#### **Thorsten Leemhuis**

#### The Linux Kernel: We have to finish this thing one day;) Solving big problems in small steps for more than two decades

twentieth (F)OSDEM already? time flies... big round of applause please:

# for organizers and all other volunteers!

you made and make this great conference happen! many thx!

warning: this talk is part of the history track but no, won't be a boring history class I promise!

# everything I mention is kinda relevant for today and tomorrow

### there will be a moral of the story in the end so let's get started....

= the stage =

# the first (F)OSDEM happened in 2001

# Linux 2.4 had just been released

# had about all important features it needed back then

all needed to conquer the world!

proper Posix support X was running (0.95) arch portability (1.2 & 2.0) SMP (2.0) proper performance this and many other important things

#### since then it got tons of improvements... this talk will only give a glimpse into what happened

# = growing up =

#### 2.4 likely would not run too well on today's computers due to missing drivers, obviously, but also...

# numbers of CPU cores would be problematic

#### back then, uniprocessor systems were the norm today, we have CPUs with 12 or 16 cores not that expensive

and even smartphones often have at least four cores

#### Linux was SMP capable since 2.0 (Jun 1996) was realized with the help of a big hammer

### Big Kernel Lock / BKL only one CPU core is allowed to execute kernel code at any time

with obvious performance impact ;-)

## finer graded locking followed in 2.2 even more in 2.4

that made Linux better at scaling still: in the 2.4.x days, other Unixes were known to scale better *by 2.6 (Dec 2003):* Linux got thousands of finer-grained locks





In 2.2, and from that to individual queue locks in 2.6. The kernel now has thousands of locks, and some people had assumed that the BKL would be gone by 2.6.

As it turns out, there are still over 500 lock\_kernel() calls in the 2.6.6 kernel. For the curious\_here are some of the places which

https://lwn.net/Articles/86859/ (May 2004)

2.6.6 still had about 500 lock\_kernel() calls :-/

### many more steps where needed and taken



Weekly Edition

Content

Archives

Search

Kernel

Security

#### <u>Reconsidering unprivileged BPF</u> (August 10, 2019)

#### **Big kernel lock**

The Big Kernel Lock lives on (May 26, 2004) The Big Kernel Semaphore? (September 15, 2004) ioctl(), the big kernel lock, and 32-bit compatibility (December 15, 2004) The new way of ioctl() (January 18, 2005) The big kernel lock strikes again (May 13, 2008) Kill BKL Vol. 2 (May 21, 2008) The BKL end game (March 30, 2010) Might 2.6.35 be BKL-free? (April 27, 2010) BKL-free in 2.6.37 (maybe) (September 20, 2010) Shielding driver authors from locking (October 20, 2010) KS2010: Lightning talks (November 2, 2010) The real BKL end game (January 26, 2011) Events calendar Unread comments big.LITTLE

LWN FAO Write for us

Distributions

Linux support for ARM big.LITTLE (February 15, 2012) A big.LITTLE scheduler update (June 12, 2012) KS2012, ARM, A big LITTLE undate (September 5, 2012)

https://lwn.net/Kernel/Index/#Big\_kernel\_lock

## Linux finally got rid of the BKL in 2011 after about 15 years

#### thx to heroic efforts by various developers esp. Arnd Bergmann, who took on the task of eliminating the BKL entirely!

# the BKL might be history, but... scalability is something still being worked on



**Content** Weekly Edition Archives Search Kernel Security Distributions Events calendar Unread comments

LWN FAQ Write for us which mesystem for Samba4? (November 25, 2004)

#### <u>Scalability</u>

Kernel Summit 2006: Scalability (July 19, 2006) Too many threads (April 10, 2007) KS2007: Scalability (September 9, 2007) Toward better direct I/O scalability (March 31, 2008) The state of the pageout scalability patches (June 17, 2008) The lockless page cache (July 29, 2008) Tangled up in threads (August 19, 2008) KS2009: How Google uses Linux (October 21, 2009) JLS: Increasing VFS scalability (November 3, 2009) Big reader locks (March 16, 2010)  $\underline{CPUS*PIDS = mess}$  (April 27, 2010) One billion files on Linux (August 18, 2010) VFS scalability patches in 2.6.36 (August 24, 2010) Dueling inode scalability patches (October 20, 2010) Resolving the inode scalability discussion (October 26, 2010) KS2010: Scalability (November 3, 2010) Dcache scalability and RCU-walk (December 14, 2010) Dcache scalability and security modules (April 27, 2011) LSFMM: Lock scaling (April 23, 2013) Split PMD locks (September 25, 2013) Optimizing CPU hotplug locking (October 9, 2013) Revisiting CPU hotplug locking (October 16, 2013) Scalability techniques (October 29, 2013) Memory-management scalability (March 13, 2015)

