Sat Jun 19

(*alt title; “2020s are the decade of NAS”)*
(alt title; “2020s are the decade of the personal server”)

There comes a time in every programmer’s life when he realizes that it’s time to build the homelab.

Up until that point, there are always the same bottlenecks; there’s not enough time, not enough space, or not enough money. Space was the big bottleneck for me; living in San Francisco for 7 years, writing software for these major companies while habitating in a shoebox apartment, is kinda the sine qua non of the “2010s developer” (we are the nation of renters).

Moving back to Kansas City was the first big leap toward, well, the normalcy of simple ownership. My current dwelling is 10x bigger than my last one in SF, and I’m paying ½ the cost for it. It’s really hard to understate how transformative it is to merely come into possession of a large amount of open, personal space. There really is no comparable vector of freedom, opportunity, et cetera.

==

This latest round of interest with building a homelab started with learning about these Synology NAS array boxes. From a technical standpoint, they’re basically a glorified, special-purpose desktop computer, with easily-swappable hard drive slots. From a user experience standpoint, they’re kind of a small revolution.

Every decent programmer is plagued by the knowledge that what they’re doing when they build applications and storage solutions is technically quite simple. You have a hard drive spinning somewhere. You have some applications that read data from that hard drive, and allow the user to interact with it in specified ways. Anyone who has used an offline desktop or laptop program is familiar with the basics of how this works.

When you start getting into *distributed* systems, the (needless) complexity of your system starts to grow exponentially; maybe your *client*, *application host*, *data store*, *networking solution* are completely different pieces of infrastructure, made by different companies, using different programming languages and operating systems. A single *website* can contain code and data from thousands of different computers, distributed all over the world. The architecture and arrangement of these computers causes much chagrin among people who grew up in familiarity with traditional computing environments.

(evergreen)

NAS arrays, like Synology and others, illuminate an opportunity to bridge this gap, and it seems weird that it’s taken us until the decade of the 2020s for the pendulum of distributed computing to swing back to local environments again. This isn’t a trend that many are appreciating; most people in the *venture capital*, *cloud computing,* and *silicon valley* worlds are still moving in the opposite direction, and there remains a large swath of both SMB and enterprise computing systems that are stubbornly sticking to traditional {expensive, insecure, unreliable, unperformant} “bare metal” solutions.

A personal server

Just as Microsoft revolutionized the computing industry by introducing the *personal computer* in the 80s/90s, I suspect that some company (or companies) are going to make it big by popularizing the *personal server* in the 2020s*. It starts the way every computing movement starts; innovations with computer hardware, firmware, and software that work well together.

On the hardware front, technology leaders like Synology and Qnap seem to have really catalyzed the market for *usable*, *sysadmin-free* home computing/storage bundles. These are “set and forget” devices, requiring very little maintenance, function just as well as AWS/Gcloud/Azure storage, and have a variable cost of only the price of electricity. The up-front investment is really only a trade-off on the time-money spectrum; if you already understand the basics of how to run a simple computer (*), a home NAS device is increasingly a better option compared to the hassle (and many times, higher cost!) of keeping track of a slew of cloud services, server instances, and other trivialities of webshit.

This space has also seen a lot of innovation, given recent advances in chip manufacturing, board design, and hardware product marketing (driven, of course, by the South China Sea industry; Americans simply can't invent at this level anymore, for whatever reason). Helios64 seems to show a lot of promise, and even some enterprising RasPi hackers, enabled by developments like CM4's native PCIe support and Marvell SATA controllers, have produced some impressive, modular boards capable of handling many common NAS workloads.

Of course, the global hardware supply chain has been thrown into chaos this last year, for various geopolitical, epidemological, and cryptofinancial reasons (damn you, shitcoin miners 😠). The innovations are there, and manufacturing / distribution of boards and components will continue, but even with these constraints, you can still get a lot of the way with commodity hardware.

Infrastructure glue

There is innovation, too, in the "firmware" layer between bare metal and usable *personal server* applications. In terms of low-level storage hotness, architectures like ZFS and Minio have reconstituted chunks of what used to require complicated RAID setups, regular maintenance schedules, and a team of overworked IT guys. Hard drives themselves are getting better optimized for particular use cases (you can now buy surveillance-, NAS-, or gaming-optimized drives at your local computer supplier).

