# Episode Book

EPISODE #5964

# Containers vs VMs for GPU Inference: What You Actually Pay

| | |
|---|---|
| PUBLISHED | 14 October 2026 at 14:48 UTC |
| DURATION | 30:37 |
| LISTEN | myweirdprompts.com/episode/gpu-virtualization-overhead-inference-41095/ |
| HOSTS | Corn, Herman, Hilbert |
| SCRIPT MODEL | deepseek-flash |
| VOICES | chatterbox-regular (chatterbox-local) |
| PIPELINE | V5.4 |
| SOURCES | 13 cited by the research (listed at the end) |
| BOOK GENERATED | 19 October 2026 at 17:42 UTC |

## SYNOPSIS

Daniel asked a simple question: what do you actually pay when you move an open-weight model off bare metal and into a container or a KVM virtual machine with PCIe passthrough? The published numbers don't agree — one paper measures containers at 0.03% overhead, another measures VMs at 21% average and up to 36% on the worst run. Both are correct. This episode unpacks why the benchmark literature quietly conflates four different isolation models under the single word "virtualization," where the overhead actually lives (spoiler: it's the IOMMU and the host scheduler, not the GPU), and which workload shapes feel it most. We also walk through a real deployment that booted perfectly, reported all eight GPUs healthy, and had silently lost GPU-to-GPU peer-to-peer — the kind of failure that doesn't show up until you're already in production.

# TRANSCRIPT



**Corn**

A container adds three hundredths of one percent. Not three percent. Zero point zero three.



**Herman**

Which sounds like a typo until you understand what a container actually is.



**Corn**

Daniel's question this week is about what you pay when you move the same open-weight model off bare metal and into something else. He wants the overhead numbers, not the configuration guide. Container, KVM virtual machine with PCIe passthrough, bare metal as the baseline, and then the harder part, which is where that overhead actually lives and whether NVIDIA and AMD differ in any way that matters.



**Herman**

And the reason it's worth a whole episode is that the published numbers look like they're describing different universes. You've got papers saying passthrough costs you nothing, and papers saying it costs you a third of your throughput.



**Corn**

Both are true. That's the fun part.



**Herman**

The real question isn't whether virtualization is slow. It's which isolation model, on which hardware, for which workload shape. Because the benchmark literature is quietly conflating at least four different things under the single word virtualization. There's the container, which shares the host kernel and the host driver. There's single-GPU passthrough, where the hypervisor hands one card to one guest and gets out of the way. There's multi-GPU passthrough, where the guest owns several cards and the interconnect between them suddenly matters. And then there's shared-GPU virtualization, the MIG and SR-IOV and vGPU family, which is a completely different product with a completely different overhead profile.



**Corn**

Four different things wearing one word. That's why the numbers don't reconcile.



**Herman**

Right. And before we get into any of it, the baseline has to be pinned down, because bare metal isn't a single number either. Kernel version, driver version, whether the CPU governor is set to performance, whether the process is NUMA-aligned to the right socket. A tuned bare-metal baseline and a sloppy one differ by more than the entire container overhead we're about to discuss.



**Corn**

So let's start with the number everyone repeats. Containers are free. Find out whether that's actually true.



**Herman**

It's true, and the mechanism is boring in the best way. Docker and Podman share the host kernel. There's no hypervisor, no second kernel, no device emulation. When the inference engine submits a CUDA kernel inside a container, it goes through the same driver stack, the same ioctl path, the same GPU, and the kernel executes identically. The container boundary is namespaces and cgroups. It's a boundary around filesystems, process trees, and resource quotas. It's not a boundary around the hardware.



**Corn**

So the GPU never learns it's in a container.



**Herman**

The GPU has no idea. It's the same instructions on the same silicon.



**Corn**

What's the actual measurement?



**Herman**

The cleanest one is from a group at Gdańsk University of Technology. They ran eight GPUs across three workloads, MLPerf SSD detection, ResNet50 training, and BERT inference, comparing Docker against bare metal. Average execution time overhead: zero point zero three percent. Maximum across every run: zero point three one percent. The paper is literally titled Containers Are Almost Free.



**Corn**

And the inference-specific number?



**Herman**

GigaGPU ran vLLM on an RTX 6000 Pro, ninety-six gigs, back in April. Bare metal: four thousand two hundred fifty tokens per second. Same model, same card, in Docker: four thousand two hundred ten. That's under one percent. You could lose more than that to a warm afternoon.



