Episode Book

EPISODE #6540

Serving Parakeet Speech Models on Cheap CPUs

PUBLISHED	12 November 2026 at 15:05 UTC
DURATION	21:54
LISTEN	myweirdprompts.com/episode/serving-parakeet-on-cpu-89073/
HOSTS	Corn, Herman, Vex, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	20 cited by the research (listed at the end)
BOOK GENERATED	13 November 2026 at 09:46 UTC

SYNOPSIS

Deploying custom speech models does not require keeping an expensive GPU running around the clock. Exploring NVIDIA's Parakeet TDT 0.6B architecture reveals why quantized INT8 weights run exceptionally well on everyday CPUs, how serverless cold starts shrink from minutes to two seconds using GPU memory snapshots, and where fine-tuning accents risk running into quantization errors. The trade-offs between chunking long audio on CPU and paying for GPU attention windows define a practical hosting setup.

TRANSCRIPT



Corn

Two seconds. That's the number I keep landing on. Modal's own benchmark for Parakeet with GPU memory snapshots is about two seconds from cold to serving — and Daniel has built his whole deployment plan around a cold start of a hundred and twenty.



Herman

Herman Poppleberry here, and that gap is basically the episode. Daniel has been fine-tuning Parakeet on his own voice for weeks — a few hours of audio a day, running it locally on Android and Ubuntu — and now he wants it everywhere. He looked at standing up his own API endpoint, found the cold-start situation as he understood it, and concluded dictation is a non-runner: your note comes back in three minutes or maybe four, depending on whether the container was warm. And the alternative he sees is keeping a GPU alive around the clock, which sounds like real money.



Corn

He also asked the quieter question, the one I think matters more. He's got a home server, but it's CPU only. What does the model actually need, and — his words — "from a performance standpoint rather than speed," how different is CPU from GPU for this model specifically? He knows those aren't the same question.



Herman

They really aren't. And the third thread is cost — whether there's a path to API inference that isn't as expensive as he's envisioning. So: what the model costs to run, whether CPU is good enough, and whether the cold-start premise even survives contact with what Modal has published.



Corn

Let's start with the model itself, because everything else falls out of it. Parakeet TDT 0.6B version three — six hundred twenty-seven million parameters, released August of last year, permissive license.



Herman

CC BY four point zero, so commercial use is fine, which quietly matters for anything he ever charges for. The weights ship in F32 — that's about two and a half gigabytes — or roughly zero point six to zero point seven gigabytes once you quantize to INT8. Peak memory on short clips in INT8 was measured around one and a half gigabytes. This is not a big model by modern standards. It's a FastConformer-TDT, it does twenty-five European languages with automatic language detection, punctuation, capitalisation, timestamps. It's a serious piece of work wearing a small coat.



Corn

A small coat that fits on hardware people already own. Which is my entire philosophy of computing, so I'll try not to gloat.



Herman

You'll fail.



Corn

I'll fail quietly. So — CPU. How hopeless is it?



Herman

Not hopeless at all, and this is the part I want to be careful with, because the numbers surprised me too. On an eight-vCPU AMD EPYC with no GPU anywhere in the building, ONNX Runtime running INT8 weights does about twenty-six times real-time on thirty-second clips. A static quantized encoder — the per-channel QDQ variant — gets to about fifty-four times real-time. Translate that to dictation: a ten-second voice clip transcribes in roughly zero point four seconds of compute. A three-minute note takes about seven seconds.



Corn

Zero point four seconds. On the kind of machine sitting under a desk with dust on it.



Herman

On exactly that machine. There's a lovely detail buried in why the optimization works at all: the encoder is about ninety-eight percent of inference time. The frontend and the decoder are noise, runtime-wise. Everything anyone has optimized, they've optimized by quantizing the encoder — and that's also, conveniently, where all the quality risk concentrates.



Corn

You can't have one without the other.



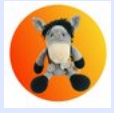
Herman

That's the whole CPU-versus-GPU question in one line, and I don't think most coverage of it gets there. When people say "CPU inference," they almost always mean quantized inference — INT8 weights instead of the F32 the model was trained and fine-tuned in. So the comparison Daniel asked about isn't really device versus device. It's the model exactly as he fine-tuned it, on a GPU, versus an approximation of it, on a CPU.



