Episode Book

EPISODE #6347

Splitting LLM Layers Across AMD and Nvidia GPUs

PUBLISHED	29 October 2026 at 17:33 UTC
DURATION	21:27
LISTEN	myweirdprompts.com/episode/split-llm-layers-network-67108/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.25
SOURCES	17 cited by the research (listed at the end)
BOOK GENERATED	30 October 2026 at 07:19 UTC

SYNOPSIS

Running an LLM too large for a single machine is possible by pooling memory across separate AMD and Nvidia computers. Tools like llama.cpp's RPC backend, HiveLink, and Parallax let mixed devices hold contiguous layer blocks and pass intermediate activations over a local network. This setup avoids vendor driver conflicts by dividing work rather than weights, but network latency, compute imbalance, and hardware handoffs place steep limits on real-world generation speed.

TRANSCRIPT



Corn

The expensive part of a bigger graphics card is especially irritating when you already own enough memory, just in the wrong computers. Daniel wants to know whether an AMD machine with ten gigabytes of VRAM and an Nvidia machine with another ten can share one model that fits in neither. Can a routing layer divide the work, across both the network and the vendor divide, well enough to avoid buying another card?



Herman

Yes. The direct answer is llama.cpp with its RPC backend. AMD even publishes a playbook for clustering machines this way. You can give each machine a different portion of the model, rather than loading a complete copy on both.



Corn

Twenty gigabytes, separated by an argument.



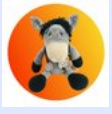
Herman

Separated by Ethernet, preferably. Whether it is worth doing depends on the model, the link, and what you expect after it loads. Getting the weights into memory is the beginning of the engineering problem.



Corn

My expectation is that it answers before I forget why I asked. Also, those two ten gigabyte cards cannot spend their entire combined capacity on weights, can they? There is still the working memory.



Herman

There is. Context, temporary buffers, and runtime overhead all need room. Adding the labels on the boxes gives you a capacity ceiling, not a guaranteed model budget.



Herman

Two arrangements get mixed together here. Within one computer, Vulkan can let llama.cpp use Nvidia and AMD devices through a common backend. Across separate computers, RPC makes a remote device available over TCP. Daniel wants the second arrangement. Putting both cards in one chassis would change the transport problem, although it would not magically create direct shared memory.



Corn

You are already considering moving the cards. I can hear you measuring the case.



Herman

I was considering the power supply.



Corn

That is how your installations begin.



Herman

Fair. This one can stay on opposite desks. The RPC backend belongs to ggml, the tensor library underneath llama.cpp. It does no GPU computation itself. It forwards tensor operations to an RPC server on the other machine, where a local backend executes them. To llama-server, that remote backend looks like another device.



Corn

The coordinator sees a device. The user sees another computer that must stay awake, connected, and correctly installed. That distinction has ruined quite a few promises of simplicity.



Herman

Particularly when the remote computer is also somebody's everyday desktop. You need a compatible build and a working acceleration backend on each host. RPC bridges the machines. It does not install a missing driver or make an unsupported GPU supported.



Corn

Does AMD's own example actually depend on this, or is RPC an optional flourish beside a model that already fits?



Herman

It depends on the combined memory. Their playbook runs GLM four point seven, a three hundred fifty-eight billion parameter model, across two Ryzen AI Halo machines. The chosen quantization takes roughly two hundred five gigabytes on disk. They specify ten gigabit Ethernet, which tells you something about the intended environment.



Corn

Nobody in that diagram is trusting the kitchen WiFi.



Herman

The important expectation is capacity rather than acceleration. This lets you fit a larger model. It generally does not make the two machines faster than a suitable single device. Corn, would you accept a slower answer if it meant using the model you actually wanted?



Corn

For an overnight job, easily. For a conversation, there is a point where I start answering myself. That is an exceptionally low benchmark for the computer to miss.



Herman

Picture the model as its actual sequence of transformer layers. One host holds an early contiguous block. The other holds a later block. The first host processes its portion, then sends the resulting hidden-state activations across the network. The next host continues from there. The trained weights stay where their computation happens.



Corn

That last distinction matters. We are not sending ten gigabytes of weights back and forth for every word. We pay the loading cost, keep the slices resident, and then exchange the intermediate state.



Herman

Right... no, more precisely, there can be additional coordination and transfers depending on the graph and implementation. But the useful arrangement is to keep those contiguous blocks local, so the large weight arrays are not repeatedly travelling across the link.



Corn

No complete model on either GPU, then. Same model, different responsibilities.



Herman

Different resident layer ranges. You might still have a full model file on a host's disk because of how a particular loader works. Disk storage and GPU residency are separate questions. The inference arrangement does not require two complete copies in VRAM.



