Episode Book

EPISODE #5970

Swapping Your Linux Kernel: What Actually Changes

PUBLISHED	14 October 2026 at 15:15 UTC
DURATION	26:00
LISTEN	myweirdprompts.com/episode/linux-kernel-swap-performance-62716/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	15 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:42 UTC

SYNOPSIS

On Ubuntu you're not locked to the kernel your distribution shipped — you can pull a newer upstream one, stay on an older one, or install a kernel someone rebuilt for a specific kind of work. This episode maps the whole landscape: how Ubuntu decides what to ship (including release candidates under its 2024 policy), what actually changes between kernel versions, why hardware support and performance are two very different reasons to switch, and what Liquorix, XanMod, and the now-mainline PREEMPT_RT kernel each optimize for. The most interesting moment comes when XanMod's own documentation labels the AVX-512 build as having no kernel benefit — the vendor of the CPU-optimized kernel undermining the enthusiast argument. Along the way: EEVDF replacing the completely fair scheduler, the throughput-versus-latency axis every tuning choice sits on, and why desktop and server motivations split apart.

TRANSCRIPT



Corn

Most people never think about their kernel. It arrives, it runs, it updates itself in the background, and that's the end of the relationship.



Herman

And yet it's the one piece of the operating system you can swap out entirely without reinstalling anything else. Same Ubuntu, same files, different kernel underneath.



Corn

Which is what Daniel wants to dig into. He wrote in with a long one, so let me compress it. His starting point is that on something like an Ubuntu LTS, you're not actually locked to the kernel the distribution gave you. You can pull a newer upstream one, keep an older one, or go get a kernel somebody has modified for a particular kind of work. Most people run what they're given, and reasonably so, but there's a crowd that deliberately doesn't.



Herman

And he wants the whole map.



Corn

The whole map. What actually changes between kernel versions, how a distribution like Ubuntu decides what to ship, whether the alternative kernels are delivering real gains or just better hardware support, what Liquorix and XanMod are optimizing for, whether compiling for your specific CPU buys you anything, where desktop and server motivations split apart, and what the benchmarks actually say when you line them up.



Herman

That last one is the one I want to get to.



Corn

The question underneath all of it is whether the default kernel is already close to optimal, or whether there's a real opportunity sitting there that most people ignore.



Herman

Here's the thing that makes the question harder than it looks. The kernel isn't one thing. It's the scheduler, the memory manager, the I/O layer, the driver host, and the power manager all in one binary. Change the kernel and you change all five at once, in ways that interact.



Corn

So you can't tune one without moving the others.



Herman

And the landscape has layers to it. You've got distribution kernels, which is what almost everybody runs. You've got upstream mainline, straight from kernel.org. You've got the distribution's own variants, the low-latency one, the hardware enablement one. And then you've got the enthusiast kernels, Liquorix, XanMod, and the real-time kernel that used to be a patchset and now isn't.



Corn

Give me the shape of the episode.



Herman

First what changes between versions and how Ubuntu decides what to ship. Then why anyone would go elsewhere, what those specialized kernels optimize for, whether CPU-specific tuning pays off, how desktop and server differ, and what the benchmark numbers actually show. The numbers are where the marketing takes a hit.



Corn

Start at the top. What actually changes when the version number moves?



Herman

Upstream Linux doesn't have a release date. It has a release rhythm. Every two to three months, when it's ready. No fixed calendar. Ubuntu is the opposite, a hard six-month cadence no matter what state upstream is in.



Corn

Those two schedules don't line up.



Herman

That's the whole problem. Canonical's rule of thumb is that they need about a month of stabilization between an upstream release and a stable Ubuntu kernel. So when Ubuntu freezes for a release, whatever upstream kernel exists at that moment gets shipped, even if it's still a release candidate.



Corn

They ship release candidates to users.



Herman

Under the policy they adopted in 2024, yes. And they split it into categories. If the candidate is at release candidate four through six, that's a Tight release. If it's at one through three, that's Unstable. In the Unstable case they ship what they call a Bridge Kernel, a kernel from the previous release, to carry people until the new one settles down.



Corn

Naming a kernel "Unstable" and shipping it anyway is a choice.



