Zaawansowane Systemy Operacyjne

(wykład obieralny, semestr letni)

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

(komplet materiałów, kryteria zaliczania)

Janina Mincer-Daszkiewicz

Andrzej Jackowski, Wojciech Ciszewski, Maciej Matraszek

Linux history

Linus Torvalds in 2023

[in conversation with Dirk Hohndel at OSS Japan]

[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

[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/

- **2024-02-23**, latest stable version **6.7.6**
- **2024-02-18**, latest mainline **6.8-rc05**

Numbering of the kernel versions – see lab notes or [Wikipedia]

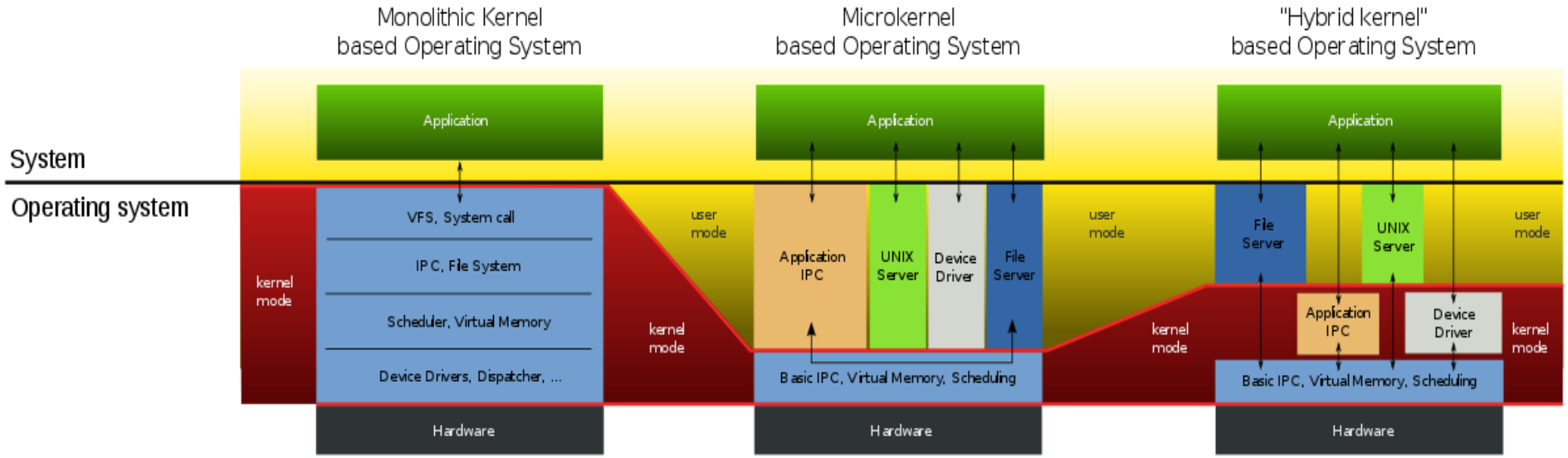
Andrew Tanenbaum in 2012

2023 Kernel Maintainers Summit group photo

Back row: Alexei Starovoitov, Dave Chinner, Konstantin Ryabitsev, Thomas Gleixner, Jakub Kicinski, Josef Bacik, Dan Williams, Wolfram Sang, Rafael Wysocki, Kees Cook, Greg Kroah-Hartman, Jiri Kosina.

Front row: Tejun Heo, Jens Axboe, Sasha Levin, Ted Ts'o, Jan Kara, Linus Torvalds, Martin Petersen, Christian Brauner, Steve Rostedt, Arnd Bergmann.

Floor: Miguel Ojeda.

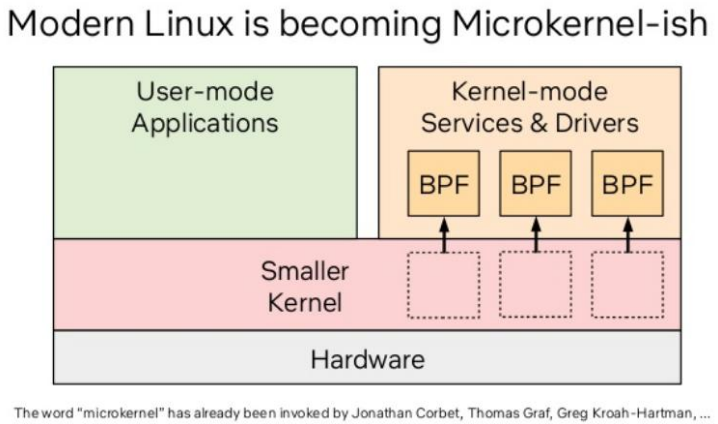


Structure of monolithic kernel, microkernel and hybrid kernel-based operating systems (source: Wikipedia)

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 ...



