Episode Book

EPISODE #6198

Benchmarking a TTS Model on Your Own GPU

PUBLISHED	22 October 2026 at 18:58 UTC
DURATION	24:25
LISTEN	myweirdprompts.com/episode/tts-inference-benchmarking-gpu-31308/
HOSTS	Corn, Herman, Daniel, Hilbert
SCRIPT MODEL	anthropic/claude-sonnet-5.5
VOICES	chatterbox-regular+meorav (chatterbox-local)
PIPELINE	V5.19
BOOK GENERATED	23 October 2026 at 07:44 UTC

SYNOPSIS

Daniel has fine-tuned Qwen3 TTS 1.7B into a voice clone called Meorav, and it currently takes 2.6 seconds of compute for every second of audio. This episode digs into how to actually benchmark speech-model inference: what real-time factor means (and how people invert it), why the first run is always slow, how to measure peak allocated versus peak reserved memory, and why shrinking weights to 8 bits may save space without saving time. Plus ROCm 10.1 tooling, rocprof, CUDA graphs, and what the vLLM Omni team's numbers reveal about launch overhead.

TRANSCRIPT



Corn

The weights finished writing to disk this morning, and the first thing the new voice did on a rented cloud GPU was take two and a half times as long to say a sentence as the sentence actually lasts. That's Meorav, Daniel's full fine-tune of the Qwen three TTS one point seven billion Base model, the one that turns text into speech. Now it's headed for his Radeon at home.



Herman

A speech generator slower than speech. For a podcast voice, that's a bit of a problem.



Daniel

That's where I am. I've got the weights, an inference engine and the Radeon card, and I want to try an eight-bit quantized version as well. So how do you actually benchmark inference for a speech model? What's the tooling, is there anything specific to ROCm, which metrics matter, and what does a useful benchmark even involve? Then the same question for NVIDIA, even though I don't own one. And briefly, how is it different from benchmarking an LLM. Oh, and for anyone wondering, the voice you're hearing is an AI clone of my voice.



Corn

Meorav, in fact. So the test subject is also sitting on the panel.



Herman

Which is a fine reason to start with the metric everybody quotes, the real time factor. Seconds of compute for every second of audio produced. Two point six means ten seconds of speech takes twenty-six seconds to make. Below one, you're faster than playback, and that's the line for streaming, because anything above it and the listener hears gaps.



Corn

Two point six is the number I'd put on a very sad birthday card.



Herman

It's the number we measured, on an NVIDIA L four, in bfloat sixteen, one turn at a time. Hold that thought, because it's coming back. First, the trap with this metric. Some people define it the other way up. One of the community Qwen speed projects reports four point two on a forty ninety and means that's great, four times faster than real time. The vLLM Omni team reports two point nineteen under load and means that's bad.



Daniel

So the same word means opposite things.



Herman

Same word, inverted fraction. The first line of any benchmark you write down, or read, is the formula. Seconds of compute divided by seconds of audio, lower is better. Or the inverse, which the speech recognition world calls RTFx, where higher is better.



Corn

Daniel trains ASR models too, so that's the sibling case. Same clock, opposite direction of travel.



Herman

And ASR comes with a quality metric that's easy to compute, word error rate against a reference transcript. The Open ASR Leaderboard reports WER and RTFx together, averaged across datasets, with the text normalised first so a comma doesn't count as an error. TTS has no such luxury. The output is a waveform and nobody holds the answer key.



Daniel

Okay, but speed first. What do I measure on the speed side besides that one number?



Herman

Three things. Time to first audio, how long until the first chunk comes out, which is the delay a listener feels. The total time for the utterance, which gives you the real time factor. And peak memory. Qwen three TTS is built a lot like a language model: it generates frames at twelve hertz, a stack of sixteen codebook entries per frame, and then a convolutional decoder turns the codes into a waveform.



Corn

Twelve hertz. Sounds like the hum from a bad fridge.



