Episode Book

EPISODE #6345

How vLLM Squeezes More Requests into GPU Memory

PUBLISHED	29 October 2026 at 15:10 UTC
DURATION	29:55
LISTEN	myweirdprompts.com/episode/vllm-gpu-memory-paged-attention-20743/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.25
SOURCES	12 cited by the research (listed at the end)
BOOK GENERATED	30 October 2026 at 07:19 UTC

SYNOPSIS

Language model serving often stalls not from slow GPU arithmetic, but from greedy memory allocators reserving space for tokens that never arrive. Explore how UC Berkeley's vLLM project borrowed virtual memory and paging concepts from operating systems to create PagedAttention. By breaking KV caches into non-contiguous physical blocks, sharing prompt states through copy-on-write, and enabling prefix caching, the engine slashed memory waste and enabled dramatically higher request throughput on identical hardware.

TRANSCRIPT



Corn

Faster inference usually gets sold as a contest to make a GPU do more arithmetic. vLLM became important by making it reserve less empty memory. Daniel keeps seeing this engine mentioned wherever people serve open models, and he wants to know what actually earned it that position. Beyond the familiar recommendation to just use it, what did it introduce, and why does that still matter?



Herman

The original insight was wonderfully unglamorous. You could have expensive accelerators, a working model, and enough theoretical compute, yet struggle to admit another request because your memory allocator had promised space to conversations that might never use it.



Corn

Speculative property development, inside a graphics card.



Herman

With a remarkably expensive address. vLLM borrowed an operating system idea, virtual memory, and applied it to the growing attention state of language model requests. That let more requests share the same hardware without changing the model architecture.



Corn

You are already doing the voice you use when somebody discovers a block table.



Herman

Corn, somebody discovered a very useful place to put a block table.



Corn

There it is.



Herman

The important part is what happened after that discovery. A Berkeley project became something model developers actively expect their users to run.



Corn

We should pin down what you actually get. Downloading vLLM does not download an intelligence upgrade. It gives you machinery for running weights you already chose, taking incoming requests, scheduling their work, and getting answers back out.



Herman

It is an Apache two point zero licensed Python library for inference and serving. It started in February twenty twenty-three at UC Berkeley's Sky Computing Lab. You can use it inside a Python application for offline batch inference, or run its server and send requests over an API.



Corn

Offline meaning I have a pile of documents and want them processed, rather than somebody waiting in a chat window.



Herman

Yes. The library's LLM class handles that batch workflow. Online, the familiar entry point is its OpenAI compatible server. That matters because an application already written around that interface can often switch endpoints instead of rebuilding its whole model integration. It also supports the Anthropic Messages API and gRPC.



Corn

Same request shape does not guarantee identical behavior.



Herman

Keep that objection. It becomes important when we get to tool calls. The engine handles scheduling, memory allocation, execution, and the plumbing around model outputs. Every one of those can affect what the application experiences.



Corn

Its reach is broader than the name suggests, too. People hear language model engine and picture a single kind of decoder on an NVIDIA card.



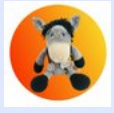
Herman

It supports more than two hundred model architectures, encompassing mixture of experts, attention and state space hybrids, multimodal systems, embeddings, and reward models. Hardware coverage includes NVIDIA, AMD, and Intel GPUs, several CPU architectures, and plugins extending to TPUs, Gaudi, Ascend, Apple Silicon, and other accelerators.



Corn

That is a large collection of things to disappoint simultaneously.



Herman

Supporting them does not mean every combination has equal maturity or equal performance. It means there is a common project where those integrations can live. That is a substantial attraction for hardware vendors.



Corn

For an application team, too. I do not want my business logic to depend on whether one accelerator vendor remembered my model exists.



Herman

Would you trade the fastest possible implementation for that broader compatibility?



Corn

Some speed, yes. Not correctness. If I have to rebuild the application when I change the machine, the cheap machine has acquired a consulting department.



Herman

That distinction explains a lot of its adoption. First, though, the memory saving has to be real.



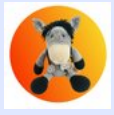
Herman

In the original serving problem, the weights were a large fixed commitment and each active conversation added a growing KV cache. Take the paper's thirteen billion parameter model example. About sixty-five percent of device memory held weights, and close to thirty percent was available for the cache.