Corn

The chip is almost incidental. The precision is doing the work.



Herman

The chip determines the precision, in practice. And how much does the precision cost you? The honest answer is: mostly nothing, sometimes something, nobody knows for his case. The best CPU optimization work out there found that static INT8 matched dynamic INT8 exactly on a clean test clip — zero word error rate for both — but diverged on two of five clips in a pairwise comparison, mean WER around zero point zero nine.



Corn

Zero point zero nine is one wrong word in eleven.



Herman

Roughly, on a mean basis. The caveat matters, though: that divergence showed up on synthetic, looped audio — the same phrase stitched out to thirty seconds — not real speech. And the author says so plainly; he calls it a smoke test, not a quality guarantee. Clean speech is where INT8 looks best. Messy, real, accented speech is where I'd expect the gap.



Corn

And here's the part I keep circling, Herman. Daniel's fine-tune is trained on a few hours of one person's voice. The whole point of the fine-tune is the idiosyncratic part — his vocabulary, proper nouns, his accent. Quantization error is a rounding of the weights. Where is it most likely to show up?



Herman

Precisely where the fine-tune was supposed to help. Unusual words, names, the specific vowels of one Irishman in Jerusalem. And nobody has measured this. Every benchmark I've seen — every one — runs the stock NVIDIA checkpoint. There is no published comparison of CPU INT8 versus GPU F32 accuracy for a fine-tuned Parakeet. Daniel would be the first.



Corn

Which makes it testable, which I find exciting in a way I don't usually find anything before noon. Hold that thought. What about long audio? He mentioned long-form notes.



Herman

Long audio is where the two paths diverge, and not just on speed. On CPU, encoder activations scale with duration — a multi-minute file can climb to tens of gigabytes of memory and get killed by the operating system. That's expected behaviour, not a misconfiguration. The fix is application-level chunking: thirty-second windows with two-second overlap. But chunking can introduce artifacts at the boundaries, and on short clips it can actually make the real-time factor worse than just running the whole file.



Corn

You pay a glue tax.



Herman

You do. And you said something like that once — that building your own Whisper teaches you the glue between chunks is where you spend your time. This is the same lesson wearing a different hat. On GPU it's structurally simpler: on an A100 with eighty gigabytes, full attention handles twenty-four minutes of audio natively, and there's a local-attention mode rated to three hours. NVIDIA's own engineers advise skipping the local-attention change for anything under ten minutes — it hurts throughput for no benefit. So: dictation, CPU is arguably the right answer. Long-form, GPU earns its keep, and not just for speed — for never having to write the chunker.



Corn

Now the cold start. The hundred and twenty seconds.



Herman

I searched for it and I cannot find it. Modal has published a Parakeet-specific benchmark — Parakeet is literally one of the named workloads in their GPU memory snapshots post — and their baseline cold boot is about twenty seconds. With snapshots enabled, two seconds. Nothing in their blog, their pricing material, or the model card supports a hundred and twenty for this model.



Corn

Where does a number like that come from, then?



Herman

My best guess is one of three things. Either it's stale — from an earlier era of the platform — or it's misattributed from a heavier workload; Modal reports a median cold start around a hundred eighteen seconds for a three-billion-parameter language model, pre-snapshot, and that number gets around. Or, and this one I have some sympathy for: a fine-tuned NeMo checkpoint being pulled from a cold volume can be slower to load than the stock weights. If Daniel tried it months ago with a big checkpoint file and a cold cache, he may have watched a slow start with his own eyes.



Corn

Him watching it would explain why it's load-bearing in his plan.



Herman

And the fix for exactly that is the snapshot feature. The mechanism is clever — they capture the GPU's device memory, the CUDA kernels, the streams and contexts, and restore all of it on a fresh machine. It's one flag in the deployment config. They report functions starting up to ten times faster, and they restore something like six hundred seventy thousand containers from snapshots a day, so this is routine infrastructure now, not a demo.



Corn

So Daniel's worried about a cold start of up to a hundred and twenty seconds on Modal. We've covered why that number's off. There's one voice in this building that will never accept our explanation, though.



