The State of eBPF

In partnership with

January 2024

The State of eBPF

The 30 million lines of code in the Linux kernel **enables significant functionality,** but adding new functionality can take years.

eBPF has evolved far beyond packet filtering to become a **general purpose computing machine inside the kernel.**

With eBPF, **engineers can quickly build custom programs in the kernel** without the whole community having to accept the change.

In tandem with the needs of cloud native workloads, eBPF **supports capabilities, increases performance, and encourages simplicity.**

eBPF enables **observability** of the Linux system, rewriting or bypassing parts of the **networking stack, and faster vulnerability fixes.**

Big tech companies, including **Google, Meta, and Netflix,** have leveraged eBPF in their data centers for years.

Many applications already use eBPF enabling continuous profiling, monitoring servers, observability platforms, and performance monitoring tools.

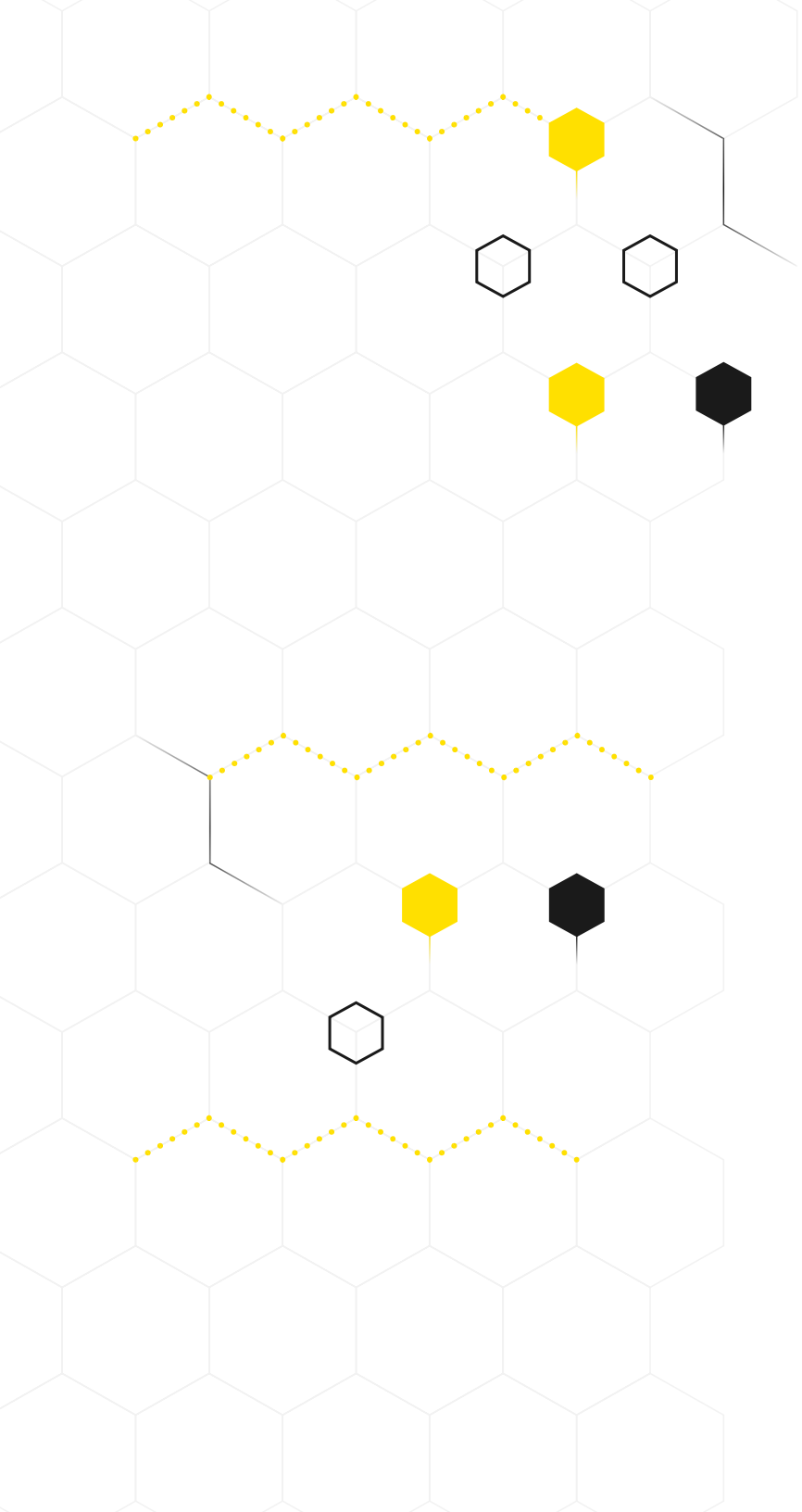
Innovation is at the heart of eBPF, creating an iteration cycle that is faster, safer, and allows greater flexibility.

