MY WEIRD PROMPTS

Podcast Transcript

EPISODE #205

The RAID Survival Guide: Managing Massive Data in 2026

Published January 08, 2026 • Runtime: 25:41

https://myweirdprompts.com/episode/raid-storage-rebuild-risks/

EPISODE SYNOPSIS

In this episode of My Weird Prompts, Herman and Corn Poppleberry tackle the high-stakes world of data storage in 2026. As 30TB Heat-Assisted Magnetic Recording (HAMR) drives become the new standard for home labs, the brothers revisit the 1987 Berkeley paper that revolutionized how we think about disk reliability. They break down the mechanics of striping, mirroring, and the elegant XOR math of distributed parity, while issuing a stark warning about the "rebuild nightmare" facing modern arrays. From the blistering speed of RAID 0 to the mission-critical reliability of RAID 10, learn why the storage configurations of the past might lead to catastrophic data loss in the age of massive drives.

DANIEL'S PROMPT

Daniel

I’d like to discuss RAID technology, which is widely used in NAS and servers to allow storage systems to survive hard drive failures. While there is a trade-off between storage capacity and redundancy, it is an important way to preserve data. I am currently running RAID 5 on my computer with five disks. I would love to learn about the different types of RAID, how the technology actually works mathematically to spread bits across systems, who invented it, and where else it is deployed in the real world besides NAS.

TRANSCRIPT

Corn

Hey everyone, welcome back to My Weird Prompts. I am Corn, and I am joined as always by my brother.

Herman

Herman Poppleberry, ready to dive into the bits and bytes. It is January eighth, twenty twenty-six, and we have a classic technical topic today that is near and dear to our household.

Corn

It really is. Our housemate Daniel was just telling us about his latest computer build. He is finally moving away from those old ten-terabyte drives and has stepped up to a five-disk RAID five array using next-generation, roughly thirty-terabyte Heat-Assisted Magnetic Recording, or HAMR, drives that are now starting to hit the market. He sent us a prompt asking about the history, the math, and the different flavors of this technology.

Herman

I love that he is running RAID five with thirty-terabyte drives. It is such a classic choice, but in twenty twenty-six, it carries some really terrifying trade-offs that most people do not think about until a drive actually clicks its last click. We are talking about roughly one hundred twenty terabytes of nominal usable capacity before formatting and filesystem overhead, which is incredible for a home lab, but the rebuild math on drives that size is a nightmare.

Corn

Exactly. And I think it is important to set the stage here. We have talked about storage before, like back in episode one hundred fifty-one when we discussed network setups, but RAID is specifically about how we handle the inevitable failure of physical hardware. Herman, for the uninitiated, what are we actually talking about when we say RAID?

Herman

Well, the acronym stands for Redundant Array of Independent Disks. Interestingly, when it was first conceived in the late eighties, the I stood for Inexpensive. The idea was that instead of buying one massive, incredibly expensive mainframe-grade hard drive, which they used to call a SLED—or Single Large Expensive Disk—you could string together a bunch of cheap, consumer-grade disks and make them act like one big, fast, reliable unit.

Corn

That is a great bit of historical context. It was a shift from relying on the perfection of a single component to building a system that could handle imperfection. Who actually gets the credit for that shift?

Herman

It mostly traces back to a nineteen eighty-seven paper from the University of California, Berkeley. David Patterson, Garth Gibson, and Randy Katz published A Case for Redundant Arrays of Inexpensive Disks. They presented it at the SIGMOD conference in June of nineteen eighty-eight. They basically laid out the blueprint for everything we use today. They realized that as disk drives got smaller and cheaper thanks to the PC revolution, the real bottleneck was not just capacity, but performance and reliability. If you have one big SLED, and it fails, you are done. If you have five disks working together, you have options.

Corn

I love the term SLED. It sounds so heavy and industrial. So the Berkeley team basically said, let's use the power of the many rather than the power of the one. But they didn't just invent one way to do it, right? They came up with a whole list of levels.

Herman

They did. They originally defined RAID levels one through five. Over the years, we have added RAID zero, RAID six, and various nested levels like RAID ten or RAID sixty. But before we get to the popular ones, we should probably mention the ones that time forgot. RAID levels two, three, and four are basically history lessons now.

Corn

Why is that? What happened to them?

Herman

RAID two was super complex. It striped data at the bit level and used something called Hamming codes for error correction. It required a massive number of disks just for the parity information. RAID three striped at the byte level, and RAID four striped at the block level but used a single, dedicated parity disk. The problem with RAID four was that every single time you wrote data to any disk, you had to update that one parity disk. It became a massive bottleneck. RAID five solved that by spreading the parity across all the disks, which is why RAID four is mostly a ghost in the machine today.