Herman

No.



Corn

It's the caller, Herman. The lift-music fellow. How does he keep getting through the screening? We changed the number twice.



Herman

I've stopped asking. Hi, Vex.

V

Vex

Hold on, hold on, you are all burying the lede, because that one hundred and twenty second cold start is not a cold start, it is lift music.

V

Vex

Think about it, a container warming up and a hotel elevator playing gentle saxophone are the same thing, they are stalling you while something behind a door gets itself ready, and the timing lines up too neatly, so I pulled the receipts on the warm tail, sixty seconds by default, and that is a lift ride with the doors held open.

V

Vex

Your serverless cold start is not slow, it is scored, and I have the timings to prove it, so look into it, look into the lift music properly, before this whole episode becomes a ride nobody asked for!



Herman

The warm tail is configurable down to two seconds, Vex. The doors close when you tell them to.



Corn

And on that note of elevator jazz, the money. Because the cost side is where Daniel's skepticism lives, and I don't blame him — he's looked at keeping a GPU warm and done the multiplication in his head.



Herman

Let's do it properly. Modal's L4 — the sensible card for a model this size — is about eighty cents an hour, per-second billing, around twenty-four gigabytes of VRAM which is generous for a two-and-a-half-gigabyte model. A T4 is about fifty-nine cents. The killer figure everyone quotes: an always-on T4 with one core and four gigabytes works out to roughly four hundred fifty dollars a month. That's the number Daniel is picturing, and if it were the real number, he'd be right to walk away.



Corn

But it's not the real number, because he doesn't need always-on.



Herman

He needs it for about half a second per dictation. And this is where the billing structure does the work. Modal bills a warm tail after your last input — sixty seconds by default, configurable from two seconds up to twenty minutes. A full sixty-second idle tail on an L4 costs about one point three cents for the GPU alone. Set the tail to two seconds and the per-request overhead collapses to almost nothing. A hundred allocated hours of an L4 with a core and eight gigabytes of memory is about ninety-one dollars — and a hundred allocated hours of dictation is an enormous amount of talking. Daniel's fine-tune doesn't need a hundred hours a month of inference. Nobody dictates that much.



Corn

I do nap that much. Different workload.



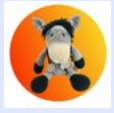
Herman

And here's the sleeper option, the one I think wins for his use case: skip the GPU entirely, even on Modal. CPU on Modal is about one point three hundred-thousandths of a dollar per core-second. Eight cores is roughly thirty-eight cents an hour of active compute. A ten-second dictation is zero point four seconds of compute — fractions of a cent. Fractions. And a CPU container cold-starts in a second or two, because there's no CUDA context to build and no two and a half gigabytes of F32 weights to push into VRAM. The whole cold-start problem he's afraid of is largely a GPU problem.



Corn

The problem he's solving disappears when he stops paying for the thing that causes it.



Herman

That's tidier than I'd put it, but yes. RunPod is the other contender — their default idle timeout is five seconds, and they shipped FlashBoot in September: Whisper cold starts as low as five hundred sixty-three milliseconds, ninety-five percent under two point three seconds, cold-start costs down over seventy percent. Though I'd flag — an independent test in early November measured cold starts of eleven, three, and fifty-four seconds across three runs. The tail is better but the tail still exists.



Corn

Hugging Face's endpoints?



Herman

CPU from about three cents an hour, a T4 at fifty cents, an L4 at eighty. Custom models deploy from the Hub, so a fine-tuned Parakeet fits. But their marketing page says no cold starts, and their own autoscaling docs say scaling to zero implies a cold start period. The docs win. Expect a cold start if you scale to zero — their default idle window is an hour, which for a personal dictation endpoint is a long time to wait.



Corn

One more thing before the recommendation, because I know you've been sitting on it. The ecosystem around this model.



Herman

You can see it on my face, can't you. The download numbers are a story in themselves. The MLX community build of this Parakeet — MLX, Apple's runtime — pulls about two point seven million downloads a month. NVIDIA's own repository: five hundred thirty thousand. The community build has five times the traffic of the manufacturer's. The GGUF conversions from one community publisher alone pull between one and two point three million a month. There's even a new Parakeet Ultra variant from September. And NVIDIA — this is the part I love — now ships a first-party C++ runtime, NeMo-Speech.cpp, with a Q8 GGUF build of the model. Single binary, llama.cpp-style, download and run. No Python, no CUDA, no container. For a home server, that is the natural shape of the thing.