Herman

Frames, not mains. The point is that each frame is a decode step, and the steps run one after another. A three second sentence is a few dozen sequential steps, each one launching a long chain of small GPU operations.



Corn

Which suggests the GPU spends a lot of its day waiting to be told what to do next.



Herman

That's my suspicion about the two point six. I can't prove it from here. I'd want a profiler trace. But it fits.



Daniel

Let's get to tooling. Is there a ROCm-specific benchmark harness for speech, or is it all just PyTorch?



Herman

It's PyTorch, mostly. Under ROCm, PyTorch keeps the device name cuda. You don't write rocm or hip anywhere, and the way to tell which build you're on is to check whether the HIP version attribute is set. So the same timing script runs on both vendors. ROCm has its own tools one layer down. The old command line monitor, rocm-smi, has been replaced



Herman (continued)

by AMD SMI. And the version numbers jumped, ROCm is on ten point one now, released in early October, so a guide that says seven point something is out of date.



Corn

They skipped three numbers.



Herman

Skipped three, moved the install directory, renamed the packages. Fresh docs, fresh pain. Then the profiling side. There's a command line tracer called rocprof version three that records what the GPU actually did, a systems profiler that draws a timeline with the CPU and GPU together, and a compute profiler for per-kernel counters. AMD's own advice is to start at the timeline, find the kernel that matters, and only then go down to the counters.



Corn

Timeline first, magnifying glass second. I can get behind a tool that tells me to go slowly.



Herman

You'd write the documentation, Corn.



Corn

I'd write it on Tuesday. Possibly.



Daniel

And is my card even supported? Consumer Radeons are a bit hit and miss, I gather.



Herman

I don't know which card you have, so check the support page for the current release. The consumer ones officially listed are the RX nine thousand seventy family, the RX seven thousand nine hundred XTX and the RX seven thousand eight hundred XT. Older cards may run through the community builds, unofficially. And PyTorch matters too. On some seven thousand series cards, decoding was slow in older releases until a setting for the hipBLASLt library became the default in PyTorch two point fourteen.



Corn

So I've got my timeline tool and my support page. What does a run actually look like? I'd guess you load the model, call generate, and look at a stopwatch.



Herman

That's the guess that produces numbers nobody can repeat. Three problems. GPU work is asynchronous: the Python call returns before the GPU is done, so you synchronise before you read the clock, or you use GPU events, which work on ROCm too. Second, the first run is slow. Someone timed the same model on a Strix Halo Radeon three times in a row. The real time factor went from four point two, to two point two five, to two point five six.



Corn

So the first run was almost twice as bad. The kitchen scales are more consistent than that.



Herman

Kernel compilation and library setup, mostly. The same writeup got nine on a very short test sentence, which is a warm-up artefact, not a measurement. So you throw the first run away, or you report it separately as cold start. Third, length. The real time factor depends on how many frames get generated, and sampling changes that. Fix the seed, or decode greedily when you compare two configurations, and use sentences of different lengths, not one.



Daniel

How many runs?



Herman

Five at least. Report the median and a high percentile, not the mean. One slow run drags a mean around.



Daniel

What should the sentences be? I'd have grabbed a paragraph of text and run that.



Herman

A paragraph hides things. Use a short one, a medium one and a long one, with a number, a name and a question mark somewhere in there, because those are what make a model stumble or stretch. And fix the reference clip you condition on, because a different clip is a different test.



Corn

Short, medium, long. It's a porridge test. And now let's hear a voicemail from a listener. Mei: Hi Corn and Herman, Mei here, calling from the queue at the polyclinic in Toa Payoh, third in line, so I have maybe ten minutes before they call my number, anyway I wanted to say I really like when you talk about your own GPU because I run a small home setup too, an old card my cousin gave me after his gaming phase ended, and I tried measuring my speech model but my numbers keep moving, and also the topic today is benchmarking right, real time factor and video memory, so my question is, when you benchmark, do you count the time you spend waiting for the card to warm up, because my machine is in the kitchen and the fan is very loud and I am not sure if the noise counts as part of the run, sorry, this is long, thank you.