Corn

Okay, so let's walk through the ones people actually use. The different RAID levels can be a bit of a soup of numbers. Most people hear RAID zero, RAID one, RAID five, and their eyes glaze over. Let's start with the simplest one, even if it is a bit of a misnomer when it comes to the redundant part of the name.

Herman

Right, RAID zero. I always say the zero stands for how much data you have left if one drive fails. RAID zero is all about striping. Imagine you have a long sentence to write down. Instead of writing the whole thing on one piece of paper, you write the first word on paper A, the second word on paper B, the third word on paper A, and so on.

Corn

So you are basically splitting the data across two or more disks. The benefit there is speed, right? Because you can read from both disks simultaneously.

Herman

Exactly. You are doubling your throughput. In twenty twenty-six, with PCIe Gen six NVMe drives, a RAID zero array can reach speeds that are frankly ridiculous—we are talking about tens of gigabytes per second. But, as you pointed out, there is zero redundancy. If disk A dies, you have half of every sentence missing. The entire array is toast. It is not a matter of if you lose data, but when.

Corn

It is a high-wire act. Great for temporary scratch space or video editing where you have the original footage elsewhere, but terrifying for a primary storage volume. So, what is the polar opposite? That would be RAID one, right?

Herman

Exactly. RAID one is mirroring. This is the simplest form of actual redundancy. You take two disks, and you write the exact same data to both of them at the same time. If disk one dies, disk two is a perfect clone. You just keep on working.

Corn

It is the ultimate peace of mind, but the cost is high. If you buy two thirty-terabyte drives, you only get thirty terabytes of usable space. You are paying a fifty percent tax on your storage capacity.

Herman

It is a steep price, which is why we do not see RAID one used for massive data sets. It is great for your operating system drive where you just need it to work no matter what, but if you are Daniel and you have a massive library of media or research data, you want something more efficient. That brings us to the king of the home server world, which is RAID five.

Corn

This is what Daniel is running with his five disks. And this is where the math gets really cool. RAID five uses something called parity to provide redundancy without losing half your space. Herman, how do you explain parity without a whiteboard?

Herman

It is actually quite elegant. Think of the exclusive or operation, or XOR. In the world of binary math, XOR is a simple rule. If you have two bits, and they are the same, the result is zero. If they are different, the result is one. So, zero XOR zero is zero. One XOR one is also zero. But one XOR zero is one.

Corn

Okay, I follow that. But how does that help me recover a lost file?

Herman

Imagine you have three disks. Disk A has a bit of data, let's say a one. Disk B has a zero. To create the parity bit for Disk C, you XOR them together. One XOR zero equals one. So Disk C stores a one. Now, imagine Disk A fails. We look at what we have left. We have a zero on Disk B and a one on Disk C. If we XOR those together, zero XOR one equals... one. We just recovered the missing bit from Disk A! It is beautifully symmetrical. You can have four, five, or even ten disks, and as long as you XOR all the data bits together to get the parity bit, you can always solve for the missing variable if one disk fails.

Corn

That is incredible. It is like an algebraic equation where you can always solve for X as long as you only lose one variable at a time. So in a RAID five setup, you are basically spreading those parity bits across all the drives in the array.

Herman

Precisely. We call it distributed parity. In Daniel’s five-disk setup, the system treats the capacity of four disks as usable space, and the equivalent of one disk's worth of space is used for parity. But it is not just one dedicated parity disk like in RAID four. The parity information is rotated across all five disks. This prevents a single disk from becoming a bottleneck during writes, though there is still a performance hit because every write requires a read-modify-write cycle to update that parity bit.

Corn

So he gets eighty percent of his total raw capacity, and he can survive any single drive failing. That feels like a great deal. But I have heard you mention before that RAID five has a bit of a dark side, especially as hard drives have gotten massive. It is twenty twenty-six, and we are seeing thirty and forty terabyte drives now. Why is RAID five risky there?

Herman

It comes down to the rebuild time and the math of unrecoverable read errors, or UREs. When a drive fails in RAID five, the array is in a degraded state. To get back to safety, you have to put in a new drive, and the system has to read every single bit on the remaining four drives to calculate the missing data for the new fifth drive.

Corn

And with a thirty-terabyte drive, that is a lot of reading.

Herman

It is a staggering amount of reading. Even at a sustained speed of about two hundred megabytes per second, it would take on the order of forty hours—getting close to two full days—of non-stop reading to rebuild a thirty-terabyte drive. In reality, with system overhead and other tasks, it can take many days; on heavily used systems it can stretch toward a week or more. During that entire time, your remaining disks are working at very high utilization, generating heat and stress. If a second drive fails during that rebuild process, the entire array is lost.

Corn

And the probability isn't zero, right?