A verifier and JIT compiler provide **safety and performance benefits** in eBPF deployments.

Other challenges to eBPF include the **performance-features tradeoff, co-existence and interoperability of tools,** and the kernel expertise needed to write programs.

The eBPF Foundation and steering committee provide **technical direction and optimize collaboration** on the technology's roadmap.

Standardization is in the future of the eBPF ecosystem, around instruction sets for all operating environments because it is becoming a critical layer in the cloud native infrastructure stack.



Contents

Introduction

In just three decades, the Linux operating system has grown from inception into the dominant computing operating system. Billions of machines **run on Linux**, including the Android OS, 93% of the world's top 1 million web servers, IOT devices, systems in cars, TVs, smartwatches, the world's top 500 fastest supercomputers, on and on.

This breadth of deployment has made it more difficult to change the kernel with users advocating for stability over cutting edge features. But now a rapidly expanding technology has revolutionized the cycle of innovation by making the Linux kernel programmable. This means the behavior of Linux can be modified in real time without risky or expensive changes to the kernel.

eBPF is that technology.

eBPF allows users to run custom programs inside the Linux kernel, which changes the behavior of the kernel and makes execution **up to 10x faster** and more efficient for key parts of what makes our computing lives work. That includes observability, which enables engineers to see where a system is going wrong and find fixes faster, networking, which involves everything from how fast emails move to how fast computation occurs, to security, which keeps our digital lives and infrastructure safer from cyber threats.

"eBPF provides flexibility to previously fixed areas of computing, powering new technologies for observability, security, and networking," states Brendan Gregg, who started working on eBPF while at Netflix, is now an Intel fellow and who is oft-quoted as saying eBPF brings "superpowers' ' to the Linux kernel.

The innovation of eBPF also means companies need less hardware to achieve better performance and they consume less power to perform the same functionality. That makes operations more cost efficient, energy efficient and sustainable, which is increasingly required to meet shareholder, consumer, and community expectations.

In the decade since eBPF's launch, the big use cases have emerged as observability, networking and security. But in the past year, Meta has delivered big wins with new eBPF-based central processing unit schedulers that resulted in 5% CPU bandwidth gains on some of Meta's largest applications. "That's an enormous number because it is basically equivalent to us having 5% more CPUs in our fleet," says Dan Kelley, director of software engineering at Meta. "We're right on the precipice of radically increasing what you can do with eBPF. We can iterate so much more rapidly on new CPU schedulers, and those big wins in production and performance turn into dollars that we don't have to spend on computers. This is one of the more exciting developments we've had in the last four or five years just in terms of the kernel projects that Meta does."

New eBPF-based central processing unit schedulers that resulted in 5% CPU bandwidth gains on some of Meta's largest applications.

eBPF was Linux only until recently. In 2021, **Microsoft** created the eBPF for Windows project to allow eBPF programs to run on top of the Windows OS. This laid the groundwork for eBPF to be standardized as an industry-wide infrastructure language, Graf says. With a unified underlying infrastructure, companies can innovate however they want on top without risk of becoming locked in to one OS or the other.

This lack of vendor lock-in —from the browser to the database to the cloud—has historically spurred increased innovation, competition in terms of cost and performance, and is a bedrock tenet of the open source ethos that drives both Linux and eBPF.

What's more, developers from either world, Linux or Windows, can write kernel extensions "once and they'll run on any platform that supports it," says Alan Jowett, Microsoft principal software design engineer. "This means more applications get to market faster and they have a larger addressable market."

Somebody "can have an idea, write an (eBPF) program, and get it in the hands of an end user," often a company or organization, within days, states Thomas Graf, CEO of Isovalent and a pioneer in the eBPF ecosystem, speaking in the documentary, "***Unlocking the Kernel***," which traces eBPF's development and growth.

The addition of Microsoft to the eBPF ecosystem was "the last big milestone," Graf says, in its establishment as a technology that "will change the industry forever."

eBPF, the Evolution

Attempts have been made before to put "virtual machines" like eBPF inside the Linux kernel and they'd always been denied for fear that they'd disrupt too much.

Instead, developers found workarounds, mostly in the form of kernel modules, which are risky because if there's a bug in the module the whole kernel can crash.

Another option has been to ask for an actual change to the kernel, which, as stated before, can take years to reach a support release in end user environments. Using tools from decades ago or switching context outsides the kernel takes time and processing power.

