the beautiful later guesses are waiting behind a closed door.



Herman

The accepted prefix determines how much progress you keep. The verifier supplies the correction at the rejection point, and drafting continues from there. Good agreement can amortize a network exchange over several final tokens. Poor agreement wastes local work and gives little amortization.



Corn

My guess is code does well when it repeats structures. A drafter can predict a familiar closing sequence without understanding the whole program.



Herman

That matches the n-gram lookahead results in the privacy-aware implementation. Average acceptance is only around one point two to one point three tokens per step, but code can reach seven.



Corn

Quite a distance between the ordinary answer and the demonstration screenshot.



Herman

Its Mistral run gets roughly nine tokens a second over an eighty millisecond WAN connection. For NeMo twelve billion, the local memory requirement is only four point nine gigabytes. That's a meaningful capacity advantage for a modest local device.



Corn

The model isn't local, though. That's enough local memory to participate, not enough to keep working when the rented machine disappears.



Herman

Right. The reported outputs match sequential greedy decoding token for token. The latency model projects fifteen to nineteen tokens a second if round-trip latency drops to twenty milliseconds, but that projection shouldn't be presented as the measured WAN result.



Corn

Does a better drafter always mean a better system? I can imagine you replacing a tiny model with something so grand that it takes longer to suggest the answer than the remote machine takes to answer.



Herman

Shard hits that exact inversion. Once communication is largely hidden, drafting accounts for ninety-four percent of the loop in one reported profile. The coordinator holds the embedding and output head plus a GLM four nine billion drafter, rather than a block of the big model's transformer layers.



Corn

You finally silence the network and discover the assistant won't stop thinking.



Herman

CUDA graphs reduce overhead from repeatedly launching the draft's GPU work. In their measurements, draft time drops from about fifty milliseconds per token to about thirteen. The limiting resource moves as you optimize.



Corn

Drafting lots of candidates could occupy my GPU continuously while the remote verifier still performs almost all the authoritative computation.



Herman

Would you count that as success if the cloud invoice fell?



Corn

If answer quality and completion time stayed acceptable, yes. I don't need a medal for local effort. I need the rented resource's charge to decline by more than the local power and added waiting cost.



Herman

Then uncertainty-aware hybrid inference is closer to the exact motive. U-HLM lets the local model skip sending a token for remote checking when it estimates that token is sufficiently likely to be accepted.



Corn

That's a different bargain from exact verification. The local model is now allowed to be confidently wrong without the large model catching that particular choice.



Herman

It trades fidelity for less remote work. The simulation reports about forty-six percent fewer uplink transmissions and large-model computations, while retaining about ninety-eight percent of the reference model's accuracy. Throughput is roughly two and a half times higher.



Corn

Those are simulation results, not a package Daniel can connect to a rental account tonight.



Herman

Wireless base-station simulations, specifically. Related attention-filtering work reports about forty-one percent energy savings, but throughput of zero point three seven tokens a second.



Corn

A sentence delivered on an instalment plan.



Herman

Energy savings relative to a defined baseline can be real and still describe an unusably slow interactive system. We need the absolute throughput alongside the improvement percentage.



Corn

Also the task. Accuracy on the evaluated workload doesn't promise that an uncertainty shortcut preserves the answer to my unusual technical question.



Herman

Prefill and decode disaggregation has another appeal. Prompt processing can use substantial parallel compute, while generating one token at a time often leans heavily on memory bandwidth. Different devices can excel at different phases.



Corn

I'd expect the handoff to happen once, which sounds vastly kinder to the internet than crossing it for every word.



Herman

Once per handoff, yes. Except the object being handed over is the KV cache, which grows with context and model architecture. It can be enormous compared with a candidate token sequence.



Corn

One delivery. Entire contents of the warehouse.



Herman

Alex Ziskind tried a DGX Spark GB ten for prefill with a Mac handling decode. The Spark processed prompts at roughly five hundred fifty to nine hundred forty tokens a second, up to fourteen times the Mac's prefill speed.



Corn

You have the expression you get before ordering an adapter.



Herman

The adapter is where the enthusiasm gets punished. A twenty-five thousand token context produced a cache transfer that took twenty-five seconds over his USB Ethernet connection.