Corn

That does not leave a heroic amount of room for waste.



Herman

Especially when a single LLaMA thirteen billion sequence could have a cache reaching about one point seven gigabytes. The exact size depends on sequence length and architecture, but the consequence is simple. Reserving too much for each conversation can exclude other conversations entirely.



Corn

I would have guessed the weights were the whole problem. Once they fit, you were mostly arguing about how fast the GPU could multiply.



Herman

That intuition works poorly for loaded autoregressive serving. Memory capacity limits how many sequences you can keep active, while memory bandwidth often constrains decoding speed. Prefill can be compute intensive, so saying all inference is memory bound would be too broad. The original breakthrough targeted a very specific bottleneck in serving concurrent requests.



Corn

The request also refuses to tell you how long it will be. Someone asks for a brief answer, then the model develops literary ambitions.



Herman

Right... the allocator must accommodate a future it cannot predict. Earlier systems commonly reserved contiguous space based on maximum sequence lengths or allocation policies designed around contiguous storage. That creates two different kinds of waste.



Corn

Space inside an allocation that never gets used, and gaps between allocations that cannot fit the next request.



Herman

Internal and external fragmentation. The first is particularly painful when you reserve for a long possible response and get a short actual response. The second appears as requests arrive, grow, and finish at different times. Enough total free memory can exist without a suitably placed contiguous region.



Corn

Empty seats, but none together.



Herman

The original profiling found actual token states occupying only about twenty to thirty-eight percent of the cache memory in the systems examined. The launch description put the wasted share at sixty to eighty percent.



Corn

Most of the cache was a reservation.



Herman

Physical memory committed to possibilities rather than tokens already present. That is why this was more consequential than shaving a little overhead from allocation.



Corn

Walk through the block table without giving it a personality. I know you are tempted.



Herman

A request sees its cache as a logical sequence. vLLM divides that sequence into fixed size blocks. Each logical block maps through a table to a physical block in device memory. Those physical blocks do not need to sit beside one another.



Herman

When a sequence grows enough to need another block, the system allocates one. The attention implementation follows the mapping to read the appropriate state. You have separated the order the model needs from the physical layout the allocator can conveniently provide.



Corn

My first objection is that following the table costs something. We have exchanged waste for indirection.



Herman

It does cost something. The trick required an attention implementation designed for that layout, rather than merely adding a general purpose lookup to an existing path. You accept controlled bookkeeping to recover much more usable capacity and keep a larger batch active.



Corn

Does the last block still contain unused space?



Herman

Yes. A sequence generally leaves some slack in its final block. You have not abolished rounding. You have bounded the waste instead of reserving the entire hypothetical future. The project reported memory waste below four percent.



Corn

A rare allocation policy with modest ambitions.



Herman

What would you expect to happen to throughput once more sequences fit, Corn?



Corn

Higher total output, assuming we have enough simultaneous demand. Not necessarily a faster answer for the person alone on the server. In fact, under load, the scheduler might spend that recovered space admitting other people rather than spoiling me.



Herman

That is the distinction people lose in the headline multipliers. Serving throughput is aggregate work completed under particular latency conditions. It does not automatically mean each isolated request becomes proportionately faster.



Corn

I am comfortable sharing. Your insistence on instantaneous replies remains a personal problem.



Herman

You would defend a frozen progress bar if it looked restful.



Corn

A frozen progress bar has made no false promises.



Herman

This one does make measurable progress. In the original paper, an OPT thirteen billion workload using ShareGPT conversations supported roughly thirty concurrent requests, against seven under one of Orca's allocation policies. The exact improvement changed dramatically with the request length distribution.



Corn

Short responses punish maximum length reservation more severely. You pay for a mansion and use the porch.



Herman

Yes, and that is why I would not export that ratio to every production workload. Long inputs, long outputs, different architectures, different scheduling targets... they change the result.



Corn

The operating system borrowing goes further than scattering blocks around memory. You can share them.



Herman

This is the part I like most. Multiple sequences can refer to the same physical blocks when they share state. A reference count records that sharing. If one branch needs to modify shared state, copy on write creates the necessary private copy instead of duplicating everything at the outset.