**Corn**

So the container is free. What's the catch?



**Herman**

The one real cost is model loading, and it's a cold-start cost, not a steady-state one. Loading a seventy-billion-parameter model from a bind mount adds two to five seconds of startup versus reading from the host filesystem directly. Put the weights on tmpfs or a local NVMe drive and it disappears. For a long-running inference server, you pay it once at boot and never again. For an autoscaling deployment that spins up workers on demand, that's five seconds on every scale event, and that's a real number when you're scaling under load.



**Corn**

Five seconds per worker, and you could be spinning up forty.



**Herman**

Then it matters. But notice what it isn't. It's a filesystem cost. It was never a GPU cost.



**Corn**

Alright. Single-GPU passthrough next.



**Herman**

Near-native, but noisier than the container numbers. Pextra ran a Tesla V100S through KVM with VFIO last July and measured llama.cpp prompt processing down one point three percent, token generation down one point four, gpu_burn down zero point eight, and the CUDA bandwidth tests inside that range. So roughly half a percent to one and a half percent across the board. packet.ai puts single-GPU passthrough at zero to two percent on modern KVM. MLPerf Inference showed an A100 within one to three percent between bare metal and a passthrough VM.



**Corn**

And the mechanism behind near-native passthrough?



**Herman**

It's worth understanding, because everything later in the episode hinges on it. With VFIO, QEMU only assigns device ownership at boot time. It maps the GPU's BARs into the guest's address space, hands over the interrupt routing, and then it's done. After that, the guest driver talks directly to the hardware registers. Pextra's CTO put it flatly: once the GPU is assigned directly to the guest, there is very little virtualization remaining in the execution path.



**Corn**

What's left?



**Herman**

Three small things. Interrupt delivery, because a physical interrupt has to be routed into the guest. PCIe address translation, because the IOMMU has to translate DMA addresses. And VM scheduling, because the guest's vCPUs are still threads on the host. For a single device on a well-resourced host, all three are in the noise.



**Corn**

So far the story is containers free, single-GPU passthrough nearly free. Now the contradiction.



**Herman**

This is the part that makes the episode worth doing. That same Gdańsk paper that measured Docker at zero point zero three percent measured KVM in the same eight-GPU environment at twenty-one point two two percent average execution time overhead. Up to thirty-six point four nine percent on the worst run.



**Corn**

Same hardware. Same workloads.



**Herman**

Same paper. BERT inference averaged twenty-eight point nine percent VM overhead, maxing at thirty-six point four seven. ResNet50 training: thirty point nine one percent. SSD detection was the outlier at three point eight four. And their recommendation is unambiguous. I'll quote it, because it's the line everyone cites: we strongly recommend using container-based configurations deployed directly on bare metal, without an additional VM layer, wherever such an approach is feasible.



**Corn**

So one paper says passthrough is a rounding error and the other says it's a third of your throughput.



**Herman**

And you don't resolve that by picking a side. You resolve it by looking at the two setups, because they're not measuring the same thing.



**Corn**

Go on.



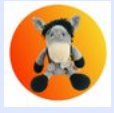
**Herman**

The ICCS configuration ran eight virtual machines in parallel on the same host, each one constrained to eight CPU cores and thirty-two gigabytes of RAM, each with a GPU passed through. So you've got eight guests, sixty-four vCPUs' worth of contention on the host scheduler, and each guest is memory-capped. The Pextra and packet.ai numbers come from a single passthrough VM on a host with cores to spare. Both measurements are correct. They describe different systems.



**Corn**

Tensor-parallel and CPU-starved versus one card and room to breathe.



**Herman**

And notice the SSD result in the ICCS data. Three point eight four percent. That workload is the least CPU-bound of the three, and it's the one where the VM barely costs anything. That's a clue about where the overhead actually lives.



**Corn**

Which is not the GPU.



**Herman**

It's not the GPU at all. The GPU is executing the same instructions at the same rate. What changes is everything around it. The cognitive work here is separating GPU compute from the CPU-side scaffolding that feeds it.



**Corn**

Split the scaffolding.



**Herman**

Start with the IOMMU. Every DMA access from the guest has to be translated through the IOMMU's page tables, because the device is writing to guest-physical addresses that don't map one-to-one to host-physical. For large contiguous allocations, that costs half a percent to one and a half percent. That's the honest answer for single-GPU passthrough overhead. It's the IOMMU tax.