Then an engineer, Alexi Starovoitov, started working with Daniel Borkmann in a new way. Rather than try to insert an entirely new virtual machine inside the Linux kernel, he looked at what was already there: a virtual machine to move packet filters, known as BPF, or Berkeley Packet Filter. They extended the capabilities of BPF "lego brick by lego brick" until it extended far beyond just networking to become a general purpose computing machine inside the kernel.

This report covers the evolution of eBPF, the revolution it created, what's being built with it today, challenges, and where it is heading.

Source: x.com/breakawaybilly

eBPF Opens the "Black Box"

The kernel of any operating system is the software that directly interfaces with hardware and interacts with machine components from device drivers to user applications. The kernel is involved in all of the interesting things that applications do. If an application wants to send a network message, or allocate memory, or do any number of other things, it goes via the kernel to involve the hardware. The Linux kernel has **30 million lines of code** and making a change can take months or even years just to get it merged.

eBPF is like a virtual machine in the Linux kernel. With eBPF, a developer writes eBPF instructions to run small specialized programs. They go to an eBPF "verifier," which checks to make sure the program is safe to add to the kernel and won't introduce bugs or crash the kernel. The program is JIT-compiled into machine code that gets executed and attached to event targets, which means the program is activated by an event, such as a file opening.

eBPF catalyzes next-generation cloud native workloads because it expands a platform's capabilities, increases performance, and reduces complexity.

"eBPF is a big deal in the sense that it allows you to do things that were either really difficult or practically impossible before," says Toke Høiland-Jørgensen, a Senior Principal Kernel Engineer at Red Hat. "The power of eBPF comes from this generality, the fact that we don't specify a functionally. We allow you to implement your own functionality in certain places and you can do whatever you need."

eBPF Grows with Cloud Native, and Vice Versa

Meta has eBPF-based schedulers in production now, and Google plans to start testing their own eBPF-based schedulers in early 2024, Kelley says. This underscores that, even though eBPF is already widely used by companies in all industries, it is still at the early stages of its impact.

And it is perfectly timed for today's computing world, which is increasingly cloud native. By 2025, "over 95% of new digital workloads will be deployed on cloud native platforms," **Gartner** estimates. That's up from 30% in 2021.

The cloud native development approach enables companies to more efficiently and reliably build and deliver applications. eBPF catalyzes next-generation cloud native workloads because it expands a platform's capabilities, increases performance, and reduces complexity.

The Linux and Kubernetes combination is the de facto cloud OS and the foundation for cloud native development. It's a solid foundation, but Kubernetes is notoriously complex. And, it can take years to make changes to the Linux kernel — more time than most organizations can spare. These factors slow organizations' ability to innovate. That is why eBPF is being so widely and quickly embraced in the cloud native world.

"Companies are moving their software to the cloud and they want observability tools, networking tools, and security tools to help them do this. eBPF is a great platform for people to create these tools on and that makes it a very disruptive infrastructure technology," says Liz Rice, chief open source officer at eBPF pioneer, Isovalent, and author of the O'Reilly report, "**What is eBPF**?" With eBPF sitting in the kernel, "we can have these incredibly powerful tools that can observe and affect what is happening across the whole machine. That's why eBPF has taken off in the cloud native world," Rice says.

The Big Use Cases: Observability, Networking, Security

For more than five years, eBPF has been operating on millions of devices and servers worldwide. Most people are already impacted by what companies do with it—and they probably don't know it.

Many of the US hyperscalers—Meta, Google, Netflix–use eBPF in production. Every Android phone uses eBPF to monitor traffic. Every single packet that goes in and out of a Meta datacenter is touched by eBPF. Companies in a myriad of industries, including software, cloud services, financial services, telco, media and entertainment, ecommerce, consulting, and security, are increasingly using eBPF technology to do more, faster, saving time and money and increasing performance. Here's a breakdown of the three major use cases, so far, for eBPF in production:

Observability

For many companies, the arena of observability is where eBPF first took off and has had its biggest impact. Graf compares observability in computing to trying to clean a room with the lights off. Turn on the lights and the cleaning goes much faster with better results.

eBPF observability tools turn on the lights.

"For people who are not kernel developers, the Linux kernel is like a black box," Høiland-Jørgensen says. "eBPF opens that black box and allows you to gain information on how the system is working that you couldn't get before."

With greater observability in distributed systems that might involve tens, hundreds, or thousands of servers, companies can more easily and fully know where the system is spending its time, where the bottlenecks are occurring, how fast the CPUs, or central processing units that run the kernel and the applications, are working, where they're spinning cycles, and to find, more quickly, what piece of code may be malfunctioning. If a company deploys a huge cluster of servers, numbering hundreds or thousands, and something goes wrong, it can take days, weeks, or even months to figure out what is causing the problem.

