How ZFS Made Its Way into FreeBSD

BY PAWEL DAWIDEK

The story of how the ZFS file system made its way into the FreeBSD operating system is a tale of passion for programming, love for technology, and a journey that led to my most valuable contribution to the FreeBSD project. It was the summer of 2005. Although I'm not great with dates, I do remember the circumstances surrounding my first encounter with ZFS. I was with my friends in the Masurian region of Poland, near one of its 2,000 beautiful lakes. One of my friends worked at a Polish telecommunications company at the time, and they used a lot of Sun Microsystems hardware and Solaris. He brought a printed copy of the announcement they had received from Sun, describing a new file system that had been in development for some time and was about to be released as part of OpenSolaris. But before I continue, let's take a step back in time for a bit of context…

I fell in love with programming at first sight. I was 12 years old and my cousin introduced me to Basic on a C-64..

Love at First Sight!

I fell in love with programming at first sight. I was 12 years old, and my cousin—Tomek—introduced me to Basic on a C-64. I was hooked. I felt like a young god: you take this soulless piece of hardware, create a program, and watch it come to life! This was the coolest thing; I couldn't think of anything better. As a result, I was never into video games. Back then, growing up in a small Polish town, it wasn't easy to find people interested in programming, so I was pretty much on my own (well, except when I was testing the limits of Tomek's patience).

When I switched to the Amiga 500, I finally found some friends from the demo scene with whom I could exchange my work using 3.5-inch floppy disks through the postal service. Latency was not the best, but I didn't complain. When my next computer—the Amiga 1200—started to show its age, it was clear it was time to move on. I knew Microsoft Windows was not for me. I'd tried Linux briefly, but I still wasn't convinced. Finally, a friend pointed me to FreeBSD. The installation was a breeze, I couldn't ask for a better experience. Ha! If you didn't cringe at that last sentence, then you clearly haven't had the "pleasure" of using sysinstall. No, the installation wasn't a breeze—it took multiple attempts for me to finally enjoy my first FreeBSD system. My understanding was that sysinstall must be like Navy Seals Hell Week, where the strong are separated from the weak, where real hackers are forged! And I made it! I set my next goal and dream to not only be a hacker who can install FreeBSD but to be a kernel hacker and a FreeBSD committer.

I accomplished my goal in 2003 when I officially joined the FreeBSD project as an src committer. Yes, I threw a party to celebrate that. Since I joined, I have worked in many areas of the system, but mainly with the GEOM framework. The GEOM framework in FreeBSD sits between disk drivers and file systems, allowing plug-ins of various transformations, like mirroring, RAID, block-level encryption, etc. I really liked the GEOM design and loved working with it, so I had decent experience with at least part of the storage stack. As for file systems, I knew enough to stay away from VFS, which is one of the most complicated parts of the kernel.

The curse. UFS has been the default file system in FreeBSD since the very beginning. In fact, UFS is much older than FreeBSD itself. UFS2, which was introduced in FreeBSD 5.0, addressed some shortcomings of UFS1, but some important ones were still not addressed. The main issue was the fsck time after a system crash or power outage. With disks getting bigger and bigger, fsck could take many hours to complete. The solution to this problem was obvious—we either needed to add journaling to UFS or port some other journaled file system to FreeBSD. Easier said than done. In Linux, there were plenty of file systems to choose from, and many people tried to port them to FreeBSD, but for some weird reason, those ports were never finished, so we ended up with extfs without journaling, read-only ReiserFS, and read-only XFS. There was even a read-write HFS+ port from Mac OS X, but, of course, no journaling, and I remember at least one failed attempt to add journaling to UFS. What was this mystery? Had the UNIX gods turned their backs on us?

Let's come back to my vacation in Masuria. My friend starts to read the ZFS announcement while my eyes and my mouth open wider and wider: Pooled storage—you can create as many file systems as you want, and they will all share available space. Unlimited snapshots that take no time to create. Unlimited clones. Built-in compression. End-to-end data verification. Self-healing of corrupted data. Transactional copy-on-write model—always consistent—no need for fsck. EVER. How? How is that even possible? This is not an evolution, but a clear revolution in file systems. I remember dreaming about this perfect marriage: the best file system running on the best operating system… Wouldn't that be amazing?

A few months later, ZFS was released, and it took not only the open-source community but the entire storage industry by storm. Some people hated it, most loved it, others feared it, but nobody was ignoring it. It was called the last word in file systems. It was called a rampant layering violation. However, it was never called just another file system. Almost every operating system wanted

ZFS: Linux userland port started under FUSE, DragonFlyBSD announced ZFS would be ported soon, and Apple started to port ZFS to Mac OS X. ZFS will be everywhere soon, just not in our beloved FreeBSD...