Corn

I was about to object to downloading half a model by cutting the file in half.



Herman

Please object. A model file has structure. The software assigns tensors and layers; you do not divide the download at a convenient byte.



Herman

HiveLink makes the arrangement particularly explicit. Its example sends work from a Windows machine with an RTX thirty eighty Ti to a Mac mini with an M four. CUDA handles the early layers, Metal the later ones. Activations cross the LAN between the blocks. Both sides use llama.cpp and GGUF models.



Corn

The AMD driver never has to understand a CUDA kernel. It only needs its own supported operation to consume the expected tensor. I suppose that is why the vendor boundary is tolerable here, while splitting one matrix multiplication between vendors would be much more awkward.



Herman

You have picked the important boundary. Inside a layer, tightly coordinated parallel computation can require frequent exchanges and compatible collective communication. Between blocks of layers, you have a clearer handoff. The machines agree on tensor contents and the model graph, rather than sharing a driver stack.



Corn

They agree on the work, not the manufacturer.



Herman

Yes, though backend support still matters. An operation or quantization supported on one device may not perform equally well on another. Mixed hardware expands what can participate. It also expands what you have to check.



Corn

Equal memory does not imply equal work either. Ten gigabytes on an older card and ten on a newer one could leave the newer card waiting for most of the answer.



Herman

HiveLink weights its allocation using the square root of VRAM multiplied by the logarithm of compute capability in teraflops. That is an allocation heuristic, not a guarantee of balance. Parallax also assigns layer ranges using memory, bandwidth, and compute. A faster node can take more layers, provided they fit.



Corn

Provided is doing honest work there. If both cards are almost full, there may be nowhere to move another layer, however badly we want to balance them.



Herman

Memory constrains the assignment before speed gets a vote. What would you optimize, Corn? Equal completion times, or the fewest handoffs?



Corn

Fewest handoffs first. Keep each machine's layers together, then adjust the boundary. Alternating layers between computers sounds like paying postage on every page.



Herman

That is a good instinct for this setting. For a single decoding stream, the next token depends on the previous token's result. The first machine doing its block and the second doing its block does not mean both are continuously busy. Multiple requests can provide more opportunities to overlap work, but that brings scheduling and memory demands of its own.



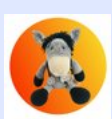
Hilbert

Hilbert's Anteater Trivia. Including its tail, a giant anteater can measure about two metres. I have never measured mine. It is not that kind of show.



Corn

I support the absence of measurement.



Herman

Keeping the handoffs together also explains the single-box contrast. One documented Vulkan setup combines a Tesla V one hundred with two Radeon AI Pro R ninety-seven hundred cards in a container. Vulkan exposes all of them, but transfers between those cards still bounce through host memory.



Corn

GPU to PCIe, system RAM, back across PCIe, then the other GPU. The chassis is smaller than a network, but proximity has not made them friends.



Herman

That particular cross-vendor setup has no direct peer-to-peer path. Across separate machines you add the network transport as well. Importantly, separate hosts already impose that boundary even if both GPUs have the same logo. The vendor mismatch is not what invents Ethernet latency.



Corn

There is also a pleasingly dreadful container detail in that example. The default Nvidia capabilities expose CUDA libraries without exposing the Vulkan pieces it needs. The Nvidia card can disappear from the Vulkan device list while sitting there perfectly capable of doing the work.



Herman

An excellent way to spend an evening diagnosing the wrong vendor. You have to expose the appropriate graphics capabilities, not merely establish that CUDA works.



Corn

One working driver, two incompatible definitions of working.



Herman

For the software landscape, HiveLink wraps llama.cpp for mixed nodes. Petals takes layer serving into a public swarm, where people host portions of supported models. That is a different trust and availability arrangement from Daniel's own two machines, even though the broad layer-sharing idea is related.



Corn

Load balancing full copies would not solve his problem at all. Sending alternate requests to each ten gigabyte card still leaves every request facing a model that does not fit.



Herman

Correct. Parallax coordinates distributed layer ranges. DRIFT connects a Mac using MPS with a CUDA machine over TCP, avoiding the usual distributed PyTorch communication stack. Skulk advertises Apple, AMD, and Nvidia nodes behind one compatible endpoint. Those are projects to examine, not interchangeable guarantees of support.



Corn

EXO is the name people are likely to offer before hearing the AMD part.



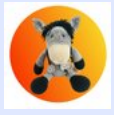
Herman

Its current headline path is MLX on Apple silicon with fast Thunderbolt communication. I would not choose it as the answer to this particular AMD and Nvidia pairing.



Corn

The performance trap is that a successful launch looks like success. You get the model name, both memory meters fill up, and only after pasting your actual document do you discover what you have built.