Corn

Generate several possible continuations from the same prompt, and you do not need several copies of the prompt's entire cache.



Herman

Parallel sampling benefits, and beam search can benefit even more because branches share substantial histories. The original beam search experiments reported memory savings roughly in the forty to sixty percent range, depending on workload. That changes how many alternatives you can explore on a fixed device.



Corn

Useful even if I do not use beam search. A service with a repeated system prompt has the same basic opportunity.



Herman

Prefix caching. Reuse the already computed cache for an identical eligible prefix rather than compute and store it independently for every request. This is where block management becomes a foundation for other optimizations, not just a better way to fit requests.



Corn

Eligible is hiding the actual restriction. Similar meaning does not count. A rewritten prompt does not earn a cache hit merely because I think it says the same thing.



Herman

Token identity and the implementation's cache rules matter. Sharing also requires appropriate isolation boundaries. A provider cannot simply assume every request is allowed to reuse every other request's cached state.



Corn

It makes repeated application scaffolding economically interesting. All those long instructions that never change between requests.



Herman

They can be a substantial reusable prefix. Though you still pay to decode the new answer, and a high cache hit rate does not mean every part of inference vanished.



Corn

The memory savings are real, but the maintenance is what keeps it on top.



Hilbert

Hilbert's Mid-Show Haiku. To the spare graphics card in the rack.



Hilbert

You hold no weights now. Your fans still turn for no one. I dust you on Thursdays.



Corn

Better maintained than most deployments.



Herman

The first deployment evidence was unusually tangible. LMSYS used FastChat with vLLM underneath for Vicuna and Chatbot Arena. Its serving GPU requirement fell by half while it handled about thirty thousand requests a day, with peaks around sixty thousand.



Corn

Halving the hardware is more persuasive than winning a chart drawn on your own website.



Herman

The launch chart was striking too. Up to twenty-four times the throughput of Hugging Face Transformers, and up to three point five times TGI. Those were launch configuration comparisons, not permanent properties of the project names.



Corn

Otherwise we would have to believe a benchmark stays valid while everything inside both systems changes.



Herman

The paper's equal latency comparisons are a more useful framing. It reported roughly two to four times the throughput against FasterTransformer and Orca. Equal latency matters because you can always make aggregate throughput look attractive by asking users to tolerate a much longer wait.



Corn

Or by quietly changing which requests count.



Herman

Workload definition is part of the result. Prompt lengths, output lengths, concurrency, hardware, and latency objectives all matter. vLLM's initial success combined a credible mechanism, public measurements, and a live service where saved capacity mattered.



Corn

That explains the beginning. It does not explain why everybody still points to the same project after competitors have had time to copy the idea.



Herman

Especially because the original PagedAttention implementation has itself been replaced. The July release, version zero point twenty-five, moved dense models to Model Runner V two and retired the original implementation. That should not be read as abandoning block based cache management.



Corn

The first code retired. The memory stopped being contiguous and stayed that way.



Herman

The underlying approach persists through prefix caching, cache quantization, and newer host resident cache tiers such as HiSparse. Software can preserve an architectural idea while replacing the implementation that first demonstrated it.



Corn

Good. I would hate to discover every database stopped using indexes because somebody rewrote the indexing code.



Herman

The continuing advantage is the work around the idea. New kernels, new hardware, new model architectures, new output conventions, and fixes to all the combinations.



Corn

The least photogenic part of an open source success story.



Herman

Also the part nobody gets to finish. A model release can introduce unfamiliar attention behavior, a different expert layout, a new quantization scheme, or custom multimodal processing. Loading the weights successfully is only the beginning.



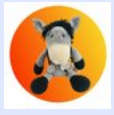
Herman

vLLM is producing roughly biweekly releases carrying seven hundred to eight hundred commits. The latest release had eight hundred fourteen commits from three hundred sixty-three contributors. That is a considerable amount of simultaneous engineering.



Corn

You hear release velocity. I hear your evening disappearing.



Herman

I do install things too early.



Corn

Usually before the phrase known issues has cooled.