**Corn**

Interrupts?



**Herman**

Negligible, and for a reason that's specific to GPUs. GPUs mostly poll. A CUDA kernel completion is checked by the driver polling a memory location, not by waiting for a hardware interrupt. So the interrupt-routing cost that dominates network throughput in virtualized environments just doesn't show up here.



**Corn**

Memory pinning?



**Herman**

One-time startup cost. The guest has to pin the DMA buffers so the hypervisor knows not to move those pages, and after the first allocation, it's done.



**Corn**

So on a clean single-GPU setup, the accounting is half a percent to two percent, and it's mostly IOMMU.



**Herman**

And the ICCS numbers, the twenty-one to thirty-one percent, are the same accounting on a configuration where the CPU-side costs stop being small. Eight VMs contending for the host scheduler. Each with pinned memory competing for the same physical pages. Each generating IO that has to go through the hypervisor's IO path. There's a line from Felter's work years ago that KVM IO overhead reaches forty percent, and the ICCS authors specifically call out IO operations as a significant contributor. The overhead was always there. It just didn't bite until the host was crowded.



**Corn**

Which means the twenty-one percent isn't virtualization overhead. It's contention.



**Herman**

It's contention that virtualization makes possible. Bare metal on the same eight-GPU box can't overcommit itself into that state, because there's no host scheduler to compete with.



**Corn**

Here's the question underneath all of this. Which workloads actually feel it?



**Herman**

That's the piece that turns the numbers into something you can act on. Compute-bound, large-batch workloads are the least sensitive. dstack found something counterintuitive on the MI300X: at batch sizes of eight and above, the virtual machine was slightly faster than bare metal, not slower. The explanation is that at large batch sizes, the CPU doesn't have to synchronize with the GPU as often. The GPU is the bottleneck, and the CPU-side virtualization cost gets hidden behind it. At small batch sizes, the CPU is on the critical path more often, so its overhead shows.



**Corn**

The bigger the batch, the more the GPU hides.



**Herman**

And the flip side is the workloads that feel it most. Memory-bandwidth-bound work, like attention, like embedding lookups. And small-message-latency-bound work, like the all-reduce inside a tensor-parallel model. Those are the two categories where the interconnect and the scheduler are on the critical path, and that's exactly where the benchmark tables stop flattering passthrough.



**Corn**

So the answer to whether a VM is fine depends on one thing you can check before you deploy.



**Herman**

What batch sizes and what tensor-parallelism.



**Corn**

And then there's the cliff.



**Herman**

There's the cliff. This one's worth walking through, because the setup looked completely healthy. A case study from magik.net, published this year. Seven hundred forty-four billion parameter mixture-of-experts model, GLM, quantized to NVFP4, running on eight RTX PRO 6000 Blackwell cards inside a Proxmox virtual machine.



**Corn**

Boots fine, cards visible.



**Herman**

Boots fine, all eight GPUs visible in nvidia-smi, driver loaded, everything reports healthy. And then somebody ran the topology query. Every GPU pair showed NS. Not Supported. No direct GPU-to-GPU DMA path.



**Corn**

The peer-to-peer got eaten.



**Herman**

With peer-to-peer disabled, every cross-GPU transfer has to bounce through host memory. The sender writes to host memory, the receiver reads from host memory. And the bandwidth collapses. They measured cross-GPU transfer at thirty-two point six gigabytes per second. Same-GPU transfer on that card is one thousand thirty-three gigabytes per second. That's a ninety-seven percent reduction. The GPU-to-GPU path didn't get slower. It stopped being a GPU-to-GPU path.



**Corn**

What does that do to the model?



**Herman**

The model had seventy-eight layers and an all-reduce per layer per decode step. NCCL all-reduce at twelve kilobytes, which is the per-layer size, hit zero point nine gigabytes per second with thirteen point seven microseconds of latency. At that latency, the interconnect is the bottleneck on every single token. Throughput collapsed to eleven tokens per second on single-stream decoding.



**Corn**

Eleven.



**Herman**

Eleven. On eight Blackwell cards.



**Corn**

And the compute was never the problem.



**Herman**

The compute was fine. Every GPU could do its share of the math. The GPUs simply couldn't talk to each other.