Herman

Far from it. Many consumer and NAS-class drives have an unrecoverable read error rate specified around one in ten to the fourteenth to ten to the fifteenth bits. That sounds like a lot, but ten to the fourteenth bits is only about twelve and a half terabytes. If you are rebuilding a hundred-terabyte array, the statistical likelihood of hitting a read error on one of the surviving drives is actually quite high. If you hit that error during a rebuild, the controller often just gives up and marks the whole array as failed. This is why many enterprise admins say RAID five is effectively dead once you get into multi-terabyte drives, especially above two to four terabytes, depending on your risk tolerance.

Corn

This is why I have seen more people moving toward RAID six, right?

Herman

Exactly. RAID six is like RAID five but with double parity. It uses two different mathematical equations—usually XOR and a more complex Reed-Solomon code—to store redundant data. This means it can survive the simultaneous failure of two drives. You lose the capacity of two disks, but you gain a massive amount of safety during that long rebuild process. If Daniel had six disks instead of five, I would have strongly suggested he go with RAID six. In twenty twenty-six, many storage engineers consider RAID six the practical minimum for large spinning disks built from very high-capacity drives.

Corn

It is interesting how the technology has to evolve just to keep up with the physical scale of the hardware. Speaking of evolution, there is another level that people often confuse with RAID zero or one, and that is RAID ten. Or is it RAID one plus zero?

Herman

It is RAID one-zero, or a stripe of mirrors. This is the gold standard for performance and reliability if you can afford the disks. You take four disks, and you pair them up. Disk one and two are a mirror. Disk three and four are a mirror. Then, you stripe data across those two mirrors.

Corn

So you get the speed of RAID zero because you are striping, but you get the redundancy of RAID one.

Herman

Correct. And the rebuild is much faster than RAID five or six because you aren't doing complex math. You are just copying data from the surviving twin of the mirror. It is very popular in high-performance databases where every millisecond of latency matters. The downside, of course, is that you lose fifty percent of your capacity. If Daniel used RAID ten, he would only have sixty terabytes of nominal space instead of one hundred twenty.

Corn

I want to circle back to something Daniel mentioned in his prompt. He talked about how the system can grow, like a starfish regrowing a limb. But he also touched on something that I think a lot of people get wrong. He said RAID is not a backup. Herman, why do we need to shout this from the rooftops?

Herman

Because it is the most dangerous misconception in all of computing! RAID protects you against hardware failure. It does not protect you against human failure, software failure, or environmental failure. If you accidentally delete a file, RAID will dutifully delete it from all your mirrored or parity disks simultaneously. If a piece of ransomware encrypts your data, RAID will help it encrypt that data across your entire array with maximum efficiency.

Corn

Or if your power supply surges and fries every disk in the chassis at once.

Herman

Exactly. RAID is about uptime. It is about making sure your server stays online even if a mechanical part breaks. Backup is about recovery. You need a separate copy of your data on a different device, preferably in a different physical location. We usually talk about the classic three-two-one rule, and these days some data-protection folks extend that into a three-two-one-one-zero rule: three copies of your data, on two different media, one off-site, one offline or immutable, and zero errors after verification. If your data only exists on one RAID array, you do not have a backup. You have a very sophisticated single point of failure.

Corn

That is a sobering thought. But let's look at the brighter side. Where else is this technology being used? Daniel mentioned NAS units, which are common for home labs and small businesses, but RAID is everywhere, right?

Herman

Oh, absolutely. If you use any cloud service, whether it is for photos, email, or document storage, your data is living on some form of RAID or its modern successor, erasure coding. In massive data centers like the ones run by Amazon or Google, they aren't just using five-disk arrays. They are using massive distributed systems that act like RAID on a global scale. They might break a file into ten pieces and add six parity pieces, then spread those sixteen pieces across sixteen different servers in three different buildings. That way, an entire building could burn down and your data would still be safe.

Corn

I imagine the financial sector is a huge user as well.

Herman

Huge. Think about high-frequency trading or even just your local bank's transaction ledger. They cannot afford for a single hard drive failure to halt trading or lose a record of a deposit. They use hardware RAID controllers with dedicated processors and battery-backed cache memory. This ensures that even if the power goes out in the middle of a write operation, the data stays in the controller's memory until it can be safely committed to the disks.

Corn

That brings up a technical nuance I wanted to ask you about. I have heard the term write hole in relation to RAID five. What is that, and why should Daniel care?

Herman

The write hole is a classic RAID five problem. Imagine the system is writing data and calculating the new parity bit. If the power cuts out after the data is written but before the parity bit is updated, you now have a mismatch. The next time the system tries to use that parity bit to recover data, it will calculate the wrong result. Modern systems use things like journaling file systems or non-volatile RAM in the controllers to bridge that hole, but it is a reminder that these mathematical systems are still bound by the laws of physics and electricity. This is why a good Uninterruptible Power Supply, or UPS, is mandatory for any RAID setup.