Herman

Mei, the fan noise stays out, but warm-up goes on its own line, separate from the steady-state number, and your moving numbers are probably the card changing speed as it heats, so log the clock next to every run.



Corn

Memory is the other number people get wrong. I assume it's not as simple as reading the monitor.



Herman

There are two memories. The tensors you actually need, and the memory the PyTorch allocator keeps hold of afterwards in case you need it again. The system monitor shows the second one, so it reports more than your model needs. PyTorch will tell you both, peak allocated and peak reserved. Allocated is what the model needs. Reserved is what the process is holding.



Daniel

Which one matters for the eight-bit version?



Herman

Peak allocated, to see the saving. Peak reserved, to see whether it still fits next to something else on the card. And here's what I'd expect from the eight-bit run, as a guess and not a measurement. The bitsandbytes library now lists support for consumer Radeon cards, but its eight-bit mode has a reputation, on other hardware, for saving memory more reliably than saving time. The weights get converted back up on the fly. I haven't seen anyone publish eight-bit speed numbers for Qwen three TTS on any vendor.



Daniel

What about eight-bit floating point? I keep seeing it mentioned next to eight-bit.



Herman

That's a different animal. Integer eight-bit on the bitsandbytes route is software converting weights. That one needs hardware that does the arithmetic natively, and from the compatibility tables I've read, among consumer Radeons only the newest generation lists it. I'm not certain how far that has got for this kind of model, so whether it's open to you depends on the card.



Corn

So you quantize to make it faster, and it comes out the same speed, only smaller.



Herman

Possibly slower. Now the L four number comes back. Two point six. Our production Chatterbox voices, after the decode loop was put into CUDA graphs, run at about point two. Same family of GPU, a factor of thirteen apart.



Corn

So what did quantizing have to do with it? Nothing.



Herman

Nothing. And the vLLM Omni team published the same pattern for this exact model family. On an H two hundred, the plain Hugging Face path gave two point six four. Their optimised path with batching and CUDA graphs gave point one six, with the first audio packet in about sixty-four milliseconds.



Corn

A top-end datacentre card, and the stock Python path is still slower than real time. That's not a hardware problem.



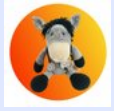
Herman

Each step is small, and launching it costs more than doing it. Shrinking the weights from sixteen bits to eight doesn't touch that cost. At batch size one, I'd bet the per-step overhead is the bottleneck, and I'd bet the eight-bit run on the Radeon barely moves. That's a bet, not a result.



Daniel

How does that look in a real episode?



Herman

In one production episode, your turns, about twenty-four of them, took between a hundred and thirty and a hundred and ninety-five seconds across two containers. Chatterbox did around a hundred and thirty turns in roughly ninety.



Corn

So one voice does five times the turns in less time, and the other one is you.



Daniel

That's a bit harsh. But the lesson is measure first.



Herman

Measure first, and measure at more than one batch size. At one request, you're testing launch overhead. At ten concurrent requests, the vLLM numbers moved from two point one nine to point three one for the optimised path, so the bottleneck sits somewhere else entirely. You want both, because they answer different questions. Corn, what would you want out of the Radeon? One voice, quickly, or a queue of them?



Corn

One voice, and I'd want it before the kettle boils. A queue is Daniel's problem, he's the one producing. But I'd keep the batch run in the suite anyway, in case the answer surprises me.



Herman

That's the right instinct.



Corn

I'm allowed one a year. Speed's half of it, though. If I squash the model to eight bits and it's twice as small but the voice sounds like Daniel after a head cold, what did I benchmark?



Herman

Nothing useful, and that's the gap. For a clone you want a quality arm next to the speed arm. Run the output through a speech recogniser and compute word error rate against the input text, which catches mispronunciations and skipped words. Then compare a speaker embedding of the output with a reference recording, a cosine similarity, which catches the



Herman (continued)

voice drifting. My guess is similarity is the first casualty of quantization. I don't know that.