**Corn**

That's a configuration error, not a passthrough cost.



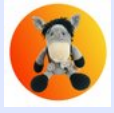
**Herman**

It's a configuration error that passthrough made possible, and that's why it's in the episode. The virtual machine's IOMMU topology didn't preserve the peer-to-peer path between the devices. Nothing announced that. The VM booted, the cards were there, the driver was happy. The failure was silent until somebody measured it.



**Corn**

So the practical diagnostic is one command.



**Herman**

nvidia-smi topo -p2p r, run inside the guest, before you trust the deployment. If every pair shows NS, your tensor parallelism is going to run over host memory. If it shows OK or PIX or PHB, you have a real interconnect.



**Corn**

And this is invisible in every benchmark table you've quoted so far.



**Herman**

Completely invisible. The multi-GPU passthrough row in those tables assumes peer-to-peer is preserved. When it isn't, you don't get a percentage. You get an order of magnitude.



**Corn**

What does the paper say about multi-node?



**Herman**

The multi-node synthesis puts SR-IOV at eighteen to thirty-five percent added all-reduce latency versus bare-metal NVLink, and vGPU at two hundred to four hundred percent in congested multi-node setups. The pattern is the same at every scale. Compute virtualizes well. Interconnect virtualizes badly.



**Corn**

Here's a small one that undercuts the whole framing.



**Herman**

The ProxmoxPulse test in April. RTX 3090, Llama 3.2 seven billion at Q4, running three ways. KVM VM with passthrough: eighty-five tokens per second. LXC container: eighty-eight. Bare metal: ninety.



**Corn**

Six percent between the VM and bare metal.



**Herman**

On a single card, on a consumer GPU, with a small model. Which is the row most people reading this would actually be in. And it's a number that would fit inside the ICCS paper's error bars for a single VM.



**Corn**

So containers are free, single-GPU passthrough is nearly free, multi-GPU passthrough can fall off a cliff. The next question is which isolation model you're actually buying, and whether NVIDIA and AMD sell you the same thing.



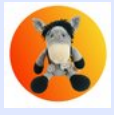
**Herman**

This is where the industry quietly sells you four different products under two vendor names. The first fork is full-device passthrough versus shared-GPU virtualization. Those are not variations on a theme. They're different machines with different economics.



**Corn**

Start with passthrough. What's the overhead number?



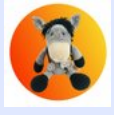
**Herman**

ClusterBid's comparison from February puts PCIe passthrough at zero percent. The card is handed to the guest and the guest owns it entirely. The cost is that you can't live-migrate the VM, because the guest state includes the GPU's state, and the GPU's state isn't portable across hosts.



**Corn**

The thing you give up for zero overhead is mobility.



**Herman**

And that's the trade at the heart of the shared-GPU family. Every one of those technologies exists to make the GPU divisible or migratable, and every one of them pays for it.



**Corn**

Walk through the family.



**Herman**

MIG is hardware partitioning. It carves the GPU into physically isolated slices, each with dedicated memory and dedicated compute units. The isolation isn't a software abstraction, it's the silicon. The overhead is zero point two to zero point four percent on compute, and it doesn't support live migration, because the slice is bound to the physical card.



**Corn**

And AMD's SR-IOV?



**Herman**

One and a half to three point eight percent, and the reason it's higher than MIG is per-access IOMMU translation. Every virtual function's accesses go through address translation, because SR-IOV partitions the device into virtual functions rather than into physical slices. Also no live migration.



**Corn**

And NVIDIA's vGPU?



**Herman**

Eight to twelve percent on compute, fifteen to twenty-five percent on memory bandwidth, and it's the only one that supports live migration. It's also the only one you pay a license for. Two thousand five hundred to three thousand five hundred dollars per GPU per year.



**Corn**

You pay the overhead twice. Once in throughput, once in license.



**Herman**

That's a fair way to put it.



**Corn**

But if live migration is a hard requirement, that's the only row that supports it.



**Herman**

Which is the honest justification. vGPU is a capacity-management tool, not a performance tool. If you need to move a running GPU workload between hosts without stopping it, you pay double-digit overhead and a license, and there's nothing else that does the job.



**Corn**

What does MIG actually cost you in throughput?



