Why was Tanenbaum wrong in the Tanenbaum-Torvalds debates?

I was recently assigned reading from the Tanenbaum-Torvalds debates in my OS class. In the debates, Tanenbaum makes some predictions:

1. Microkernels are the future
2. x86 will die out and RISC architectures will dominate the market
3. (5 years from then) everyone will be running a free GNU OS

I was a one year old when the debates happened, so I lack historical intuition. Why have these predictions not panned out? It seems to me, that from Tanenbaum's perspective, they're pretty reasonable predictions of the future. What happened so that they didn't come to pass?

operating-systems kernel

15 Answers

up vote
159
down
vote
accepted

Microkernels are the future

I think Linus hit the points on monolithic kernels in his debate. Certainly some lessons learned from microkernel research was applied to monolithic kernels. Microsoft sometimes used to claim that the Win32 kernel was a microkernel architecture. It's a bit of a stretch when you look at some textbook microkernels, but the claims had some technical justification.

x86 will die out and RISC architectures will dominate the market

If you back up from desktops and servers, RISC dominates the processor market by any measure. ARM (R stands for RISC) outsells x86 in number of processors, there are more ARM processors than x86 processors in use, and there is more total ARM computing capacity than x86 computing capacity. This year, a single ARM vendor (yeah, Apple) may outsell all x86 vendors combined. Only in the desktop and server space does x86 dominate. So long as Windows is the dominant platform for desktop computers and Linux for servers, this is likely to continue to be true for a while.

There's a part *b* to this as well. Intel engineers did some amazing work to squeeze life out of their instruction set, even to the point of making a RISC core with an opcode translator that sits on top. Compare to one of the dominant RISC desktop chip makers, IBM, who could not get a power-efficient and high performance G5 for Apple laptops in a reasonable timeframe.

(5 years from then) everyone will be running a free GNU OS

I think the various OS vendors still offer compelling value propositions on their OSes. GNU isn't even necessarily the most important player in the Open Source community, so even a more widespread adoption of open source software didn't necessarily translate into GNU OSes. Yet, there's a whole lot of GNU stuff out there (all Macs ship with GNU's Bash, for example. There's probably some GNU system tools on Android phones). I think the computer ecosystem is a lot more diverse than Tannenbaum foresaw, even when you restrict your view to desktop computers.

up vote 58 down vote

Software Experts Ignored the Economics Of Hardware

...or "Moore was right and they were both wrong"

The biggest thing that was overlooked in this debate was the impact of CPU manufacturing technology and economics, driven by shrinking transistor sizes as expressed in Moore's Law (not surprising as though they knew a lot about CPU hardware, these guys studied and debated software, not CPU manufacturing or economics). Fixed manufacturing costs which are amortized over CPU's (e.g. ISA design, CPU design and CPU production facilities) have grown rapidly, thereby increasing the value of economies of scale; with per unit CPU costs (in terms of "bang for the buck" & "bang for the watt") plummeting, the cost of a CPU needn't be amortized over such a broad selection of functions to provide value, so computing in products with fixed function has exploded; CPU transistor budgets have grown exponentially, therefore wasting a fixed number of transistors due inefficiencies in ISA design is of no consequence.

1. CPU Scale Wins Over CPU Diversity

The importance of economies of scale has made the benefits of a ISA/CPU targeting a larger (therefore broader) market outweighs the potential benefits from design choices which narrow the market for an ISA/CPU. OS's can address larger and larger portions of the market per supported ISA/CPU, so there is little need (or even no need) for porting exercises to allow an OS ecosystem to thrive. The problem domains ISA's & CPU's target tend to be so broad that they mostly overlap, so for any software beyond a compiler, the size of porting exercises has also diminished. Arguably, **both Torvalds & Tanenbaum** overestimated the portion of kernel design and implementation that now needs to be ISA or even CPU specific. As Tanenbaum described, modern OS kernels do abstract out the distinctions between CPU's & ISA's. However, the CPU/ISA specific code in modern OS's is much smaller than a microkernel. Rather than implementing interrupt handling/scheduling, memory management, communication & I/O, these non-portable bits address only a tiny fraction of the implementation of those services, with the vast majority of the architecture of even these core OS functions being portable.

