Zaawansowane Systemy Operacyjne

(wykład obieralny, semestr letni)

http://students.mimuw.edu.pl/ZSO

(komplet materiałów, kryteria zaliczania)

Janina Mincer-Daszkiewicz

Łukasz Sznuk, Wojciech Ciszewski, Maciej Matraszek

Wsparcie: Adam Cichy

Linux history

Linus Torvalds in 2024
in conversation with Dirk Hohndel at OSS Vienna

Linus Torvalds, Finland, born in the same year as UNIX, i.e. 1969, creator of the Linux kernel and the Git version control sysem.

Linus Torvalds announcing Linux 1.0, 30.03.1994

Currently hired by the Linux Foundation

Richard Stallman, founder of the GNU project and the Free Software Foundation, co-creator of the GNU GPL license, creator of the Emacs editor, GCC compiler, GDB debugger.

Richard Stallman in 2019

May 1991, version 0.01: no support for the network, limited number of device drivers, one file system (Minix), processes with protected address spaces

The Linux Kernel Archives – https://www.kernel.org/

- **2025-02-21**, latest stable version **6.13.4**
- **2025-02-16**, latest mainline **6.14-rc3**

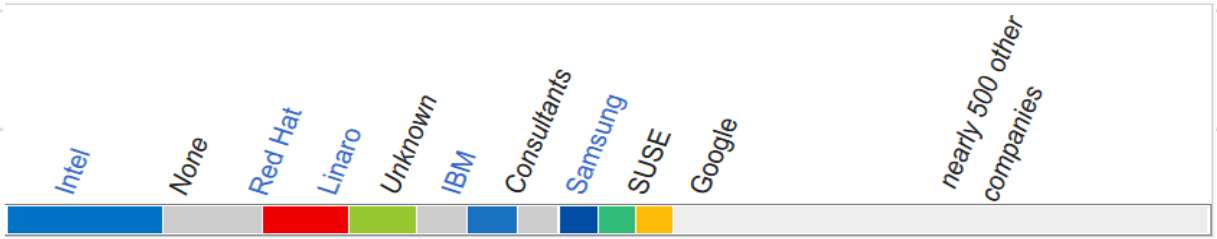
Numbering of the kernel versions – see Wikipedia

Andrew Tanenbaum in 2023

2024 Kernel Maintainers Summit group photo

Recent release history Oct. 2024

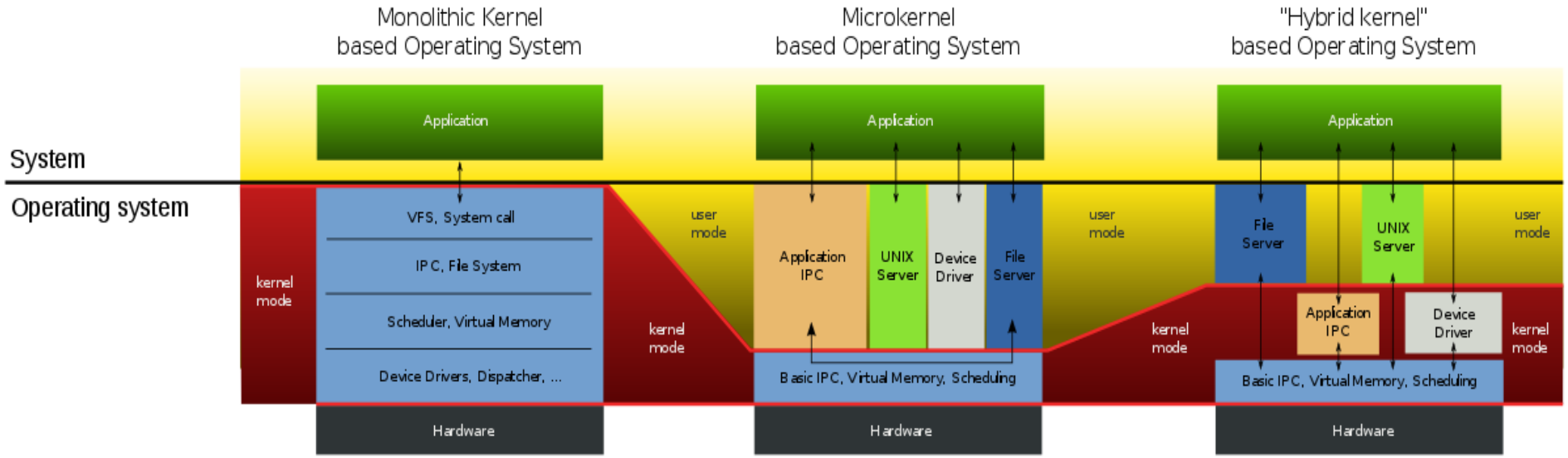
Release	Date	Commits	Devs	1st time
6.6	Oct 29	14,069	1,976	249
6.7	Jan 7	17,284	1,973	270
6.8	Mar 10	14,405	1,938	245
6.9	May 12	14,480	2,026	281
6.10	Jul 14	13,312	1,918	242
6.11	Sep 15	13,890	1,970	250



Corporate affiliation of contributions to the Linux kernel, 4.8–4.13.[218]

Roughly **14%** of the code is part of the "core" (arch, kernel and mm directories), while **60%** is drivers.

Linux kernel versions (source: [Wikipedia)](Wikipedia)

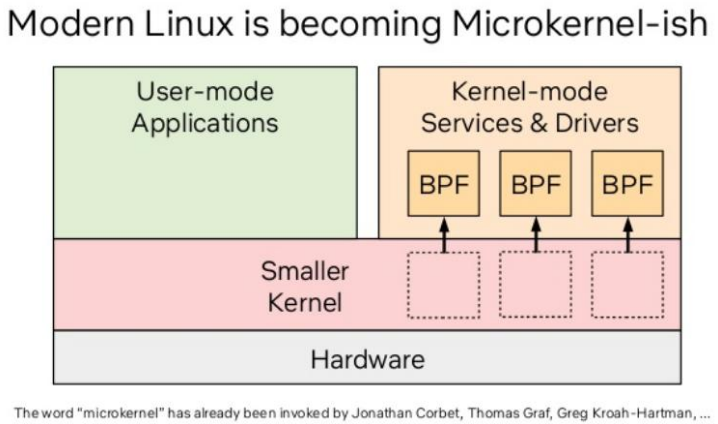


Structure of monolithic kernel, microkernel and hybrid kernel-based operating systems (source: Wikipedia)

Linus Torvalds :

"As to the whole 'hybrid kernel' thing - it's just marketing. It's 'oh, those microkernels had good PR, how can we try to get good PR for our working kernel? Oh, I know, let's use a cool name and try to imply that it has all the PR advantages that that other system has'."

But – eBPF makes a change ...



The Good:

- Open and transparent process
- Excellent code quality
- Stability
- Available everywhere
- Almost entirely vendor neutral

The Bad:

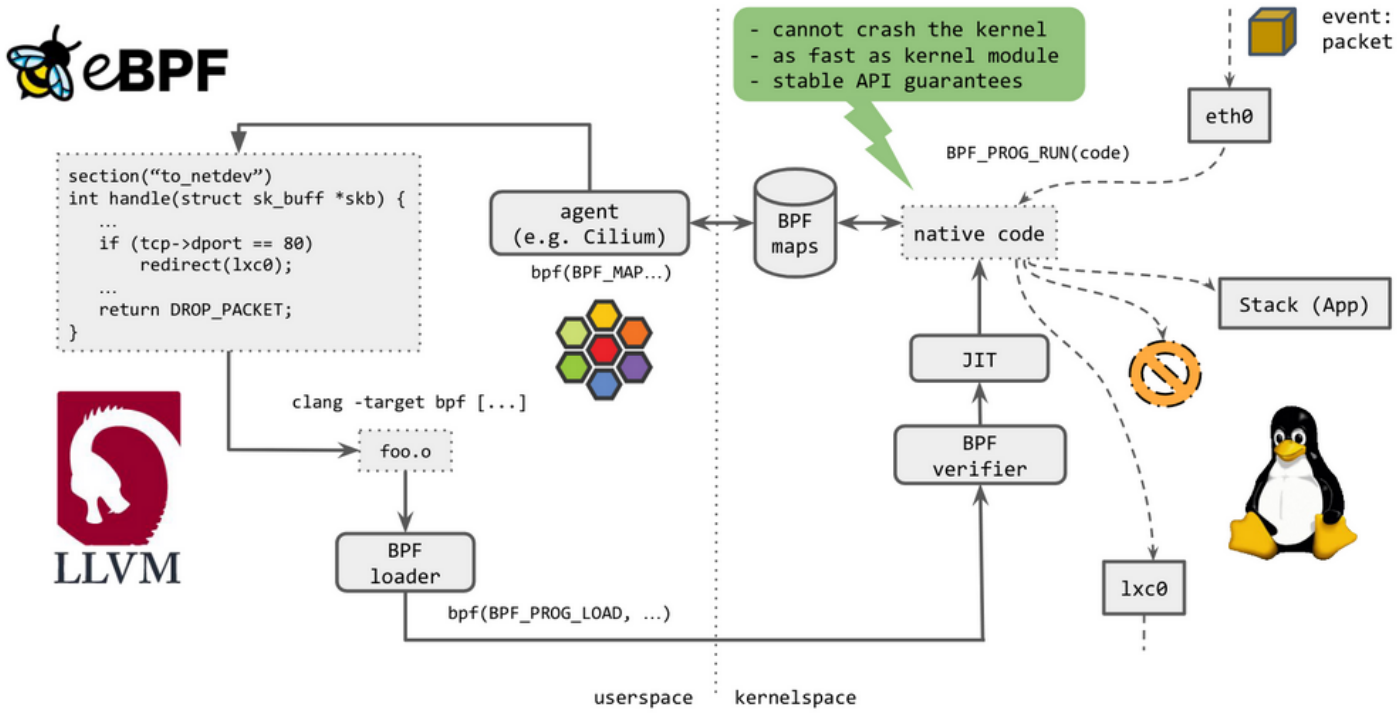
- Hard to change
- Shouting is involved (getting better)
- Large and complicated codebase
- Upstreaming code is hard, consensus has to be found.
- Upstreaming is time consuming
- Depending on the Linux distribution, merged code can take years to become generally available
- Everybody maintains forks with 100-1000s backports

Linux Development

2018, new Code of Conduct

What is BPF?
Highly efficient sandboxed virtual machine in the Linux kernel making the Linux kernel programmable at native execution speed.

Programmability allows to continuously adapt to changing requirements and innovate quickly.