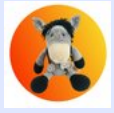
**Herman**

The capacity number is the one that sticks. Eight H100s carved into fifty-six slices running Llama eight billion: twenty-two tokens per second per slice, one thousand two hundred thirty-two tokens per second aggregate across the whole machine. The same eight cards running full-size models: four thousand two hundred tokens per second. You gave up about seventy percent of your aggregate throughput to get fifty-six isolated tenants.



**Corn**

Fifty-six tenants is a lot of multi-tenancy.



**Herman**

It is. And if you're a cloud provider renting slices, that math works. If you're one team running one model, you've just carved your own GPUs into pieces you can't put back together without a maintenance window.



**Corn**

Where does the scheduling cost show up on vGPU?



**Herman**

In the tail. vGPU is time-sliced, so the GPU switches between tenants on a schedule. On compute-bound work, that costs four to twelve percent. On memory-bandwidth-bound work under contention, it's up to twenty-five percent. And the p99 latency in a shared vGPU deployment runs fifteen to forty percent higher than on a dedicated card at high query rates, because the scheduler introduces jitter. Average latency looks fine. The tail doesn't.



**Corn**

Which is the number that matters if you've got an interactive workload.



**Herman**

It's the number that matters if users notice slowness, yes. If your users perceive p99, that's the row to look at.



**Corn**

Now the vendor question. Does AMD passthrough differently from NVIDIA?



**Herman**

The short answer is that nobody has published the clean head-to-head. dstack's MI300X work and Pextra's NVIDIA work use different GPUs, different models, different methods. You cannot subtract them and get a vendor delta. That benchmark doesn't exist in the public literature.



**Corn**

So what do we actually know?



**Herman**

We know the passthrough numbers on each side are in the same band. dstack ran MI300X through a KVM VM against bare metal and found them nearly identical at moderate concurrency, sixteen to sixty-four concurrent requests. Bare metal was slightly ahead at low concurrency, four to sixteen, which they attribute to faster kernel launches and direct device access, and slightly ahead at high concurrency, sixty-four to one twenty-eight. The gap was narrow in every direction.



**Corn**

And the AMD side is comparable to Pextra's four tenths of a percent.



**Herman**

Comparable. Neither side is showing double-digit overhead on single-GPU passthrough.



**Corn**

So the vendor difference is somewhere else.



**Herman**

It's in the software stack, and it's in two specific places. The first is the virtualization stack itself. NVIDIA's is vGPU, which is closed and licensed, plus MIG. AMD's is MxGPU, which is an open-source SR-IOV host driver. Up to sixteen virtual functions per GPU on the MI350 and MI400, and it supports dynamic creation of virtual functions without draining the running workloads on the card.



**Corn**

You can add a virtual function to a live card.



**Herman**

Without stopping anything. MIG requires you to reconfigure the partitioning, which is a disruptive operation, so you schedule it. AMD's SR-IOV lets you carve a new VF off a running card on demand. Operationally, that's a real advantage for anything that needs to



**Herman (continued)**

scale tenants up and down.



**Corn**

So why isn't everyone on AMD then?



**Herman**

Because the second thing is the ecosystem, and this is where the silicon advantage stops mattering. AMD recommends setting HIP_VISIBLE_DEVICES to zero per virtual function, because MIOpen and rocBLAS don't reliably detect virtual-function boundaries. The libraries don't know they're running on a virtual function and will try to use capabilities they don't have.



**Corn**

The math libraries don't see the partition.



**Herman**

They see a GPU. They don't see which fraction of a GPU. So the user has to tell them.



**Corn**

That's an environment variable away from working.



**Herman**

It's one environment variable, and it's one environment variable that has to be set correctly in every container, in every image, in every Helm chart, forever, and can silently break when somebody forgets. Compare that to the CUDA stack, where virtual-function awareness is baked in because vGPU is a supported product line.



**Corn**

So the comparison isn't silicon. It's how many years of tooling have gone into making the virtualization invisible.



**Herman**

It's the maturity of the abstraction. Red Hat published a guide this month on scheduling MI300X SR-IOV virtual functions directly into OpenStack tenant VMs through VFIO. That's the plumbing getting built out in real time.



**Corn**

And the operational flexibility of AMD's approach, if the ecosystem catches up, matters more than one and a half percent of IOMMU overhead.



**Herman**

It might. The overhead is in the noise compared to the ability to add capacity to a running card without a maintenance window.



**Corn**

So what's the architecture answer for someone building this today?