2. Open Source Won the Battle, But Lost the War

More bang for the buck means that a larger share of computing is performed by fixed function products, where the ability to modify the product is not part of the value proposition for the customer. Now ironically, open source has flourished in these fixed function devices, but more often than not, the benefits of those freedoms being realized more by those making the products rather than end users (which actually was true of the software market even back then: Microsoft was a big consumer of open source software, but their customers were not). Similarly, one could argue that open source has struggled more in the general purpose desktop space than anywhere else, but as the web and cloud computing has grown, desktop computing has increasingly been used for a narrower purpose (primarily running a browser), with the remaining functions running in the cloud (ironically, primarily on open source platforms). In short: open source does really own the general purpose computing space, but the market has become more sophisticated; computing product packaging less often stops at general purpose function, but continues along to product intended for fixed functions, where much of the advantage of open source computing is in conflict with the product goals.

3. 2^n Growth Means Fixed k Savings Are Not Important

The exponential growth of transistor budgets has brought with it the realization that the transistor budget cost of a CISC architecture is almost completely fixed. RISC's strategic advantage was that it moved complexity out of the CPU's instruction set and in to the compiler (no doubt partly motivated by the fact that compiler writers benefited far less from complex ISA's than human developers coding in assembly, but compilers could much more easily reason mathematically about, and therefore exploit, a simpler ISA); the resulting transistor savings could then be applied to improving CPU performance. The caveat was that the transistor budget savings from a simpler ISA was mostly fixed (and overhead in compiler design was mostly fixed too). While this fixed impact was a huge chunk of the budget back in the day, as one can

imagine it only takes a few rounds of exponential growth for the impact to become trivial. This rapidly declining impact combined with the afore mentioned rapidly increasing importance of the CPU monoculture meant a very small window of opportunity for any new ISA to establish itself. Even where new ISA's did succeed, modern "RISC" ISA's are are not the orthogonal ISA's described by the RISC strategy, as continued growth in transistor budgets and broader applicability of SIMD processing in particular has encouraged the adoption of new instructions tuned for specific functions.

4. Simple: Separation Of Concerns. Complex: Separation Of Address Space.

The modern Linux kernel (along with most other kernels) fits the rather loose definition of a macrokernel and not the rather than the narrow definition of a microkernel. That said, with its driver architecture, dynamically loaded modules and multiprocessing optimizations which make kernel space communications increasingly resemble a microkernel's message passing, it's structure more closely resembles a microkernel design (as embodied by Minix) than the macrokernel design (as embodied by Linux's design at the time of the discussion). Like a microkernel design, the Linux kernel does provide generalized communication, scheduling, interrupt handling, and memory management for all other OS components; its components do tend to have distinct code and data structures. While modules are dynamically loaded, loosely coupled pieces of portable code, that communicate through fixed interfaces, the don't employ one remaining property of microkernels: they aren't user space processes. In the end, Moore's Law ensured that issues motivated by hardware concerns like portability (a concern of Tanenbaum's) & performance (a concern of Torvalds') diminished, but software development issues became of paramount importance. The remaining unrealized advantages that a separation of address spaces could provide are outweighed by the additional baggage imposed on the OS software due to design limitations and increased complexity of component interfaces.

Interestingly, what **has** been a strong trend is the emergence of the hypervisor, which much like microkernels, abstracts out the hardware. Some claim that hypervisors are microkernels. Hypervisor architecture is different though, as responsibilities meant to be owned by microkernels are handled by the "guest" kernels sitting atop, with hypervisors multiplex between them, and the hypervisor abstraction is not generic messaging and memory address space, but predominantly actual hardware emulation.

In Conclusion: The Future Favours Those Who Adopt Least Strict Semantics

*..or "nitpickers suck at predicting the future"

In practice a lot of the rightness/wrongness in the debate is a matter of semantics (and that was part of what Torvalds was arguing and IMHO Tanenbaum failed to fully appreciate). It's hard to make precise definitions about the future because there are so many factors outside of the argument that can come in to play; looser semantics means your predictions are a larger target on the dartboard than the other guy's, giving you way better odds. If you ignore semantics, the arguments advanced by both Torvalds and Tanenbaum were right about a lot of things and wrong about very little.

tl;dr

Most ISA's don't fit the semantic definition of RISC, but harness most of the design advantages which were distinctive of RISC CPU's at the time; the amount of OS which is CPU specific is less than Tanenbaum expected, let alone Torvalds; open source does dominate general purpose computing, but the consumers of that market are now primarily those who package computing in to more fixed function products where much of the benefit of open source software isn't realized; separating out OS function across address spaces didn't prove to be beneficial, but separating out OS function across "virtual" hardware has. If you want to claim your predictions proved right, leave yourself as much semantic maneuvering room as possible, just like Mr. Torvalds.