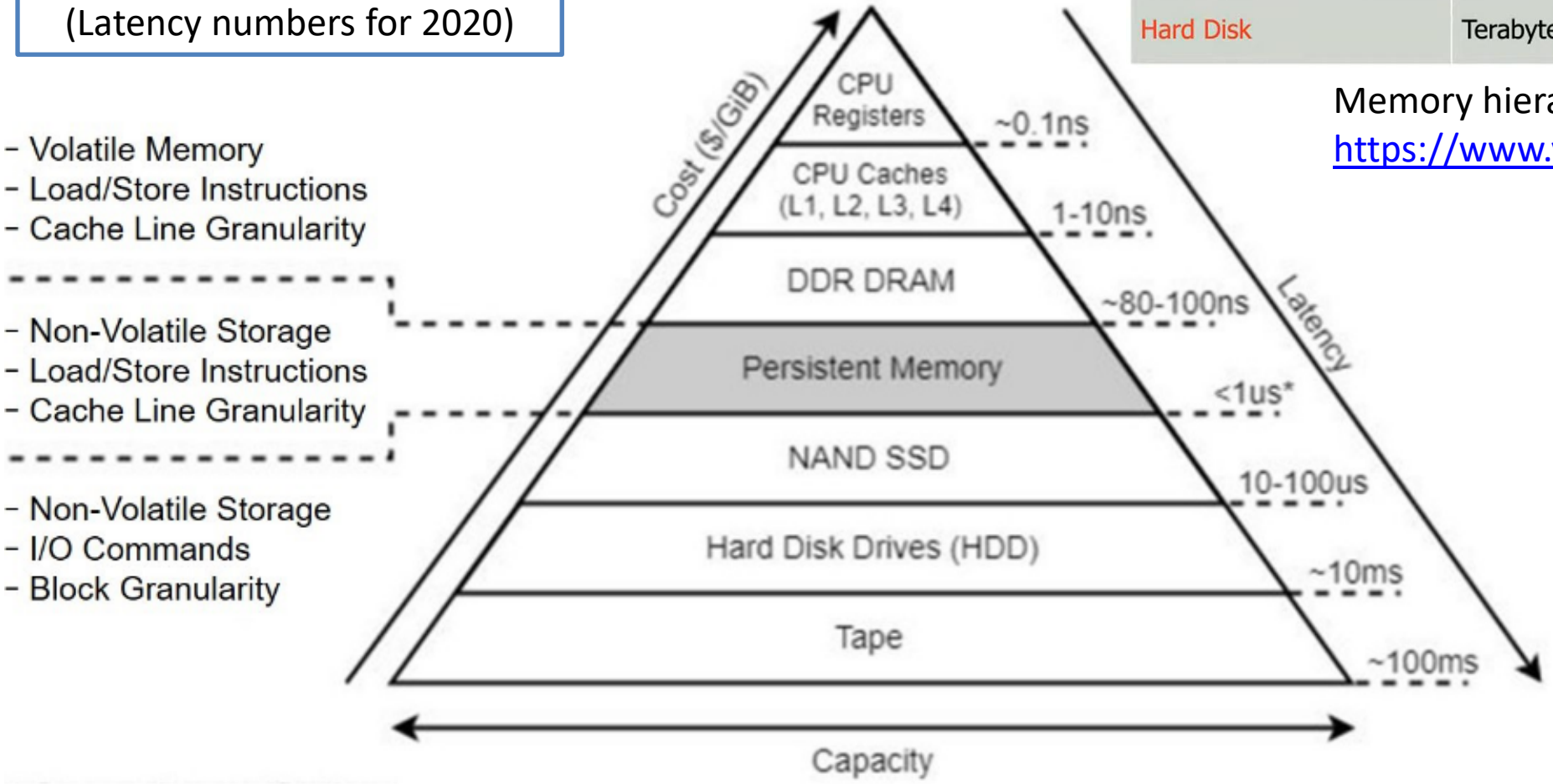


How to Make Linux Microservice-Aware with Cilium and eBPF, Thomas Graf, QCon 2018, (presentation, transcript)

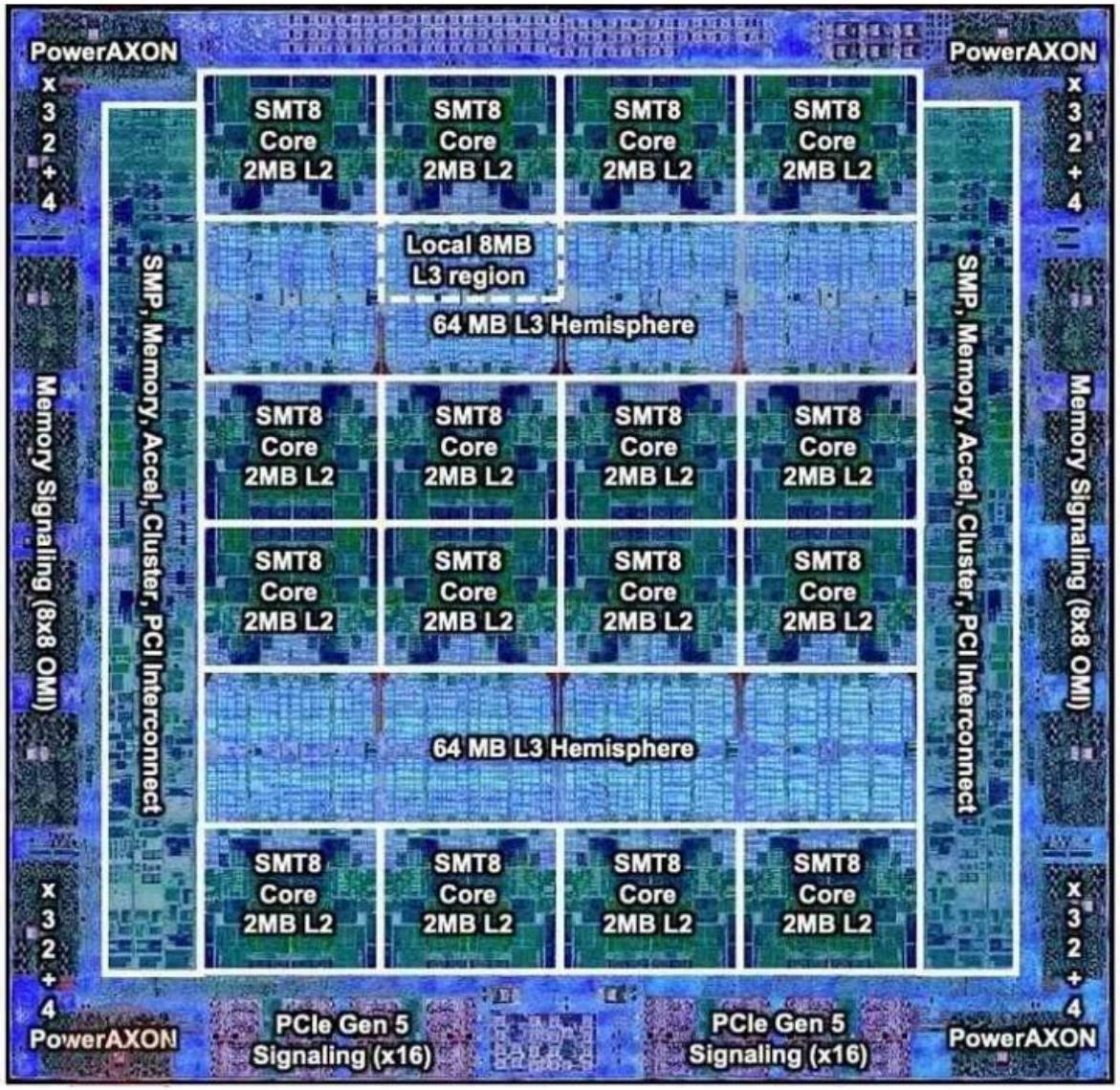
Memory Device	Capacity	Latency	Cost per Megabyte
SRAM	512 Bytes	sub-nanosec	
SRAM	KByte~MByte	~nanosec	< 0.3$
DRAM	Gigabyte	~50 nanosec	< 0.03$
PCM-DIMM (Intel Optane DC DIMM)	Gigabyte	~200 nanosec	< 0.004$
PCM-SSD (Intel Optane SSD)	Gigabyte ~Terabyte	~10 µs	< 0.001$
Flash memory	Gigabyte ~Terabyte	~100 µs	< 0.00008$
Hard Disk	Terabyte	~10 millisec	< 0.00003$

[Memory hierarchy, Intel](#)
(Latency numbers for 2020)

Memory hierarchy, sample values ~2021
https://www.youtube.com/watch?v=J6jkrDlgflo



IBM POWER10, 2020



Cores:

15-16 cores,
8 threads/core

L2 Caches:

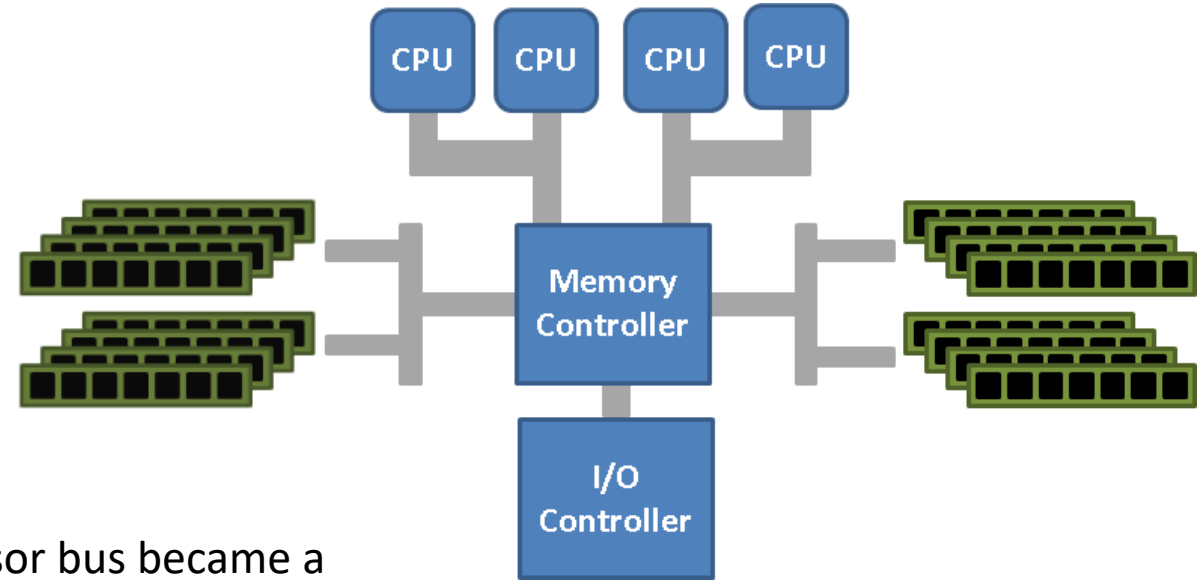
2 MB per core

L3 Cache:

120 MB shared

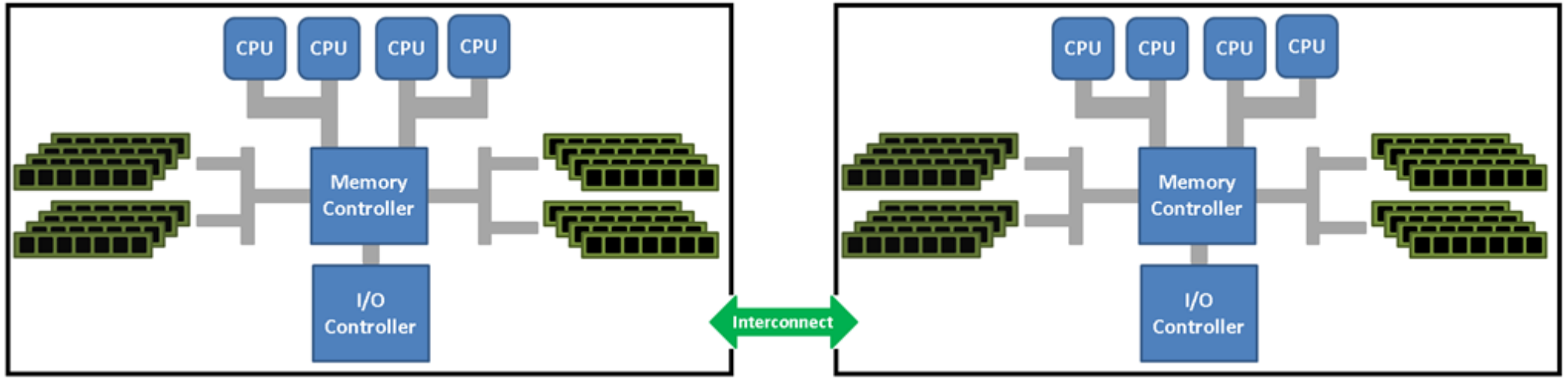
https://www.youtube.com/watch?v=J6jkrDlgflo

UMA vs NUMA



In **Symmetric Multiprocessing** (**SMP**) Systems, a single memory controller is shared among all CPUs (**Uniform Memory Access—UMA**). All of the processors have equal access to the memory and I/O in the system.

As more processors were added to the system the processor bus became a limitation to the overall system performance.



To scale more, **Non-Uniform Memory Architectures** (**NUMA**) implement multiple buses and memory controllers.

The interconnect between the two systems introduced latency for the memory access across nodes.