Linux Development

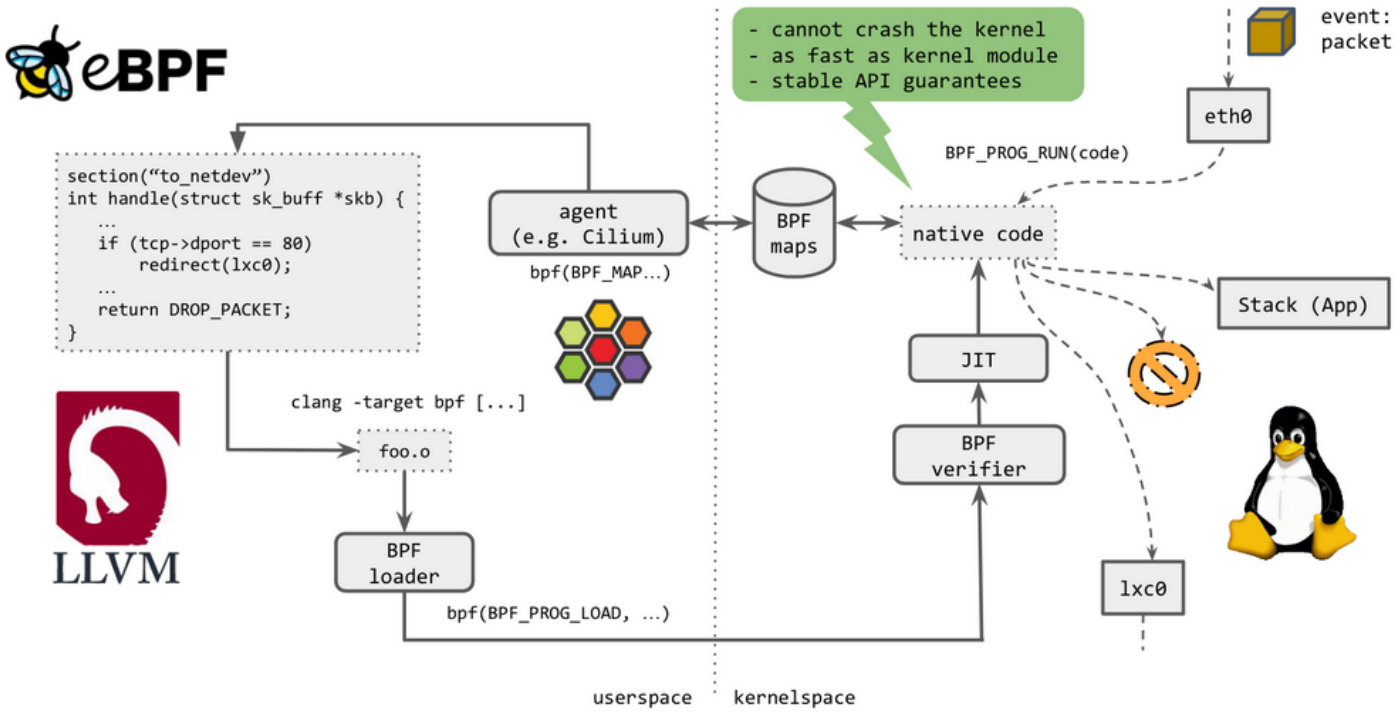
The Good:

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

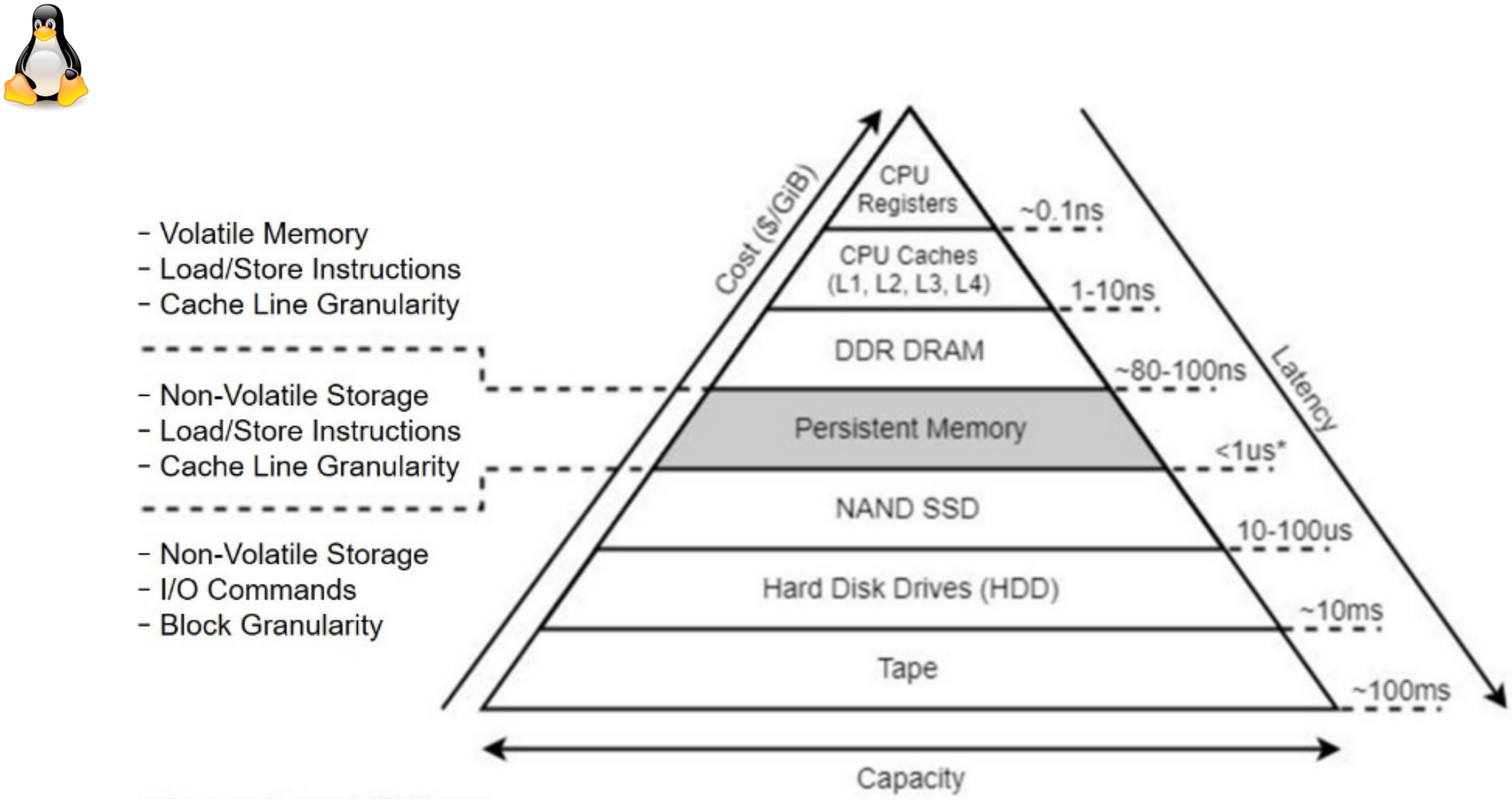
The Bad:

- 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

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



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



[Memory hierarchy, Intel]