Scatter/gather chaining

https://lwn.net/Kernel/Index/#Scalability

many small improvements over time never ending story

#### quite a few mm optimizations lately

#### new scheduler load balancing core in Linux 5.5

scheduling for asymmetric systems got improved recently

### most people do not notice any of this mostly flies under the radar

# thx to all these small steps Linux is and stays one of the best scaling OS kernels

= being a good host =

getting rid of the BKL was one of the first big achievements reached in many small steps

## something everybody worked towards not always like that
#### more often, there is some competition which can lead to interesting results

## something important today was absent in the early FOSDEM days:

## builtin virtualization capabilities

in the mid 2000s: virtualization with x86 Linux got famous Xen (~2005) made it popular and x86 processors started getting virtualization capabilities (2006)

## Xen looked like the obvious and fitting solution the Linux world

one that everyone seemed to agree on only problem: support for running as Host (Dom0) or Guest (DomU) was out-of-tree

## and Xen was a Kernel underneath the Linux kernel

#### then suddenly, out of nowhere, in Oct 2006: KVM merged already into 2.6.20 in Feb 2007

because it was so small

## in the beginning compared to Xen worse performance, less features, CPU support required a toy?

KVM was quickly improved in small steps various people and companies made it better and better

a we know today: turned out to be a game changer used basically everywhere these days and made Linux rule the cloud Xen still around Dom0 and DomU support only merged in 3.0 days (2011!) and small when compared to KVM

## why did KVM succeed?

some might say: because it took Xensource too long to upstream their code definitely a factor, but I doubt it would have changed much

the real reason: KVM had a better, more flexible, and future-proof design

#### built into Linux, not underneath it

## reuse things already there that suited Linux more and left it in control

which obviously is in the interest of Linux developers that's why a lot of people were willing to help which in the end resulted in a better solution

#### history lesson relevant today, as every now and then we have similar situations like Xen vs KVM

#### DPDK (Data Plane Development Kit) a technique to make network packages bypass the Linux kernel

Linux developers started to fight back with the eXpress Data Path (XDP), whereupon the AF\_XDP socket (XSK) builds seems XDP & AF\_XDP can mostly keep up with DPDK these days *likely more future proof* 

#### another similar situation Asynchronous I/O (AIO) common in the Windows world, unusual in Linux

## these days io\_uring finally brings proper AIO to Linux

An answer to the SPDK Storage Performance Development Kit – a I/O bypass technique that started to gain territory



SSDs are getting crazy performance. We so need async IO to overcome the syscall overhead.



https://twitter.com/mjpt777/status/1215209572681515008

just as KVM:

both XDP/AF\_XDP and io\_uring started small and got and get improved in small steps

## = hosting differently =

another thing Linux still lacked during the early days of FOSDEM Support for Containers other Unixes supported them already FreeBSD jails (1999), Solaris Zones (2004)

#### Linux containers only became famous ~2014

# so why did it take so long?

#### kernel simply lacked required features impossible to build something like Jails or Zones easily & reliable

#### features got built, one step at a time took years...

## some for exactly this use case

various namespaces (2002 - now)

#### some for nearly this use case cgroups (2007) (initially often used for Virtualization with KVM)

#### some for different USE CASES capabilities (~2003), seccomp (2005),

Docker combined features in a new, more attractive way
...and made Linux containers popular these small steps thus in the end changed the computer world funny detail: LXC was designed to become the preferred container solution

Virtuozzo/OpenVZ became small; Linux-Vserver nearly forgotten they came earlier, but used out-of-tree patches

### LXC still around, but not as big as Docker *ChromeOS and Canonical use it*

imagine for a moment what if just one company had been working towards LXC?

### might have been a pretty bad return of investment...

those things show companies investing money into developing complex new features bears risks...

## a problem for the kernel, but still Linux, the OS, got a better and more flexible solution

### thx to the small steps as they lead to features that Docker could combine in new, attractive way

### = unexpected, but welcomed surprise =

docker shows: sometimes things surface nobody aimed for thx to kernel improvements in small steps, that lead to individual features you can recombine in various ways

# Linux recently started a trip into the unknown

### since ~2014 and 3.15+ people improved the Berkeley Packet Filter (BPF, these days often called Classic BPF/cBPF)

### the in-kernel mini-VM (like a Java VM, not an emulated computer)

# tcpdump relied on it to only get the packets it was interested in

### for performance reasons

(copying everything over to userland first is way too much work...)

improved cBPF got called eBPF called BPF for short these says :@

## faster and much more powerful VM

### run small programs in kernel mode

20 years ago, this idea would likely have been shot down immediately

### network devs scratched itches with eBPF and improved it again and again

## XDP & AF\_XDP build upon it

### other kernel subsystems started to use it, too and more and more will soon



Content
Weekly Edition
Archives
Search
Kernel

What is rather more difficult is moving information between operations. In Metzmacher's case, he would like to call <u>openat()</u> asynchronously, then submit I/O operations on the resulting file descriptor without waiting for the open to complete.