[Understanding Non-Uniform Memory Access/Architectures (NUMA)]

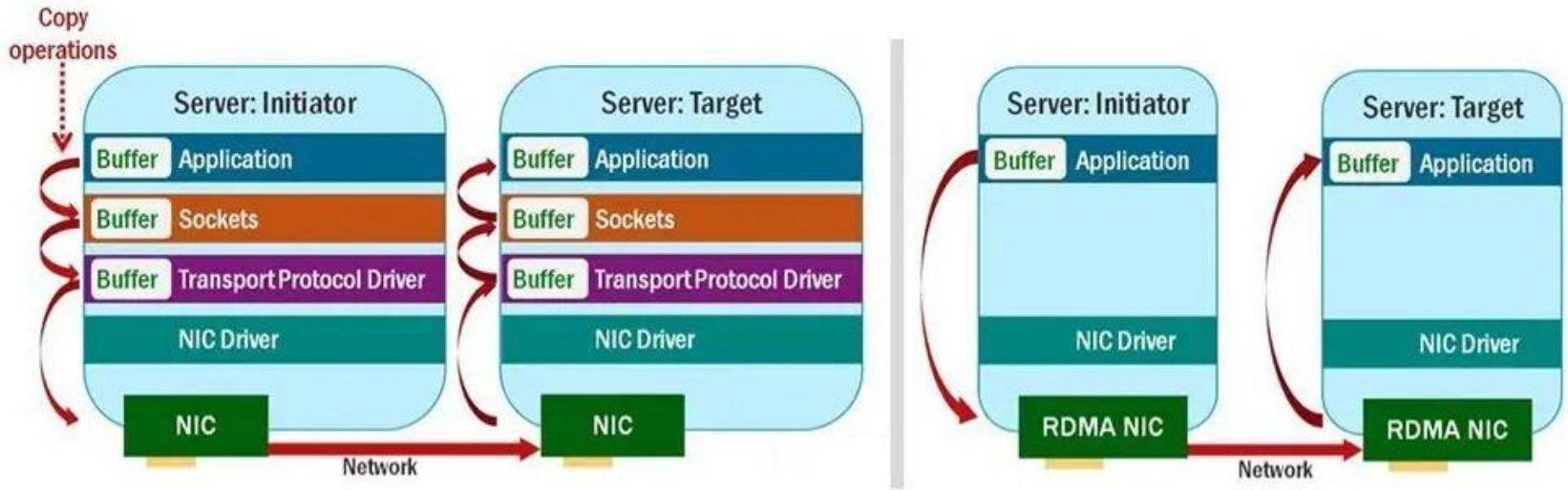
Remote Direct Memory Access (RDMA)

Remote Direct Memory Access (RDMA) provides direct memory access from the memory of one host to the memory of another host without involving the remote Operating System and CPU, boosting network and host performance with lower latency, lower CPU load and higher bandwidth. In contrast, TCP/IP communications typically require copy operations, which add latency and consume significant CPU and memory resources.

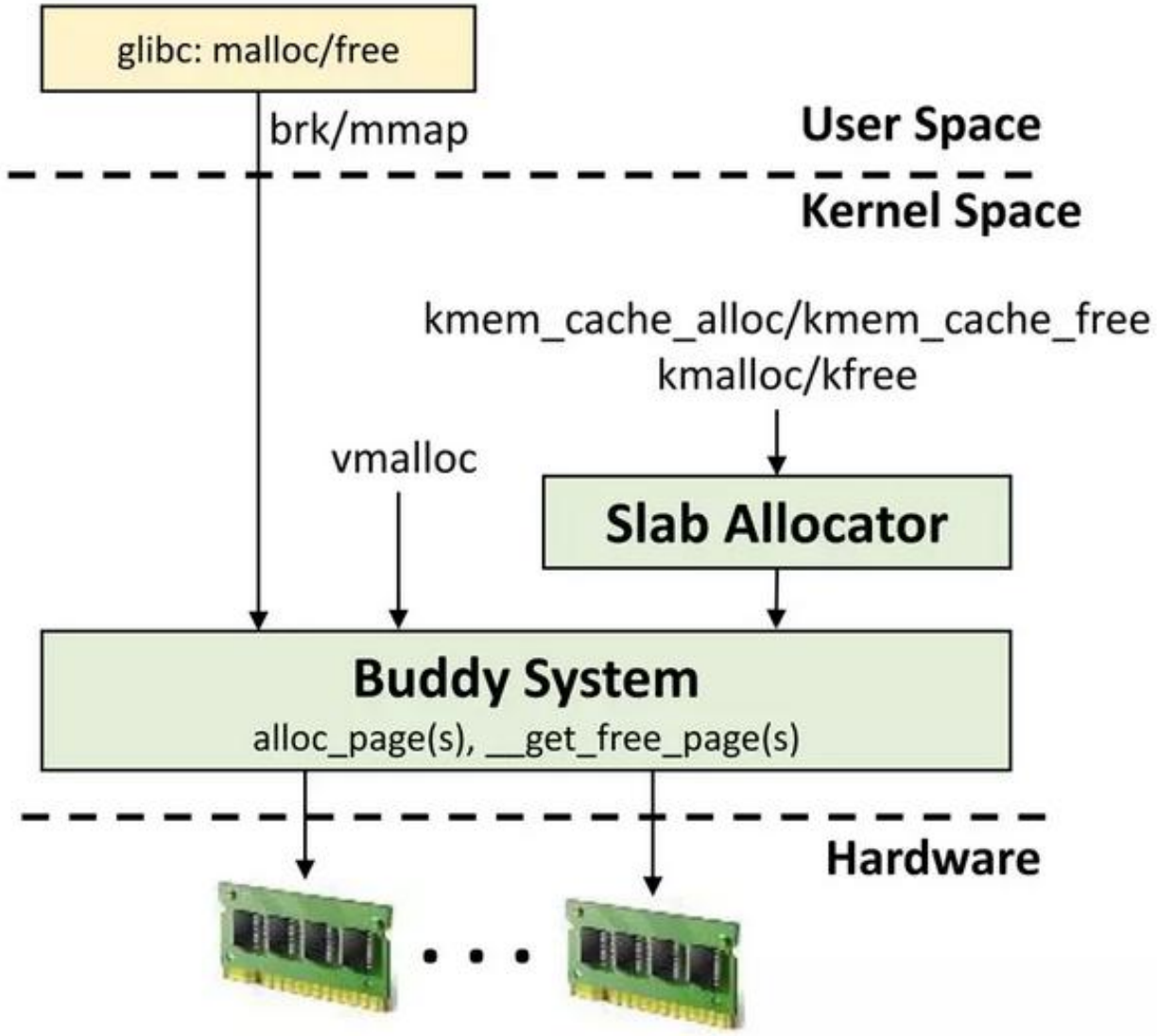
RDMA supports zero-copy networking by enabling the network adapter to transfer data directly to or from application memory, eliminating the need to copy data between application memory and the data buffers in the operating system. Such transfers require no work to be done by CPUs, caches, or context switches, and transfers continue in parallel with other system operations. When an application performs an RDMA Read or Write request, the application data is delivered directly to the network, reducing latency and enabling fast message transfer.

Where used: High Performance Computing, Machine Learning, Big Data

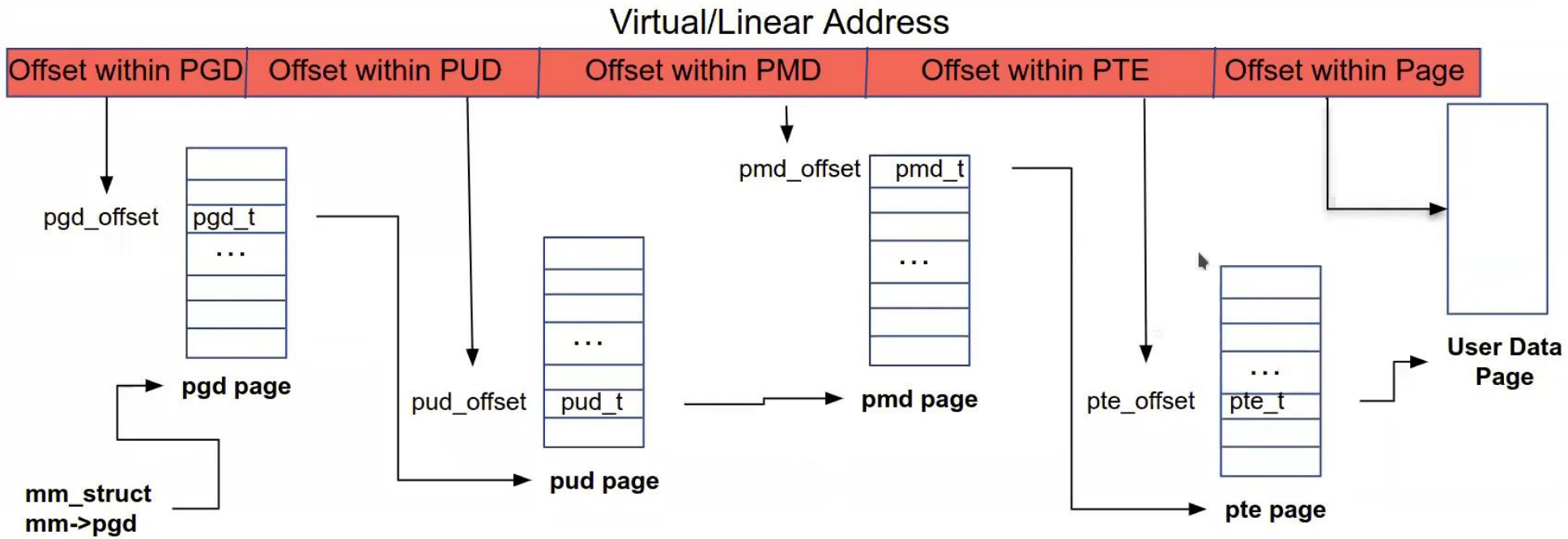
https://www.teimouri.net/review-whats-remote-direct-memory-access-rdma/



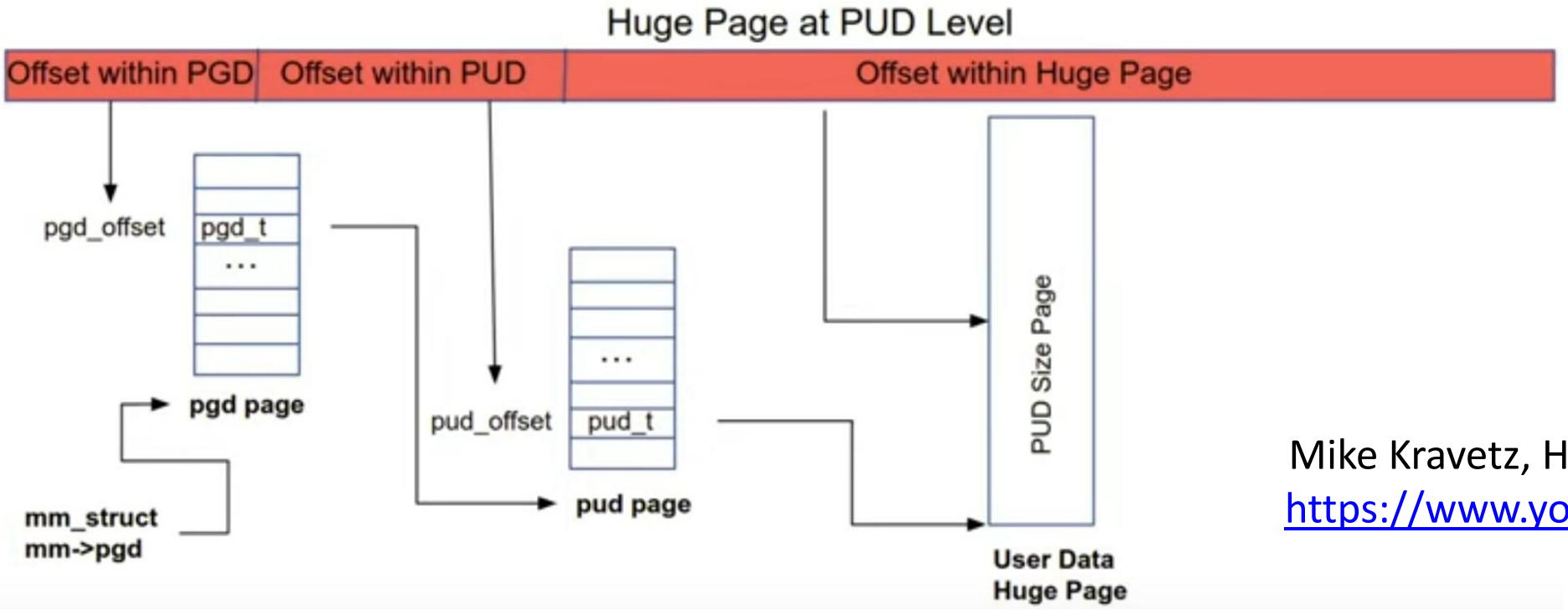
Why do we need yet another memory allocator?