P.S. A final ironic observation: Linus Torvalds is one of the strongest proponents of keeping as much new functionality as possible up in user space and out of the Linux kernel.

up vote 41 down vote

> - *Microkernels are the future*

He got that wrong, seems everything is converging into using hybrid kernels. Linux formally still is monolithic, but adding stuff like FUSE etc. make it look bit hybrid too.

> - *x86 will die out and RISC architectures will dominate the market*

Ok, so x86 didn't die out. But doesn't RISC dominate the market? Billions of smartphones using ARM, all gaming consoles using RISC processors, most network hardware using MIPS processors.

Besides, by early-2000s RISC and CISC have converged so much, that there were no clear cut differences in internal design. Modern x86 processors basically are internally RISC with CISC interface.

> - *(5 years from then) everyone will be running a free GNU OS*

If by GNU OS he meant GNU Hurd, then indeed a totally failed prediction. What people do massively use is Android. Android is Linux, however it is **not** GNU, as it doesn't use GNU libc. Instead it uses Google's own Bionic. And the market share of standard desktop Linux is still below 2%. But did he really mean consumer PCs? On server market Linux absolutely dominates with 70-90% of share depending on the segment.

---

up vote 21 down vote

1. Not sure.
2. Half-right. Today's "x86" chips *are* RISC under the hood, with basically a "CISC interface". x86 didn't die out because Intel had enough market share and enough revenue to make that transition and get it right before other RISC solutions captured significant market share from them.
3. Two main reasons: compatibility and usability.

   Again, the existing systems (Windows and, to a lesser extent Mac OS) have a very large installed base. That means lots of users using lots of programs. Free GNU OSes can't duplicate that. The WINE project has done a lot of work in that direction, but it's still no substitute for an actual Windows system, and the WINE developers don't even try to claim that it is. People don't want to use an OS that won't run their favorite programs, no matter how theoretically awesome it is. (And without an installed user base, no one wants to develop for the system. It's a chicken-and-egg problem.)

   And then we get to usability. My mom has a Windows system. She can use it just fine for her purposes. Everything she needs to work with her computer is available from the Windows interface, and if I said the words "command line" to her, she wouldn't even know what I'm talking about. As far as I know, it's **still** not possible to do that on any free GNU OS. In fact, Linux is so hard to work with that even the greatest of the community demigods have serious troubles with simple tasks sometimes. And they never seem to "get it" and work to fix the issues, which is why they never gain market share. (The linked article should be required reading for anyone who ever tries to produce any mass-market program!)

---

up vote 13 down vote

I think there are a couple of reasons that are pretty serious, but haven't been mentioned.

The first is Tanenbaum's rather blind assumption that technical superiority would lead to market dominance. People have argued for years about whether microkernels (nanokernels, picokernels, etc.) are technically superior, but for the moment, let's just assume they are. We're still left with a question of whether that technical superiority is likely to translate to market dominance. I will posit that it doesn't. For most people, Windows, Mac OS, etc., are good enough. Worse,the improvements that would make a significant difference to most users would be in the user interface, not the kernel.

The second is drastically over-estimating the *rate* of change in a (reasonably) mature market. It's pretty easy to change things in a hurry when you have a new market. To get essentially everybody changed over in five years, the migration would had to have been happening at full speed even as he predicted it.

I'd also note another way in which he was right that I haven't seen mentioned. People have already noted the ubiquity of RISC (e.g., in cell phones). What they haven't mentioned is the ubiquity of what I'd call "microkernel 2.0". This is now more often known as a "virtual machine" or "hypervisor". They really are pretty much microkernels though.