Herman

That is why production teams pin releases instead of treating my curiosity as deployment policy. Velocity gives you access to fixes and new support. It also gives you a stream of changes that needs qualification.



Corn

I accept the value, Herman. I do not accept that more commits automatically means a better service. You could spend eight hundred commits repairing seven hundred commits.



Herman

Commit counts are evidence of activity, not a quality metric. The more meaningful result is that support for new architectures lands quickly. Recent releases include DeepSeek V four point one Flash, GLM five point three Flash, K two Horizon, and Kimi K three. For someone trying to serve a newly released model, responsiveness is a product feature.



Corn

Before the benchmark even happens, the engine has to understand the model.



Herman

What would make you pick the less fashionable engine?



Corn

A workload it serves more reliably. Or one specific latency requirement it meets better. Default is a starting assumption, not a commandment. If my service handles one interactive user, a high concurrency throughput trophy does not settle my purchase.



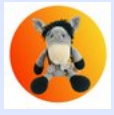
Herman

The comparative work against TGI supports that distinction. It found vLLM reaching much higher throughput under high concurrency, while TGI delivered lower tail latency in the single user interactive setting.



Corn

The last unlucky request matters when it belongs to you.



Herman

There is no universal faster engine independent of workload. I also would not turn vendor case studies into a settled ranking against SGLang. I have not seen an independent peer reviewed comparison that establishes one winner across the relevant workloads.



Corn

Yet the ecosystem has made a choice in a less numerical way. Hugging Face moved TGI into maintenance mode and directed users toward vLLM.



Herman

Maintenance mode in December, followed by archival in March. Mistral's own Small four model guidance recommends vLLM for production ready inference, while listing SGLang as work in progress for that model. Those recommendations are stronger evidence of default status than stars.



Corn

A model developer has an interest in telling customers how to make its product work. That recommendation carries more weight than somebody collecting logos.



Herman

There is a reinforcing cycle. Model developers target the engine users already deploy. Hardware vendors optimize the engine model developers target. Application teams get more reasons to stay. The repository now has roughly ninety-six thousand stars and twenty-four thousand forks, with more than two thousand contributors.



Corn

The forks at least give everybody somewhere to put their emergency patch.



Herman

They also make vendor neutrality more plausible. Apache licensing allows commercial use and modification without asking a central company for permission. The project lists major production users across model companies, cloud infrastructure, and application businesses.



Corn

Permission to fork does not reproduce the people maintaining it, though. If one team loses momentum, a license cannot answer the issue queue.



Herman

Community concentration can be both efficient and fragile. Everyone benefits from sharing the work, then discovers everyone depends on the same work being done.



Corn

Especially now that efficiency means coordinating more than one server. A cache hit on the wrong machine is no cache hit at all.



Herman

That is the move toward an inference control plane. vLLM executes model requests. Something above it must decide where those requests should go, whether a useful prefix is already resident, how loaded each replica is, and whether moving cached state is worth the network cost.



Corn

A round robin router could send my identical prompt to a cold replica while the next machine holds everything it needs.



Herman

Or a cache obsessed router could send it to an overloaded warm replica and make it slower anyway. You are balancing reuse against queueing. I do not think there is one routing policy that wins everywhere.



Corn

The useful state has a location. That sounds obvious until somebody draws all the GPUs as interchangeable boxes.



Herman

llm-d version one point zero explicitly supports multiple model servers, including vLLM and SGLang. It has backing across Red Hat, IBM, Google Cloud, CoreWeave, and NVIDIA, and CNCF governance. The layer above the engine is becoming an important shared project in its own right.



Corn

Keeping more than one engine underneath it seems healthy. Otherwise an abstraction becomes a very elaborate extension cord for one implementation.



Herman

It also lets placement and routing evolve without requiring a single engine to own every cluster level decision. Disaggregated prefill and decode add another dimension. Different stages can benefit from different resource arrangements, but moving state between them has a cost.



Corn

The network receives the invoice for our architectural elegance.



Herman

Exactly... no, more specifically, the network and reliability machinery become part of the performance problem. You have to measure transfer costs, recovery behavior, and contention alongside GPU execution.



Corn

That makes the spectacular cost claims harder to attribute to one component.