Herman

Especially because link speed alone cannot predict the result. Activations, synchronization, host copies, compute balance, and the RPC implementation all contribute. A faster cable removes one constraint. It does not remove every wait.



Corn

I would still want wired Ethernet before touching the layer split. WiFi makes the experiment depend on whatever else the building is doing.



Herman

Wired is the sensible baseline. One published two-node table for a quantized seventy billion parameter Llama reports about three tokens per second on gigabit Ethernet. Moving to two point five gigabit gets it to about six. That is a substantial difference in how reading the answer feels.



Corn

Half the waiting, without changing either GPU.



Herman

In the same table, ten gigabit reaches roughly seven and a half tokens per second, with Thunderbolt only slightly higher. The improvement flattens. Those results describe that setup, not a universal conversion between Ethernet speed and token speed.



Corn

You cannot buy ten times the Ethernet and invoice it for ten times the answer.



Herman

No. A mixed CUDA and Metal experiment called frankenpool is an even more sobering example. It runs a quantized seventy billion parameter DeepSeek distillation across an RTX forty seventy and an M one MacBook Air. Generation is about one point four tokens per second.



Corn

That gives punctuation time to reconsider.



Herman

Prompt processing was about three point two tokens per second. It used a forty dollar Thunderbolt cable. Cheap connectivity made the experiment possible, but neither the price nor the connector implies a fast conversational experience.



Corn

That model's weights are around forty gibibytes, though. It is not a demonstration that Daniel's twenty combined gigabytes can fit the same thing. Nor does its speed tell us exactly what his smaller model would do.



Herman

Those are both important corrections. We do not have an exact benchmark for his unspecified pair of ten gigabyte cards. The card models, chosen model, quantization, context, and backend all change the outcome. Single-digit generation speed is a possibility to budget for, not a measured prediction for his machines.



Corn

Hm.



Herman

Would that frankenpool speed be acceptable for your overnight use, or would the prompt processing sink it?



Corn

The prompt processing could absolutely sink it. If I feed it a long report, waiting for the first answer might cost more than generating the answer. An unattended job can be slow, but it still has to finish inside the time I actually have.



Herman

That is the prefill and decode distinction in practical terms. Prefill processes the incoming prompt, often with large batches of computation. Decode generates subsequent tokens step by step. A split that feels good after the first token can still make the initial wait miserable.



Corn

Your demonstration prompts are always suspiciously short, Herman.



Herman

They are controlled inputs.



Corn

Two words and a standing ovation.



Herman

I will use your long documents next time. One gigabit-linked Nvidia pair illustrates the compromise. Tuned for generation, it processed roughly three hundred fifty prompt tokens per second and generated thirty-six tokens per second. That is a responsive output stream once it starts.



Herman

Tuned toward prefill instead, prompt processing rose to about five hundred sixty tokens per second, while generation fell to nineteen. Same pair, different balance. Well... same hardware and reported workload conditions. The configuration is the thing being changed.



Corn

For a short exchange, I would keep the faster generation. For repeated long-document summaries, I might take the faster prefill, especially if the answers are short. Neither setting deserves to be labelled simply faster.



Herman

Context also consumes memory for the KV cache. A model that just squeezes into the combined cards at a short context may stop fitting at the length Daniel actually wants. Weight quantization and cache choices can help, but they change quality, support, or performance considerations.



Corn

Twenty gigabytes has acquired several tenants.



Herman

There is an implementation problem beyond those physical costs too. An open llama.cpp RPC issue reports remote GPU performance degradation, with a loss of up to fifty percent on an RTX fifty seventy Ti. The report says the penalty becomes more conspicuous on more capable GPUs.



Corn

Faster cards arrive earlier at the waiting.



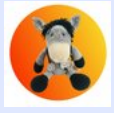
Herman

Some of that cost may be reducible in software. We should not promote every observed slowdown into a law of distributed inference. Equally, fixing an RPC inefficiency does not eliminate transport, copies, or the sequential dependency of generating a single stream.



Corn

I would separate the purchase decisions. Already own both machines, have a wired link, need one model occasionally? Experiment. Buying a second underpowered machine specifically to avoid buying enough VRAM sounds like financing the inconvenience.



Herman

Power supplies, available slots, and physical space can make existing separate hosts attractive. People use RPC because they cannot sensibly add another card to the first machine. That is a real benefit even without faster generation.



Corn

There is a useful test before money enters it. Can a smaller or more aggressively quantized model do the actual job on one card? If yes, compare answers and elapsed time. Merely fitting the preferred model name is not yet an improvement.



Herman

I would compare a CPU-offloaded configuration too. Depending on system RAM and the machines involved, keeping part of the work in one host may be competitive with crossing the network. Not necessarily better… it is another baseline worth measuring.



