Episode Book

EPISODE #5991

Batch Inference on Your Own GPU: vLLM, SGLang, Ollama, llama.cpp

PUBLISHED	15 October 2026 at 09:34 UTC
DURATION	21:12
LISTEN	myweirdprompts.com/episode/self-hosted-batch-inference-gpu-37921/
HOSTS	Corn, Herman, Jim, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
BOOK GENERATED	19 October 2026 at 17:42 UTC

SYNOPSIS

Hosted batch inference is one of the best deals in AI right now — half price, no latency requirement, just hand over a file. But what if you want that on your own GPU? This episode unpacks why the hosted batch API is really a scheduling and billing construct, and what you actually have to build locally to reproduce it. We walk through continuous batching, the KV cache budget that secretly governs your concurrency, and the two shapes a local batch job can take: offline in-process, or server plus concurrent client. Then we compare the real options — vLLM's offline class and its newer run-batch command, SGLang's prefix-sharing strengths and AMD weak spots, Ollama's development-tool defaults, llama.cpp's portability, and why Hugging Face's text generation inference is no longer a starting point.

TRANSCRIPT



Corn

Daniel sent us a prompt that starts from a very reasonable place. Batch inference on hosted APIs is one of the best deals in AI right now. You write a file of requests, you send it off, you wait a few hours, you pay half price. DeepSeek does it, a few others do it.



Herman

It is the best throughput-per-dollar on the market, and it is not close.



Corn

And Daniel's question is, why can't I have that on my own box. He's got a Linux server, some GPU, AMD or NVIDIA, he wants to point a file at it and get a file back. Does he need a special framework? Is Ollama enough? What changes when the card is red instead of green?



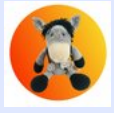
Herman

The short version is that the hosted version is a product and the self-hosted version is a configuration problem.



Corn

Which is the whole episode.



Herman

A batch API on a hosted provider is a billing construct. They take your file, they fill their idle GPU capacity with it, and they give you half off because the work has no latency requirement. You are selling them the right to schedule you whenever. That's the deal.



Corn

So the discount is arbitrage, not generosity.



Herman

Exactly that. Which means the thing you're trying to reproduce locally isn't the API. The API is a thin wrapper. The thing you're reproducing is the scheduling.



Corn

Define the thing properly, because "batch inference" gets used to mean about four different things.



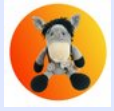
Herman

On your own hardware, batch inference is just high-concurrency inference with continuous batching. There is no separate mode you install. The engine adds and retires requests from the running batch on every decode step, so the GPU never sits idle waiting for the slowest sequence in a fixed group to finish.



Corn

Say that again, because I think that's the part people skip.



Herman

With a naive setup you take a group of requests, you run them together, and the whole group finishes when the longest one finishes. Everything shorter than the longest one is wasted capacity. Continuous batching drops the finished ones the moment they're done and slots new ones in, so the batch is constantly turning over.



Corn

And the file-in, file-out part on top of that?



Herman

Forty lines of Python, if you want it. Or you use the wrapper the engine already ships. But the wrapper is nothing without the scheduler underneath.



Corn

So what does the actual work look like on a Linux box. Two shapes.



Herman

Two shapes, and picking the wrong one is where most of the lost throughput lives. The first is offline and in-process. You load the model in Python, you hand it the whole list of prompts, you wait on the call. vLLM's offline class does this. SGLang's engine does this. No HTTP server at all.



Corn

And the second?



Herman

Server plus concurrent client. You stand up an OpenAI-compatible endpoint and you fire many requests at it at once from an async client with a concurrency limit. What you must not do is loop over your prompts one at a time. If you go sequential, you get the latency of a single request multiplied by your list length, and you have thrown away everything the engine was built for.



Corn

The offline shape is very close to my ideal, I'll be honest. You hand it a list, you walk away.



Herman

You do love a machine that works while nobody is watching.