Herman

They are system results. IBM reports an eleven point five fold inference cost reduction against an all proprietary configuration, with roughly ninety-eight percent prefix cache hits and a cost around eleven cents per million tokens. That is a very cache friendly workload,



Herman (continued)

not a universal tariff for open models.



Herman

A repeated coding agent workload can expose enormous shared prefixes. In the large GLM deployment, peak output exceeded six point six million tokens per minute, with up to three thousand concurrent coding agents and no preemptions reported in that run.



Corn

No preemptions meaning requests did not have to be pushed out and resumed because the system ran short of cache capacity.



Herman

In that run, yes. It demonstrates coordinated capacity, not a promise that no deployment ever runs out. Separate DeepSeek optimizations pushed throughput per H two hundred from roughly fifteen hundred to twenty-two hundred tokens per second with wide expert parallelism.



Corn

More arithmetic helps after all.



Herman

It certainly does. Better memory use made the original opening. The modern engine also needs serious compute optimization. Reducing everything to PagedAttention would erase years of work.



Corn

There is a nastier measurement problem than a throughput chart with the wrong concurrency. The server can make a good model look incapable.



Herman

Ubicloud's tool call example is unusually clear. Serving Kimi K two point six through an earlier vLLM release produced ninety-two percent tool call schema accuracy. A newer release with the appropriate constrained decoding setting brought that to one hundred percent in their test.



Corn

Valid shape, not perfect decisions.



Herman

Correct. Schema conformity does not prove the model chose the right tool or supplied truthful arguments. It measures whether the output obeyed the required structure. That distinction matters, but the improvement still exposes a serving layer effect masquerading as model quality.



Corn

Somebody tests an open model, sees malformed calls, and concludes the closed model is smarter. They may have compared a polished hosted stack against a misconfigured engine.



Herman

They may. The reverse mistake is also possible, treating constrained output as evidence of better reasoning. An engine can guarantee a permitted structure without guaranteeing a useful answer.



Corn

The weights deserve credit and blame for what the weights did. The sampler, template, parser, and constraints deserve their own entries.



Herman

This is where API compatibility stops being enough. Two endpoints can accept the same request while interpreting tool constraints differently. A benchmark needs the serving configuration to make the result reproducible.



Corn

You once trusted an endpoint because it returned a successful status. Now you distrust anything that looks comfortable.



Herman

I trust the successful status to tell me the request completed. I want separate evidence that it completed correctly. Would you count malformed tool output as a model failure when the application cannot use it?



Corn

For the application, yes. For comparing model capability, I would separate it. The application experiences the whole stack. The experiment should explain which part failed. Otherwise we fix the wrong thing.



Herman

That separation becomes especially important when a default server makes the same configuration choice across many deployments.



Hilbert

Default is a generous word for it, Herman. My cousin calls it the installation. He runs a small serving setup. One customer got quicker replies than another, and he spent weeks saying the model had favorites. It was prefix caching. Their prompts started with the same long instructions. The other customer's did not.



Corn

The slower customer was paying for originality.



Hilbert

They were waiting for it. He kept changing the sampling settings. I told him the cache was doing its job. Nobody had told the customer there was a cache. You explained the block table twice, Herman. He got through several calls without explaining that much.



Herman

Identical prefixes can avoid repeated prefill work. Changing sampling would not remove that advantage.



Hilbert

He gave each customer a separate cache salt in the end. Stopped them sharing across customers. Most expensive way of settling an argument I have heard this month. He called it equal service. Both customers still had different prompts.



Corn

Equal waiting was available sooner than equal understanding.



Hilbert

I have the release counts written in a little notebook. I add them while his installation updates. Eight hundred fourteen commits in the latest one. That has spoiled the average. Seven hundred something was perfectly serviceable. Eight hundred fourteen is an outlier.



Herman

Within that little run of releases, it is the highest. More importantly, separate salts can preserve reuse inside each customer's workload. They do not necessarily erase all the caching benefit.



Hilbert

It still got slower for the customer who telephoned. He was the one who had complained on somebody else's behalf.



Corn

Your cousin could have kept the speed and explained the difference, but he bought an observable form of fairness instead.



Hilbert

He bought fewer telephone calls. They have stopped asking him about it.