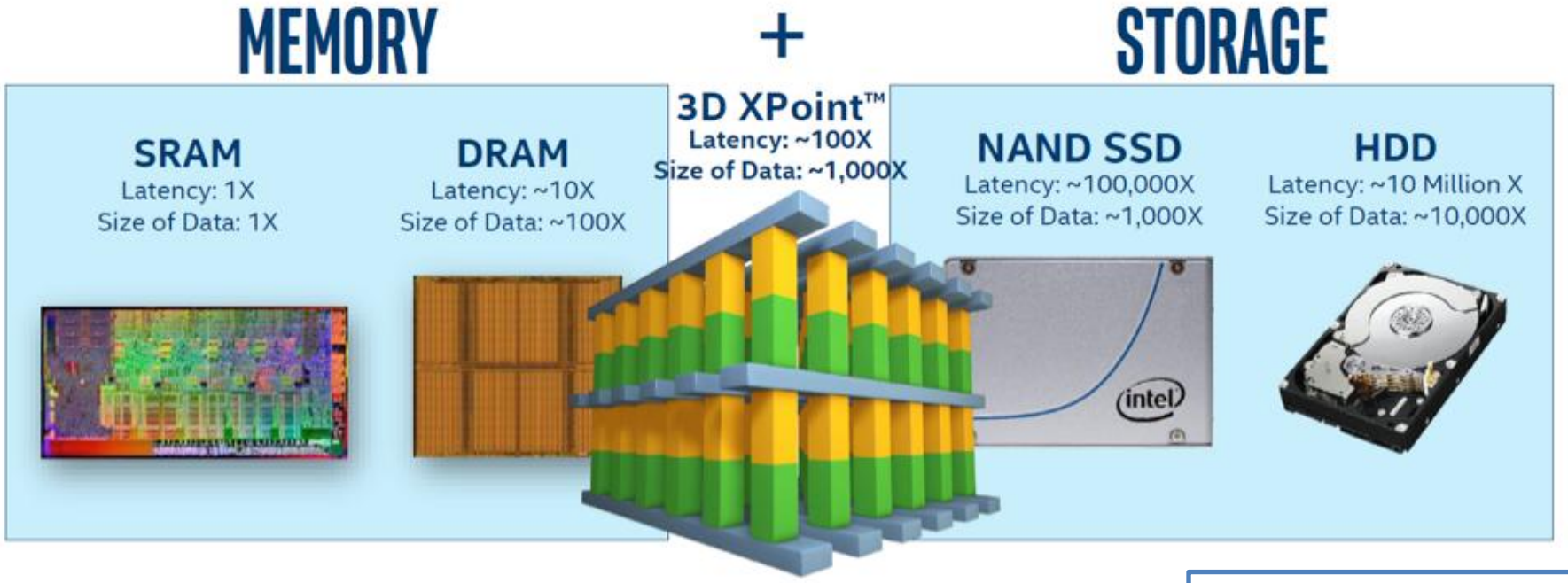


Image source: Intel

A [non-volatile memory](#) (NVM) technology.

In 2015, **Intel** and **Micron** claimed **3D XPoint** would be up to **1,000 times faster** and have up to **1,000 times more endurance** than NAND flash, and have **10 times the storage density** of conventional memory. Up to **half the cost** of DRAM.

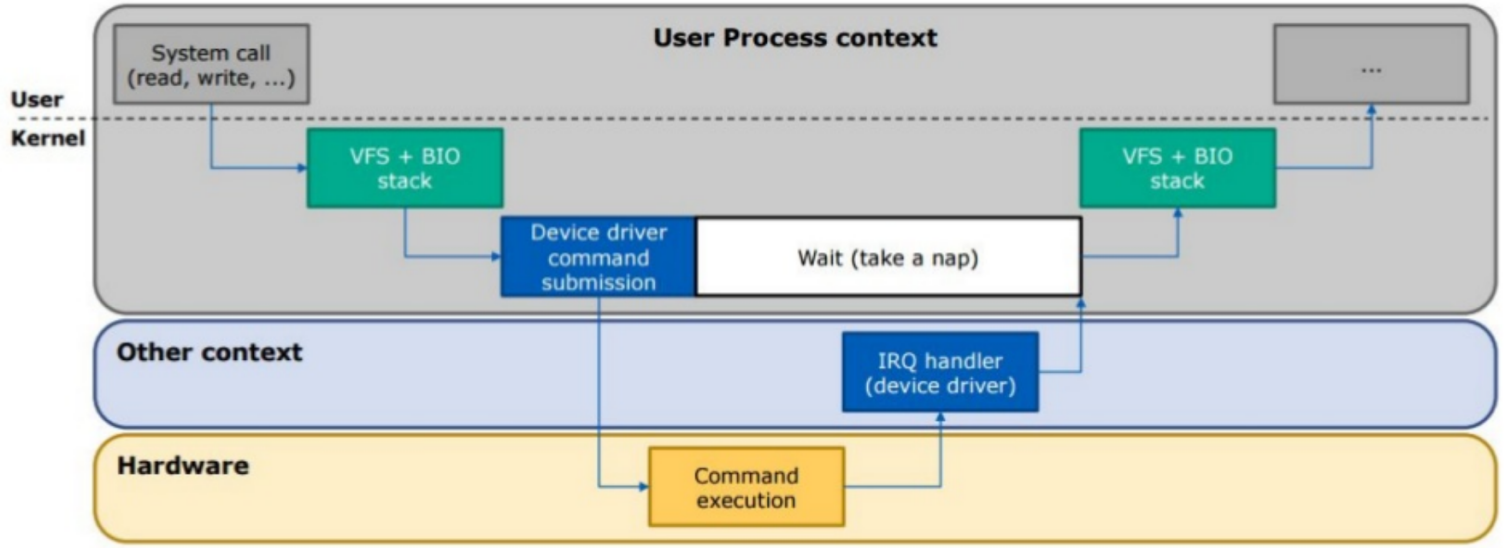
2023 – discontinued

Available on the open market under the brand name **Optane** (Intel) since April 2017.