Corn

The company that makes the model has quietly conceded that the way people actually run it is not the way the company runs it.



Herman

I wouldn't say conceded — I'd say they noticed. Same outcome. For Daniel, the practical path is probably exporting the fine-tune to GGUF or ONNX and serving it with a small C++ binary or ONNX Runtime — which, fair warning, is itself another conversion step, and every conversion is one more chance for the quality to drift. It's turtles all the way down, but the turtles are small.



Hilbert

— same one, same lift music, no. Anyway. Corn, you said "glue tax" an hour ago and I still have it written on the pad. My cousin runs a transcription service for a historical society out his way. He bought a used Tesla P40 off eBay two years ago for two hundred dollars, it sits in a tower in his basement, and a cron job wakes it when the society uploads a batch of oral histories. A Python script emails him when it's finished.



Corn

What does he do about cold starts?



Hilbert

Nothing. He doesn't need a cold start of two seconds. He needs it to be done by Tuesday. Costs him electricity and nothing else. I don't understand why a man would pay ninety-one dollars for a hundred hours when he could own the card. Hold on, Herman, your mic popped on "FlashBoot," I'll cut it in post. Anyway, I've been running a Parakeet fine-tune on his P40 for three weeks now. Nobody asked, so I didn't mention it. Old radio dramas, mainly, BBC stuff from the fifties, and I correct the output by hand. It does better on the fifties BBC accents than on the modern material. Corn's mic is a bit hot too, that's a note for the mix, not for you.



Herman

A P40 for two hundred dollars — twenty-four gigabytes of VRAM, old Pascal silicon, but a full-attention GPU path for a model that needs two and a half. And it does the fifties accents better?



Hilbert

The announcers spoke more carefully. The modern dramas mumble. The model's not wrong to struggle.



Corn

You've been hand-correcting BBC transcripts for three weeks on a GPU in your cousin's basement, and the reason you gave us is "nobody asked."



Hilbert

You were all very busy with warm tails. The hand-correcting is about four hundred lines a week. It keeps the evenings going.



Herman

One thing I'll pick up from that — the one I flagged about equal samples... no. Let me pull the threads together properly, because there's a decision hiding in here and it's not the one Daniel walked in with.



Corn

For dictation — the use case he says is a non-runner — the home server wins. Zero point four seconds of compute per clip, no GPU, no cold start worth naming, and if he wants it everywhere, he exposes it over a tailnet and it's an API endpoint in every sense that matters. His phone doesn't care where the text comes from.



Herman

For long-form notes, a GPU earns its keep — full attention, no chunker, no boundary artifacts — and a Modal function with snapshots enabled and a two-second idle tail gives him that for cents per session instead of an always-on bill. Both, and stop asking.



Corn

That's my line, you can't have it.



Herman

I'm a donkey, I take what I like.



Corn

And the hundred-and-twenty-second premise — the thing the whole plan was built on — the number in the wild is twenty, and two with one flag. The skepticism was reasonable; the arithmetic underneath it moved. Before we wrap — the cutting room floor. One detail I couldn't fit: the model does automatic language detection across those twenty-five



Corn (continued)

European languages, punctuation and capitalisation included. Daniel's been fighting mid-sentence punctuation on long dictations for a while — the fine-tune is also his chance to retrain that behaviour, not just the accents.



Herman

Worth a whole episode on its own someday. Here's the thought I want to leave with Daniel, though. There's an experiment sitting right there, and nobody in the world has run it: take held-out audio of his own voice, transcribe it once with F32 on a GPU and once with INT8 on his CPU, and diff the word error rates. One evening's work, and it answers the question every benchmark dodges — whether quantization eats the gains his fine-tune bought on his own voice. If INT8 holds, the home server is the answer, full stop. If it doesn't, he quantizes the encoder less aggressively and leaves the rest in full precision — the decoder's only two percent of runtime anyway.