(source: Adrian Huang, Slab Allocator in Linux Kernel, 2022)



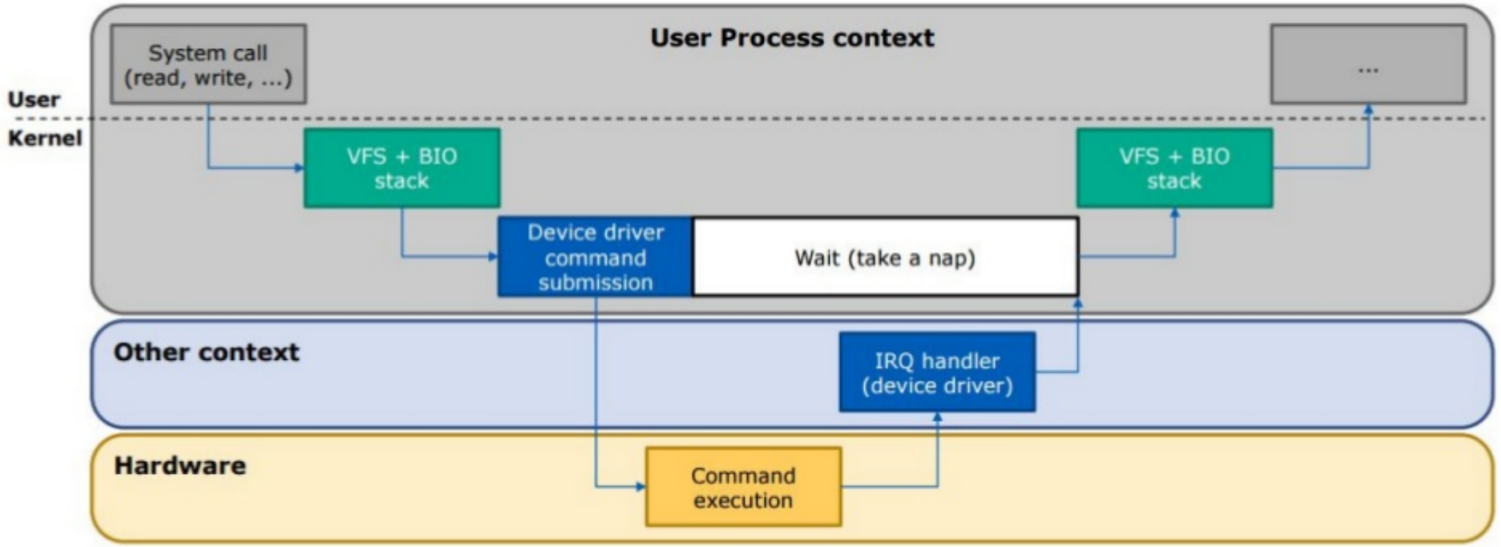
Page size
4KB



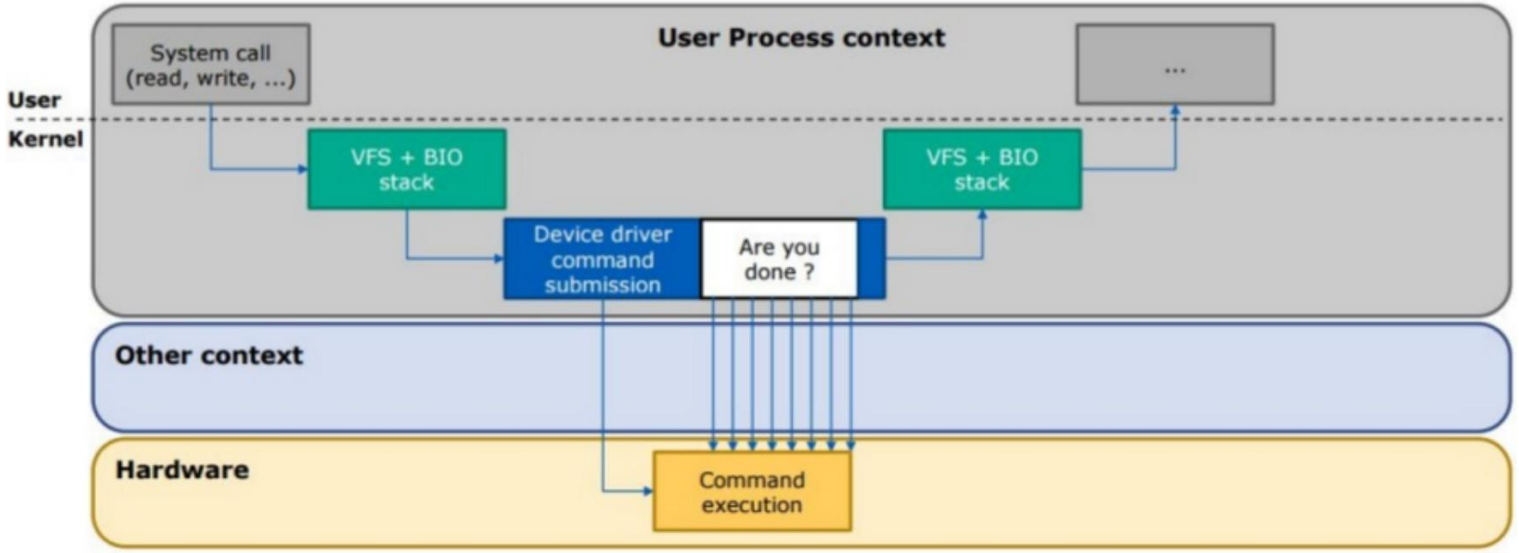
Page size
1GB

Mike Kravetz, Huge page concepts in Linux, https://www.youtube.com/watch?v=n67gCNiKVcw

Interrupt based I/O completion model



Polling based I/O completion model

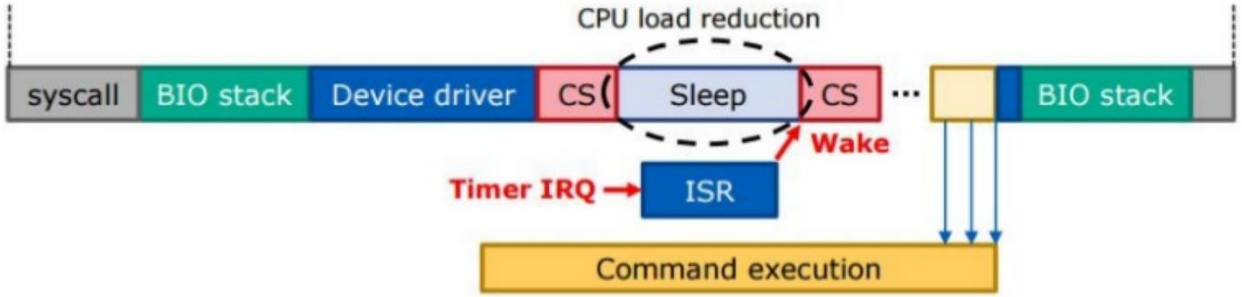


[The Linux Block Layer. Built for Fast Storage](), Sagi Grimberg, June 2018

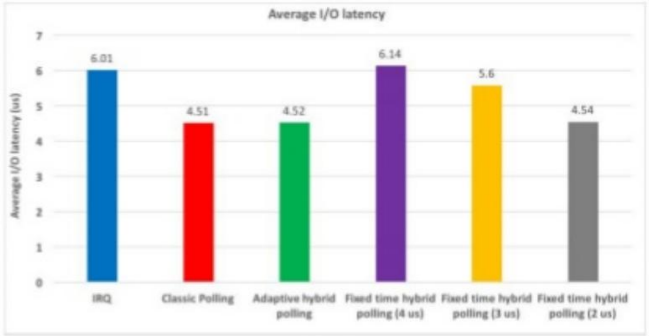
But what about CPU% - can we go hybrid?

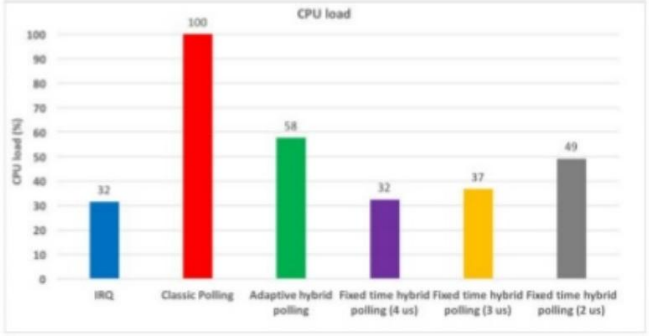
- Yes!
  - We have all the statistics framework in place, let's use it for hybrid polling!
  - Wake up poller after ½ of the mean latency.