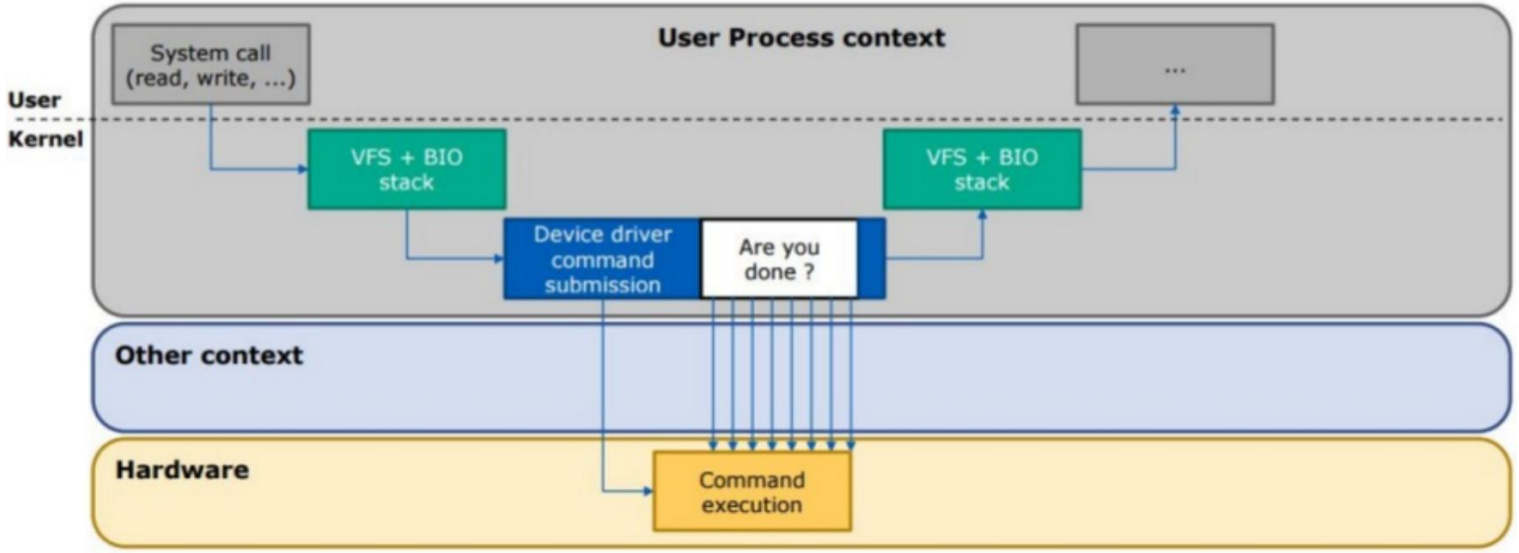
[Intel 3D XPoint Technology](#), Disruptive Technologies Session, 2015 HPC User Forum

[3D XPoint™ Technology Revolutionizes Storage Memory](#)