Corn

Twenty-five seconds.



Herman

He described ninety-six percent of that interval as network waiting. The GPU was idle. Faster prompt computation had created a state-transfer bill that swallowed the advantage.



Corn

This is still a local cable experiment. If that loses, putting the machines across an ordinary internet uplink needs an unusually persuasive explanation.



Herman

He upgraded to fifty gigabit Mellanox networking. Time to first token for the disaggregated arrangement was two point four seconds, against two point three for the Spark alone. Almost parity after substantial networking improvement.



Corn

Congratulations on purchasing the missing tenth of a second.



Herman

The decode comparison was worse. Thirty-four tokens a second when disaggregated, versus fifty-two on the Mac Mini alone. With a Mac Studio, the corresponding results were eighty-four against one hundred six.



Corn

Both cases slower than letting the decoding machine do its own complete job. Although comparing against the Spark alone would make the combination look much better.



Herman

It did. The Studio combination was substantially faster than the Spark alone. That's why the baseline matters. The relevant alternative might be the strongest complete machine you already own, not whichever individual device makes the split look best.



Corn

Matching GPU types removes that convenient mismatch. Two identical cards don't give you a special prompt-processing card and a special generation card. They give you more memory, another rental, and a network.



Herman

There isn't a controlled identical-card comparison in these examples. My inference is that layer splitting still buys aggregate capacity, while phase splitting loses the obvious hardware specialization advantage. Context sizes and concurrent loads can still change the result.



Corn

For a model that fits locally, I'd want a very specific reason to involve the rented twin. For one that doesn't fit either card, the question becomes whether their combined capacity beats renting a complete larger box.



Herman

Average ping tells you the network's one-way delay, and that gives you a starting point for the comparison. More importantly, billing may continue through every stall. Shard's fleet logs put six scattered GPUs at roughly two dollars fifty an hour.



Corn

That sounds inexpensive until we know utilization and the alternative. Thirty tokens a second on a huge model is a capability result. It's not automatically cheaper than a hosted endpoint with batching, or a complete machine hired only while it's needed.



Herman

Nor do rented verifier seconds necessarily shrink because verification takes fewer serial steps. A dedicated instance is usually yours for elapsed time. Idle capacity only becomes useful if your serving system can put something else on it.



Corn

The unused milliseconds don't come back as coins.



Herman

A shared service could schedule other work there, but then its billing model decides who receives the benefit.



Hilbert

That part is right. Corn, my neighbour Avi and I are trying a two-machine setup this month. His machine is two streets over. The average ping looks respectable. The pauses don't. I am writing every spike above two hundred milliseconds in a paper notebook. Avi has the notebook on his kitchen table because his children get home while I'm out. They're using the connection then.



Corn

You've given the busiest household the paperwork.



Hilbert

They can read a clock. The entries cluster after school. Avi keeps sending me the average again. I don't want the average again. I want the answer to stop hanging halfway through. Herman, one small correction. Ping measures the round-trip, not the one-way delay. You use the right meaning everywhere else, then switch it just now.



Herman

I did. Round-trip is the right term. Your average also conceals the long stalls. The pipelined work can absorb some variation, but a sufficiently late dependency still holds up progress.



Hilbert

That's what I'm writing to the Shard people about. Their pipelining section needs a paragraph on the tail. I don't mean their results are wrong. I mean my line behaves well until it doesn't. Avi wants the notebook off his table. I'm asking him to photograph the pages first. I write the full time, including seconds, even when the spike is unremarkable. Otherwise the gaps look like I stop watching.



Corn

Your handwriting is now competing with their upload.



Hilbert

The photographs arrive fine. The setup is still running. Nothing is fixed yet. I'm not mailing the notebook until I have the last afternoon. Avi's latest message says the longest pause today is at three seventeen.



Herman

A cost comparison needs those stalls included, rather than discarding them as an untidy measurement. At fixed hourly billing, interrupted output reduces useful tokens per rented hour. For interactive use, it also changes whether the system feels usable.



Corn

I wouldn't accept a calculator that deletes the difficult afternoons. The bill includes them.



Herman

Privacy is the other price we haven't finished discussing. Keeping original tokens local doesn't make intermediate activations harmless. In the privacy-aware split experiments, attacks recover about fifty-nine percent of tokens with a two-layer split.