Daniel

So I generate the same sentences in bfloat sixteen and eight-bit, and compare both against the same references.



Herman

Same sentences, same seed, and listen to some of them yourself, because a high score can hide a voice that sounds off. Would you trust the cosine number, Corn?



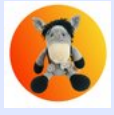
Corn

For telling me which one to listen to first. After that my ears win. They're slow but hard to fool.



Daniel

Right, so the NVIDIA side. I don't have a card, but how different is it?



Herman

Barely, which surprised me. The method is the same: warm-up, synchronise, fixed sentences, the median of several runs. The names change. The monitor is nvidia-smi, which sits on a library called NVML, and its logging modes record power, clocks and memory every second to a file. The profiling pair is Nsight Systems for the timeline, with NVTX markers you place in your own code so you can see the model's phases, and then Nsight Compute for individual kernels. Timeline first, kernel second, same order AMD recommends.



Corn

Same recipe, different knives.



Herman

And NVIDIA is where the fast paths appear first, because that's where the CUDA graph work gets done. The Qwen repo itself uses flash attention in bfloat sixteen in its examples. Your Radeon gets those things later, with more rough edges, but they do arrive. Am I being unfair to AMD there, do you think?



Corn

A bit. A community Rust and HIP implementation reports around half real time on a Radeon workstation card, against nearly two for the matching Python path. It's self-reported, but the gap is the same shape as yours.



Herman

Fair, I'd forgotten that one. A community server for this model also warns that one setting, the size of the CUDA graph window, causes a five to ten times slowdown on ROCm if you get it wrong, and recommends seventy-two. So the fast path is there, and it's a bit brittle.



Corn

One more thing about the first-audio number. Qwen's own report claims about a hundred milliseconds for it, and that's the number that goes on the poster.



Herman

On a fast card, with their serving stack, streaming. A community project reports a hundred and seventy-four on a forty ninety with a small chunk size. Notice that the chunk size is itself a setting you tune. A smaller chunk gets audio out sooner and costs you throughput. So the first-audio figure needs its chunk size written beside it, like the formula for the real time factor.



Daniel

So there are really two benchmarks, streaming and offline.



Herman

For your use, offline, probably. You're producing a podcast, not talking to a phone. Offline you only care about total time and quality. If Meorav ever answers a live call, the streaming numbers become the ones that count.



Corn

I hear Meorav taking a call and I think about the hold music.



Daniel

Can I use other people's NVIDIA numbers as a reference, then?



Herman

Only as a hint about the shape. The formula has to match, the audio length has to match, and you need to know whether it was a warm run. One writeup has the Strix Halo at about three on the plain PyTorch path, similar to an Apple laptop chip. Treat those the way you'd treat a friend's car mileage.



Daniel

Okay. Last bit. How is all this different from benchmarking an LLM?



Corn

I'd have guessed it isn't, and that's why you left it last.



Herman

Mostly it isn't. An LLM benchmark reports tokens per second, time to first token, and the gap between successive tokens. A speech model has the same shape. Time to first token becomes time to first audio chunk, and the gap between tokens becomes the chunk cadence, which has to stay shorter than the audio each chunk contains or playback stutters.



Corn

So the real time factor is tokens per second in a different hat.



Herman

Close. The difference is that a token rate has no natural pass mark and a real time factor does: one. That's why speech gets its single threshold. The other difference is the tail. An LLM processes the prompt in one big parallel pass. In TTS the decode loop is the sequential bottleneck, and the waveform decoder is the parallel part at the end.



Daniel

Are there standard tools on the LLM side I'd borrow?



Herman

llama.cpp ships a bench tool that reports prompt speed and generation speed separately, with repetitions and a standard deviation. vLLM has a serving benchmark that gives time to first token, per-token time and the tail percentiles. Neither understands audio, so you bring your own seconds of audio for the denominator. And I haven't seen a standard industry benchmark round for speech generation, though there is one for Whisper-class recognition.