It <u>turns out</u> that there is a plan for this: inevitably it calls for <u>... wait for it ... using BPF</u> to make the connection from one operation to the next. The ability to run bits of code in the kernel at appropriate places in a chain of asynchronous operations would clearly open up a number of interesting new possibilities. "There's a lot of potential

https://lwn.net/Articles/810414/

eBPF still gets improved a lot with each new version

starts to change the kernel fundamentally

## Linux gains more aspects of a microkernel

### that's what Europe's biggest computer magazine wrote the German c't magazine





Gratis-Bordmittel aktivieren und optimieren

### Windows absichern

Virenschutz, Privatsphäre, Extra-Schutz für unterwegs

#### Kfz-Diagnose, Fahrtenbuch, Notruf, Hotspot OBD2-Dongles: Nützliche Spione

#### Profi-Gerate mit 15 und 17 Zoll Erste Notebooks mit Hexa-Core



Turbo-LAN: NBase-T-Switches Lüfterloser Mini-PC mit Power Falt-Orohne Parrot Anafi Handwerker-Handy Cat S61

Desinfec't: Fotos und andere Daten retten Webdienste per REST anzapfen Flexible Heimautomation mit Node-Red Fotos optimieren mit Gimp 2.10

### Luxus-Boards für AMD Ryzen

Vollausstattung für Gamer, Übertakter und Kreative

Hintergrund | Linux-Firewall-Technik

#### Flexibler filtern

#### Neue Firewall-Technik für Linux bringt Elemente von Microkerneln

Der Linux-Kernel erhält eine weitere Firewall-Technik, bei der zur Laufzeit maßgeschneiderter Code den Netzwerkverkehr filtert. Zum sicheren und abgeschirmten Erzeugen dieses Codes hat Linux eine Infrastruktur bekommen, die den Kernel erheblich verändern könnte.

**Von Thorsten Leemhuis** 

ASTONGO

Der im August erwartete Linux-Kernel 4.18 bringt erste Teile des Bpfilter, einer neuen Palet-Filter-Technik für Firewalls. Admins brauchen sich allendings nicht um ihr Iptables- oder Milables-Know-how zu sorgen: Bpfilter soll lediglich den Unterbau ersetzen, den das altbekannte Iptables und sein designierter Nachfolger nutzen. Das soll sie deutlich schneller machen. Noch liegt dieses Ziel aber in weiter Ferne, denn bei 4.18 wurden nur Teile des Fundarnents gelegt. Das hat dem eher monolithischen Linux-Kernel ganz nebenbei zueiner Infra struktur verholfen, die prinzipiell eine Modularisierung in der Art von Microkerneln ermöglicht.

#### Genau passender Code

Der Clou von Bpfilter: Die Entscheidung über die Handhabung von Netzwerkpaketen erfolgt dusch lokal erzeugten Programmcode, den der Kernel mit dem BPF ausführt. Dabei handelt es sich um eine prozesbasierte Virtu al Machine, die allerdings simpler gestrickt ist als jene von. NET oder Java. Die VM ist ist auch als eBPF bekannt, da sie in den lettren Jahren aus dem Berkeley Packet Filter (BPF) hervorgegangen ist; mit ihm hat der aktu elle und bereits an vielen Stellen des Kernels gerutztie BPF aber kaum noch etwas gemein.

Der BPF-Programmcode zum Filtern von Netzwerkpaketen mit dem Bpfilter wird für die jeweiligen Anforderungen maßgeschneidert.

Das reduziert Verzweigungen im Gode und verspricht, Overhead zu vermeiden. Damit der Prozessor den BPF-Gode möglichst schnell ausführt, übersetzt der Kernel ihn auf gängigen Prozessor-Architekturen noch per Just-in-Time (JIT) in Maschinenbefehle. Darüber hinaus kann der Bpfilter den Netzwerkverkehr schon beim eXpress Data Path (XDP) abgreifen; der BPF-Gode kann die Pakete dadurch schon kurz nach Empfang durch die Netzwerkhandware verarbeiten, bevor der mächtigere und daher trägere Netzwerkstack von Linux übernimmt.