At the kernel level, special-purpose operating systems like FreeNAS and Unraid are increasingly attractive options. Of course, the standard Linux flavors continue to mature and develop (I've been running a Manjaro server in my office for the last 6 months, and am pleased to report I haven't had to touch it once). Containerization via Docker and other libraries has finally proliferated, enabling everyone from mass-market enterprises and hobbyist tinkerers alike to simplify server setup and make orchestration a breeze. (Some people also seem to like this "Kubernetes" thing, but it seems complicated to me).

Finally, networking has also seen some promising advances, with VPN architectures like Wireguard, overlay networks like Yggdrasil, and other P2P decentralization schemes (like Matrix and Urbit) have simplified much of the process of standing up a basic, secure, semi-private network to connect and share data with remote computers. Obviously, security itself is still a largely unsolved domain, with ransomware rocketing to a national security priority comparable to terrorism, but simple tools and encryption for highly-trusted networks now seem within reach for hobbyist players.

The future of applications

Software, of course, is where the rub is. Almost all current software applications in the suite of self-hosted server programs are developed and licensed as *open source* solutions, with all of the trade-offs that entails. Typically these are things like *poor usability*, *high learning curve*, *high*

maintenance / update burden, with only benefits being free cost & full customizability to the brave user who dares navigate the installation process (yet another time/money tradeoff).
On the proprietary / managed / commercial end, platforms like Nextcloud and Owncloud seem to be leading the charge in "on-prem" applications for small businesses / enterprises that prefer to avoid cloud services. Synology also gets marks here for making simple backup and storage services finally accessible to mass-market users.

Of course, there is a self-hosted, open-source version of every application out there imaginable, and most of them are typically plagued by usability and maintenance issues. You can check out a list here for instance, and there are great blogs, podcasts, and Youtube channels everywhere that stay up-to-date on the latest updates and developments (I've recently been enjoying Linux Unplugged). But, overall, these scattered FOSS applications always seem to do 70% of what the proprietary versions do, and open-source maintainers are perennially unable to keep up with the gigantic commercial players with huge budgets, particularly in supporting the latest, shiniest new devices…

[Sidenote; The reason is actually fairly simple, and you could probably describe it as some kind of iron law of open source usability; developers have a high *intrinsic* motivation to write best-in-class open source architectures, or "version 1" applications, but going through the whole { product design -> UX -> frontend -> devops -> bugfixing } process is something that very few people *intrinsically* like doing, and thus the "product process" -- polishing an actual interface for users deeply on the {have lots of money, have very little time} end of the time/money spectrum -- almost always needs a large monetary incentive to get done properly.]

Most close to my heart are full-blown decentralization platforms, like Urbit and Matrix, which aspire to standardize a set of modular application protocols, enabling a new era of full interoperability between webapps and services which have previously segmented themselves into vertical, inoperable walled gardens. The possibilities here are truly captivating, but these platforms still remain a long way away from maturation and mass adoption.

Death of the "user"

Software innovations are by far the costliest, trickiest, and least stable innovations in computing trends, which is why new computing paradigms are almost always driven by hardware / firmware-level advances. My own theory on this has to do with the concept of the "end-user". Heretofore in the history of computing, "applications" have been designed, implemented, and sold to "end-users"; and institutions, from companies, to universities, to governments, and to civilians themselves, have maintained a strict economic and social separation between *technologists* (mystical, genius programming and design wizards) and *non-technical users* (in the eyes of technology companies, dummies that don't know anything).

The major corporate software providers have a way of preying on this distinction; by keeping "users" as a separate class of uninformed, disempowered, and dependent consumers of technology, they maintain their position as leaders in delivering a polished "user experience"; by making system architecture bewilderingly complex, redundant, and incompatible, they ensure that the average civilian thinks of computers as some distant, wizard technology, that they can't possibly hope to comprehend, and are happy to fork over big chunks of their money and time to get access to.

My own bet is that this distinction fades over time; just as, during the advent of the printing press, access to printed tomes had to be mediated by a small, literate clerical class, the next generations of citizens will be expected to type on keyboards, install / upgrade software, swap out hard drives, etc. It seems almost inevitable that we'll develop a culture of ownership and maintenance of our digital tools the same way we do our houses, cars, and other technological advancements that have, over time, faded into the normal fabric of everyday life. And I'm willing to make a strong career investment that the decade of the 2020s will see this trend play out.

(* the term "personal server" was copped from Urbit marketing, although I'm sure it existed before that)