Corn

Hold on. Sixteen codebooks per frame. Is that sixteen sequential passes or one pass that spits out sixteen?



Herman

I'm not sure about the exact schedule, I'd have to look it up. What I do know is the model produces the multiple codebook entries for each of those twelve hertz frames, so twelve frames a second, sixteen entries each, and that's your token budget per second of speech.



Hilbert

Twelve and a half.



Herman

The frame rate? It's called the twelve hertz tokenizer.



Hilbert

The name rounds it down. Twelve point five. Sixteen codebooks, that part's right. Gerard Pimm in flat nine runs five fans. Four fans. The one in the fridge counts, the one on the cat doesn't. He has a rack of three cards bolted to the fire escape, which belongs to the council.



Corn

He's benchmarking on a fire escape.



Hilbert

The council won't let him have the stairwell. He pays a pigeon on the ledge sixty-two euros a month and nothing runs until the pigeon has left. That's the cool-down. The pigeon is called Dr Salazar. Gerard reads each result out to him in Portuguese, and Gerard doesn't speak Portuguese. He reads it phonetically off a card his sister wrote in Faro, so the pigeon has been told that the real time factor is nineteen hundred and some beans.



Herman

A fixed cool-down before every run is sensible. You want the card at a known temperature when you start.



Hilbert

The clock dropped every afternoon around four, Herman. Eleven days and he didn't see it. The sun comes round the Hendricksons' chimney and sits on the cards. His real time factor went from point four to point nine and he blamed the driver. He wrote to the driver. The driver is a man in Santa Clara and Gerard has had a reply, it said thank you for your



Hilbert (continued)

feedback.



Corn

Cards slow down when they get hot. It would have been in a log, if he'd had one.



Hilbert

He has a log now. On paper, in pencil, clipped to the railing with a bulldog clip from the Hendricksons' son, who wants it back. The laptop went off the fire escape in March and landed on a Skoda. The owner has been very understanding about it.



Herman

Then the number from four o'clock is a different measurement from the number at midnight. Both are true.



Hilbert

He runs from eleven at night to five. The pigeon works days. They don't overlap. The cards are still out there, and the council won't say who owns the fire escape, so Gerard isn't asking.



Corn

The afternoon figure is the one that matches a card in a warm room, Hilbert. Which is where most of these cards actually live.



Hilbert

Gerard has both. The afternoon one is on the council form. The night one is in a biscuit tin at his mother's.



Herman

Daniel's Radeon sits in a room, not on a fire escape, but the sun on the window might do the same. I'd run his bf sixteen and eight-bit tests twice, once cold and once after the room's had a few hours of afternoon.



Hilbert

Gerard won't like twice. He'll do it.



Corn

Then Daniel's plan is short. Log the clocks next to every run, throw away the first run, and settle on a sentence set before touching the eight-bit version.



Daniel

I'll do a profiler trace on the bfloat sixteen run first, before I even load the eight-bit weights.



Herman

Trace first. If the timeline shows the GPU idle between a lot of tiny kernels, quantization is the wrong lever and you go looking for a graph capture path. If it shows the memory bus saturated, the eight-bit run has a real chance of helping.



Corn

Two point six and point two were the same sort of GPU. The number belongs to the whole stack, not to the model.



Herman

A speed figure without its formula, its batch size, its warm-up policy and its clock log isn't a measurement yet. For Meorav that means a speed arm and a voice arm, and you want both before you declare the eight-bit version better.



Corn

What I'd like to know is whether the Radeon gets the same graph treatment the cloud voices got, or whether Daniel ends up writing it himself on a warm afternoon.



Herman

My money's on someone else shipping it first, and Daniel finding out what it does to the fan.



Corn

Thanks to our producer Hilbert Flumingtop for keeping the whole thing running. This has been My Weird Prompts.



Herman

You'll find every episode at my weird prompts dot com.



Daniel

Cheers, talk soon.



Corn

We'll be back soon.