Durch diese und andere Vorteile verspricht der neue Ansatz, effizienter zu arbeiten als die bislang von Iptables und Nftables genutzte Infrastruktur des Netfilter-Subsystems. Die ist zwar in C.geschrieben, aber als universeller und flezibler Palet-Filter ausgelegt. Beim Abarbeiten des Regelsatzes muss der Code daher auch auf Eventualitäten gefast sein, die lokal verwendete Regeln gar richt nutzen. Das macht den Codepfad komplex, schlieftlich bietet der Netfilterimmens viele Möglichkeiten, von denen Firesalls meist nur einen Bruchteil nutzen.

#### Userspace-Helferlein

Der Umstieg auf Bpfiher ist aber noch Zukunftsmusik, denn in 4.18 flossen lediglich Vorarbeiten ein. Sie schaffen eine Infrastruktur, mit der der Kernel den Filtercode lokal erzugen soll. Das erfordert einen komplexen Übersetzungsvorgung, daher wollen die Entwickler die dazu nötigen Funktionen nicht im Code sehen, der mit Kernel-Rechten ausgeführt wird. Da es um eine sehr sicherheitskritische Kernel-Funktion geht, wollen sie die Aufgabe aber auch nicht an Werkzeuge delegieren, die unabhängig vom Kernel entwickelt werden und als normale Anwendungen laufen.

Die Programmierer haben daher für den Bpfilter einen Zwischenweg geschaffen: Erweiterung am User Mode Helper (UMH) und Module-Code, um den Kernel-Quellen beiliegende

Copyright by Heine Weller Facebildes FOF For House to Maker and F

c't 2018, Heft 15

24

Disclaimer: it was me who wrote that ;-)



knurd | Log out | [Renew]

Content

Weekly Edition Archives Search Kernel Security Distributions Events calendar Unread comments

LWN FAQ Write for us

#### Edition

Return to the Front page By Jonathan Corbet November 6, 2017 2017 Maintainers Summit

#### Kernel regression tracking, part 2

The tracking of kernel regressions was <u>discussed at the 2017 Kernel</u> <u>Summit</u>; the topic made a second appearance at the first-ever Maintainers Summit two days later. This session was partly a repeat of what came before for the benefit of those (including Linus Torvalds) who weren't at the first discussion, but some new ground was covered as well.

Thorsten Leemhuis started with a reprise of the Kernel Summit discussion, noting that he has been doing regression tracking for the last year and has found it to be rather harder than he had expected. The core of the problem, he said, is that nobody tells him anything about outstanding regressions or the progress that has been made in fixing them, forcing him to dig through the lists to discover that information on his own. He had, though, come to a few conclusions on how he wants to proceed.

First, he will try again to establish the use of special tags to identify regressions. His first attempt had failed to gain traction, but he agreed that he perhaps had not tried hard enough to publicize the scheme and get developers to use it. He will be looking into using the kernel Bugzilla again, even though it still seems like unpleasant work to him. He'll try to improve the documentation of how regressions should be

tracked and handled. There is a plan to create a new mailing list on vger.kernel.org, with the idea that regression reports would be copied there. He will put more effort into poking maintainers about open regressions.

The discussion quickly turned to the problem (as seen by some) of the many kernel subsystems that do not use the kernel.org Bugzilla instance for tracking bugs. Peter Anvin said that many developers don't see much value in that system. Reported bugs tend to say something like "my laptop doesn't boot" with no further information; that tends not to be useful for the identification of any actual bugs. Beyond that, many bugs reported against the core kernel or x86 architecture turn out to be driver bugs in the end.

Users, it was suggested, should be explicitly directed to the mailing lists when



# others compared it to microkernels, too



#### knurd | Log out | (Subscriber)

### **Bpfilter (and user-mode blobs) for 4.18**

By **Jonathan Corbet** May 30, 2018 In February, the <u>bpfilter mechanism</u> was first posted to the mailing lists. Bpfilter is meant to be a replacement for the current in-kernel firewall/packet-filtering code. It provides little

functionality itself; instead, it creates a set of hooks that can run BPF programs to make the packet-filtering decisions. <u>A version of that patch set</u> has been merged into the netnext tree for 4.18. It will not be replacing any existing packet filters in its current form, but it does feature a significant change to one of its more controversial features: the new user-mode helper mechanism.

[...]

The replacement of netfilter, even if it happens as expected, will take years to play out, but we may see a number of interesting uses of the new user-mode helper mechanism before then. The kernel has just gained a way to easily sandbox code that is carrying out complex tasks and which does not need to be running in a privileged mode; it doesn't take much effort to think of other settings where this ability could be used to isolate scary code. Just be careful not to call the result a "microkernel" or people might get upset.



https://twitter.com/srostedt/status/1177147373283418112



**Toke Høiland-Jørgensen** @toke\_dk · Dec 14, 2019 Another step on the path towards Linux becoming a BPF-powered microkernel? Fascinating to watch!