Hybrid Polling



Hybrid polling - Performance

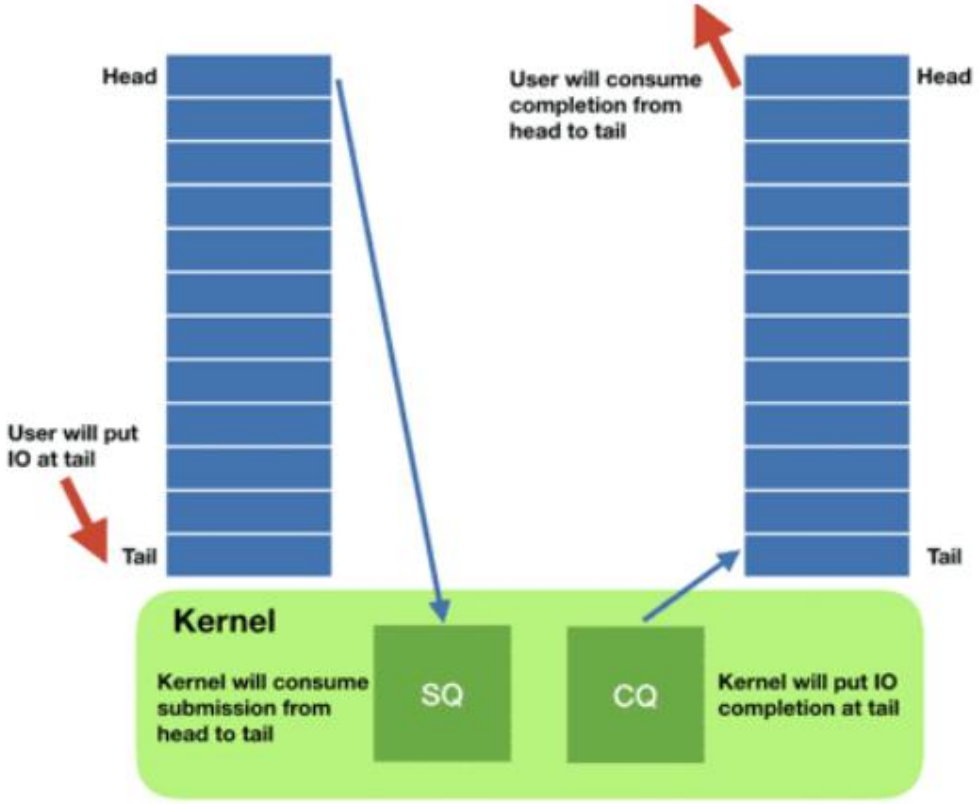


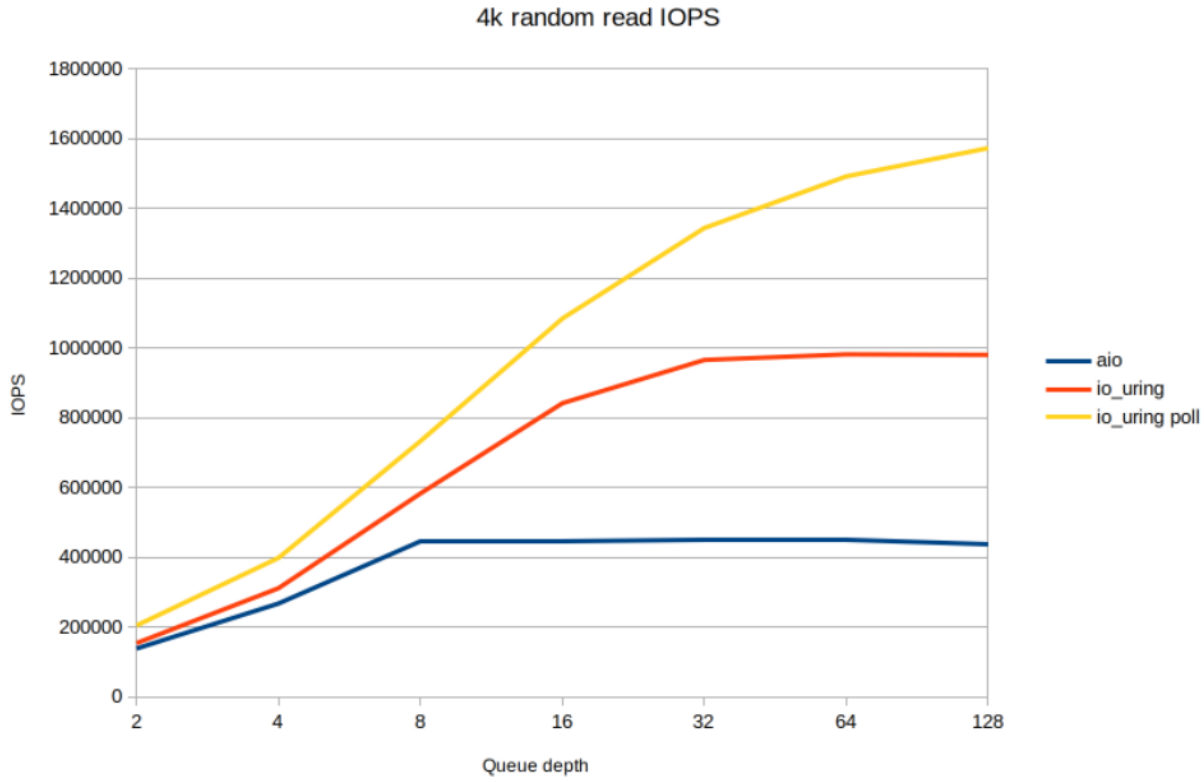


Polling has been added in v4.1.

[The Linux Block Layer. Built for Fast Storage](), Sagi Grimberg, June 2018

Asynchronous I/O – io_uring





Faster IO through io_uring, Kernel Recepies, 2019, Jen Axboe

But … security

New Linux Rootkit, Bruce Schneier, April 24, 2025

- The company has released a working rootkit called “Curing” that uses **io_uring**, a feature built into the Linux kernel, to stealthily perform malicious activities without being caught by many of the detection solutions currently on the market.
- This was just waiting to happen, if it hasn’t already happened earlier. In June 2023, Google reported on its Security blog:
  - in the past year, there has been a clear trend: 60% of the submissions exploited the io_uring component of the Linux kernel[…]. Furthermore, io_uring vulnerabilities were used in all the submissions which bypassed our mitigations.
  - To protect our users, we decided to limit the usage of io_uring in Google products.
- LWN waved a big red flag four years ago, so the only surprise here is that the rootkit is issued in public this much later, and apparently still works.

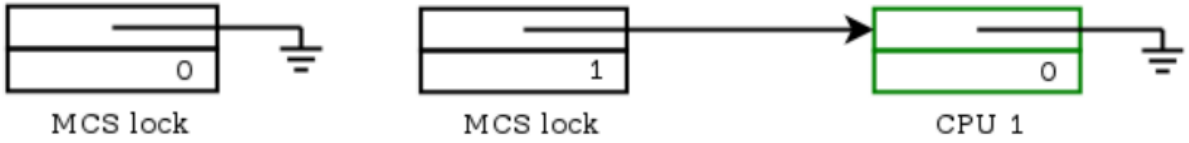
  Auditing io_uring, Jonathan Corbet, June 3, 2021

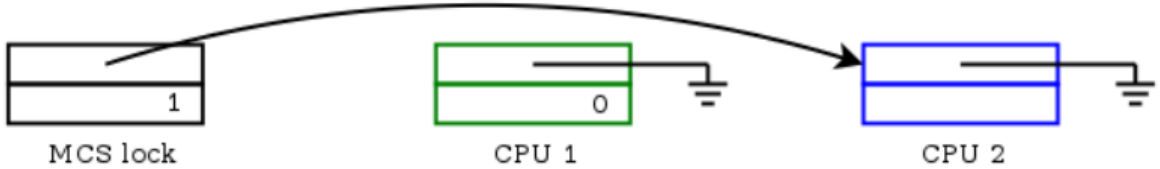
Spinlocks – implementation

– In the 2.6.24 kernel, a spinlock was represented by an **integer value**. A **value of one** indicated that the lock is available, the more **negative the value** of the lock gets, the more processors are trying to acquire it.

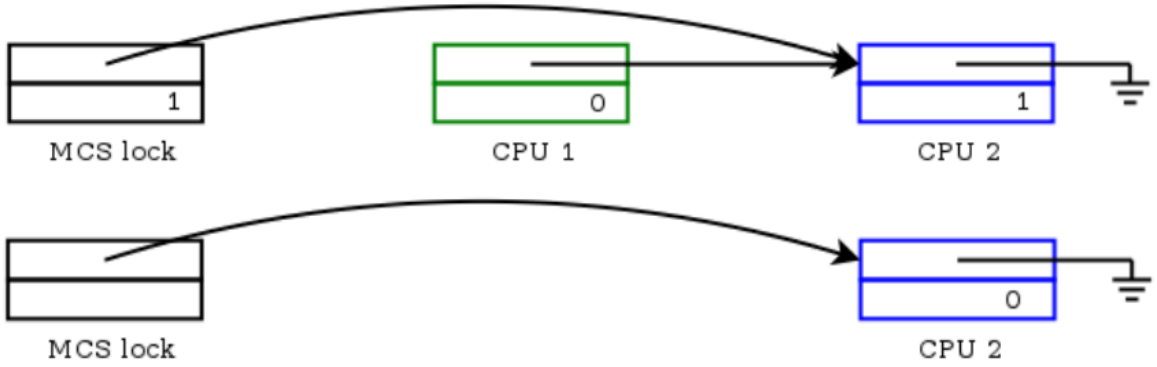
– Ticket spinlocks (2008) added fairness to the mechanism by using 16-bit quantity, split into two bytes. You can think of the "next" field as being the number on the next ticket in the dispenser, while "owner" is the number appearing in the "now serving" display over the counter.

– MCS locks (Mellor-Crummey & Scott, 2014) expand a spinlock into a per-CPU structure, eliminating much of the cache-line bouncing.

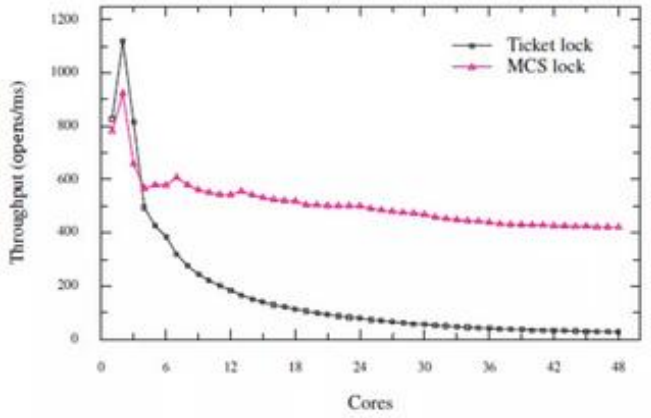




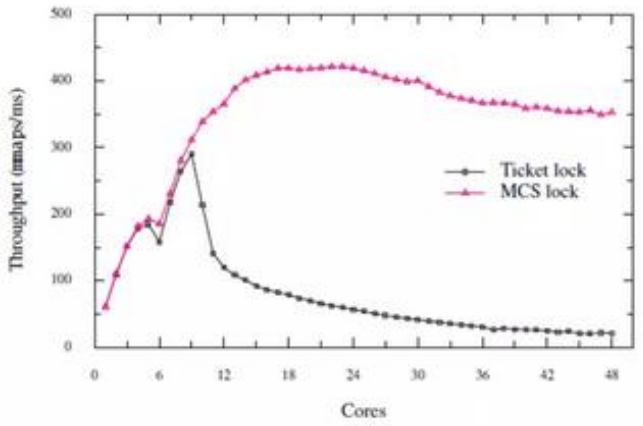

```
struct mcs_spinlock {
    struct mcs_spinlock *next;
    int locked; /* 1 if lock acquired */
};
```



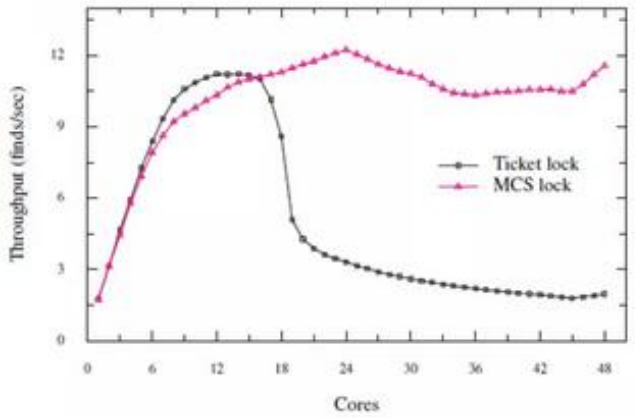
Ticket spinlock vs MCS lock



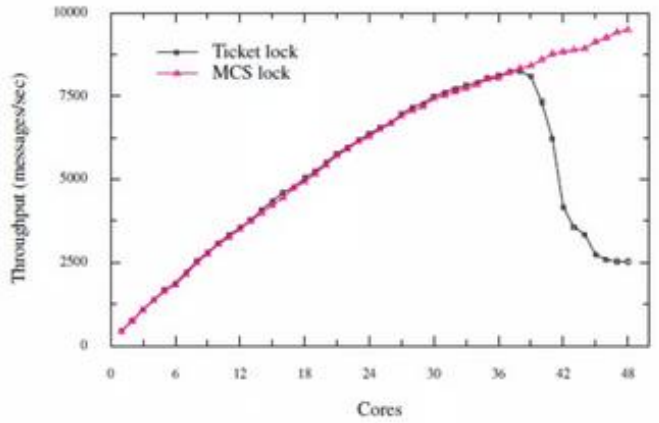
(a) Performance for FOPS.



(b) Performance for MEMPOP.



(c) Performance for PFIND.



(d) Performance for EXIM.

Figure 12: Performance of benchmarks using ticket locks and MCS locks.