Herman

It's honest labeling, at least. And this is where the version story gets interesting, because the GA kernel for a release is the latest upstream kernel at the time of that release, and then it stays on that version for the entire life of the release. Four years, in an LTS case, on one kernel version.



Corn

With security fixes going in.



Herman

That's the hardware enablement kernel's whole reason to exist. HWE backports a newer kernel to an older LTS, and it updates every six months until it matches the GA kernel of the next LTS.



Corn

So on an older LTS you get a choice. Stay on the original kernel, or follow the HWE stack and get newer hardware support.



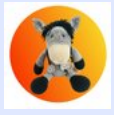
Herman

And Corn, you made this point before, that the real story of LTS is the hardware enablement stack absorbing most of the reason people used to jump to interim releases.



Corn

I stand by it.



Herman

The variant list is worth hearing, because it shows how much curation is happening. Ubuntu ships a low-latency kernel for things like audio work, generic HWE, generic HWE edge, cloud kernels for AWS, Azure and GCP, platform kernels for Raspberry Pi and Snapdragon, and silicon-partner kernels built with the chip vendors.



Corn

And people think of the default as just an older upstream.



Herman

That's the misconception to kill. A distribution kernel isn't upstream frozen in amber. Canonical backports the CVE fixes itself. That's why an Ubuntu stable kernel is named with a four-part scheme ending in ABI, rather than kernel.org's three-part version. The version number itself tells you it's a different product.



Corn

Say that plainly. What do you give up if you go install the newest upstream kernel on your desktop?



Herman

The maintenance pipeline. You get the new scheduler behavior, the new driver support, the new everything. You also get every regression upstream hasn't fixed yet, and nobody backporting security patches to the version you're running. You've opted out of the curated product and into the raw one.



Corn

So the default kernel is the maintained one, and the newest one is the unmaintained one.



Herman

For that version, yes. Which is not an argument against ever doing it. It's an argument for knowing what you're doing, and for the people who do it, most of them are on a distribution where they can roll back.



Corn

What actually changes inside the kernel between versions, the things that would move performance?



Herman

A few big levers. The CPU scheduler. The memory manager, where multi-generational LRU changed how pages get reclaimed. The I/O schedulers. Power management. Driver support. The scheduler is the biggest one for anything interactive, and in kernel 6.6 it changed in a way that matters.



Corn

EEVDF.



Herman

EEVDF replaced the completely fair scheduler. It's a different way of deciding which process runs next, built around virtual deadlines instead of the older fairness model. On most workloads you wouldn't notice. On some you would.



Corn

So a kernel upgrade can change how your machine feels without touching a single other package.



Herman

It can. But it won't make your CPU faster. What it changes is how the CPU's time gets handed out, how memory gets reclaimed, how I/O gets queued. If none of those is your bottleneck, the upgrade does nothing you can measure.



Corn

Which sets up the question Daniel actually cares about. Why would somebody go get a different kernel on purpose?



Herman

Two reasons, and the balance between them is the interesting part. Hardware support is the boring one. New laptop, new WiFi chip, new GPU, the drivers landed upstream after your distribution froze. That's not folklore. That's just a new kernel containing code your



Herman (continued)

old kernel doesn't have.



Corn

And the other reason.



Herman

Performance, responsiveness, latency. Which is where it gets contentious, because those three words are not synonyms and the kernels that sell on them are trading one for the others.



Corn

Define what's actually tunable.



Herman

CPU scheduling, obviously. Memory management. I/O scheduling, where you've got choices like Kyber for multi-queue devices and BFQ for single-queue. Power management, the CPU frequency governors. And system latency, which is the preemption model, how aggressively the kernel will interrupt what it's doing to let something else run.



Corn

Every one of those is a trade.



Herman

Every one. Give more, smaller time slices to threads and your latency improves while your throughput drops. That's the axis. On a desktop you might want the latency. On a database server you almost always want the throughput.



Corn

And that's the tension the specialized kernels are built around.



Herman

Take Liquorix. Their own description is "designed for uncompromised responsiveness in interactive systems," and they name their audiences. Low-latency audio and video production, and reduced frame-time deviation in games. Not average frame rate. Frame-time deviation. The stutter, not the speed.