Brendan Gregg @brendangregg · Dec 14, 2019

Facebook's Martin KaFai Lau has developed BPF STRUCT\_OPS to allow implementing tcp\_congestion\_ops (and more) in BPF. marc.info /?l=linux-netde...



https://twitter.com/toke\_dk/status/1205824686426378240

## maybe the beginning or middle of a small revolution

makes Linux more error-resistant, flexible, and powerful and most people don't notice anything

happening in a lot of small steps

# = longstanding wishes =

### another area where Linux was behind from the early FOSDEM days until recently
#### a proper tracing solution similar to DTrace published 2005, built for Solaris

Linux finally got something better quite recently: BCC and bpftrace www.brendangregg.com/blog/2018-10-08/dtrace-for-linux-2018.html

#### called "DTrace 2.0" by Brendan Gregg "one of the leading experts on DTrace" (Wikipedia)

# BCC and bpftrace can do more than DTrace

## pretty cool, see Brendan website, his talks, or his book

www.brendangregg.com

#### BPF Performance Tools (book)

This is the official site for the book BPF Performance Tools: Linux System and Application Observability, published by Addison Wesley (2019). This book can help you get the most out of your systems and applications, helping you improve performance, reduce costs, and solve software issues. Here I'll describe the book, link to related content, and list errata and updates.

The book is available on <u>Amazon.com</u> (<u>paperback</u>, <u>kindle</u>), <u>InformIT</u> (paperback, PDF, etc), and Safari (<u>here</u> and <u>here</u>). The paper book was released in December 2019 **but sold out immediately**; more copies printed soon. ISBN-13: 9780136554820.

The Amazon Kindle preview shows the first 100 pages out of this 880 page book.

As an example new tool from the book, readahead.bt provides a new view of file system read ahead performance: the age of read-ahead pages when they are finally referenced, and unused read-ahead pages while tracing:

1	,			
	# readahead.	bt		
	Attaching 5 probes			
	^C			
	Readahead unused pages: 128			
1				
-	Readahead used page age (ms):			
-	@age_ms:			
-	[1]	2455	000000000000000000000000000000000000000	
-	[2, 4]	8424	000000000000000000000000000000000000000	
	[4, 8)	4417	000000000000000000000000000000000000000	
-	[8, 16)	7680	000000000000000000000000000000000000000	
-	[16, 32)	4352	000000000000000000000000000000000000000	
-	[32, 64)	0		
-	[64, 128)	0		
	[128, 256)	384	66	
1				







New tools developed for this book colored red.

### just like containers: took 10 to 15 years to build everything into the Linux kernel

the cool thing: happened without a design that had exactly BCC or bpftrace in mind they emerged thx to evolution

various building blocks got developed in the past 10 to 15 years with smaller goals perf, ftrace, tracepoints, kprobes, uprobes, kretprobes, uprobes, ... features someone developed to scratch a specific itch

### those are one part of the solution; the other:

eBPF ;-)

# eBPF and tracing/perf tools got combined

#### and people developed BCC and bpftrace

and "ta ta", finally, after many years and many small steps

#### Linux got a DTrace 2.0 15 years after people called for it...

#### = something impossible =

#### Linux soon will offer an important new feature one almost nobody would have expected in the early FOSDEM days

### realtime capabilities control your Laser cutter with Linux

reminder: Realtime is primary about predictability, not performance

very vague and kinda crazy idea back then by a few people



https://youtu.be/BTak9U6vuc0?t=512



https://youtu.be/BTak9U6vuc0?t=799

#### still

#### the developers behind the idea didn't give up worked towards realizing the idea ever since in small steps

### they made Linux better for all of us

realtime systems hit many problems and scalability issues first

#### RT developers had lots of body blows one of the worst afaics:

after going 90 to 95% of the route, they needed money for the rest most of those that used RT patches didn't help much with development

luckily, the RT people were successful

Linux Foundation helped and founded a project 2015

# soon the main trip will finally be finished

#### CONFIG\_PREEMPT\_RT already in mainline but not exposed yet!

#### main thing missing a printk() rework https://lwn.net/Articles/800946/

differences got settled recently, just need to be implemented looks like it will be ready this year realtime, for real, this year, too?