Benchmarks

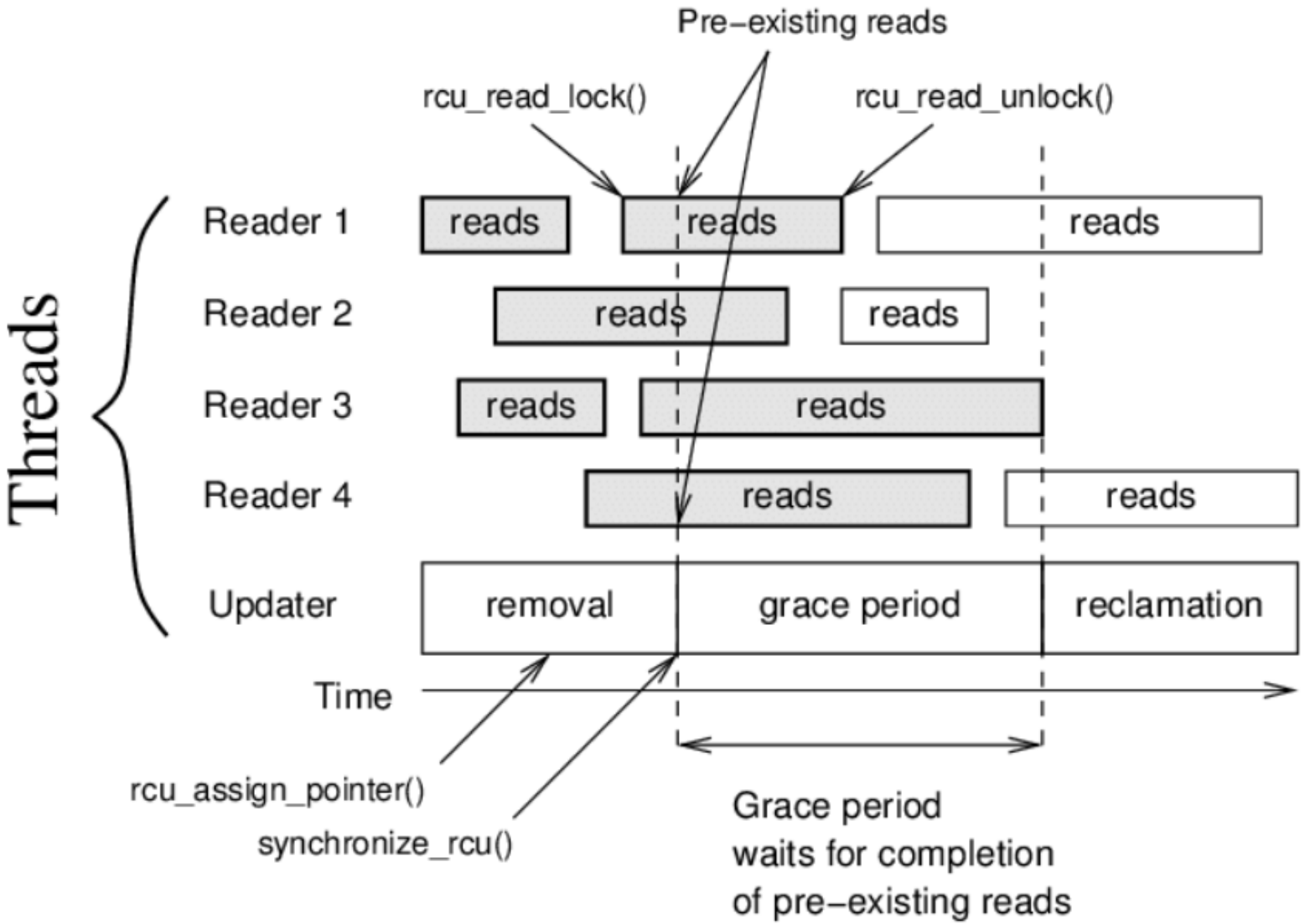
- FOPS: creates a single file and starts one process on each core. Repeated system calls "open() and close()".
  - ✓ Executing the critical section increases from 450 cycles on two cores to 852 cycles on four cores
  - ✓ The critical section is executed multiple times per-operation and modifies shared data, which incurs costly cache misses
- MEMPOP
  - ✓ One process per core
  - ✓ Each process mmaps 64 kB of memory with the MAP_POPULATE flag, then munmaps the memory
- PFIND: searches for a file by executing several instances of the GNU find utility
- EXIM (mail server): A single master process listens for incoming SMTP connections via TCP and forks a new process for each connection

Reference Paper: Boyd-Wickizer, Silas, et al. "Non-scalable locks are dangerous." Proceedings of the Linux Symposium. 2012.

Read-Copy Update (RCU)

Grace period – time period when every thread was in at least one quiescent state.

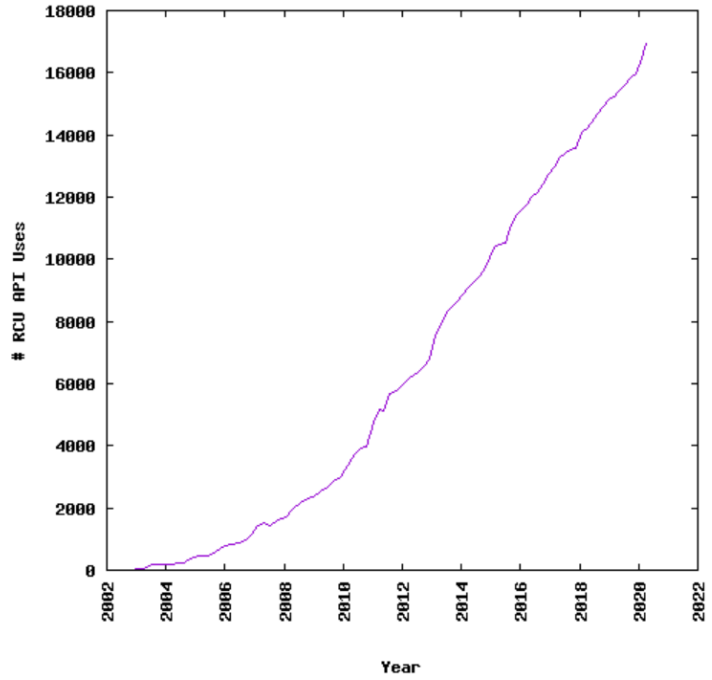
Quiescent state – any point in the thread execution where the thread does not hold a reference to shared memory.

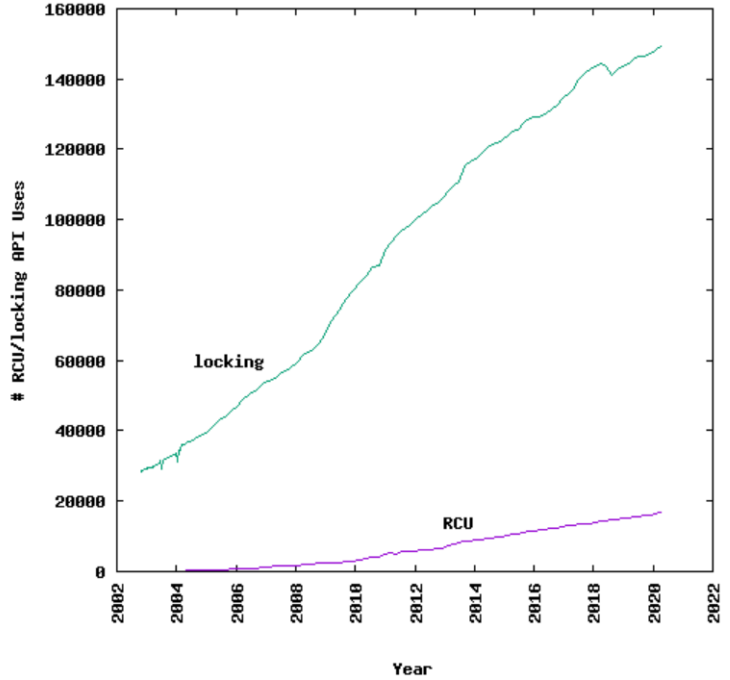


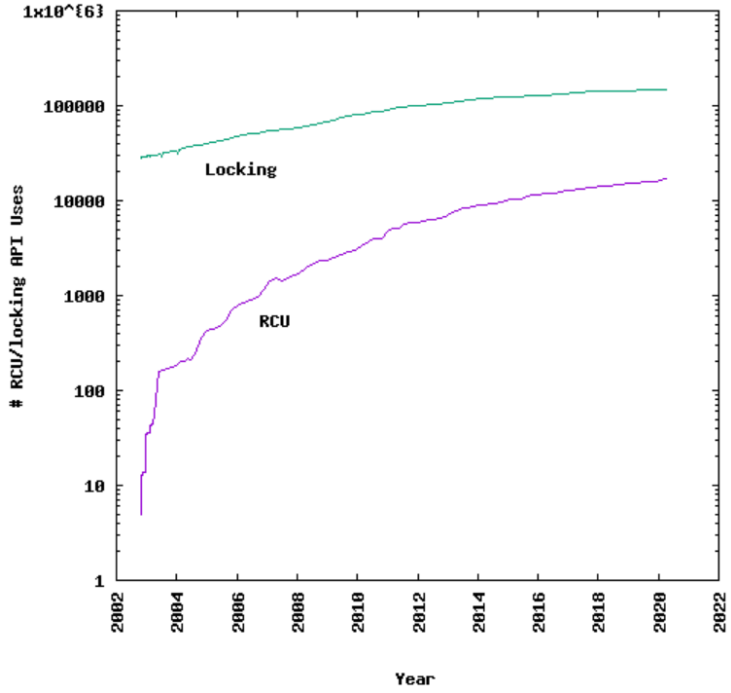
Schematic of RCU Grace Period and Read-Side Critical Sections

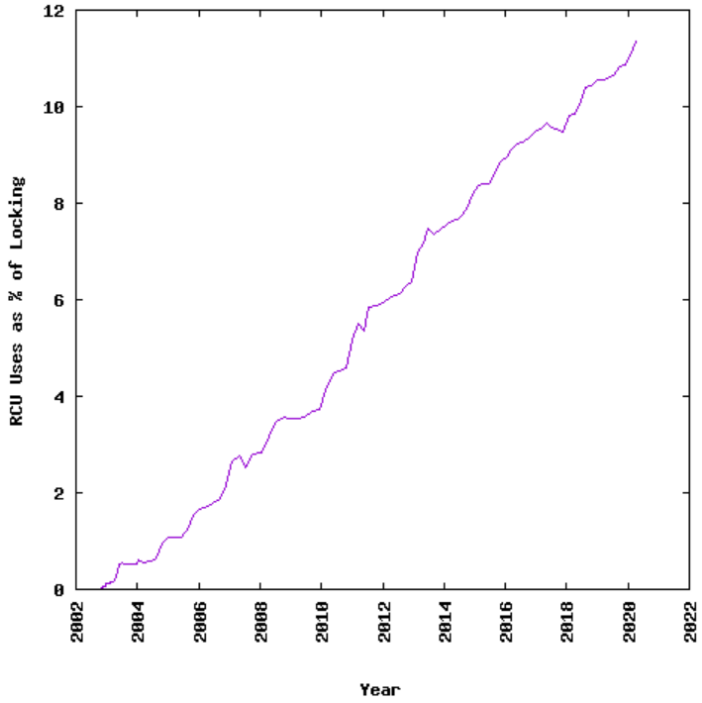
Paul McKenney

http://www.rdrop.com/users/paulmck/RCU/linuxusage.html









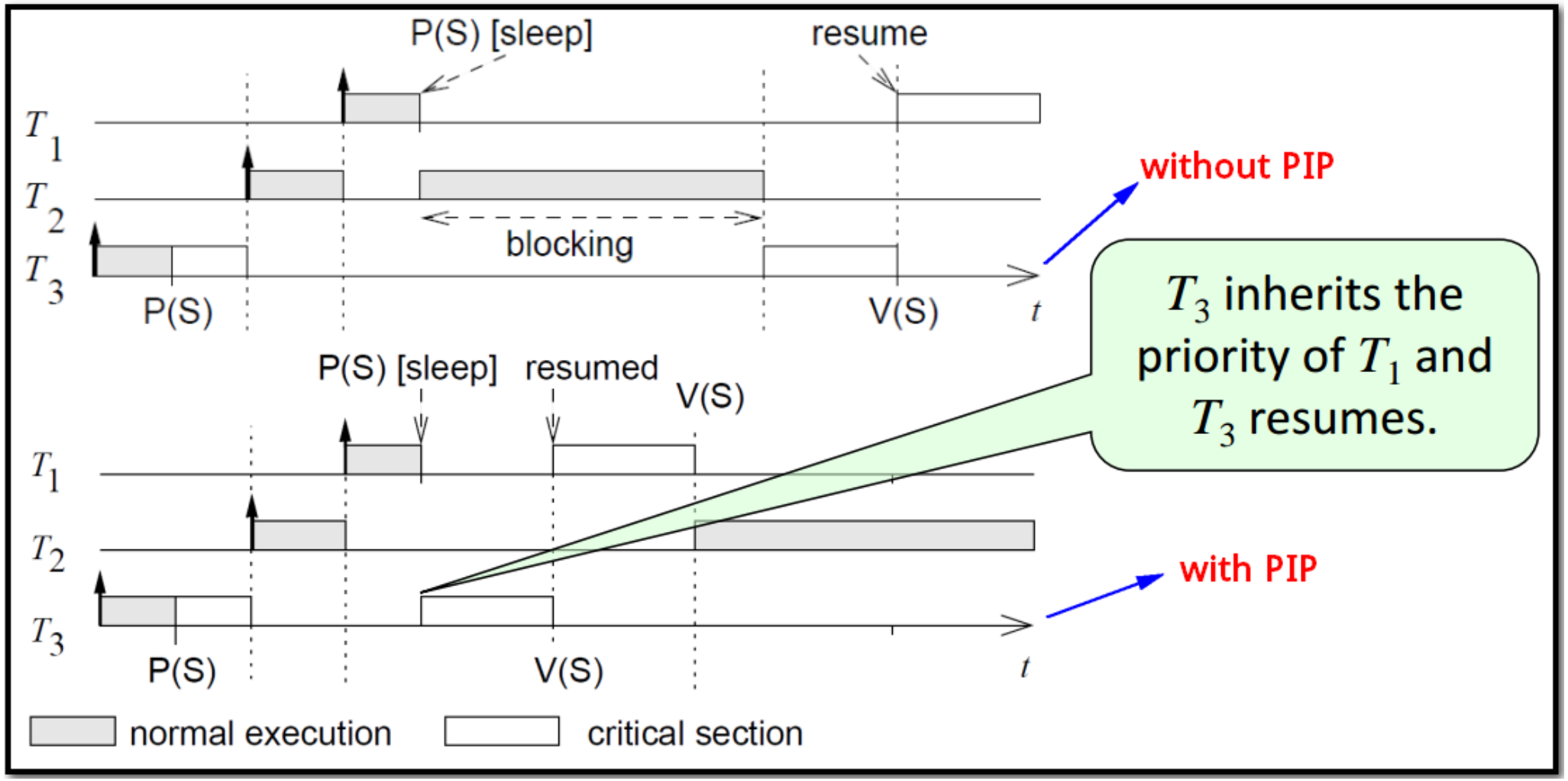
Read-Copy Update (RCU)

RCU is a very specialized primitive, and it is exceedingly important to **use the right tool for the job**.