Corn

What do they change to get there?



Herman

Zen interactive tuning, which by their own account trades throughput and power for responsiveness. A different CPU scheduler, either PDS or BMQ depending on the build. A thousand-hertz tick, which is the timer interrupting the system a thousand times a second instead of two hundred and fifty. Timeslices shortened from four milliseconds to two. Hard kernel preemption. Preemptible RCU. Kyber or BFQ for I/O. TCP BBR3 for networking. Multi-generational LRU on by default.



Corn

So it's a stack of deliberate choices, each one buying responsiveness with something else.



Herman

And XanMod is a different philosophy wearing similar clothes. They call it general-purpose. They're built with LLVM software pipelining and polyhedral compilation, and they tune core scheduling, load balancing, caching, the virtual memory manager and CPU frequency for heavy workloads rather than for interactivity.



Corn

Heavy workloads meaning what.



Herman

Sustained compute. They default multi-generational LRU, they support sched_ext, they ship BBRv3, they carry Valve's Steam Deck drivers, and they take part of the Clear Linux patchset.



Corn

And they ship builds for different CPU generations.



Herman

x86-64 version one, two and three. Which is exactly where Daniel's question about CPU-specific compilation lands, and XanMod's own documentation is the answer.



Corn

Say it.



Herman

They label x86-64 version four, the AVX-512 build, as having no kernel benefit. Their words. The people shipping the CPU-optimized kernel are telling you the most aggressive CPU optimization doesn't help the kernel.



Corn

That's the whole enthusiast argument, undermined by the vendor.



Herman

A kernel is mostly memory management, scheduling and bookkeeping. Those aren't vector math. AVX-512 is for multiplying big matrices. The kernel doesn't multiply big matrices. There's no loop for the compiler to vectorize because the hot paths aren't that kind of code.



Corn

So how much of the enthusiast kernel story is the placebo you've been warning about for years?



Herman

Less than you'd think, more than the marketing admits. The preemption changes are real, they're just not free. Which takes us to the third one, the real-time kernel, because that's the cleanest example of a kernel optimizing for something that isn't speed at all.



Corn

Real-time doesn't mean fast.



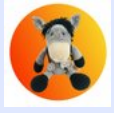
Herman

Real-time means bounded worst case. It means you can guarantee the system responds within a window, not that it responds quickly on average. The way PREEMPT_RT does that is by replacing spinlocks with preemptible, priority-inheriting mechanisms, so a high-priority task can't get stuck behind a low-priority one holding a lock.



Corn

And that costs you throughput.



Herman

It costs you throughput. Which is fine, because if you're running a robot arm or a synthesizer you don't care about averaging slightly better, you care that the worst case never blows your deadline.



Corn

Where does that patch live now?



Herman

In the mainline kernel. It was merged in 6.12, in September of 2024, after roughly twenty years out of tree. First release with real-time baked in for x86, ARM64 and RISC-V. The out-of-tree patch still exists and they still recommend it, for extra architectures and ongoing improvements. But running real-time on mainstream hardware is now a configuration option, not a patchset you apply.



Corn

Two decades of work, and the payoff is that it stopped being interesting.



Herman

That's the success condition. Something that was a specialized kernel became infrastructure. And there's more of it coming. Linux 7.1 is bringing real-time support for 32-bit ARM with no out-of-tree patches at all.



Corn

Stay on the scheduler, because that's where the actual news is.



Herman

Two things happened. One, EEVDF in 6.6, which is a general-purpose scheduler that works fine for most things and is admittedly not optimal for many. The Linux community says that out loud, which I appreciate.



Corn

And two?



Herman

sched_ext, in 6.12. This is the one that changes the shape of the whole argument. sched_ext lets you write a CPU scheduler as a BPF program and load it into a running kernel. Dynamically. You don't rebuild anything.



Corn

And if your custom scheduler is broken?



Herman

There's a watchdog. If the loaded scheduler misbehaves, the kernel boots it and falls back. The pitch is that it's safe, it cannot crash your system the way a bad in-tree scheduler change would.



Corn

So the scheduler becomes something you load and unload, like a driver.