#### describing all the steps taken would fill hours



Realtime

Content Weekly Edition Archives Search Kernel Security Distributions Events calendar Unread comments

LWN FAQ Write for us The ongoing realtime story (October 27, 2004) Schedulers, pluggable and realtime (November 3, 2004) Merging the realtime security module (January 11, 2005) Low latency for audio applications (January 26, 2005) Realtime preemption and read-copy-update (March 30, 2005) The beginning of the realtime preemption debate (June 1, 2005) Realtime and interrupt latency (June 14, 2005) Kernel Summit 2005: Realtime capabilities (July 20, 2005) A new approach to kernel timers (September 20, 2005) Priority inheritance in the kernel (April 3, 2006) Kernel Summit 2006: Realtime (July 19, 2006) Read-copy-update for realtime (September 26, 2006) KS2007: Realtime and syslets (September 9, 2007) What's in the realtime tree (October 3, 2007) Fair user scheduling and other scheduler patches (October 16, 2007) Realtime adaptive locks (March 5, 2008) SCHED FIFO and realtime throttling (September 1, 2008) A new realtime tree (December 9, 2008) Interview: the return of the realtime preemption tree (February 16, 2009) The realtime preemption endgame (August 5, 2009) The realtime preemption mini-summit (September 28, 2009) Scenes from the Real Time Linux Workshop (October 5, 2009) KS2009: Realtime preemption (October 21, 2009) The state of realtime Linux (June 15, 2010) Realtime Linux: academia v. reality (July 26, 2010) Realtime group scheduling doesn't know JACK (December 19, 2010) ELC: A PREEMPT RT roadmap (April 27, 2011) Per-CPU variables and the realtime tree (July 26, 2011) Software interrupts and realtime (October 17, 2012) The 2012 realtime minisummit (October 24, 2012)

Approaches to realtime Linux (October 12, 2004)

Realtime preemption, part 2 (October 20, 2004)

LCE: Realtime, present and future (November 13, 2012) The future of realtime Linux (November 6, 2013) The future of the realtime patch set (October 21, 2014)

Realtime mainlining (November 3, 2015) Time-based packet transmission (March 8, 2018)

Deadline scheduling

Deadline scheduling for Linux (October 13, 2009)

https://lwn.net/Kernel/Index/#Realtime

shows:

crazy goals that look unreachable can be achieved in small steps

### that's how most kernel big features evolve

as new kernel features often are not designed by some company often it are individuals that want to realize an idea or a dream they might have to (ab)use companies to realize their ideas
# or find money in other places

but with a good idea and commitment big & crazy dreams can be realized

# = working differently =

# *containers, bpftrace, realtime, ...* Linux learned a lot since the early FOSDEM days

it took quite long to get those features realized that's just how the Linux world is

#### you can't just hire ~50 developers and make them build a feature you want in two or three years

like Sun could for Zones, DTrace or ZFS

# bears costly risks Linux developers might reject the outcome

#### they want to see small incremental, steps which take more work, time, and might have a bad return of investment

served them very well as often lead to one of the best or the best solution on the market

#### but it has disadvantages, too

#### political and licensing issues aside

#### Is ZFS (2005) the most sophisticated filesystem in the \*nix world?

hands up if, you agree!

#### work on "ZFS for Linux" already started in ~2008 Btrfs

#### but hasn't reached that goal yet doesn't look like it will become a Linux-ZFS anytime soon

Ability to handle swap files and swap partitions

#### Implemented but not recommended for production use [edit]

- Hierarchical per-subvolume quotas<sup>[45]</sup>
- RAID 5, RAID 6<sup>[46]</sup>

#### Planned but not yet implemented [edit]

- In-band data deduplication<sup>[32]</sup>
- Online filesystem check<sup>[47]</sup>
- RAID with up to six parity devices, surpassing the reliability of RAID 5 and RAID 6<sup>[48]</sup>
- Object-level RAID 0, RAID 1, and RAID 10
- Encryption<sup>[8][49]</sup>

In 2009, Btrfs was expected to offer a feature set comparable to ZFS, developed by Sun Microsystems.<sup>[50]</sup> After Oracle's acquisition of Sun in 2009, Mason and Oracle decided to continue with Btrfs development.<sup>[51]</sup>

#### Cloning [edit]

Btrfs provides a clone operation that atomically creates a conv-on-write snapshot of a file. Such cloned files are

https://en.wikipedia.org/wiki/Btrfs#Implemented\_but\_not\_recommended\_for\_production\_use see also: https://btrfs.wiki.kernel.org/index.php/Status

# so what went wrong?

one thing for sure it was overhyped still needed a lot of improvements after the groundwork was done and that as always, was... done in small steps that took (and take) a lot of time

## shows how quick things improve mainly depends on...

(1) how complex the problem is and (2) how many individuals or companies back development