For a great many jobs, normal **locking** remains the **best tool**.

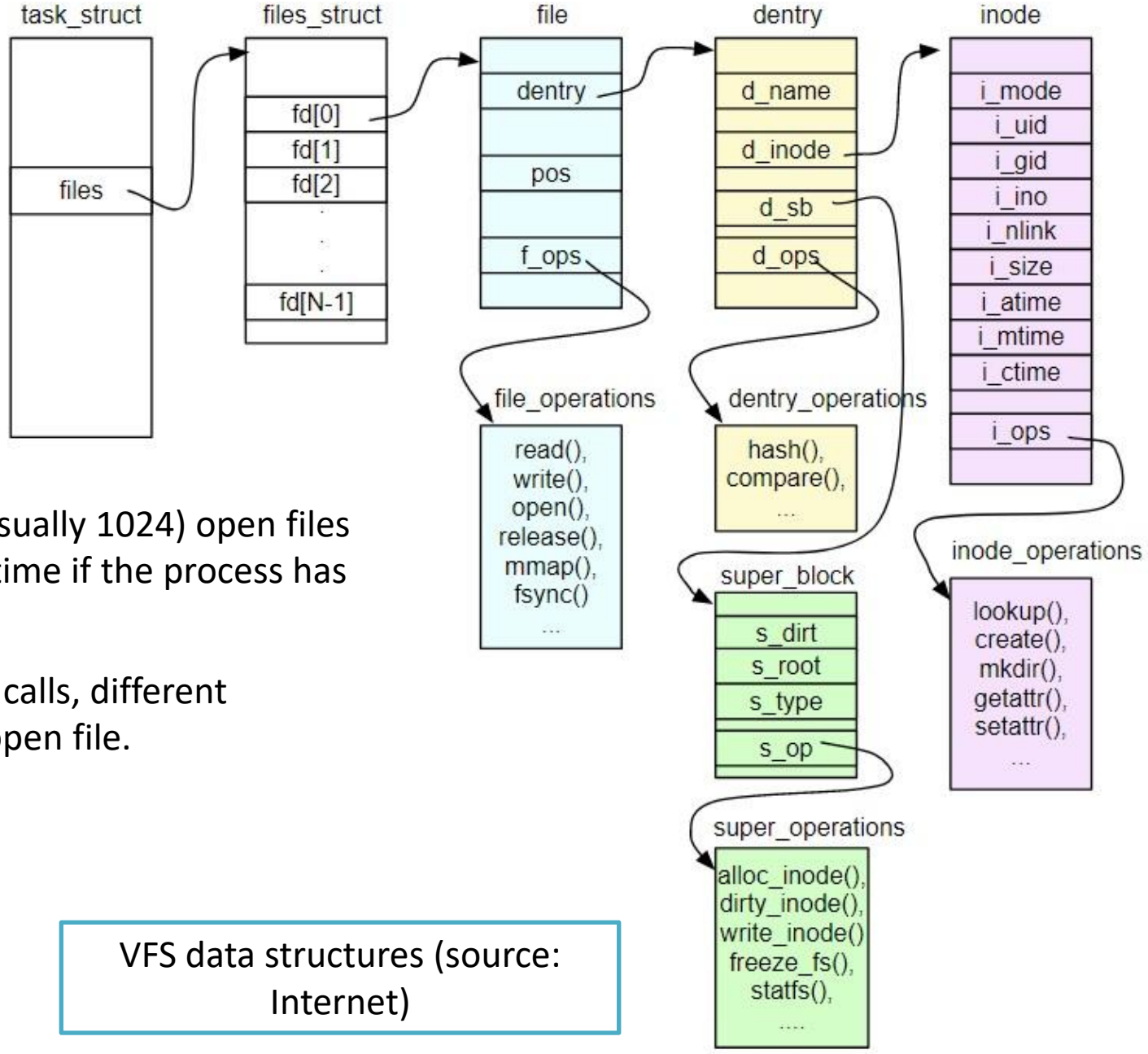
Almost all RCU uses in the Linux kernel use locking to **protect updates**, which does place a hard **upper limit** on **RCU's fraction of synchronization primitives**.

Priority inversion and priority inheritance



[Solutions for Priority Inversion in Real-time Scheduling]

Process data structures with file information



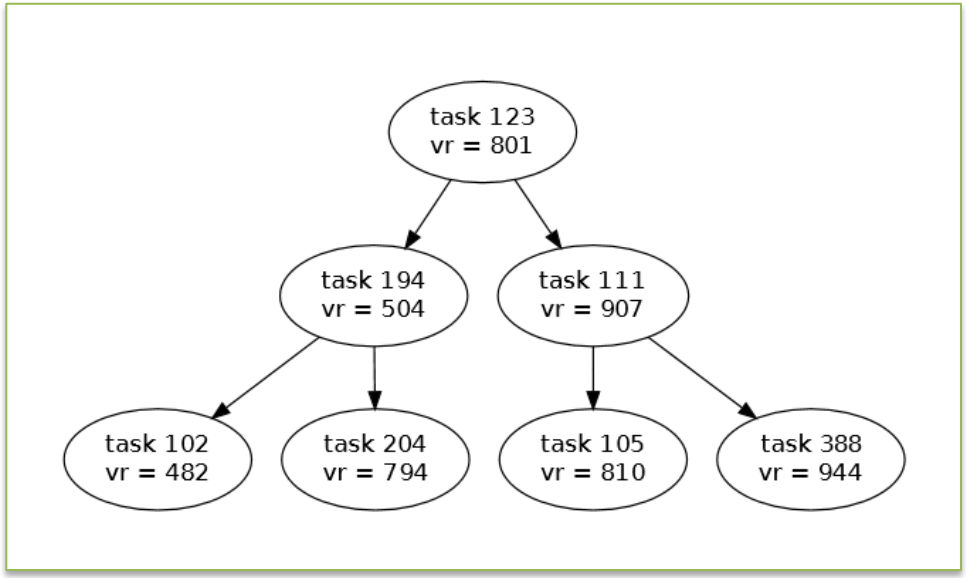
The process can have **NR_OPEN** (usually 1024) open files (this limit can be **increased** at run time if the process has superuser privileges).

Thanks to **dup()** and **fork()** system calls, different descriptors can refer to the same open file.

VFS data structures (source: Internet)

Completely Fair Scheduler

Red-black tree
for CFS scheduler
process selection **O(1)** insertion
O(log(n))



- When a task has finished running on the CPU, all of the other tasks in the tree need to have their **unfairness** increase.
- To prevent having to update **all** of the tasks in the tree the scheduler maintains a per-task **vruntime** statistic.
- This is the **amount of total nanoseconds** that the task has spent **running** on a **CPU weighted** by its **niceness**.
- Thus, instead of updating all other tasks to be more **unfair** when a task has **finished** running on the CPU, we update the **leaving** task to be more fair than others by increasing its **virtual runtime**.
- The scheduler always selects the **most unfairly treated** task by selecting the task with the **lowest vruntime**.

http://cs.unm.edu/~eschulte/classes/cs587/data/bfs-v-cfs_groves-knockel-schulte.pdf

A short story of sched_ext

David Vernet

- Patchset v1: 2022-11-30, patchset v2: 2023-01-27
- Kernel Report 2023: The extensible scheduler class (write complete CPU schedulers in BPF)
  - It allows users to write a custom scheduling policy using BPF without modifying the kernel code.
  - BPF provides a safe kernel programming environment.
  - BPF verifier ensures that your custom scheduler has neither a memory bug nor an infinite loop.
  - BPF scheduler can be updated without reinstalling the kernel and rebooting a server.
  - Developed by engineers from **Meta** and **Google**.
  - Why: easy experimentation, faster scheduler development, ad hoc schedulers for special workloads.
  - Why ***not***: added mainteance burden, benchmark gaming, vendors may require specific schedulers, ABI concerns, redirection of work on core scheduler.
  - Rejected by scheduler maintainer (**Peter Zijlstra**).
- September 30, 2024: Linus has released 6.12-rc1 – sched_ext got merged.
- https://github.com/sched-ext/scx/

Peter Zijlstra

Changwoo Min's introduction to the sched_ext scheduling class

Earliest Eligible Virtual Deadline First (EEVDF)

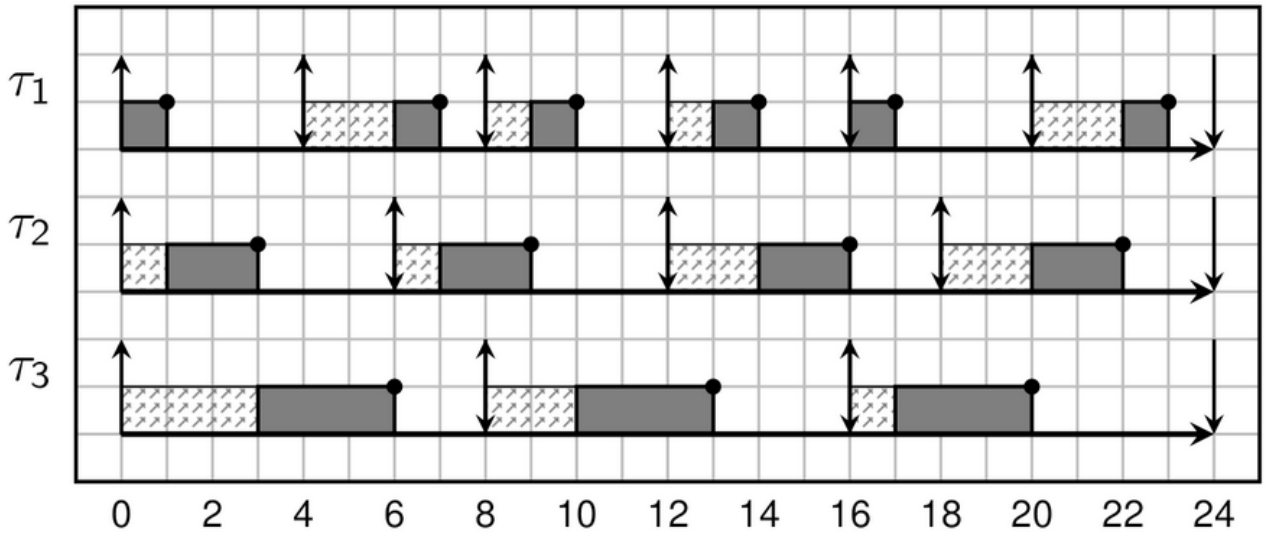
Peter Zijlstra

- An EEVDF CPU scheduler for Linux, Jonathan Corbet, March 9, 2023.
- Completing the EEVDF scheduler, Jonathan Corbet, April 11, 2024.
- Work by **Peter Zijlstra**.
- (**Oct 30, 2023**) Merged as an option for the **6.6 kernel**.
- It should provide **better performance and fairness** while **relying less on fragile heuristics**. The merge message notes that there may be some rare performance regressions with some workloads, and that work is ongoing to resolve them.
- One place where there is a desire for improvement is in the handling of **latency**.
- Scheduling algorithm is not new; it was described in this 1995 paper by Ion Stoica and Hussein Abdel-Wahab.
- (**April 05, 2024**) Peter Zijlstra posted a patch series intended to finish the EEVDF work. Beyond some fixes, this work includes a significant behavioral change and a new feature intended to **help latency-sensitive tasks**.
- The amount of CPU time given to any two processes (with the same nice value) will be the same, but the low-latency process will get it in a **larger number** of **shorter slices**.
- Currently a **default scheduler** (replaced CFS).