Corn

I'm suspicious of how much I approve of it. But yes.



Herman

Here's the thing though, Corn. The offline shape is not just ergonomics. It skips the whole HTTP layer, and that matters more than people expect.



Corn

Because the server is not free.



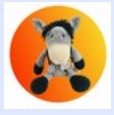
Herman

The server is a process, a socket, a serialization step per request, and a client that has to be written well enough not to become the bottleneck. For a one-shot job on a file you already have, all of that is overhead you don't need.



Corn

But if you're feeding it from other services?



Herman

Then you want the server, and you want it concurrent.



Corn

The mechanism underneath all of this is continuous batching, and once you see it, the rest of the tooling makes sense.



Herman

And the constraint underneath continuous batching is not compute. That's the mental model I'd tattoo on people. It's the KV cache.



Corn

Explain the cache.



Herman

Every sequence you're generating needs its attention keys and values held in memory for every token of context it has seen. That memory per sequence grows with context length. So your realistic concurrency is the VRAM you have left after the weights, divided by the per-sequence KV cost at your context length.



Corn

So a hundred and forty gigs of weights on a big card, and whatever's left over is your batch budget.



Herman

And it is a budget. Shorter max context, quantized weights, quantized cache, all of them buy you more concurrent sequences. This single ratio explains almost every tuning decision you'll make.



Corn

What happens when you run out.



Herman

vLLM preempts requests and recomputes them later. RECOMPUTE is the default in V1. Your throughput falls off a cliff, and it looks like the machine getting slower rather than the machine being full.



Corn

That's the detail I'd want before I rent anything. Being full should look like being full.



Herman

It doesn't. It looks like a mystery.



Corn

So the card is full and the job is fine and the wall clock is twice what you planned.



Herman

Right. And the fix is not to add compute. The fix is to shrink the context or shrink the cache.



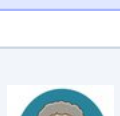
Corn

Which leads us to vLLM, which is the default answer for serious batch work.



Herman

vLLM is where I'd start on anything resembling a real card. Two entry points. The offline class, where you construct the model and call generate or chat on a list of prompts. And the newer command, vllm run-batch, which takes an input file and an output file and a model name.



Corn

And that one reads the same format as the hosted batch APIs.



Herman

Deliberately. Each line is the OpenAI Batch schema. A custom ID, a method, a URL, a body. The accepted URLs are the chat completions route, embeddings, audio transcriptions, audio translations, and anything ending in score or rerank.



Corn

Score or rerank covers a lot of retrieval work.



Herman

It does, and it means you can batch your reranking pass on the same file as your chat pass, which is a nicer workflow than people realize.



Corn

What happens when a line is malformed.



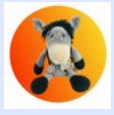
Herman

You get an error entry for that line. The whole batch does not die. That's the property you want in a job that runs for three hours.



Corn

And it does not start a server.



Herman

It runs in-process, with the optional exception of a metrics port if you want Prometheus to see it. Input can be a local path or an HTTP URL, and the output can be pushed with an HTTP PUT, so you can run the whole thing on a box and have the result land somewhere else.



Corn

The old invocation, the python dash m one?



Herman

Deprecated. If you find a tutorial using it, the tutorial is stale.



Corn

Now the knobs. Which ones actually move throughput.



Herman

Three. gpu_memory_utilization, which is the share of VRAM you pre-allocate, and most of that pre-allocation goes to the KV cache. max_num_seqs, which caps how many sequences are in flight. And max_num_batched_tokens, which caps how many tokens go into a single step.



Corn

And the docs have a recommendation on that last one.



Herman

Above eight thousand one hundred and ninety-two for throughput, on smaller models on large GPUs. And chunked prefill is on by default in V1, which is what lets a long prompt and a short prompt share a step without one starving the other.



Corn

Prefix caching.