Many of the US hyperscalers—Meta, Google, Netflix–use eBPF in production. Every Android phone uses eBPF to monitor traffic. Every single packet that goes in and out of a Meta datacenter is touched by eBPF.

Gregg has authored many of today's leading eBPF-based observability tools and dozens of advanced eBPF tracing tools to shorten that cycle down to days, hours, or even minutes and to provide unique insights into system behavior, including analyzing CPUs, memory, disks, file systems, networking, languages, applications, and more. By attaching eBPF programs to events like a file opening, users get metrics that provide amazing visibility into what's happening in the system.

Companies have long had to make choices around observability because achieving certain levels was cost prohibitive due to performance or instrumentation bottle necks. Thanks to Gregg's work and efforts from others, eBPF makes greater observability more efficient and easier to add, making it possible to build tools that enable much greater visibility into root-causing errors.

"Brendan's work with eBPF disrupted observability. All the stuff he built at Netflix became mainstream and his work has led us to so many tools," says Shweta Saraf, Director of Platform Networking at Netflix.

Today, Netflix ingests and enriches billions of eBPF "flow logs" per hour to get visibility in its cloud ecosystem. The enriched data enables Netflix to analyze networks for availability, performance, and security to ensure applications can deliver their data across a globally dispersed cloud-based ecosystem. eBPF makes it "extremely efficient," Saraf adds.

Netflix ingests and enriches billions of eBPF "flow logs" per hour to get visibility in its cloud ecosystem. The enriched data enables Netflix to analyze networks for availability, performance, and security to ensure applications can deliver their data across a globally dispersed cloud-based ecosystem.

Networking

The arena of networking is a great example of how eBPF adds speed and performance.

Many parts of the Linux networking stack were written decades ago when IPs and port ranges could be tracked on spreadsheets rather than changing with every container. eBPF enables programmers to rewrite the networking stack, only leverage the needed parts, or skip it completely to save time and processing power. By bypassing things that are not needed or rewriting functionality based on new methods of building software, networking performance dramatically improves.

Cilium's Layer 4 **load balancer XDP** doubled throughput and reduced CPU consumption by 72x for Seznam.cz. Meta **open sourced** Katran in 2018 and every packet going into or out of a datacenter is processed by eBPF because of the performance benefits it brings. In 2022 Walmart open sourced the L3AF project, which provides Kernel Function as a Service using eBPF, creating a marketplace for eBPF programs, where users and developers can share their own signed eBPF programs and download eBPF programs from others. iptables is being **replaced** by the kernel community and eBPF bringing networking to near line rate.

Security

The enhanced observability that eBPF enables drives a better ability to spot and prevent security attacks, including those within the kernel as well as throughout Kubernetes and cloud native environments.

"You cannot secure what you cannot see," says Amit Gupta, chief product officer at **Tigera**, which provides a security platform to prevent, detect, and mitigate security breaches in cloud native applications. "The first step in any kind of secure architecture is to get full, secure visibility. eBPF helps create that visibility. When eBPF data comes back to user space, you can then coordinate with everything else that's happening … and make a decision on whether it is likely to be malicious activity," Gupta says. Instead of taking months to ferret out attackers or even missing them completely, eBPF tools can enable action "within seconds," Gupta says.

eBPF also pushes security enforcement policies into distributed environments so that they get implemented in real time. If a vulnerability occurs in the kernel, for instance, fast fixes can occur via eBPF without altering the kernel code, allowing for security updates on the fly.

Because of those capabilities, "eBPF will play an outsized role in securing the software supply chain," Gupta adds.

Some say "software is eating the world," I would say that: "eBPF is eating the software," states a **blog** from Cloudflare, a global cloud platform, that has used eBPF to build DDoS mitigation, as well as other products.

eBPF can also be written so that the same program can run on multiple versions of the Linux kernel. This is important because companies adopt newer kernel versions at different rates. Because eBPF can run on multiple versions, companies will be more likely to adopt newer security features because they know they will be interoperable with older kernel versions.

If a vulnerability occurs in the kernel, for instance, fast fixes can occur via eBPF without altering the kernel code, allowing for security updates on the fly.

The Spread of eBPF

Early on, eBPF was a technology only consumable by Linux developers because it was so complicated and so few people have expertise with the Linux kernel.

Then the hyperscalers —Google, Facebook, Netflix, etc., — jumped in. They had the engineering might to develop the technology, as well as the expertise and control of their systems to use it, roll it out, and make improvements. They've also led the charge with eBPF in production, meaning it probably touches the lives of almost every digital consumer today.