Interrupt based I/O completion model



Polling based I/O completion model

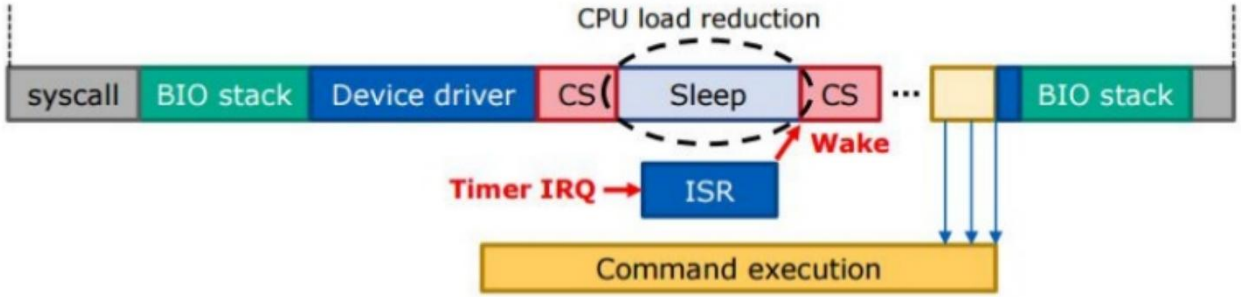


[The Linux Block Layer. Built for Fast Storage](), Sagi Grimberg, June 2018

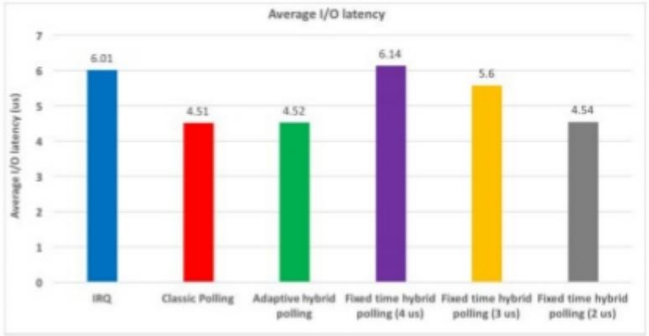
But what about CPU% - can we go hybrid?

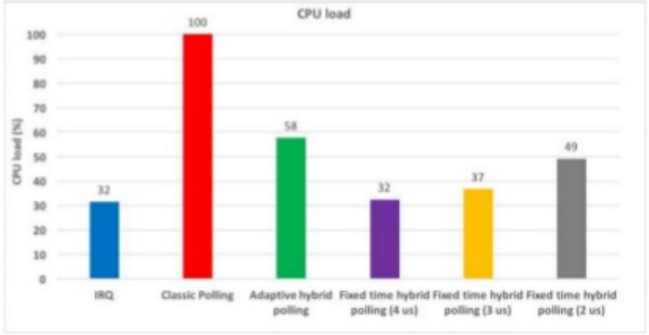
- Yes!
  - We have all the statistics framework in place, let's use it for hybrid polling!
  - Wake up poller after ½ of the mean latency.

Hybrid Polling



Hybrid polling - Performance

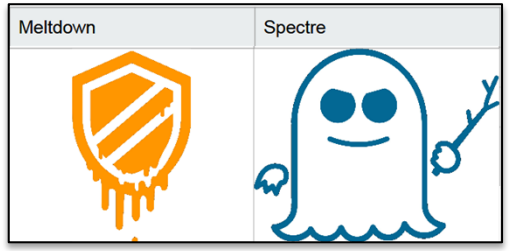




Polling has been added in v4.1.

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

Process address space KASLR and KAISER (KPTI)

