# WeGet letters

by Michael W Lucas

Dear FreeBSD Letters Columnist,

How are you? I am fine. Well, mostly fine. Sort of fine. Okay, no, I am not fine at all. I am a new sysadmin and utterly lost and am hoping you can offer guidance. I'm setting up a new web server, just like this issue is about, and I'm stopped in the installer. I hear all these things about optimizing filesystems for different applications, and there's lots of blog posts about ways to optimize, but I'm not sure which advice to take. Whatever I set up I must live with for years! Please help me make less bad choices.

—New Sysadmin

Dear NS,

Your letter is something of a relief, as it provides ample distraction from the horrors of administering the web server I set up in 2017, or my Sendmail configuration from 1992. It's like getting my mind off my abscessed tooth by busting a few of my ribs. Well done!

You might be a new sysadmin, but at least you understand that you must live with your bad decisions throughout the server's lifespan. Yes, you could get all DevOps and dynamically redeploy hosts with improved settings, but all you're doing is reducing the time you must live with one set of decisions before replacing them with a different set of equally bad ones. A change of poor choices is not as good as a rest.

**How do you optimize a filesystem for a database at install time?**

How do you optimize a filesystem for a database at install time? My answer is: don't. Premature optimization is the root of all evil, along with poor privilege management and nano. You have no idea how your database will interact with the filesystem until you run the application under real load. The only sensible choice is to arrange your new system so that you will have empty disks to move your database to. Yes, this is pretty much the same thing as devopsing to a new host, except it's not a new host and you don't need Ansible. If you're using one of those virtual host providers that offers block storage, that's dandy, except you'll be formatting those blocks with a filesystem. Where The Cloud is really "other people's computers," Block Storage is "other people's cast-off hard drives arranged in a Redundant Array of Inexpensive Crap." The main advantage is you're not the person who needs to trace the alarm beep to a drive tray.

If you insist on optimizing your filesystems, well, here's what you do.

First off, understand that storage devices are lying liars that lie. The newest solid state storage maintains a malformed compatibility with hard drives released in the previous century, which were built on standards designed for punch cards, which had their roots in 17th-century looms and the Luddites, so every time you plug in a storage device you're putting someone out of work but there's no ethical data storage under capitalism so go for it. The main lie that needs to concern you is the sector size. Today's drives overwhelmingly use 4K sectors, except for some NVMe devices that support multiple sector sizes but I don't have any of those so I'll pretend they don't exist. You need to make sure that the partitions—not the filesystems, the *partitions*—on your drives align with those 4K sector sizes. If the fancy boot loader you like requires a 98K GPT partition, it fills nineteen and a half disk sectors. Drives claim that they'll save you, but that's another lie. That next partition better begin at 100K, a nice multiple of 4, or all your filesystem blocks will be split between drive sectors and every interaction with the hardware will take twice as long and burn out the drive twice as quickly.

**What about the filesystems? Filesystems need tuning! Balderdash, I say!**

Once you have partitions, the filesystem blocks need to also be multiples of 4K. ZFS defaults to 128K stripes. UFS defaults to block sizes of 32K, which lets it use 4K fragments, so it should be good as-is but don't get clever and think that smaller blocks mean better performance because—no matter what a bunch of old blog posts say—they don't.

There you go. You can report to management that your filesystems are tuned for your hardware. Return to playing nethack.

*But that's the partitions*, I hear some of you whine. *What about the filesystems? Filesystems need tuning!* Balderdash, I say! You wouldn't tuna fish, why tune a filesystem? Filesystems are written for lazy people. Leave them alone, they have only caused you as much pain as their programmers insisted on, and that was only because marketing insisted on planned obsolescence to compel upgrades. BSD operating systems are not driven by profit, and user pain increases support requests, so the amount of filesystem agony has been methodically reduced until there's hardly any. Why, using filesystems barely qualifies as "torment" anymore. Any changes are likely to increase your pain.

You're still here? You still want advice?

Huh. You do know that therapists build careers out of helping people like you, right?

Fine.

Your filesystem should reflect your data. If you know that your data consists of, say, many 64KB files, you can set your blocks to that size. You know your own data, you should be able to figure this out. If your data is less predictable, don't optimize.

Databases can tempt even the most jaded sysadmin into optimizing their filesystem. Databases have predictable block sizes. MySQL uses a 16K block, so you could configure the underlying ZFS dataset to use a recordsize of 16K. MySQL can compress its data, as can ZFS. Attempting to compress already-compressed data wastes system resources. Study your application and pick a place to compress data.

Don't configure UFS to use 16K blocks, even when it's supporting MySQL. A UFS block has eight fragments, and thanks to the underlying disk each fragment has a minimum size of 4K. Putting two UFS fragments on each disk drive sector would shatter performance. Tuning your MySQL install to use larger blocks on UFS might make sense, but again, leave the filesystem alone.

Postgres? 8K blocks everywhere by default. Again, tuning the database to match the disk might make sense, but 8K blocks on either ZFS or UFS ruin system performance. If your data demands 8K blocks, however, your best option is to use ZFS and set an 8K record size.

But in general, leave bad enough alone unless you want to make things worse. Which is a very common impulse among people who won't get the therapy they need. But rest assured, a few months of struggling to understand the interactions between applications and filesystems will soon make you an *experienced* system administrator.

You poor slob.

Have a question for Michael?
Send it to letters@freebsdjournal.org

**MICHAEL W LUCAS** has written over fifty books, and the UN's "Special Ambassador To Make Lucas Shut Up" was accidentally-on-purpose crushed beneath a stack of them. He wrote one book about UFS and two on ZFS. As the ZFS books were co-written with Allan Jude, they might even contain something helpful.