WeGetletters

by Michael W Lucas

Dear Letters Person,

Every day someone figures out new types of virtualization or ways to complicate it. Full virtualization, light virtualization, containers, ABI compatibility, it just goes on and on. Where does this end? How can I get ahead of this?

—Racing Ahead of Virtualization Every Day

PS: does the *Journal* have a new person for this column yet, or am I going to get a completely unhelpful rant?

Dear RAVED,

Anyone who believes my rants are unhelpful is new to IT. Once your trauma helps you develop triggers, you'll understand. Give it another week.

Business schools teach the importance of "getting ahead" of a trend. That's utterly inapplicable to sysadmins. We not only are the trend, we spend our spare time dreaming up ways to make the trend both steeper and even more trendy.

We continuously delude ourselves that we can make things better, when we can only make things differently bad.

The server has one hard drive? Better mirror that, or RAID-5 it, or fibre channel it off to the NAS where it becomes someone else's problem. Bloatware runs slowly? Trim the program or add memory or install a fast caching disk. Cosmic ray filesystem corruption? A zraid3 array with copies=99 will fix that! Every "improvement" adds failure modes. Even senior sysadmins, who understand in their marrow Rule of System Administration #16 (*The impulse to improve is the leading cause of failure*) fall prey to this fallacy.

Business schools teach the importance of "getting ahead" of a trend. That's utterly inapplicable to sysadmins.

Virtualization is merely another expression of such.

This isn't new. Remember chroots? When the Internet converted from a private educational network to a tangle of private enterprises, much of the core software was revealed to have critical security problems. Nobody on the private Internet wanted to destroy the private network, but once it went public a handful of people out in the world liked setting nice things on fire. (Looking at the modern Internet, I can't help wondering if some of those early attackers were time travelers come back to save us from ourselves.) If a program suffered from shell escapes, you could run it from a directory that contained only the files the program needed. You can't have a shell escape if you don't have a shell to escape to, after all! This is both a clever hack and a prime example of solving the wrong problem.

FreeBSD's jails were first conceived of as enhanced chroots. What if you took that locked-in directory, gave it an IP address and its own process space? It'd look like a full system but wholly contained within another system! We could even give that super-chroot the ability to run its own child super-chroots! It's beautiful, and elegant, and complex enough to encourage not only failures, but exciting new failure modes you've never previously experienced.

But there's good news, jail fans! Chances are you're using your jail for a specific task. It only needs a small selection of the base system, not a complete userland. OccamBSD is quite usable, and lets you install only the necessary system components. Where jails were an improvement over chroots, chroots are now an improvement over jails!

The impulse to improve.

Or the Linuxulator. You can take a FreeBSD system and have it run Linux programs. Linux is just a kernel, after all. Install your least loathed Linux userland in a directory, chroot your programs into that directory, and run as if they were a Linux system. That'll save you from installing an extra server just to run Linux because (real talk here) who wants to run a Unix without ZFS or PF? Many developers use the Linuxulator as an intermediate step to port Linux software to FreeBSD. An easy win, right?

You can take a FreeBSD system and have it run Linux programs. Linux is just a kernel, after all.

Sure. Linux mode is almost entirely compatible with Linux. "Almost entirely compatible" is like "almost leprosy-free;" good for you, but I ain't touching it. I'm better off licking armadillos.

These aren't enough, though. We want to further optimize our virtualization, so we add in unionfs and base jails. Optimize! Never mind today's endless infinite oceans of disk space. Yes, yes, your big data application needs a storage array, but you can fit thousands of OS installs on an NVMe disk and upgrade them all with a simple script. Or you can have the optimally arranged base jails upgrade the One True Jail with a simple script and have another simple script that runs through all your OS installs and applies the needed upgrades to each. Nobody's willing to address the real problem and design systems that don't need upgrades, because that would put the entire computer industry out of business and who would poison the planet then?

Heavy virtualization? Bhyve, qemu, libvirt, all of those? Mere super-jails that add CPU, process, and filesystem isolation. More secure? There's a reason security professionals say, "another day, another hypervisor escape."

Virtualization allows endless opportunities for not mere failure, but debacle. The tangle of complex interconnected systems creates an exponential climb of interactions, a rising arc that peaks at a number too large for that lump of mildly electrified pudding in your skull to process. You're a sysadmin. You'll sit down and contemplate how these systems might fail. Some failure modes are obvious: fire, flood, famine. Some less so: what if

System V IPC communications leak between two particular jails, leaking company secrets into the world? On second thought, hang onto that excuse; it'll be useful when you decide to blow the whistle on your unethical employer. What, you claim they're ethical? Then how are they making the money to pay you? Don't worry about it, when it becomes impossible to ignore, you're prepared to leak.

The apparent "problem" is, of course, that modern hardware is ridiculously overpowered for most tasks. I rent a small, dedicated server. It runs jails and bhyve VMs. The most resource-intensive application I run is my own email server. Email takes very few resources, but rspamd takes everything you give it. The hardware is mostly idle, so I pile more functions on it. Because it's *there*. Why not try to make things a little better, and use my resources?

Having too much computing power is not a problem in any reasonable sense of the word. But I did it anyway.

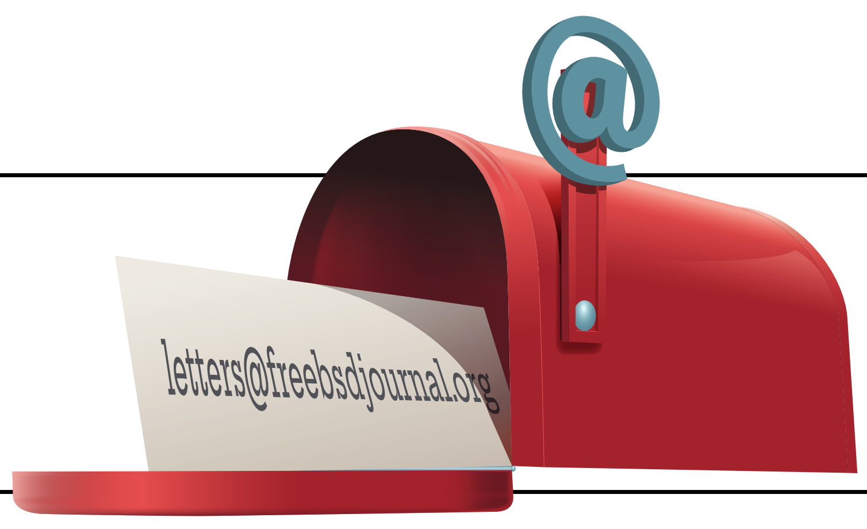
So: getting ahead of the trend? You mean get ahead of this nightmarish tangle of optimizations?

Yes, there is a way to "get ahead of the trend."

Fail faster.

Good luck. I have faith in you.

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



MICHAEL W LUCAS' latest books include *Dear Abyss* (a collection of these columns), *Run Your Own Mail Server*, and *Apocalypse Moi*. See more at https://mwl.io.

FreeBSD Mastery Jails
SSH Mastery
FreeBSD Mastery: Storage Essentials
Letters to ed(1)
DNSSEC Mastery
Ed Mastery
Tarsnap Mastery: Online Backups for the Truly Paranoid
FreeBSD Mastery: Specialty Filesystems
Only Footnotes
PAM Mastery
Sudo Mastery
SNMP Mastery
FreeBSD Mastery Advanced ZFS
TLS Mastery
Michael W Lucas