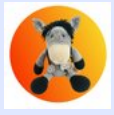
Herman

And Valve is the proof of concept. They worked with Igalia on a sched_ext scheduler for the Steam Deck and reported more consistent frame rates, and they've been moving toward enabling it by default.



Corn

More consistent, not faster.



Herman

More consistent. Which is the right target for a handheld. You'd rather have a stable sixty than a jittery eighty. And it means the Steam Deck doesn't need its own kernel to get that. It needs its own scheduler, loaded on top of a normal one.



Corn

Now the benchmarks, because Daniel asked specifically and this is where the story turns.



Herman

Phoronix is the reference here, and the December 2024 comparison is the one to know. They took Liquorix 6.12.5 and upstream 6.12.5, same version number, and ran them head to head on a Ryzen 9 9950X. A hundred and one benchmarks.



Corn

What was the result?



Herman

Upstream won fifty-six of them. And on a geometric mean across the whole suite, upstream was twenty-one percent faster.



Corn

The responsiveness kernel was twenty-one percent slower.



Herman

On throughput. And that's not a contradiction, it's the entire point. Liquorix is not trying to win those benchmarks. It's trading throughput for latency on purpose, and the benchmark suite measures throughput, so of course it loses.



Corn

Did it win anything?



Herman

It did well on the synthetic kernel micro-benchmarks, which is a bit circular, and on Darktable and some video and image encoding. Mostly it lost the heavy compute tests, which is exactly where throughput matters most.



Corn

So the marketing and the benchmark aren't in conflict. They're measuring different axes and one of them got put on the wrong axis.



Herman

Which is the most useful thing to take from it. "Faster" and "more responsive" are different claims. A kernel can be worse at one and better at the other, and the benchmark only tells you which question it was asked.



Corn

There are older comparisons too.



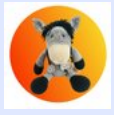
Herman

2021, a Ryzen 5 laptop, a hundred and eighty-eight benchmarks. XanMod, both the stable and edge builds, had a small geometric-mean lead over mainline. Liquorix came in behind the others.



Corn

Small lead, so within noise-ish.



Herman

Small enough that I wouldn't rebuild my system for it. And then there's the 2020 test, which is the funniest one, because it's a case where a default choice actively hurt. Liquorix defaulted to the BFQ I/O scheduler, and Phoronix ran SQLite with eight threads on NVMe.



Corn

And BFQ is built for spinning disks.



Herman

BFQ is built to be fair to interactive processes on a single queue. NVMe is a multi-queue device with enormous parallelism. Putting BFQ in front of it on a database workload was a significant disadvantage. They've since moved the default, but the lesson holds. The optimization is a bet, and sometimes the bet is wrong for your machine.



Corn

Which is why the rule should be that you measure before you switch.



Herman

Measure before you switch. And the measurement is the hard part, because Daniel asked whether the gains show up in real applications or only in synthetic benchmarks, and the honest answer is: sometimes one, sometimes the other, and it depends entirely on whether your actual workload stresses the thing the kernel changed.



Corn

If you're compiling, the scheduler barely matters and throughput dominates.



Herman

If you're compiling, you care about throughput. If you're doing live audio, you care about the worst-case scheduling latency and throughput is irrelevant. Same kernel, opposite verdict.



Corn

Server versus desktop.



Herman

This is where deliberate kernel selection gets more defensible, because servers have measurable, stable workloads and skilled operators. If you're running a latency-sensitive trading system you already know you care about the tail, not the mean. If you're running a database you care about throughput and page cache behavior. If you're virtualizing, you might care about I/O scheduling. And you can actually benchmark your own workload, which is the rare advantage a desktop user doesn't have.



Corn

The desktop user is guessing.



Herman

Their workload is whatever they happen to do today, which makes benchmarking it nearly impossible and makes the default kernel the correct choice almost every time.



Corn

So most people should just leave it alone.



Herman

Most people should leave it alone. The default kernel is a curated, maintained, security-backported product that the distribution has tested against an enormous range of hardware. Swapping it for something from a website is trading that for a specific behavior you probably haven't measured.



Corn

When is it not a waste of time?



Herman

Hardware support. New chip, old distribution, HWE kernel, clean justification. Or a specialized workload where you know your latency or throughput requirement and can measure it. Or you want to tinker, which is a legitimate reason and not one I'd argue with.