# turned out: problem scope is really complex...

# and companies did not care too much

some companies helped quite a bit Oracle, Suse, Facebook, and a few others

#### but some didn't help much or at all (no complaint)

## big question will Linux get something to compete with ZFS?

#### *I'm pretty sure:* **SOONER OF LATER IT WILL** *it might just take 10 more years...*

will it be bcachefs? a lot of people have high expectation

#### I'd say: wait and see and keep your expectations under control

#### history shows: it's a hard problem that takes a lot of effort

bcachefs right now is nearly a oneman show and not even submitted to upstream inclusion yet... unlikely to fly soon will take many years, even if big companies would start to back it = lifestyle =

#### before coming to an end, let's switch gears stop talking about features and look how the Linux kernel is developed

# during the early FOSDEM days Linux kernel development looked odd to outsiders

no central development forge like sourceforge, gitlab or github

# development driven by mail

# Dozens of mailing lists

# no tracker for patch submissions

quite a few fall through the cracks
#### no central issue tracker for neither developers nor users

long unstable development phases new features lingered in unstable tree for long

#### no predictable release cadence

no driver database no way to easily look up if Linux contains a driver for your particular hardware and see what features it supports

#### we had a overworked lead developer one reason for that:

# we did not even have a version control system (VCS)

## there were more odd aspects

#### the kernel development model improved somewhat since then

#### after a short bitkeeper journey We got git in 2005! changed the world for the better; thanks Linus!

unstable/stable model left behind we got a mostly predictable release cycle (2005/~2.6.13) new releases every 9 or 10 weeks

#### a lot called it crazy back then, but turned out very well! browsers picked scheme up

#### we also got Stable and Longterm kernels ~2005: 2.6.11.y, 2.6.16.y

but to be honest many of the other odd things are still around some even got worse...

#### we now have hundreds of mailing lists instead of a few Dozen

there is a bugzilla, which a lot of developer do not look at at all hint: official place to report a bug in most cases is a mailing list!

security became much more important, but we still have no automated code checking in a central place

### a lot of room for improvements here

#### switch to a central forge like gitlab or github? could be a major step forward, as this brings CI, issue tracker, code review, and many more things

## but no, that won't happen anytime soon

### just as with features: developers demand small steps here, too

needs someone motivated enough to drive small, boring things forward without an immediate return of investment

### as that's why quite a few things are still kinda archaic

which becomes more and more of a problem...

Developer satisfaction?		communication style :(		
lost patches :(	feeling non-productive	:(	-	
struggling with tools :(		lost of patch versions :(		
lost of "nitpicks" :(	Do I want to send a pa that I don't have to?.	tch duplicate work :(		
lost bugs :(	Do I want to finish my pa	atch?		æ
introducing regres	sions :( what's the sions :( sinconsistency :(	ne status of my patch? :( non-transparency :( 27	les)	PLIMELES CONFERENCE

lwn.net/Articles/799134/ (links to slides) www.youtube.com/watch?v=iAfrrNdl2f4



knurd | Log\_out | (Subscriber)

By Jonathan Corbet

November 1, 2019

OSS EU

#### Next steps for kernel workflow improvement

Content Weekly Edition Archives Search Kernel Security Distributions Events calendar Unread comments

LWN FAQ Write for us

Edition Return to the Front page The kernel project's email-based development process is well established and has some <u>strong defenders</u>, but it is also showing its age. At the <u>2019 Kernel</u> <u>Maintainers Summit</u>, it became clear that the kernel's processes are much in need of updating, and that the maintainers are beginning to understand that. It is one thing, though, to establish goals for an improved process; it is another to

actually implement that process and convince developers to use it. At the <u>2019 Open Source Summit</u> <u>Europe</u>, a group of 20 or so maintainers and developers met in the corner of a noisy exhibition hall to try to work out what some of the first steps in that direction might be.

The meeting was organized and led by Konstantin Ryabitsev, who is in charge of kernel.org (among other responsibilities) at the Linux Foundation (LF). Developing the kernel by emailing patches is suboptimal, he said, especially when it comes to dovetailing with continuous-integration (CI) processes, but it still works well for many kernel developers. Any new processes will have to coexist with the old, or they will not be adopted. There are, it seems, some resources at the LF that can be directed toward improving the kernel's development processes, especially if it is clear that this work is something that the community wants.

#### Attestation

Pvahiteov's first goal didn't feature strongly at the Maintainers Summit, but is an issue that he has been



#### Thread



Dmitry Vyukov @dvyukov



<u>ب</u>

Welcome #Gerrit changes for #linux kernel: linux-review.googlesource.com/c/virt/kvm/kvm...  $\sim$ 

and the mailing list version for contrast: lore.kernel.org/lkml/202001231...