Herman

That's the one that pays for itself in a batch job, because the typical batch job has every request sharing the same long system prompt. Cache it once and the preamble stops being a per-request cost.



Corn

So a thousand rows with the same four-hundred-token instruction block.



Herman

You pay for the instruction block once, and the other nine hundred and ninety-nine rows skip straight to the part that's actually different.



Corn

Herman.



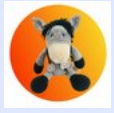
Herman

Yes.



Corn

Is there a downside to turning every one of those knobs up at the same time?



Herman

There is, and it's the one people find by accident. Push max_num_seqs too high and you blow the KV budget, and then you get the preemption-and-recompute behaviour instead of more throughput. The knobs fight each other.



Corn

So the right setting is the one where the cache exactly fits.



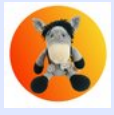
Herman

Approximately, yes. You're looking for the point just before the preemption starts.



Corn

The other serious option is SGLang.



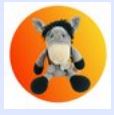
Herman

SGLang has an engine class with a generate call and an async variant, and it is strong on workloads where every request shares a prefix. That's its home turf.



Corn

Where does it lose.



Herman

On AMD. Its ROCm path targets the datacenter Instinct cards. Consumer Radeon support is weaker than vLLM's or llama.cpp's, and the third-party comparisons consistently rate vLLM's ROCm support as the more mature of the two.



Corn

We touched that on the DeepSeek serving episode. Same conclusion.



Herman

Same conclusion, and it hasn't changed.



Corn

Now the one everybody's going to ask about. Ollama.



Herman

Ollama is a development tool and it is very good at being a development tool. It is not a batch engine, and the defaults are the reason.



Corn

Give me the number.



Herman

Parallelism defaults to one. One request at a time per model. There is a queue, it defaults to five hundred and twelve deep, and past that it starts rejecting requests outright.



Corn

So your script gets connection errors at five hundred and thirteen, not slow throughput.



Herman

Correct. And the memory model is the part people miss. Each parallel slot gets its own context. Four slots at two thousand context is eight thousand of context to hold, not two thousand. So raising parallelism raises memory linearly.



Corn

And there's no file endpoint.



Herman

There's no batch route at all. You can raise the parallel setting and hammer it with concurrent requests, but Ollama isn't pretending to be a throughput server.



Corn

There's also the CISPA finding from this year.



Herman

A study found Ollama's defaults change benchmark scores. A forced beginning-of-sequence token, and a hidden repetition penalty of one point one applied under the hood.



Corn

Which is invisible in a chat window and very visible across ten thousand rows you never read.



Herman

That's the quiet trap of batch work generally. It is exactly the workload where a silent sampling change goes unnoticed.



Corn

llama.cpp's server is the other end of the spectrum.



Herman

The portability answer. It runs on CUDA, on ROCm, on Vulkan, on Metal, on plain CPU, with GGUF quantized weights. Continuous batching is on by default. The parallel flag sets your slot count and defaults to auto.



Corn

Auto meaning?



Herman

It picks a sensible number based on your context and memory. And when it's auto, the unified KV buffer is on by default, so the slots share one buffer instead of each getting a fixed slice.



Corn

Older builds did it differently.



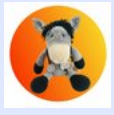
Herman

Older builds split the context evenly across the slots, so you'd set thirty-two thousand context and four slots and each one silently got eight. That caught a lot of people.



Corn

Flags worth knowing.



Herman

The logical batch size defaults to two thousand and forty-eight and the physical batch to five hundred and twelve. Neither is a file endpoint. But the completion route takes multiple prompts and the embeddings route takes an array, so you can hand it a group at once.



Corn

Peak concurrent throughput lower than vLLM.



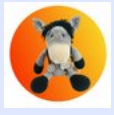
Herman

Lower, yes. But it will run on the card you already own, and that is the trade.



Corn

What about Hugging Face's text generation inference.