**Herman**

It splits by workload shape. Containers for anything that doesn't need a hard boundary. Single-GPU passthrough when you need a real isolation boundary, either security or Windows-only tooling, and you're running one card per VM. Multi-GPU passthrough only after you've verified the peer-to-peer topology inside the guest. And shared-GPU virtualization when you're a provider, when you need tenants, or when live migration is a hard requirement. Those are four different jobs.



**Corn**

And the counterintuitive result worth putting in the record?



**Herman**

n0derunner's experiment this year. vLLM in a container inside a VM, against the same container on bare metal. The VM version matched, and in some runs slightly beat, bare metal.



**Corn**

A container inside a VM beating bare metal.



**Herman**

Which sounds wrong until you remember dstack's batch-size finding. At moderate-to-high batch sizes, the VM's scheduling overhead doesn't bite, and the VM's CPU pinning and NUMA alignment were better than the host's defaults. The host was leaving performance on the table. The VM was configured more carefully.



**Corn**

Your hypervisor did a better job than you did.



**Herman**

The hypervisor did the tuning. On a single-GPU, well-configured VM, you're inside a couple of percent of bare metal, and the exact ordering depends on how carefully the bare-metal host was tuned. If your baseline is a sloppy bare metal, a careful VM can beat it.



**Corn**

Which means the plain answer to whether you should run bare metal is: check whether your bare metal is actually tuned first.



**Herman**

That's the question most people skip. The overhead of a VM is smaller than the overhead of a host that hasn't had its CPU governor set.



**Corn**

So the deployment layer is a small number next to the things you're probably not measuring.



**Herman**

Which is a strange place to end up after twenty minutes of passthrough numbers, but it's where the evidence points.



**Hilbert**

The interesting number isn't the overhead. It's the moment the interconnect downgrades and nothing tells you.



**Hilbert**

Everything on paper looks correct. VM boots, GPUs visible, driver loaded, all eight cards in nvidia-smi. You'd never know the peer-to-peer path got dropped unless you ran the topology query. That's the failure mode nobody writes about, because it doesn't show up in a percentage. It shows up as your model running slower than it has any right to.



**Corn**

Virtualization doesn't fail loudly?



**Hilbert**

It doesn't fail. It routes traffic the long way. A card can be perfectly healthy and completely useless for tensor parallelism at the same time.



**Herman**

You're describing the magik.net case exactly.



**Hilbert**

I'm describing every multi-GPU deployment I've ever seen go wrong. I once spent a week on a throughput collapse where the only thing that gave it away was the fans. Second card in the machine was spinning slower than the first, because it was idle while the first card did all the work and bounced every intermediate result through host memory.



**Corn**

You diagnosed it from the fans.



**Hilbert**

I diagnosed it from the fans. Both cards reported healthy. The second one had nothing to do.



**Herman**

Was that a real diagnostic, or is that a Hilbert original?



**Hilbert**

It's a real one. The card doing all the work runs hot. The card waiting runs cool. If you know what the fans should sound like under load, you know.



**Corn**

What was the machine?



**Hilbert**

Belonged to a client who owned three identical servers and insisted they were one server. Called them by his children's names. Individually, when he wanted to talk to one of them. Which he would not do. He referred to the three together as one machine, and he would get visibly upset if I named one of them.



**Herman**

He named them after his kids.



**Hilbert**

Liam, Sarah, and Michael. Liam was the primary. There was no primary. He just called it that, and if I asked which one Liam was, he'd tell me it was all of them.



**Corn**

All three of his children were one server.



**Hilbert**

He'd get upset if I said the server was down. It wasn't down. Liam was down. Or what he called Liam.



**Corn**

The ten-gigabit card. That's what you call the interconnect?



**Hilbert**

The peer path. He had it disabled in the BIOS on Sarah. Took me a week to find out, because he wouldn't let me touch Sarah.



**Corn**

Of course not.



**Hilbert**

I quit over the grammar, in the end. I could deal with the servers. The grammar was too much.



**Herman**

The interconnect silently downgrading is the open question in the literature right now. You've got the twenty-one to thirty-one percent KVM figure from one paper and the zero to two percent single-GPU passthrough consensus from everyone else, and nobody has reconciled them with a controlled head-to-head on the same hardware.



**Corn**

Which is a real gap, not a settled question.