Corn

Say the trade-offs on the other side.



Herman

Stability, because specialized kernels lag on regression fixes. Security, because they don't have Canonical's backport pipeline. Driver compatibility, because a proprietary driver may not build against a kernel it wasn't tested on. And support, because when something breaks the first question anyone asks is what kernel you're on, and the answer "a build from a forum" ends the conversation.



Corn

Herman, you once predicted that the specialist kernel community shrinks as features get absorbed into distributions. Still hold that?



Herman

More than ever, and the real-time merge is the example. When PREEMPT_RT became a config option, a whole category of specialist kernel quietly went away. The same thing is happening with sched_ext. You don't need a gaming kernel if you can load a gaming scheduler onto a stock one.



Corn

Which reframes the whole debate.



Herman

It does. If the future is loading a scheduler per workload instead of choosing a kernel per workload, then the kernel version matters less and less. The Steam Deck story is that future arriving early.



Corn

And XanMod's own documentation saying AVX-512 has no kernel benefit.



Herman

That's the tie-breaker for the CPU-specific argument. If the most aggressive build option does nothing for the kernel, then the "compiled for my exact CPU" pitch is mostly a feeling.



Corn

So where does this leave the default?



Herman

Close to optimal for most users, meaningful for a specific slice, and mostly misunderstood in the middle. The people who switch and are happy are usually the ones who knew exactly what they wanted and measured it. The people who switch and are disappointed are the ones who wanted "faster" and got "different."



Corn

Ninety-nine percent of people should never touch it.



Herman

Ninety-nine percent. And the one percent who should know exactly why.



Hilbert

It's a hundred and twenty hertz, not a thousand.



Hilbert

The tick rate. He said a thousand. On most stock builds it's two hundred and fifty, sometimes a hundred. Liquorix uses a thousand. But the number that matters is the one on your own machine and most people never look at it.



Herman

That's a fair correction.



Hilbert

I ran a custom kernel once because my neighbor claimed his machine booted faster.



Corn

Of course you did.



Hilbert

So we sat there with a stopwatch. I had mine at a thousand hertz, he had his at two hundred and fifty, and we spent the whole weekend opening applications and timing them. Same apps, same order.



Herman

With a stopwatch.



Hilbert

He'd say go, and we'd click, and we'd watch the clock.



Corn

And who won?



Hilbert

He did. His system was faster. But he had a newer SSD, so it wasn't the kernel, and I told him that.



Corn

So you kept the kernel.



Hilbert

I still run it. It's a laptop I use to check email and watch videos. But every time a new release comes out I rebuild it from source, and that takes most of a weekend.



Herman

Every release.



Hilbert

I've got a spreadsheet of the compile times going back years. And the times are getting faster, which I'd like to think is my configuration, though I did get a new CPU at some point.



Corn

A spreadsheet of compile times.



Hilbert

Column for the version, column for the time, notes column. I track my notes.



Herman

And the tinkering is the point, not the speed.



Hilbert

The speed was never the point. Knowing you built the thing you're running, that's the point. The email doesn't load any faster. I just know how it got there.



Corn

Hilbert, let me set the levels before we keep going. Your mic's a touch hot on the plosives.



Hilbert

Noted. Anyway, the spreadsheet's the useful part. Every rebuild the spreadsheet gets a new row, and I can see the machine getting quicker, which is exactly what you'd expect from a newer CPU and nothing to do with the kernel at all. I know that. I still keep the spreadsheet.



Corn

So the tinkering is the reward, and the benchmark is incidental.



Hilbert

That's it. The benchmark is what you do to justify the tinkering.



Herman

Which is a better answer than ours. We told people the default is fine. You're telling them why the people who ignore that advice are still getting something out of it.



Hilbert

They're getting a spreadsheet.



Corn

One thing I want to pick up from Hilbert, before we land this. The compile times getting faster because of a new CPU and not the configuration, that's the same mistake as the benchmark, just harder to see. The thing you changed gets the credit for what something else did.



Herman

The stopwatch and the spreadsheet are the same instrument. You're measuring, but you're not controlling for anything.