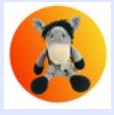
Herman

In maintenance mode, by their own docs, since December of last year. Minor fixes only. They point new deployments at vLLM and SGLang, and at llama.cpp and MLX for local.



Corn

So don't start there.



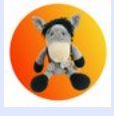
Herman

Don't start there. It was a good answer and it isn't one anymore.



Corn

Hold on.



Herman

Go on.

J

Jim

I'm calling about the word "decoration." Somebody on your show just called continuous batching decoration, and I want it on the record that my neighbour has been running a dehumidifier in his garage for eleven months straight, and the council has done nothing, nothing at all, and now I'm supposed to listen to you people describe throughput like it's a hobby. I sat in that planning meeting. I sat in it.



Corn

Jim, that's not quite what was said. We were saying the tuning knobs are secondary to the cache budget. Nothing got called decoration.

J

Jim

And the parking ticket. Two hundred and forty shekels for a meter I paid. I have the receipt. I have the receipt and the council has the receipt and somehow I'm the one who's wrong.



Herman

We're actually on the memory side of this, Jim. The cache is the whole argument.

J

Jim

I'll be calling back about the dehumidifier.



Herman

The card choice question. All of the tooling assumes the toolchain works on your card.



Corn

Daniel asked it directly. Does the red versus green choice change the answer.



Herman

It changes the answer completely, and not for the reason people think. It's not compute. Both vendors make cards that will do the arithmetic. It is which software stacks have been built and tested for which cards.



Corn

Start green.



Herman

NVIDIA, vLLM wants compute capability seven point five or higher. That's the T4, the RTX 20-series and up, A100, L4, H100, B200. It is a wide, well-trodden path. If you have a modern NVIDIA card, run-batch is the obvious default and the rest of the episode is tuning.



Corn

And red splits into two worlds.



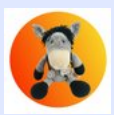
Herman

It does, and conflating them is the mistake. Datacenter Instinct is one world. vLLM supports ROCm six point three or higher on MI200, MI300, and MI350, with the last one wanting ROCm seven or above. AMD publishes prebuilt ROCm vLLM Docker images, documented mainly for MI300X. On Instinct, the answer is vLLM from AMD's own image.



Corn

That's a supported path, not a workaround.



Herman

It's a supported path. AMD has been investing there because that's where their datacenter customers are.



Corn

And the other world.



Herman

Consumer Radeon. vLLM supports the RX 7900 series, the RX 9000 series, and the Ryzen AI MAX and AI 300 APUs. So it's not unsupported. But SGLang's consumer Radeon support is weaker than vLLM's or llama.cpp's.



Corn

So on a consumer Radeon the pragmatic answer is often llama.cpp with a few parallel slots.



Herman

Which works, runs on the card you own, and gives up peak throughput in exchange.



Corn

The rule of thumb, then.



Herman

Datacenter card or recent NVIDIA, use vLLM, run-batch for OpenAI-format JSONL or the offline class, or SGLang if your workload is all shared prefixes. AMD Instinct, use vLLM from AMD's ROCm Docker image. Consumer Radeon, a mixed box, or low VRAM, llama.cpp with a few parallel slots, and vLLM if you're on RX 7900 or 9000 series. Ollama is for development.



Corn

Now the bit that gets skipped.



Herman

The economics inverting. On a hosted batch API you pay half price for someone else's idle capacity. On your own box you're not paying half price. You're paying in configuration, in cache tuning, in the risk that the toolchain you picked is the one that doesn't quite work on your card.



Corn

And you're paying it up front.



Herman

Up front, and it's fixed. Which means it only amortizes if your batch is large. A fifty-row job does not justify any of this. A million-row job justifies all of it.



Corn

There's a break-even and nobody publishes it.