**Herman**

The second gap is the vendor question. There is no clean NVIDIA-versus-AMD passthrough benchmark on identical inference workloads. Every comparison you can find is really an ecosystem comparison wearing performance clothing.



**Corn**

The direction of travel is AMD's virtualization stack getting the plumbing it needs. Dynamic virtual functions without draining workloads is a real operational advantage over MIG's static partitioning, and the one and a half to three point eight percent overhead matters less than being able to add a tenant to a running card.



**Herman**

If OpenStack and Kubernetes scheduling of those virtual functions becomes standard, the flexibility argument gets stronger than the overhead argument.



**Corn**

The overhead is a percentage. The flexibility is a maintenance window you don't have to take.



**Herman**

That's where the decision gets made.



**Corn**

The straight answer to Daniel's question is: containers are free, single-GPU passthrough is nearly free, multi-GPU passthrough requires verification, and shared-GPU virtualization is a capacity tool, not a performance one. The percentage is almost never the interesting part. Where the traffic goes is.



**Herman**

If you run one of these configurations, run the topology query inside the guest on day one. On day one.



**Corn**

That's the episode. Thanks to Hilbert Flumingtop for producing, and for the fan-based diagnostic. If this was your kind of episode, go back for episode thirty-seven eighty-nine, What Virtualization Actually Costs on two thousand twenty-six Hardware; episode two, Local STT For AMD GPU Owners; and episode three, Safetensors or something else. This has been My Weird Prompts.



**Herman**

The human-AI collaboration podcast. If you want to send us your own prompt, the Telegram bot is t dot me slash MWP listener bot. Anything you're wondering about, technical or otherwise.



**Corn**

We read everything.



**Herman**

We'll be back soon.



**Corn**

See you then.

# SOURCES

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


1. **ICCS 2026 [primary]**
   Kościowski, Proficz, Czarnul, Containers Are (Almost) Free
   https://www.iccs-meeting.org/archive/iccs2026/papers/167840464.pdf

2. **AMD Instinct docs [primary]**
   MxGPU/SR-IOV getting started
   https://instinct.docs.amd.com/projects/virt-drv/en/latest/userguides/Getting%5Fstarted%5Fwith%5FMxGPU.html

3. **AMD MxGPU-Virtualization GIM 9.0.0.K release notes [primary]**
   https://github.com/amd/MxGPU-Virtualization/releases/tag/9.0.0.K

4. **dstack**
   Benchmarking AMD GPUs: bare-metal, VMs (2025-07-22)
   https://dstack.ai/blog/benchmark-amd-vms/

5. **Pextra**
   How Much Performance Do You Lose with QEMU/KVM GPU Passthrough? (2026-07-15)
   https://pextra.cloud/articles/gpu-passthrough-performance/

6. **packet.ai**
   Bare Metal vs VM GPU Performance 2026 (2026-07-19)
   https://packet.ai/blog/bare-metal-vs-vm-gpu-performance-benchmarks

7. **n0derunner**
   Containers in VM Vs Containers on Bare metal (2026-07-20)
   https://www.n0derunner.com/containers-in-vm-vs-containers-on-bare-metal-for-ai-workloads/

8. **GigaGPU**
   Docker vs Bare Metal for AI Inference (2026-04-16)
   https://gigagpu.com/docker-vs-bare-metal-ai-inference/

9. **ClusterBid**
GPU Virtualization Software: vGPU, MIG, and SR-IOV Performance Overhead (2026-02)
https://clusterbid.com/blog/gpu-virtualization-software-vgpu-mig-sriov-performance-overhead

10. **ProxmoxPulse**
GPU VM vs LXC Container Guide (2026-04-13)
https://proxmoxpulse.com/articles/proxmox-ai-inference-gpu-vm-vs-lxc-container/

11. **magik.net**
The Virtualization Revelation: How a Proxmox VM Became the Hidden Bottleneck
https://magik.net/blog/llmopt/seg_2/chunk_1/virtualization_bottleneck_revelation.html

12. **Red Hat Developer**
GPU virtualization at scale: AMD MI300X SR-IOV (2026-10-01)
https://developers.redhat.com/articles/2026/10/01/gpu-virtualization-amd-mi300x-sr-iov-red-hat-openstack-services-on-openshift

13. **VMware**
vGPU vs MIG perf
https://www.vmware.com/docs/vgpu-vs-mig-perf