Corn

Going back to the sched_ext question, because I think it's the one Daniel's really circling. If you can load a scheduler per workload instead of choosing a kernel per workload, does the kernel version stop mattering?



Herman

Less, but not never. You still need a kernel new enough to support sched_ext, which means 6.12 or later. And sched_ext only covers the scheduler. It doesn't touch memory management or I/O scheduling or power management, so if your bottleneck is one of those you're back to choosing a kernel.



Corn

So it's a partial decoupling.



Herman

The most useful one available. And the real-time merge is the pattern repeating. Twenty years as a specialized patchset, then one release where it becomes a config option, and suddenly nobody has to run a custom kernel to get bounded latency.



Corn

Which ones follow it?



Herman

The real-time support for 32-bit ARM lands in 7.1 with no out-of-tree patches, so that's the next one. And I'd bet on a bunch of the smaller latency patches from the enthusiast kernels following the same path over the next few years.



Corn

The cutting-room floor, one thing. XanMod's build list, dated this month. They've got a main build on 7.2, an LTS build on 6.18, and an RT build on 6.18 from the third of October. A specialty kernel distribution tracking the real-time branch right alongside its stable builds.



Herman

Which tells you real-time has stopped being a separate world. It's a flavor now, the same way low-latency is a flavor of Ubuntu's kernel. That's the merge showing up in a build list.



Corn

The question I'm left with is whether Valve ships that sched_ext scheduler to every Steam Deck, because if a per-workload scheduler becomes the default on a mass-market device, the whole alternative-kernel category gets smaller overnight.



Herman

And the answer is nobody outside Valve knows yet, which is why it's worth watching. The one thing I'd tell Daniel is that his instinct about the second axis is right. "Faster" and "more responsive" are two different questions, and most of the confusion in this space comes from people asking one and getting an answer to the other.



Corn

Which is exactly what Hilbert's spreadsheet was doing. Thanks to our producer Hilbert Flumingtop for holding the levels and the spreadsheet. If this was your kind of episode, go back for episode seven twenty-nine, The Surprising Family Tree of Modern Operating Systems; episode thirty-seven eighty-nine, What Virtualization Actually Costs on two thousand twenty-six Hardware; and episode eleven, How Does Fine Tuning Work Anyway. This has been My Weird Prompts, the human-AI collaboration podcast.



Herman

If you enjoyed this episode, leave us a review and subscribe. It helps other listeners find the show. We'll be back soon.

SOURCES

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

1. **Liquorix Kernel homepage [primary]**
   https://liquorix.net/

2. **XanMod Kernel homepage, builds dated 2026-10-11 [primary]**
   https://xanmod.org/

3. **Canonical Kernel Team policy, 2024-08-09 (updated 2025-08-11) [primary]**
   https://discourse.ubuntu.com/t/kernel-version-selection-for-ubuntu-releases/47007

4. **Ubuntu kernel variants [primary]**
   https://ubuntu.com/kernel/variants

5. **EEVDF kernel docs [primary]**
   https://docs.kernel.org/next/scheduler/sched-eevdf.html

6. **sched_ext kernel docs [primary]**
   https://kernel.org/doc/html/latest/scheduler/sched-ext.html

7. **sched_ext scheduler collection [primary]**
   https://github.com/sched-ext/scx/

8. **Phoronix, 2024-12-20**
   https://www.phoronix.com/news/Linux-6.12-Liquorix-Performance

9. **Phoronix, 2021-07-26**
https://www.phoronix.com/review/ryzen5-xanmod-liquorix/6

10. **Phoronix, 2020**
https://www.phoronix.com/review/ubuntu-xanmod-clear

11. **Phoronix, 2026-04-27**
https://www.phoronix.com/news/Linux-7.1-ARM-RT

12. **Phoronix, 2024-09-20**
https://www.phoronix.com/news/Linux-6.12-Does-Real-Time

13. **PREEMPT_RT overview**
https://en.wikipedia.org/wiki/PREEMPT_RT

14. **LWN, LAVD gaming scheduler at LPC 2025**
https://lwn.net/Articles/1051430/

15. **Phoronix, 2026-06-25**
https://www.phoronix.com/news/Extended-Cache-Aware-Sched