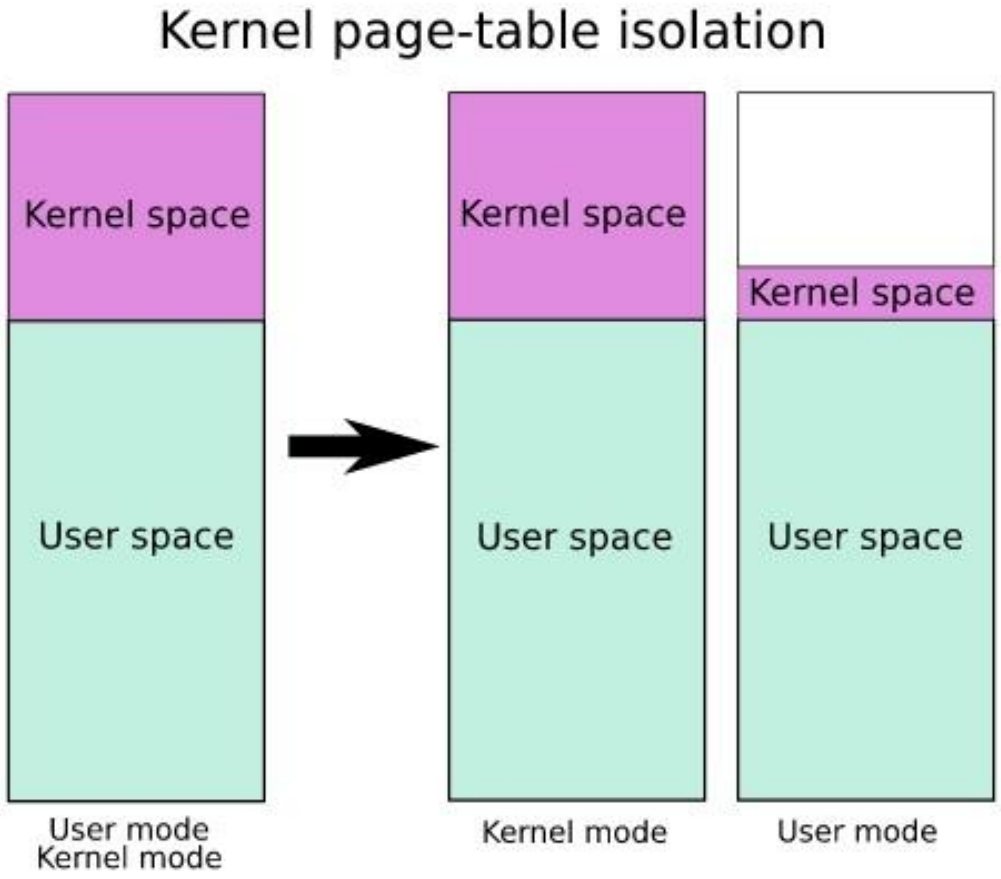


Additional reading

- KASLR is Dead: Long Live KASLR (June 2017) – paper introducing KAISER.
- KAISER: hiding the kernel from user space, Jonathan Corbet, November 2017.
- The current state of kernel page-table isolation, Jonathan Corbet, December 2017.
- Linux Documentation in GitHub, January 2018.
- Meltdown and Spectre, Piotr Zalas, March 2018.

KASLR – Kernel Address Space Layout Randomization

KAISER – Kernel Address Isolation to have Side-channels Efficiently Removed (renamed later to KPTI)