The Deadline scheduler

Worst-case execution time

Task	Runtime (WCET)	Period
T_1	1	4
T_2	2	6
T_3	3	8



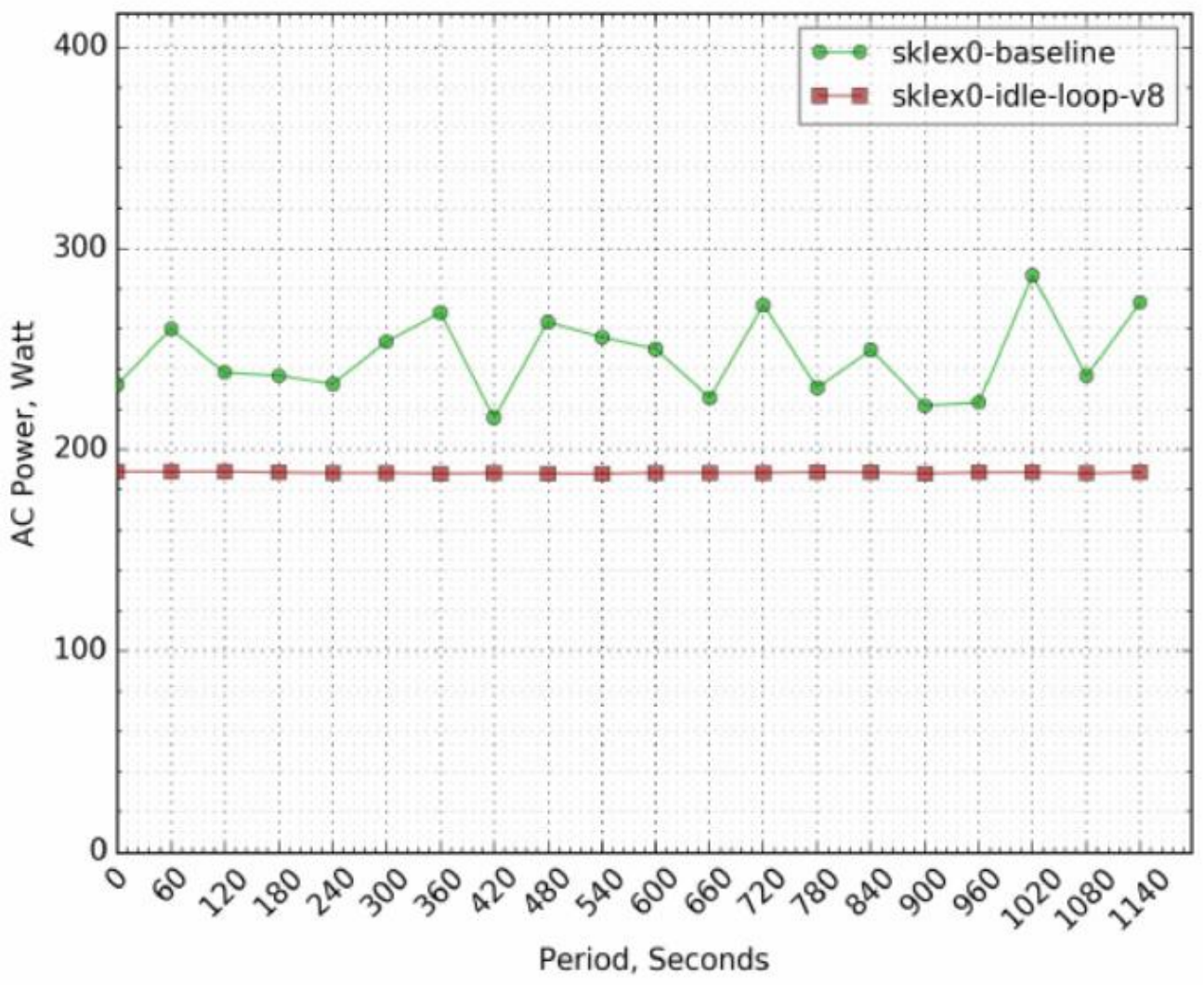
https://lwn.net/Articles/743740/

it is not possible to use a fixed-priority scheduler to schedule this task set while meeting every deadline; regardless of the assignment of priorities, one task will not run in time to get its work done.

Deadline scheduling gets away with the notion of process priorities. Instead, processes provide three parameters: runtime, period, and deadline. A SCHED_DEADLINE task is guaranteed to receive "**runtime**" microseconds of execution time every "**period**" microseconds, and these "runtime" microseconds are available within "**deadline**" microseconds from the beginning of the period. The task scheduler uses that information to run the process with the **earliest deadline first** (EDF).

Idle power (Intel OTC Server Power Lab)

The green line is with the old idle loop, the red is with the new: power consumption is less under the new scheme, and moreover it is much more predictable than before.



CPU Idle Loop Rework, Rafael J. Wysocki (Intel), 2018.

Scheduling – Arm big.LITTLE CPU chip

Scheduling for asymmetric Arm systems, Jonathan Corbet, November 2020.

The **big.LITTLE architecture** placed **fast** (but power-hungry) and **slower** (but more power-efficient) CPUs in the same **system-on-chip** (**SoC**); significant scheduler changes were needed for Linux to be able to properly distribute tasks on such systems.

Putting tasks on the wrong CPU can result in poor performance or excessive power consumption, so a lot of work has gone into the problem of **optimally distributing workloads** on big.LITTLE systems.

When the scheduler gets it wrong, though, performance will suffer, but things will still work.

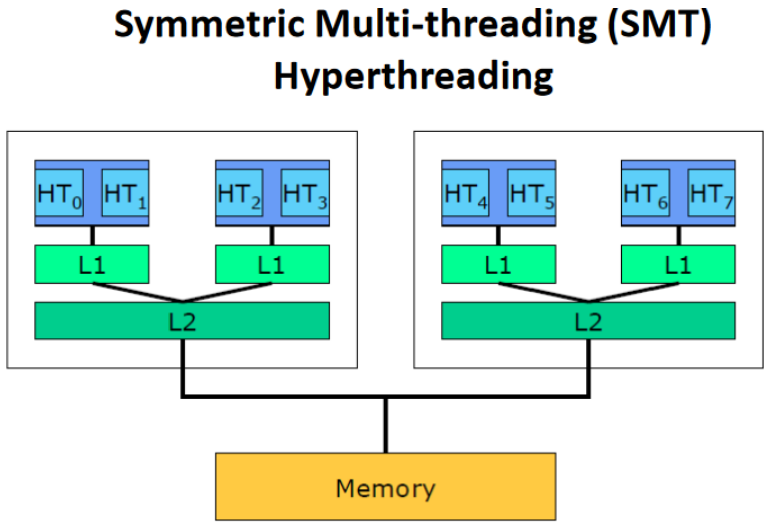
Future Arm designs, include systems where some CPUs can run **both 64-bit and 32-bit** tasks, while others are **limited to 64-bit tasks** only. The result of an incorrect scheduling choice is no longer a matter of performance; it could be catastrophic for the workload involved.

What should happen if a 32-bit task attempts to run on a 64-bit-only CPU?

- Kill the task or
- recalculate the task's CPU-affinity mask?

Cortex A57/A53 MPCore big.LITTLE CPU chip

Core scheduling



Core scheduling, Jonathan Corbet, February 2019.

SMT (**simultaneous multithreading**) increases performance by turning one physical CPU into two virtual CPUs that share the hardware; while one is waiting for data from memory, the other can be executing. Sharing a processor this closely has led to security issues and concerns for years, and many security-conscious users disable SMT entirely.

On kernels where core scheduling is enabled, a **core_cookie** field is added to the task structure. These cookies are used to define the trust boundaries; two processes with the same cookie value trust each other and can be allowed to run simultaneously on the same core. (Peter Zijlstra)

Completing and merging core scheduling, Jonathan Corbet, May 2020.

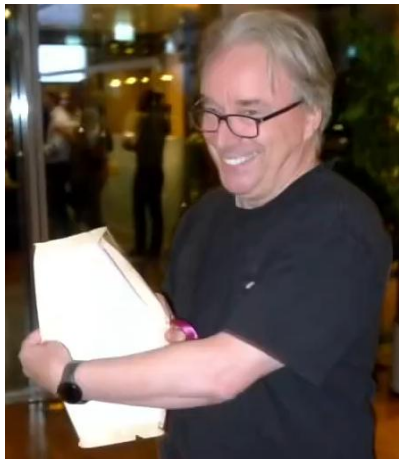
A set of virtualization tests showed the system running at **96%** of the performance of an unmodified kernel with **core scheduling enabled**; the 4% performance hit hurts, but it's far better than the **87%** performance result measured for this workload with **SMT turned off** entirely.

The all-important kernel-build benchmark showed almost no penalty with core scheduling, while turning off SMT cost 8%.

History of PREEMPT_RT

- Small group of core developers: Ingo Molnar, Steven Rostedt, Thomas Gleixner, Sebastian A. Siewior, John Ogness.
- Started delivering patches in 2004.
- Merged last brick on a road – the first ever physical pull request (ver. 6.12, September 19, 2024).
- Real time patches: https://lwn.net/Kernel/Index/#Realtime.
- Today almost all required parts of the PREEMPT_RT are part of the common Linux code base.
- Work is continued.

*Realtime kernels are unlikely to become the default, simply because there's some **small performance overhead** from using the realtime config option. But with all the necessary code being part of the mainline kernel, it's certainly possible that some distributions might turn it on by default or make it easier to turn on.*
(https://lwn.net/Articles/990985/)



On September 19, **Thomas Gleixner** delivered the pull request for the realtime preemption enablement patches to **Linus Torvalds** — in printed form, wrapped in gold, with a ribbon, as Torvalds had requested. It was a significant milestone, marking the completion of a project that required 20 years of effort. (https://lwn.net/Articles/990985/)

Admiral Grace Hopper explains the nanosecond

6,2 tys. km

1906-1992

- I called over to the engineering building and I said: „Please cut off a nanosecond and send it over to me”.
- I wanted a piece of wire which would represent the maximum distance that electricity could travel in a billionth of a second. Of course, it wouldn’t really be through wire. It’d out in space; **the velocity of light**.
- So, if you start with the velocity of light, you’ll discover that a **nanosecond** is **11.8 inches** long (**29,97 cm**)

- At the end of about a week, I called back and said: „I need something to compare this to. Could I please have a microsecond?”
- Here is a **microsecond**, **984 feet** (**29992,32 cm**). I sometimes think we ought to hang one over every programmer’s desk (or around their neck) so they know when they’re throwing away when they throw away microseconds.

https://www.youtube.com/watch?v=9eyFDBPk4Yw

Brendan Gregg

But ...

Be aware, shouting in the datacenter
is not recommended

Vibration can badly influence disk latency

https://www.youtube.com/watch?v=tDacjrSCeq4
https://www.youtube.com/watch?v=lMPozJFC8g0

The Kernel Report 2024 – what to expect in 2025?

Jonathan Corbet, Executive Editor, LWN.net & Kernel Documentation Maintainer

- Rust (Commiting to Rust in the kernel)
  - Memory-safe systems programming, eliminates whole classes of bugs.
  - First merged in October 2022 for 6.1 (as an experiment), slow development.
  - Conclusion from the Kernel Maintainers Summit in 2024: Rust in the kernel is viable.
  - Rust concerns – not easy to learn, language stability, Rust/C API correspondence, getting abstractions upstream – if they are not used yet.
  - What might be merged (lots if infrastructure, Nova – NVIDIA GPU driver, Apple GPU driver).
- CPU scheduler creativity
  - For decades there could be only one CPU scheduler.
  - Sched_ext – a new scheduling class, based on BPF, merged in November 2024 for 6.12.
    - Anybody can write a CPU scheduler.
    - Quick and safe iteration.
    - Focus on one use case.
- Security
  - The problems with CVE (Common Vulnerabilities and Exposure) numbers
    - Many vulnerabilities never get CVEs.
    - The kernel is now a CNA – Certficate Numbering Authority.