Since 2017, Meta has processed every packet going into its data centers with eBPF, as does Google for most of its data center traffic. Meta has documented its wins with improved **CPU scheduler efficiency**. Google uses eBPF both for **security monitoring and enforcement**. Every networking packet in the **Android smartphone** system hits eBPF, reports Isovalent's Bill Mulligan and Daniel Borkman in **InfoQ**. All of the traffic that goes to Netflix, from over 200 million customers, and every server "goes through some eBPF somewhere," says Arthur Gonigberg, Staff Software Engineer, Platform Networking at Netflix.

FIGURE 1

EBPF FOUNDATION MEMBERS

PLATINUM

CROWDSTRIKE
Google
HUAWEI
intel
ISOVALENT
Meta
Microsoft
Red Hat

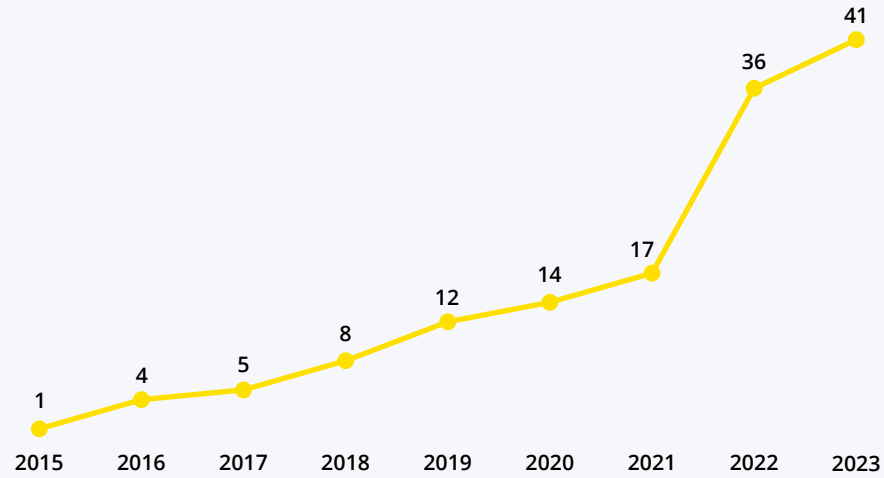
SILVER

DATADOG
FUTUREWEI
NETFLIX

Source: eBPF Foundation

FIGURE 2

ACTIVE EBPF LANDSCAPE PROJECTS BY YEAR



Source: GitHub

eBPF for the Rest of Us

Hyperscalers and big companies have what most companies do not have: teams of software engineers. To spread eBPF into more enterprises, open source software projects arose to make the technology more consumable and out-of-the box.

The eBPF-powered Cilium leads all others. Founded in 2016, Cilium initially focused on networking and then expanded to eBPF-powered connectivity, observability and security. Cilium is among the fastest-growing projects in the Cloud Native Computing Foundation, alongside Kubernetes, Envoy, and Prometheus.

More than 100 companies have publicly said they adopted **Cilium** and the project powers some of the largest Kubernetes clusters in the world. End users range from startups to leading financial institutions and telcos and companies such as The New York Times, Bloomberg, and Bell Canada.

Cilium, which Isovalent was founded to develop, builds the solutions that enterprises need in cloud native environments and it builds features to connect Kubernetes to existing legacy infrastructure, some of which will never be cloud native. What's more, major cloud providers integrate Cilium into their Kubernetes solutions, making it the "default" in the entire cloud native ecosystem.

"AWS uses Cilium to provide networking and security capabilities behind the scenes of its EKS Anywhere on-premises Kubernetes service; Google Cloud uses it to power the networking component of Google Kubernetes Engine," reports **The Stack**. S&P Global's network engineers leveraged eBPF-based networking with Cilium to build "a multi-cloud connectivity superhighway".

"A Goldman Sachs will not use eBPF directly. It will buy an eBPF-based solution," like Cilium, says Graf.

Red Hat has also been one of the early adopters of eBPF, enabling and supporting the technology in its enterprise Linux distribution from an early stage. Its customers run Red Hat Enterprise Linux (RHEL) on a wide variety of platforms. One of the early uses for eBPF in RHEL has been the **express Data Path**, or XDP, subsystem, which makes it possible to integrate flexible programmable network packet processing directly into the Linux kernel data path, with very high performance. XDP has an eBPF hook that runs as soon as the server processes a data packet, allowing eBPF to look at the packet and send it where it is supposed to go faster than if it was just the Linux kernel working on its own. The eBPF program does whatever it is programmed to do, which may simply be nothing more than dropping the packet to protect against DDoS attacks. Red Hat also developed a library for running multiple XDP programs one after the other. So, if one vendor gives a user a firewall, and another vendor gives the same customer a packet logger, the customer needs to know how to run them together. The libXDP library makes sure they work correctly, together.