The big difference is that Tanenbaum thought in terms of a microkernel and a user-mode OS emulation both designed specifically for each other. Instead, we've kept the OS essentially unchanged, and tailored the microkernel to run it as-is. This isn't as nice technically, but from a market perspective, it's dramatically superior -- instead of a whole new system top to bottom, the user can continue to use most of his existing code as-is, and just add a cool new "utility" that happens to really be a microkernel OS.

answered Mar 22 '12 at 18:29

Jerry Coffin

---

up vote 8 down vote

One big reason was Windows, especially Windows 3.1 and a little later, Windows 95 and NT 3.51. Consumers particularly loved the GUI interfaces as opposed to the old text based systems of Unix and DOS. This meant more average people would purchase computers for home use. Also, the explosion of the Internet in the mid-90's increased sales.

Prices for PC's also dropped throughout the 90's until they reached the point where they are today. This was due to the economies of scale presented by an increased consumer and business demand. For example, all five of my current computers cost less combined than the one 486 desktop I purchased in 1992.

Now, in a way he might be right but in an unexpected direction. The rise of mobile devices, smartphones and tablets, have partially brought about simplified operating systems and may reduce the prominence of x86. However, they go far beyond what was predicted in 1992.

answered Mar 21 '12 at 23:19

jfrankcarr

---

up vote 6 down vote

Ultimately it all comes down to the fact that things don't really like to change.

We didn't migrate to a better-architected microkernel because monolithic ones were easier to create, ran faster, and everyone knew how to build them. Also because Linux was developed as a monolithic kernel and became popular, there were no microkernels that achieved enough success to take off. (its a bit like the same reason we all run Windows, we run Windows because everybody runs Windows)

RISC, others have pointed out that x86 is pretty much a RISC architecture nowadays, with a CISC wrapper on top for backwards compatibility.

A lot of people are running a free GNU OS - on the server. The web is pretty much driven by it. Nobody notices because all the clients are Windows. Back in those days, you had a choice: Linux that was still very much a hobby OS; a flavour of Unix but you couldn't afford to buy it; or cheap n cheerful Windows. Linux took too long to replace Unix, and still doesn't quite have enough of a compelling solution to running on the desktop (partly because of ideological problems with different Window systems, binary graphics drivers, and the lack of a stable ABI). It's doing rather well in other non-desktop markets like embedded and mobile however.

answered Mar 21 '12 at 23:48
gbjbaanb

---

up vote 6 down vote

All are true if you don't think that a computer is something on your desktop.

1. True - microkernels never worked because they were never micro enough. If the whole of your stripped down embedded linux is smaller than the x86 specific bit of the MACH kernel is the question of micro-kernels relevent?
2. RISC is dominating the market. More ARM cpus are sold each year than X86 cpus ever. You are probably never more than 6ft from an ARM cpu.
3. Almost everybody is running linux, it's in their router, their TV setop box, their Tivo and their Android phone - they just don't know that these have an OS

answered Mar 22 '12 at 0:00
Martin Beckett

---

up vote 4 down vote

1) He was wrong on the microkernels. To my understanding the need for speed requirement trumps the separation of concerns enforced in microkernels, at least in the Linux kernel.

2) The predominant architecture in tablets and mobile phones is ARM which is a RISC instruction set. Even Windows has been ported.

3) Everyone does not run a free GNU OS. This is primarily because of patents and backward compatibility. Those not wanting Windows frequently choose OS X.

answered Mar 21 '12 at 23:17
user1249

---

up vote 2 down vote

1. Microkernels replace method calls with inter-process messaging, which adds development complexity for some hypothetical benefits. As it turns out for the most part you can get just about the same advantages from good componentization even if everything is living in one big process.
2. The question is no longer relevant. CISC architectures no longer exist, all modern processors are RISC in their hearts, but that didn't kill the x86 instruction set. x86 processors since the Pentium-Pro era (17 years ago) use op-code translation to allow an essentially RISC core to look like an x86 CPU from the outside.
3. Classic Worse is Better. Iteration, pragmatism, network and ecosystem effects beat purity every time.

answered Mar 23 '12 at 0:03

Wedge

up vote 2 down vote

1. Microkernels failed

For the reasons that Linus Torvalds stated, that on paper it looks theoretically attractive but in implementation on modern systems - which are very complex systems - the complexity becomes exponentially unmanageable. Case study is GNU Hurd, a fully microkernel system which failed to even achieve basic functions. Mac OS X is similar to Hurd in structure and it's the least stable and most constrained OS out there.