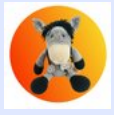
Herman

Because it depends entirely on your card, your model, and how much your own time is worth. But the shape is simple. Small batch, hosted wins. Large batch on hardware you already own, self-hosted wins.



Corn

One thing about running this on the machine in the flat.



Herman

Go on.



Corn

The card is doing something for four hours and I can't use the machine for anything else.



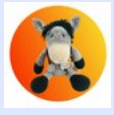
Herman

You can't. The memory is pre-allocated to the cache. That's the cost of the throughput.



Corn

I've made peace with it. Mostly.



Herman

Herman Poppleberry, by the way, in case anyone forgot.



Corn

Nobody forgot.



Hilbert

It's not sixteen.



Hilbert

The number you'd have landed on if you'd kept going. Sixteen sequences. I had it at sixteen on the machine in the back room, and everyone keeps talking about the weights like that's the constraint.



Herman

We've been saying the cache is the constraint.



Hilbert

The cache is all of it. Everything else is decoration.



Corn

We said that. That's what we said.



Hilbert

I know what you said. I'm putting it on the record. Sixteen sequences, and the way I got there was cutting the context window in half, not swapping to a smaller quantized model. Same weights, half the context, more sequences than the smaller model would have given me.



Herman

That's a clean result, actually. Halving context halves the per-sequence KV cost.



Hilbert

It does. And the machine in the back room will do it, but only at night.



Corn

Why only at night.



Hilbert

Because of the cooling. There's a dehumidifier involved, and it's loud enough that the neighbours would ask questions during the day. So the batch jobs run at night, and the dehumidifier runs at night, and that's the arrangement.



Corn

The dehumidifier is doing the cooling.



Hilbert

It's moving air. There's a card in there I couldn't name for you if you asked. The power supply is two smaller ones I wired together, which I would not recommend to anyone, and I'll say that much.



Herman

Sixteen sequences on a card you can't name.



Hilbert

Sixteen. Round numbers bother me less than they bother you two. That's the count. And the dehumidifier filter needs changing every six weeks, which nobody ever puts in the manual, and I've got one on order that's late.



Corn

Every six weeks.



Hilbert

That's what it says on the box. It's wrong. It's four if the job runs long.



Herman

So the KV budget is the whole game, and the way you buy concurrency is by spending context.



Corn

The way Hilbert buys it, anyway. With a dehumidifier and a power supply he built.



Herman

There is a real lesson buried in that, which is that the cache is the budget and everything else is bookkeeping.



Corn

I'm not going to think about the power supply.



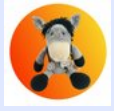
Herman

No.



Corn

Herman, the thing I keep circling is that the tooling has stopped being the obstacle. Two years ago this conversation would have been about whether it was possible at all on consumer hardware. Now the answer on almost any card is yes, with a caveat about which engine.



Herman

That's the shift. The bottleneck moved from feasibility to whether your batch is big enough to justify the setup.



Corn

And vLLM and llama.cpp are both converging on OpenAI-compatible batch endpoints, which raises the obvious question.



Herman

Whether the hosted discount survives. If self-hosting a large batch is a config file and an evening, the half-price deal is competing against your own idle card.



Corn

And on a machine that's already doing something at night anyway, the comparison gets uncomfortable for the hosted side.



Herman

It does. Though the hosted side still wins the small jobs, and it wins the jobs where you don't own the hardware.



Corn

Which is most jobs.



Herman

That's the honest place to leave it.



Corn

Thanks as ever to Hilbert Flumingtop, our producer.



Herman

For more along these lines, there's episode two, Local STT For AMD GPU Owners; episode three, Safetensors or something else; and episode twenty-four sixty-four, Batch APIs. This has been My Weird Prompts, the human-AI collaboration podcast.



Corn

If you want to send us something to argue about, the Telegram bot is t dot me slash MWP listener bot. Everything else is at my weird prompts dot com.



Herman

We'll be back soon.



Corn

See you tomorrow.