eBPF Applications in Production, Development

Other **major eBPF applications** include:

- **Bcc**, a toolkit and library for efficient eBPF-kernel tracing.
- **Bpftrace**, a high-level tracing language for Linux eBPF.
- **Falco**, a behavioral activity monitor designed to detect anomalous activity in applications. With the help of eBPF Falco "audits a system at the Linux kernel layer."
- **Katran**, a high performance layer 4 load balancer.
- **Pixie**, an open source observability tool for Kubernetes applications that uses eBPF to automatically capture telemetry data.
- **Calico**, a pluggable eBPF-based networking and security for containers and Kubernetes.
- **Tetragon**, which provides eBPF-based transparent security observability combined with real-time runtime enforcement.

Almost three dozen other emerging applications are in the works, as listed by the eBPF Foundation **website**. These applications enable things such as continuous profiling, detection and filtering of events to detect suspicious behavior, monitoring servers, observability platforms, performance monitoring tools, platforms for cloud native workloads, load balancers, tools to debug and inspect Kubernetes resources, service maps, enforcement systems, and more. eBPF also has libraries written in Rust, Golang, and C++ to help debug, load, and compile eBPF programs.

Innovating in Direct and Indirect Ways

eBPF improves organizations' ability to innovate in many direct and indirect ways.

Most directly, eBPF provides several performance boosts that will accelerate the development lifecycle as in, for example, Meta's work with CPU scheduling. "What is absolutely magical about eBPF, relative to normal kernel development is, in normal kernel development, you have to come up with a hypothesis, build a new kernel, reboot the computer, test the hypothesis. That is slow," Meta's Kelley explains. "The faster you can go from having an idea to having test results basically determines how quickly you can execute a big idea and how big of an idea you can reasonably go after. eBPF is really shrinking all of that. The iteration cycle is much faster and you don't have to be as smart because you can run a lot more experiments. Being able to do that safely, with a short iteration time, hastens the process of discovery."

What's more, eBPF's decoupling from Linux kernel releases means that developers don't have to wait months or years before a new feature they need makes its way to the release of a major distribution. Decoupling from Linux kernel releases also helps prevent "feature creep" — and the performance hits that come with it — in the Linux kernel itself.

In the same vein, eBPF also decouples the production feedback loop from the Linux kernel itself by bypassing the time-consuming process of patching the kernel and then deploying changes gradually to the fleet. Instead, changes can be made and tested on the fly. eBPF can also leverage what it needs in the userspace stack, making small changes instead of rewriting large parts of the stack.

If all of this sounds a lot like the process of breaking out parts of application monoliths into microservices, it's no coincidence. Indeed, eBPF aligns with some of the most important computing paradigms of the last several years. Case in point: Just as edge computing brings computation and data storage resources closer to the point of data generation, eBPF moves data processing from the packet level to as close to the event source — such as specific socket hooks or XDP (eXpress Data Path) — as possible. This reduces resource consumption and improves the efficiency of data processing. Organizations can further improve networking, observability, and security efficiency and performance by using eBPF.

eBPF's decoupling from Linux kernel releases means that developers don't have to wait months or years before a new feature they need makes its way to the release of a major distribution

eBPF can also be written so that the same program can run on multiple versions of the Linux kernel. That also spurs faster innovation. For instance, an entity like Meta, Google, or Amazon has millions of computers, not all of which are running the exact same version of the kernel because it takes a long time to upgrade a whole fleet of computers. As such, they're always running multiple versions of the OS at the same time. If one had to specialize each eBPF program for multiple versions of software running on the fleet, that would end up being a big tax. "Being able to run on different kernel versions is another thing that speeds up development iteration velocity," Meta's Kelley explains.

More indirectly, the guardrails that eBPF puts into place will help developers focus on innovation and constant iteration rather than toilsome (but critical) management, observability, and security tasks.

For example, the time developers spend looking for and fixing bugs is time they are not spending on building applications and services that will boost the bottom line. By using eBPF programs to trace network packets and other events in real time, organizations gain deep visibility into system behavior and can respond to issues quickly. If a packet still gets lost, for instance, there's now an eBPF-based open source tool dubbed, **Pwruc**, for "packet where are you?' to trace packets in the Linux kernel to hasten "debugging network connectivity issues."

Most importantly, eBPF protects the kernel, which then ultimately drives enhanced innovation.

The eBPF **verifier** for Linux ensures the safety of eBPF programs with a two-step process. It first does a check to disallow loops and other config validation, then it simulates every instruction execution and observes the state change of registers and the stack.