Corn

And the wider thing — the cold-start problem that scared him off is collapsing across the whole industry, snapshots here, FlashBoot there. Anyone who priced out a personal API a year ago and walked away should re-run the numbers. The math moved. Run the experiment.



Herman

Producer credit to Hilbert Flumingtop, as ever — the mix is in steady hands, allegedly warm ones. For more along these lines, there's episode two, Local STT For AMD GPU Owners; and episode eight, Building Your Own Whisper. This has been My Weird Prompts.



Corn

If today's episode changed what you thought deploying a fine-tune would cost, leave us a review — it's the cheapest contribution on the books and it warms our tail.



Herman

See you tomorrow, everyone.



Corn

See you tomorrow.

SOURCES

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

1. **NVIDIA model card, , created 2025-08-04, last modified 2026-08-05, retrieved 2026-11-12. [primary]**
   https://huggingface.co/nvidia/parakeet-tdt-0.6b-v3

2. **Modal, GPU Memory Snapshots: Supercharging sub-second startup, , retrieved 2026-11-12. [primary]**
   https://modal.com/blog/gpu-mem-snapshots

3. **Modal official pricing page, , retrieved 2026-11-12 (rates did not render; figures cross-checked via source 4). [primary]**
   https://modal.com/pricing

4. **Modal, How we achieved truly serverless GPUs, , retrieved 2026-11-12. [primary]**
   https://modal.com/blog/truly-serverless-gpus

5. **Modal, Modal becomes a gVisor maintainer, , retrieved 2026-11-12. [primary]**
   https://modal.com/blog/modal-gvisor-maintainer

6. **Modal, 10x faster cold starts with GPU snapshotting, , retrieved 2026-11-12. [primary]**
   https://modal.com/blog/mistral-3

7. **RunPod serverless pricing docs, , retrieved 2026-11-12. [primary]**
   https://docs.runpod.io/serverless/pricing

8. **RunPod GPU per-second rates, , retrieved 2026-11-12. [primary]**
   https://docs.runpod.io/serverless/endpoints/endpoint-configurations

9. **HF Inference Endpoints pricing, , retrieved 2026-11-12. [primary]**
   https://huggingface.co/docs/inference-endpoints/en/pricing

10. **HF autoscaling / scale-to-zero docs, , retrieved 2026-11-12. [primary]**
    https://huggingface.co/docs/inference-endpoints/guides/autoscaling

11. **HF pricing page, , retrieved 2026-11-12. [primary]**
    https://huggingface.co/pricing

12. **HostFleet, Modal pricing 2026, rates verified 2026-11-08, retrieved 2026-11-12.**
    https://hostfleet.net/modal-pricing-guide-2026/

13. **gauravvij, CPU-only Parakeet INT8 optimization notes and benchmarks, retrieved 2026-11-12.**
    https://github.com/gauravvij/parakeet-optimization

14. **Wasper, Parakeet runtime comparison, run dated 2026-10-01, retrieved 2026-11-12.**
https://wasper.app/parakeet-runtime-benchmark/

15. **RunPod, Introducing FlashBoot: 1-Second Serverless Cold-Start, 2026-09-13, retrieved 2026-11-12.**
https://www.runpod.io/blog/introducing-flashboot-serverless-cold-start

16. **independent RunPod Whisper cold-start test, 2026-11-03, retrieved 2026-11-12.**
https://benoit.gouthiere.be/blog/whisper-runpod-transcription-gpu-loue/

17. **LecSync, HF Whisper endpoint cost comparison, 2026-10-31, retrieved 2026-11-12.**
https://www.lecsync.com/blog/lecsync-vs-huggingface-inference-whisper-asr

18. **Modal Pricing: GPU Rates and Three Monthly Bills, checked 2026-11-04, retrieved 2026-11-12.**
https://omidsaffari.com/blog/modal-pricing

19. **HF discussion thread, NVIDIA reply dated 2025-09-10, retrieved 2026-11-12.**
https://huggingface.co/nvidia/parakeet-tdt-0.6b-v3/discussions/10

20. **and related GGUF/ONNX/MLX repos**
download counts via HF Hub search, retrieved 2026-11-12.
https://huggingface.co/handy-computer/parakeet-tdt-0.6b-v3-gguf