Herman

That is another production objective hiding outside the tokens per second chart. Customers need the service's behavior to be intelligible. Two identical model choices can yield different experiences because their requests reuse different amounts of work.



Corn

I would not call the engine's behavior a personality. I would call it something the contract forgot to mention. But I can see why the customer experiences it personally.



Herman

You also have legitimate reasons to isolate cache reuse. Latency equality is only one concern. Security boundaries and tenancy requirements can matter more than the maximum possible hit rate.



Corn

Which makes routing less like a contest to maximize one percentage. The best hit rate may violate the service you meant to offer.



Herman

That is why I hesitate over the word default myself. vLLM has earned a strong starting position through broad support, measurable performance, and an unusually large maintenance community. It does not choose your operating objectives for you.



Corn

The concentration question remains. Apache licensing gives people an exit, but maintaining hardware support and model compatibility at this pace is an expensive form of freedom.



Herman

There is a reported claim that a company called Inferact raised a hundred fifty million dollar seed round around core maintainers, with vendor neutrality described as its mission. I have not seen primary confirmation, so I would not treat the funding or mission as established.



Corn

Commercial money could fund the unglamorous maintenance. It could also influence which unglamorous maintenance gets done.



Herman

Governance and actual contribution patterns will tell us more than an unconfirmed fundraising number. The ecosystem needs sustained engineering either way.



Corn

The idea I am leaving with is that an inference engine belongs inside the measurement. The weights alone do not explain a result.



Herman

The server determines which work gets reused, how requests share capacity, and whether required output constraints are actually enforced. Model comparisons need those conditions attached.



Corn

I wonder whether this ecosystem can preserve both its maintenance speed and customers' ability to understand why the same model behaves differently across deployments.



Herman

Thanks to Hilbert Flumingtop for producing the show.



Corn

For more along these lines, there's episode three, Safetensors or something else; episode twelve, The AI Breakthrough; and episode ten, How ASR Went From Frustration To ... Whisper Magic. This has been My Weird Prompts, the human-AI collaboration podcast.



Herman

If you enjoyed it, leave us a review wherever you listen. It helps other curious people find us.



Corn

See you tomorrow.

SOURCES

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

1. **vLLM repo, README + releases v0.30.0/v0.31.0/v0.32.0, last push 2026-10-29 [primary]**
https://github.com/vllm-project/vllm

2. **Kwon et al., Efficient Memory Management for LLM Serving with PagedAttention, SOSP 2023 (full text read) [primary]**
https://arxiv.org/abs/2309.06180

3. **vLLM: Easy, Fast, and Cheap LLM Serving with PagedAttention, 2023-06-20 [primary]**
https://blog.vllm.ai/2023/06/20/vllm.html

4. **vLLM Large Scale Serving: DeepSeek @ 2.2k tok/s/H200 with Wide-EP, 2025-12-17 [primary]**
https://blog.vllm.ai/2025/12/17/large-scale-serving.html

5. **llm-d 1.0 press release, 2026-10-20 [primary]**
https://newsroom.ibm.com/2026-10-20-llm-d-1-0-advances-open,-cost-efficient-ai-inference-for-production-scale

6. **Mistral model card discussion recommending vLLM [primary]**
https://huggingface.co/mistralai/Mistral-Small-4-119B-2603-NVFP4/discussions/7

7. **Ubicloud, 2026-10-20**
https://www.ubicloud.com/blog/fixing-tool-call-accuracy-in-vllm

8. **Kolluru, Comparative Analysis of LLM Inference Serving Systems: vLLM and HuggingFace TGI, 2025-11-17**
https://arxiv.org/abs/2511.17593

9. **Sisodia, From Inference Engine to Inference Control Plane, 2026-09-19**
https://arxiv.org/abs/2609.23130

10. **ChatForest review (TGI maintenance-mode claim, Rufus/LinkedIn/Roblox usage)**
https://chatforest.com/reviews/vllm-production-llm-serving/

11. **byteiota on v0.25.0, 2026-07-11**
https://byteiota.com/vllm-v0-25-model-runner-v2-default-pagedattention-gone/

12. **HN discussion of Ubicloud post, 2026-10-26**
https://news.ycombinator.com/item?id=50242870