2. CPU architecture

This has become diversified for various use cases. One CPU architecture did not dominate because embedded, mobile, desktop, server and so forth use cases are different and required different approaches. Tannenbaum failed to see this diversification.

3. GNU vs World?

GNU did not dominate, but Linux did on server, embedded and mobile. Apple tablets and phones run iOS which is just plain old Unix. Accurate statistics are difficult to acquire for Linux deployments on desktop because there is no real core mechanism - sale of units - that can surely give an accurate value. Most Linux deployments on the desktop are sometimes recorded as Windows deployments because users buy a Windows system and then write over it with Linux. However, if you segment OS's then Linux has around 5-6% on the desktop according to http://www.w3schools.com/browsers/browsers_os.asp and this is equal to the number of Windows Vista users worldwide which is highly significant.

Based on my own estimations from various sources it seems that Linux on the desktop could actually be equal to the number of users on Windows XP - roughly 25% - if you count non-Western nations like China and India where Linux is more popular than in the USA or EU but who might not be counted in Western statistics because they only count traffic to English speaking websites aimed at Westerners.

In India most college students use Ubuntu or Fedora because this is the default OS of the Indian education systems and at the famous IIT's. Most Indian government offices use Linux also. In China Red Flag Linux is the offical OS of the Chinese government and school systems - Academies of Arts and Sciences - and is the recommended OS in China by the state run media as an effort to stop young impoverished Chinese from using pirated copies of Windows. If you counted the useage of Linux in India and China it would shock most Western tech experts and radically change the perceptions of the true dominance of Linux desktop in non-Western developing nations where it is dominant.

answered Jun 3 '12 at 8:09
AG Restringere

---

up vote 1 down vote

Production became cheaper, x86 came so close to the price of RISC, that it wasn't feasible anymore to use it. There was also a small vendor lock-in.

answered Mar 21 '12 at 23:04
Lucas Kauffman

---

up vote 1 down vote

For 2: The CISIC instruction set has a big advantage: It's more compact, like compressed machine code. Nowadays it's very cheap to decode CISC instructions into micro-ops and it's very expensive to access the RAM. So CISC has the advantage of pushing more code in the L1/L2/L3 caches

answered Mar 24 '12 at 10:34

Christian

---

up vote 1 down vote

> *Microkernels are the future x86 will die out and RISC architectures will dominate the market (5 years from then) everyone will be running a free GNU OS*

It depends how you view and define future, in the traditional sense his predictions have failed.

However the time has not yet ended *(another deeper discussion aside)*.

Thus, things still could change:

1. Microkernels or some variant could make a comeback
2. RISC/ARM may well dominate -> tablets / mobiles
3. 10 or 15 years from now. Who knows, open source is changing the world slowly..

---

up vote 0 down vote

I remember the time -- and the time that preceded it. Dunno about microkernals, but

2) The idea of RISC had two legs: that software optimizations could be done in software better than in the hardware, and that RISC chips could economically be made that were faster than CISC chips.

Both ideas turned out to be false in the short term. Intel could, and did, make CISC chips that clocked instructions faster than the competing RISC chips, at a competitive price. Intel could, and did, make CISC chips that did program optimization better in hardware than could be done in the compiler, or in the software runtime supervisor --and any software optimization could be added on top of that, just as it would with a RISC chip.

3) Computer Science, Programming, and Operations, have been completely re-invented 4 times in my career. From main frame to PC. From command line to GUI. From GUI to internet. From Internet to iPad. Revolution seems normal now, but WE DID NOT PREDICT THAT. Like all older programmers at the time, he was predicting the 'end of history'.

Very few people were running a GNU OS in five years, because the count restarted.

Perhaps it is still happening. 5 years ago you would have predicted that our Windows Server would be replaced by *nix servers (As I write, SAMBA has released an AD Domain Server, which was the missing piece of the puzzle). It's not going to happen where I work: we won't have any local servers.

answered Dec 19 '12 at 1:11

david

Your Answer

Sign up or log in

Sign up using Google

Sign up using Facebook

Sign up using Email and Password

Post as a guest

By posting your answer, you agree to the privacy policy and terms of service.

Not the answer you're looking for? Browse other questions tagged operating-systems kernel or ask your own question.