=

R

(000)

М

Gerrit has side-by-side diffs, full expandable context, non-lossy comments attached to lines. Here are docs: linux.googlesource.com/Documentation/... 7:17 PM · Jan 23, 2020 · Twitter Web App

12 Retweets 39 Likes

https://twitter.com/dvyukov/status/1220410272755671043

#### just like with features small steps are taken and it will take time; you can help!

#### should the Linux Foundation help more? not sure about that

Linux developers likely would prefer not to be governed like OpenStack or Kubernetes are

### In a nevertheless Linux development meanwhile runs at the usual pace

#### a new kernel version every 9 or 10 weeks for many years now

## each with ~13.500 commits these days

#### diffstat:

#### bringing round about +650.000 insertions and -350.000 deletions growth: ~1,5 million lines per year

#### about 15 years after Andrew Morton wrote: (who back then was #2 in the hierarchy)

From: Andrew Morton <akpm@osdl.org> To: ebiederm@xmission.com (Eric W. Biederman) Cc: torvalds@osdl.org, pavel@suse.cz, len.brown@intel.com, drzeus-list@drzeus.cx, acpi-devel@lists.sourceforge.net, ncunningham@cyclades.com, masouds@masoud.ir, linux-kernel@vger.kernel.org Subject: Re: [PATCH 2/2] suspend: Cleanup calling of power off methods. Date: Wed, 21 Sep 2005 11:24:48 -0700 Message-ID: <20050921112448.0e121a3d.akpm@osdl.org> (raw) In-Reply-To: <m1ll1qcmzr.fsf@ebiederm.dsl.xmission.com> ebiederm@xmission.com (Eric W. Biederman) wrote: > > > Famous last words, but the actual patch volume \_has\_ to drop off one day. > > In fact there doesn't seem to much happening out there wrt 2.6.15. > > Due to changes coming through git or that there will simply be fewer

```
From: Andrew Morton <akpm@osdl.org>
To: ebiederm@xmission.com (Eric W. Biederman)
Cc: torvalds@osdl.org, pavel@suse.cz, len.brown@intel.com,
        drzeus-list@drzeus.cx, acpi-devel@lists.sourceforge.net,
        ncunningham@cyclades.com, masouds@masoud.ir,
        linux-kernel@vger.kernel.org
Subject: Re: [PATCH 2/2] suspend: Cleanup calling of power off methods.
Date: Wed, 21 Sep 2005 11:24:48 -0700
Message-ID: <20050921112448.0e121a3d.akpm@osdl.org> (raw)
In-Reply-To: <m1ll1qcmzr.fsf@ebiederm.dsl.xmission.com>
ebiederm@xmission.com (Eric W. Biederman) wrote:
>
> > Famous last words, but the actual patch volume _has_ to drop off one day.
> > In fact there doesn't seem to much happening out there wrt 2.6.15.
>
> Due to changes coming through git or that there will simply be fewer
> things that need to be patched?
We're at -rc2 and I only have only maybe 100 patches tagged for 2.6.15 at
this time. The number of actual major features lined up for 2.6.15 looks
relatively small too.
As I said, famous last words.
                               But we have to finish this thing one day ;)
  As for 2 6 15 T know T have natches in the queue that T intend to send
```

## = summing things up =

Linux developers solve big problems in small steps #bigkernellock
small steps lead to better and more flexible solutions #kvm vs #xen sometimes make new, groundbreaking technologies possible #docker

## building blocks build in small steps can even help fulfilling old wishes #DTrace\_2.0

process can lead to quite unexpected, disrupting results #bpf (keep an eye on it!)

# that's what made and makes Linux so great

reaching big goals with small steps takes time and thus money

# they thus need someone really committed

# ideally and individual that wants to realize a dream

that worked great in a lot of areas #realtime – but also #BKL, #KVM, #DTace\_2.0, #BPF, ... in some areas, we are not there yet :-/

to improve things, become an individual that is committed

## and find money to get the dream realized

# then Linux will get a filesystem even better than ZFS

and developer tools and schemes even better than what we have or other things that will have a positive impact on the world

## like Linux and Git had and have

which once were just a dream in somebody's head that's it — questions? (TWIMC: this is slide #234)

#### feedback

### please provide feedback feedback welcomed, even if negative; talk to me!

mail: linux@leemhuis.info, thl@ct.de GPG Key: 0x72B6E6EF4C583D2D social media: @kernellogger, @knurd42 on #twitter & #friendica 4 more social media accounts, see www.leemhuis.info/me/

### #EOF