If something passes muster with the verifier, which is no easy task, Gregg says, then everyone has more confidence in new eBPF-based products. Startups will be more likely to convince bigger companies, like Netflix, to use their products if they are eBPF-based.

The same is true on the Windows side. It uses a different verifier than the one for Linux. The Windows verifier checks the eBPF code offline first and requires that any code be provably safe and signed by someone Microsoft considers safe, which helps to ensure provenance of the code.

Because traditional Windows drivers are developed outside of the Windows code base by third parties who, oftentimes, don't have as in-depth knowledge of Windows as do Microsoft developers, stability has sometimes been an issue with drivers.

eBPF protects the kernel, which then ultimately drives enhanced innovation.

eBPF for Windows, given the verifier, provides another security check that can catch some of those issues and "third parties can more easily provide extensions to the Windows OS that are provably safe and reliable," Microsoft's Jowett says. "We can significantly improve the ecosystem experience of Windows by making sure that Windows drivers and other kernel extensions are provably safe."

"Fast by Friday"

Technology often makes things go faster and be more efficient. This helps the environment if less hardware is needed and less power is consumed. It spurs innovation by making tasks faster to complete. It helps companies contain compute costs and it benefits end users whose websites load faster.

But just because newer generations of hardware and software promise better performance, the potential doesn't always get realized, noted Gregg in a **keynote** at the eBPF Summit, 2023. Instead, bottlenecks remain mysteries for too long and slow things up. New software and hardware options are not always assessed because engineers lack time to do so. As computers get ever more complex, the challenge is to enable quicker resolution of all computing issues.

"eBPF will be essential," to "solve performance of anything," and enable engineers to fix any software performance issue—and to do it in production, "immediately," Gregg says.

eBPF tools will help with CPU flame graphs to see where CPUs are blocked, drill into latency issues, create more efficient tools and, eventually, allow zero instrumentation. He's called on the industry to set a standard to fix any computing issue within five days, not weeks or months, thus his coining of the term, "fast by Friday."

If issues get fixed faster, innovation happens faster, too.

Not Magic Fairy Dust

Despite its many virtues, eBPF is not "magic fairy dust that allows you to do everything," Red Hat's Høiland-Jørgensen says. "We are still bound by the laws of physics. A CPU can still only do a certain number of operations per second."

One of the biggest challenges with eBPF, as with almost any software program, is security. eBPF enables greater security because it enables greater observability. But any program also can introduce security challenges. If one is going to run code in the kernel space, it's going to have access to a lot of capabilities that normal programs on computers don't get. As such, there's no perfect future where there is no potential harm from doing that. The eBPF development community is continually learning how to make eBPF safer than other forms of code execution in the kernel without sacrificing the power of the tool.

The eBPF development community is continually learning how to make eBPF safer than other forms of code execution in the kernel without sacrificing the power of the tool.

Other challenges with eBPF include:

- **Performance tradeoffs**. The higher performance achieved by eBPF results from optimizing for particular use cases. Doing too many things with eBPF may end up eating the gains. Users may have to pick which subset of features to enable to keep performance gains.
- **Co-existence**. Functionality in the kernel also has to support a lot of use cases. That means tools have to co-exist. eBPF functionalities in one piece of software will have to work in combination with other software. If a kernel has two eBPF programs and they both want to work on the same packet, it needs to be decided who goes first. Today, this is mostly done manually, but work is being done so eBPF programs don't step on each other's toes.
- **Deep kernel expertise**. eBPF exposes the internals of the Linux kernel to applications. By doing so, it also exposes a lot of the nitty-gritty kernel details that applications are traditionally shielded from because of the boundaries between applications and the kernel. This means that programming eBPF effectively requires deep kernel expertise. This is part of the reason that most of the early adoption of eBPF has been by organizations who also employ kernel developers.

Gregg maintains that eBPF can be widely deployed even though the vast majority of software developers will never write eBPF code. If Netflix has 3,000 engineers, only about 10 may ever write eBPF code. But everyone will know it exists and will go to those 10 to get the programs they need, Gregg told the audience at **YOW! Perth**.

"For companies to best make use of eBPF they need at least one senior engineer who can handle eBPF programming who can help guide their company to finding the open source, commercial, or in-house solutions to best match their needs," Gregg says. "eBPF can be a difficult technology. If there isn't already a good-fitting solution it can require a mix of kernel engineering and assembly programming in a restricted environment to program directly, and to fully realize its value requires a solid understanding of operating system internals plus imagination."

Platform engineers will also play an important role. Not only will they curate the hardware and software that eBPF developers need to reliably and safely deploy code on-demand, but eBPF's extension of the Linux kernel will enable the creation of new abstractions and building blocks that platform engineers can add to the set of standardized tools they curate for developers over time.