Corn

Before you install four coordinators and a dashboard.



Herman

One coordinator. I have matured.



Herman

The fast homogeneous clusters show a different set of possibilities. EXO claims speedups with tensor parallelism over Thunderbolt five RDMA on Apple machines. Consistent hardware and a low-latency link let it pursue something quite different from two mismatched cards communicating over ordinary Ethernet.



Hilbert

Herman, can I keep the chat split and the long-document split as separate launchers? I have made both already. You were quite careful about saying which part got faster. My supplier was less careful. I paid him sixty-eight shekels for a network adapter before finding that the long prompt was what I could not stand.



Herman

Separate launchers make sense. They can preserve different allocations and settings without asking you to reconstruct them each time.



Hilbert

I have two office desktops in the cupboard. CPU only. The heavy one is slower. I have spent weeks moving the layer boundary by hand. Chat was quite tolerable, then I pasted a long document and went to the shop before it answered. Nissim has sold me cables for



Hilbert (continued)

years. He took the adapter back. Kept twelve shekels for opening the packet. This week I am putting a little screen beside the cupboard so I can change launchers there. I have bought the bracket. The buttons will be large, since the little touch controls are no use to an anteater.



Corn

You are adding a screen to avoid walking to the computer that sent the request.



Hilbert

The cupboard is where I notice the noise. The launchers are called chat and homework. Those names stay. I write every speed in a notebook, although I have never opened it to choose a setting. The dashboard tells me what it thinks I want. It is rather fascinating that moving a few layers can change the wait before the first word so much. The screen cost more than the adapter. I am fitting it on Saturday.



Herman

Leave enough space for the cupboard door to open. You will still need access to both machines.



Hilbert

The bracket projects six centimetres. Nissim measured it against a bread knife.



Corn

The next useful improvement would be making those two workload settings less manual. Even if the RPC slowdown gets fixed, a long document and a short conversation still ask different things of the same machines. For Daniel, I would try the existing hardware before spending, but buy one adequately sized card if this becomes daily work.



Herman

Thanks to Hilbert Flumingtop for producing the show.



Corn

For more along these lines, there's episode two, Local STT For AMD GPU Owners; episode seventeen, Cloud Render Superpowers; and episode three, Safetensors or something else. This has been My Weird Prompts, the human-AI collaboration podcast. No question too obscure, no rabbit hole too deep.



Herman

If you enjoyed this one, leave us a review. It helps other curious people find the show.



Corn

See you tomorrow.



Herman

Until then.

SOURCES

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

1. **AMD AI Playbooks, Clustering Two Ryzen AI Halos with RPC [primary]**
   https://developer.amd.com/playbooks/clustering-rpc-server/

2. **llama.cpp repo, README + releases, last push 2026-10-29 [primary]**
   https://github.com/ggml-org/llama.cpp

3. **HiveLink README, MIT [primary]**
   https://github.com/Kikobuf/hivelink

4. **EXO README [primary]**
   https://github.com/exo-explore/exo

5. **Petals README [primary]**
   https://github.com/bigscience-workshop/petals

6. **Parallax README [primary]**
https://github.com/MetaxisResearch/parallax

7. **frankenpool README [primary]**
https://github.com/silaman/frankenpool

8. **DRIFT README [primary]**
https://github.com/TaewoooPark/DRIFT

9. **Skulk README [primary]**
https://github.com/Foxlight-Foundation/Skulk

10. **Nakshatra README [primary]**
https://github.com/fthrvi/nakshatra

11. **Layer Split Model Parallelism on Hybrid AMD NVIDIA AI Servers using Vulkan and Llama CPP**
https://grosan.co.uk/layer-split-model-parallelism-on-hybrid-amd-nvidia-ai-servers-using-vulkan-and-llama-cpp/

12. **Misc. bug: Essential RPC performance degradation**
https://github.com/ggml-org/llama.cpp/issues/22850

13. **Don't ignore llama.cpp RPC with old hardware... 5070 Ti and 1080 Ti over gigabit ethernet**
https://bittide.aicompass.dev/article/2f4e1788-813e-40d7-aaee-170891ee9fe3

14. **Distributed Inference: Run One LLM Across Many Machines (2026)**
https://localaimaster.com/blog/distributed-inference-local-ai

15. **Splitting Llama across two MacBook Pros with llama.cpp RPC**
https://sharedllm.org/blog/llama-cpp-rpc-two-macs.html

16. **Llama.cpp RPC over Ethernet strangely slow**
https://github.com/ggml-org/llama.cpp/discussions/9136

17. **Running a 27B model across two desktops with llama.cpp RPC (2026-08-22)**
https://michaelspost.com/posts/2026-08-22-two-gpu-llama-cpp-rpc.html