To write this article, I had to analyze a lot of IRC logs from that era. What struck me the most was how much skepticism there was about ZFS itself: too complicated, too many layers, just demoware, design flaws are going to be found soon, just wait for the first disaster story, it will never be ported to a community-developed OS, it's just hype, it's hilarious. I guess people are used to the fact that if something looks too good to be true, it often is. Fortunately, love is blind, and I didn't notice this at that time.

After waiting 10 months after the ZFS release and seeing nobody starting the work, I thought I might as well give it a shot. With almost zero knowledge about the VFS layer, I'd likely fail quickly, but who could stop me from trying? At the very least, I'd learn something new. My porting work started on August 12, 2006. To not raise people's hopes too high, the perforce branch I created had "This is not a ZFS port!" as its description. My initial estimates were six months to have a read-only prototype.

I must admit that even though ZFS was not my creation, working on ZFS was the most engaging project in my career. I love to work hard, and I love to work late. I like to hyperfocus on projects, and I have been fortunate to work on many amazing projects. For many years, I was one of the most productive FreeBSD committers while still developing my own business. But no other project kept me awake for 48 hours straight with almost no breaks and only short naps between those 48 hour periods. What I'm about to tell you sounds impossible, even for me today, but it did happen, I can assure you :)

When I work on big projects, I still like to have something that I can run as quickly as possible and then incrementally implement missing bits. The first step was to port userland components like libzpool, ztest, and zdb. This went mostly ok. The next challenge was to compile and load the ZFS kernel module. When you try to load a kernel module with missing symbols, the FreeBSD kernel linker reports the first missing symbol and returns an error. I had so many missing symbols that I had to hack the linker to report them all at once. It was taking too much time to fix them one by one. After five days, I loaded zfs.ko for the first time. In theory, there were four main meeting points between the FreeBSD kernel and the ZFS code:

1. On the bottom of the stack, we have to teach ZFS how to talk to block devices in FreeBSD, so this means connecting ZFS to GEOM, which in GEOM terms is creating a consumer-only GEOM class. Because of my GEOM experience, it was trivial.
2. On the top of the stack, we need to connect ZFS to FreeBSD's VFS, so port the ZPL layer.
3. Also, on the top of the stack, ZFS storage can be accessed through ZVOLs, and because ZVOLs are block devices, this is again GEOM, but this time provider-only GEOM class.
4. The last component is the /dev/zfs device that is used by userland ZFS tools (zfs(8) and zpool(8)) to communicate with the ZFS kernel module.

Porting the ZPL layer and attaching ZFS to FreeBSD's VFS was, of course, the hardest part. The first kernel mount on FreeBSD happened on August 19, 2006, so, one week in. After exactly ten days (and nights) of work, I had a read-write prototype ready. I could create pools, create file systems, and mount them, create ZVOLs whose behavior was really stable, create files and directories, list them, and change permissions and ownership. My initial estimates of six months for a read-only prototype turned out to be "a little" off. There was still a huge amount of work to do, but the encouragement from the community gave me the needed motivation to continue and finish the project. In 2007, ZFS was officially released with an experimental status in FreeBSD 7.0, and in FreeBSD 8.0 (2009), it was declared as production-ready.

No other port that was announced before my work came to fruition, so I guess to reverse a curse you just need to work hard enough :)

Just to be fair, a working read-write prototype in ten days wouldn't have been possible without one very important decision that was made by the ZFS creators at the early stage of ZFS development. They wanted most of the code to compile in userland, so it could be easily tested and debugged. This was an immense help in my porting efforts because most of the code was already highly portable.

When I work on big projects, I still like to have something that I can run as quickly as possible and then incrementally implement missing bits.

This was an amazing journey, and I wish every software developer a similar experience. While writing this, I'd like to recognize some people. First, I'd like to thank Jeff Bonwick, Matt Ahrens, and the whole ZFS team at Sun for creating this revolutionary technology and always supporting my work. Alexander Kabaev, for all his patience and help with VFS. Robert Watson, for all his encouragement and for being a role model I always looked up to. Kris Kennaway, for being a ruthless early tester—we didn't call him BugMagnet for nothing. Martin Matuska, for stepping up and taking over ZFS maintenance when the time came. And last, but not least, I'd like to thank the entire FreeBSD community—there is nothing that brings more satisfaction than feeling that your work is appreciated and provides real value.

A lot has happened in the 20 years since ZFS was initially released: NetApp started a legal battle against Sun. Apple discontinued the ZFS port. Licensing issues prevented ZFS from being a native component of the Linux kernel. Sun Microsystems no longer exists, and the new owner closed ZFS development. And yet, this great technology prevailed, and the project lives on under the OpenZFS flag. Long live OpenZFS! Long live FreeBSD!

PAWEL DAWIDEK is Co-Founder and CTO at Fudo Security, a security vendor building products for secure remote access. He is also involved in the FreeBSD operating system where he works on security- and storage-related projects, like GELI disk encryption, Capsicum capability and sandboxing framework, jail containers, ZFS and various GEOM classes. Pawel's passion outside of technology is training Brazilian Jiu Jitsu.