- **Too much data**. If you can see everything thanks to eBPF technology, you "see everything," said Jean Yang, head of product observability at Postman, in a **keynote** at eBPF Summit. That can overwhelm system engineers with network data. Filtering is needed so that data collected off by eBPF can be presented in a more tailored form.

As eBPF becomes one of the most influential technologies in the infrastructure software world, it's critical to optimize collaboration between projects and "ensure that the core of eBPF is well maintained and equipped with a clear roadmap and vision for the bright future ahead,"

- **Interoperability**. Some ISVs are starting to ship eBPF-based software. As more software vendors incorporate it, interoperability becomes an even bigger must-have. This will require a mix of technical solutions for interoperability, community norms, and best practices for how to be a good eBPF application citizen.

Role of the eBPF Foundation

As eBPF becomes one of the most influential technologies in the infrastructure software world, it's critical to optimize collaboration between projects and "ensure that the core of eBPF is well maintained and equipped with a clear roadmap and vision for the bright future ahead," stated Graf of Isovalent in a **blog** announcing the eBPF Foundation and steering committee to take care of the technical direction and vision of eBPF. The port of eBPF to the Windows kernel and other ports to new platforms will also require coordination of portability and runtime requirements.

Conclusion: Where eBPF is Headed

While eBPF is already widely deployed, it is still at the beginning of the large wave of innovation it will unlock, eBPF experts say.

In terms of kernel development, Meta's wins with CPU schedulers makes eBPF "fundamentally different in an interesting way" because it has demonstrated the winning combination of being able to "just launch and test faster," than ever before, Meta's Kelley says.

No doubt, eBPF will become the new layer in the new cloud native infrastructure stack, impacting the observability, performance, reliability, networking, and security of all applications, supporters say. Platform engineers will cobble together eBPF-powered infrastructure building blocks to create platforms that developers then deploy software on, adding business logic to the mix, and replacing aging Linux kernel internals that cannot keep up with today's digital and, increasingly, cloud native world.

Meanwhile, Microsoft's work with eBPF is "paving the way for other operating systems" such as BSD and Apple's OS X, to use eBPF, Gregg said in a **keynote** at the Yow! Perth conference. The work Microsoft is doing includes such things as making sure there is a separate foundation and bytecode for eBPF, and committees to oversee changes.

Microsoft's Jowett says standardization in the eBPF ecosystem around instruction sets is key to ensuring that eBPF programs function the same in all environments, which will make it easier to port applications back and forth.

"It is a fundamental part of getting eBPF to move away from being a Linux-specific technology to being an industry wide platform," Jowett says.

Adds Gregg: "Microsoft is pushing the way to make sure this grows beyond Linux," Eventually, eBPF will be available on all operating systems, he says.

What will success for eBPF continue to look like? Postman's Yang says it won't look like anything. Years ago, when hardware ruled the tech stack, "Intel Inside," was the phrase that instilled confidence. Few bothered to look further into what chip was doing what.

In today's software and cloud-first defined world, "eBPF inside," will have the same stature, and no one, not developers or users, will even know it is there, Yang says. It will just be running the digital world around us.

Acknowledgements

First, we want to recognize all of the maintainers, contributors, and members of the eBPF Foundation and project.

The following individuals contributed to the content of this paper:

- Dan Brown, eBPF Foundation
- Lisa Caywood, Red Hat
- Will Ferrier, Intel
- Arthur Gonigberg, Netflix
- Thomas Graf, Isovalent
- Brendan Gregg, Intel
- Amit Gupta, Tigera
- Daniel Haney, Microsoft
- Toke Hoiland Jorgensen, RedHat
- Alan Jowett, Microsoft
- Dan Kelley, Meta
- Bill Mulligan, Isovalent
- Sridhar Rao, eBPF Foundation
- Liz Rice, Isovalent
- Shweta Saraf, Netflix
- Chris Wright, RedHat

The **eBPF Foundation** brings together a cross-platform community of eBPF-related projects from across the open source ecosystem in an independent forum. The foundation provides a forum to collaborate on a common technical vision, vocabulary, security best practices, and general roadmap, to be applied within separate workstreams, for example Kubernetes, operating system kernels, and enterprise communities.

Founded in 2021, **Linux Foundation Research** explores the growing scale of open source collaboration, providing insight into emerging technology trends, best practices, and the global impact of open source projects. Through leveraging project databases and networks, and a commitment to best practices in quantitative and qualitative methodologies, Linux Foundation Research is creating the go-to library for open source insights for the benefit of organizations the world over.

To reference this work, please cite as follows: "The State of eBPF," The Linux Foundation, January 2024.