Corn

Vector form remains disappointingly readable.



Herman

Moving the boundary to eight layers reduces reported recovery to around thirty-five percent, with minimal throughput impact in that setup. That's an improvement under the tested attack, not a declaration that the remaining representations are safe.



Corn

It also changes how much model my local machine must hold and execute. So a privacy requirement can push the split away from whichever placement looked cheapest.



Herman

Local drafting alone doesn't automatically confer that privacy boundary either. If the verifier receives candidate tokens and context directly, it sees what verification requires. You have to inspect the actual data flow, not attach privacy to the word local.



Corn

For Daniel, I'd separate a worthwhile experiment from a purchase recommendation. The implementation exists. The exact-output path exists. The cheap-cloud-helper that demonstrably reduces the total invoice is not established by these performance charts.



Herman

We'd need the same model, quality settings, context workload, and useful output volume, comparing split execution with whole-request execution. Include rental time, transfer charges where applicable, local electricity, and the time the system spends waiting.



Corn

Plus a baseline that isn't deliberately bad. Don't compare my beautifully optimized split fleet with a complete server running the wrong configuration.



Herman

One adjacent detail that didn't fit earlier is Exo's consumer disaggregated inference. It's been announced, but the consumer feature in question hadn't been released in the account we're discussing. An announced handoff shouldn't go on the list of tools Daniel can run.



Corn

A release announcement can't carry the cache.



Herman

The recommendation, then, is to treat Shard and ShardFlow as concrete distributed experiments, Mesh-LLM as actively developed split-serving software, and split-inference as a directly relevant local-cloud implementation. Treat uncertainty-based remote skipping as research, and cost savings as something still to demonstrate.



Corn

The measurement I'd like next is dollars per million useful tokens against a complete-machine baseline, including the afternoons when the connection stalls. Until that exists, I'd rent a complete machine for savings and split a request when capacity or a specific privacy design justifies the extra machinery.



Herman

Thanks to Hilbert Flumingtop for producing the show.



Corn

If you enjoyed this one, try episode seventeen, Cloud Render Superpowers; episode two, Local STT For AMD GPU Owners; and episode three, Safetensors or something else. This has been My Weird Prompts, where no question is too obscure and no rabbit hole too deep.



Herman

If you enjoy the show, leave us a review in your podcast app. It helps other curious people find us.



Corn

See you tomorrow.



Herman

Until then.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Cunningham, Privacy-Aware Split Inference with Speculative Decoding for LLMs over WANs, 2026-02-18 [primary]**
   https://arxiv.org/abs/2602.16760

2. **working implementation of the above [primary]**
   https://github.com/coder903/split-inference

3. **Shard, Apache-2.0, README + FLEET_STATE.md, accessed 2026-10-29 [primary]**
   https://github.com/leyten/shard

4. **ShardFlow, MIT, README + ARCHITECTURE.md [primary]**
   https://github.com/rautaditya2606/Shardflow

5. **Mesh-LLM v0.78.1 (2026-10-05), 3,630 stars [primary]**
   https://github.com/Mesh-LLM/mesh-llm

6. **Mesh LLM site [primary]**
   https://meshllm.cloud/

7. **BuzzRAG on Alex Ziskind's DGX Spark / Mac Studio disaggregation tests**
https://buzzrag.com/article/splitting-llm-across-two-machines-disaggregated-inference

8. **U-HLM, uncertainty-aware hybrid inference, ICMLCN 2025**
https://arxiv.org/abs/2412.12687v3

9. **energy-efficient wireless LLM inference, 2025-08-18**
https://arxiv.org/abs/2508.12590v1

10. **ADAHI, 2025-10-03**
https://arxiv.org/abs/2510.02851v2

11. **WebGPU split-compute cost experiment, WIP**
https://github.com/tanmaysachan/splitcompute

12. **Mesh LLM launch coverage, HN #6 / 344 points**
https://stackpulsar.com/blog/mesh-inference-iroh-2026/

13. **2026 GPU rental pricing**
https://nodegrove.io/guides/cloud-gpu

14. **local vs rent vs host economics**
https://gpuearn.com/blog/local-vs-rent-vs-host-guide-2026/