Corn

It is amazing how much engineering goes into just making sure a one stays a one and a zero stays a zero. Now, looking at the year twenty twenty-six, we are seeing a shift away from traditional hardware RAID toward software-defined storage. Things like ZFS or BTRFS. How do those differ from the classic RAID Daniel is using?

Herman

That is where it gets really exciting. Traditional RAID is often invisible to the operating system. The OS just sees one big disk. But software-defined storage, like ZFS, is aware of both the disks and the file system. It uses something called checksums. Every time it reads a block of data, it verifies it against a known mathematical signature.

Corn

So it can detect bit rot?

Herman

Exactly. Standard RAID can tell you if a drive has failed completely. But what if a drive is still spinning, but it just happens to flip one single bit because of a cosmic ray or a minor magnetic flaw? A traditional RAID controller might not notice. It will just hand that corrupted data to the user. ZFS sees the checksum mismatch, realizes the data is wrong, and uses the parity information to automatically repair the corrupted bit on the fly. We call it self-healing storage. It also eliminates the write hole problem entirely by using a copy-on-write architecture. It never overwrites data; it always writes new data to a new block and then updates the pointers.

Corn

Self-healing. That really does sound like the starfish analogy Daniel used. It is a much more intelligent way of handling data integrity. So, if Daniel ever decides to rebuild his system, maybe a move to a ZFS-based setup like TrueNAS would be a good next step for him.

Herman

I would highly recommend it. Though, knowing him, he will probably want to jump straight to a ten-disk RAID-Z three array just to see if he can make the lights in the house flicker. RAID-Z three is the ZFS version of triple parity, meaning he could lose three drives and still stay online.

Corn

He does love his hardware. You know, thinking about the broader implications of this, RAID really changed the economics of the entire internet. By making it possible to use cheap, mass-produced disks for mission-critical tasks, it lowered the barrier to entry for everything from web hosting to streaming services.

Herman

You are spot on. Without RAID, the cost of storing a petabyte of data would have remained astronomical for much longer. It democratized high-availability storage. It is one of those invisible technologies that hums along in the basement of the digital world, and we only notice it when the little red light on a server tray starts blinking. In twenty twenty-six, we are seeing this evolve into AI-driven storage management where the system can predict a drive failure before it happens by analyzing vibration patterns and latency spikes.

Corn

I love those kinds of technologies. The ones that are so successful they become invisible. Herman, we have covered a lot of ground here, from the Berkeley paper in eighty-seven to XOR math and the dangers of the RAID five rebuild. What are the big takeaways for someone like Daniel or any of our listeners who are managing their own data?

Herman

First, understand your RAID level. Know the difference between striping for speed and parity for safety. Second, be honest about your drive sizes. If you are using massive thirty-terabyte drives, RAID five is simply not enough protection anymore; you need RAID six or RAID ten. Third, and I will say it again, RAID is not a backup. Have an off-site, immutable copy. And finally, look into software-defined storage. The intelligence that ZFS or similar systems bring to data integrity is well worth the learning curve.

Corn

Solid advice. And speaking of data, if you found this deep dive into storage arrays useful, we would really appreciate it if you could leave us a review on your podcast app or on Spotify. It genuinely helps the show grow and reach more people who might be wondering why their NAS is making that weird clicking sound.

Herman

Yes, please do. We love hearing from you. And if you want to see the show notes or find our RSS feed, head over to myweirdprompts.com. We have all our past episodes there, including the ones we mentioned today.

Corn

This has been a great one. Thanks to Daniel for the prompt. It is always fun to dig into the mechanical and mathematical heart of our machines.

Herman

Absolutely. It makes me want to go check the health of our own arrays right now.

Corn

Probably a good idea. Thanks for listening to My Weird Prompts. I am Corn.

Herman

And I am Herman Poppleberry. We will catch you in the next one.

Corn

See ya.

Herman

We should really talk about erasure coding next time. It is like RAID but for the cloud era.

Corn

Oh, I am sure Daniel will have a prompt about that soon enough. He is already looking at distributed file systems for his next project.

Herman

Of course he is. The man never sleeps.

Corn

Neither does his server. Alright, let's go get some coffee.

Herman

Sounds like a plan.

Corn

One last thing for the listeners, if you have a weird prompt of your own, something that keeps you up at night or something you have been tinkering with in your home lab, send it our way through the contact form at myweirdprompts.com. We might just dedicate an entire episode to it.

Herman

We love the technical ones, but we are open to anything. The weirder, the better.

Corn

Exactly. Until next time, everyone. Stay curious.

Herman

And keep your backups current!

Corn

Especially that. Bye!