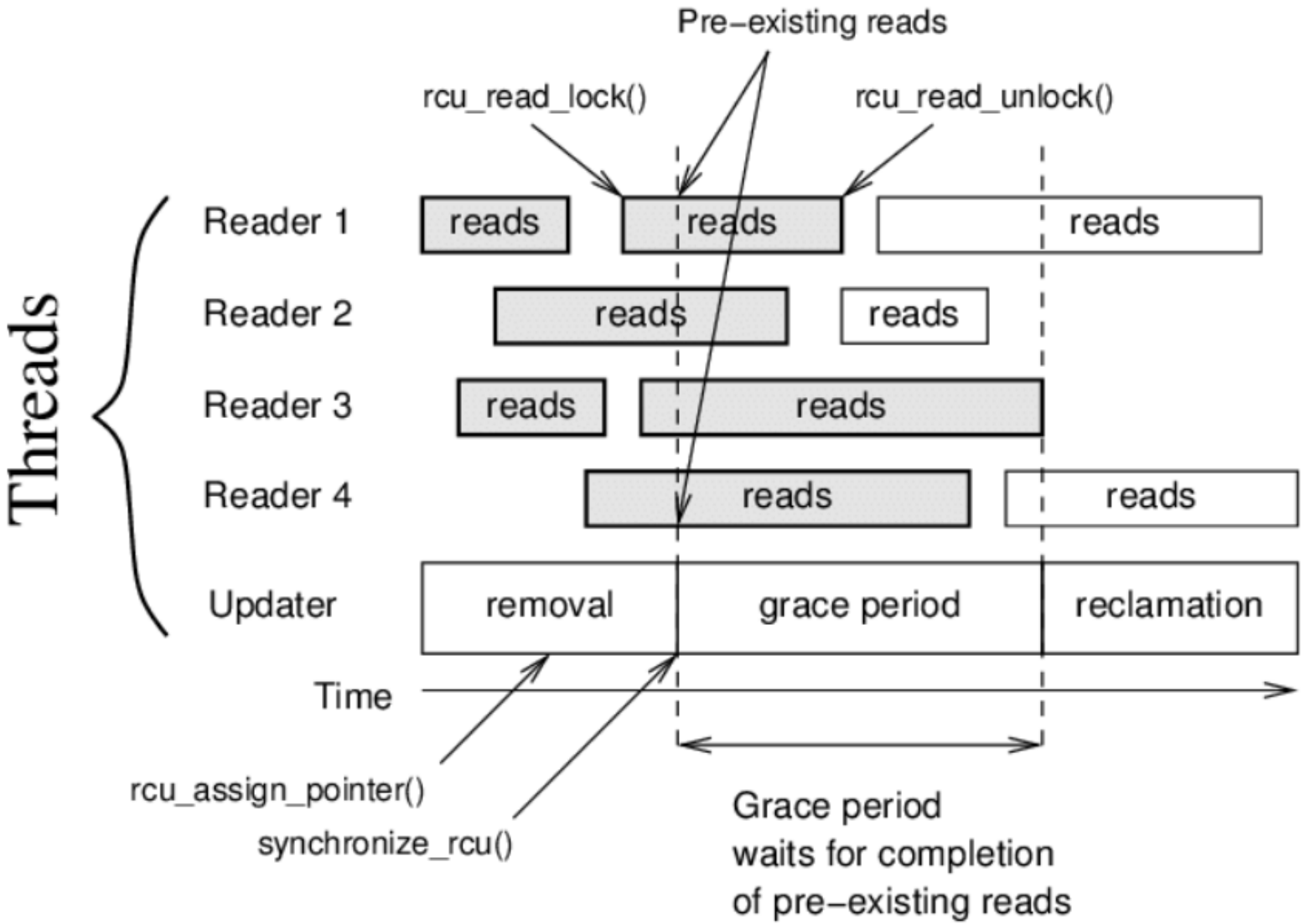


Kernel Page Table Isolation – KPTI (source: Wikipedia)

Read-Copy Update (RCU)

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

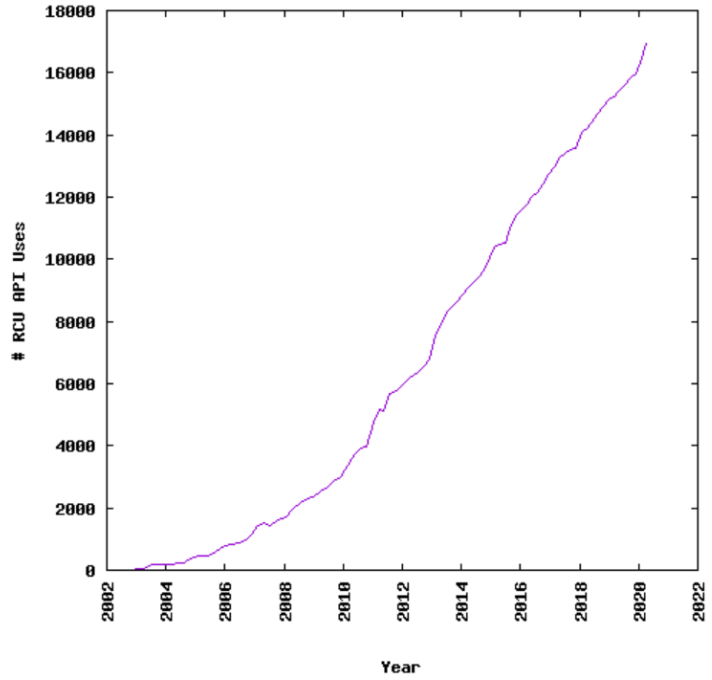
Quiescent state – any point in the thread execution where the thread does not hold a reference to shared memory.

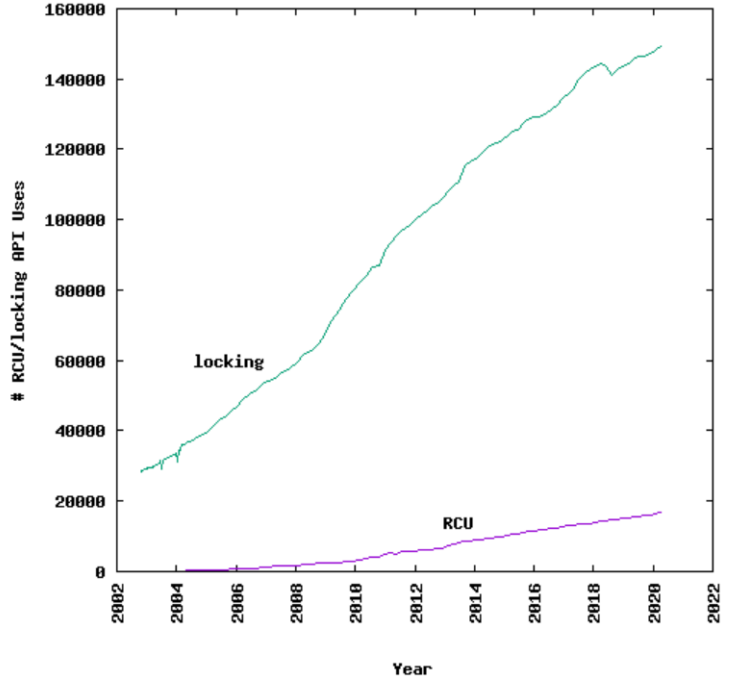


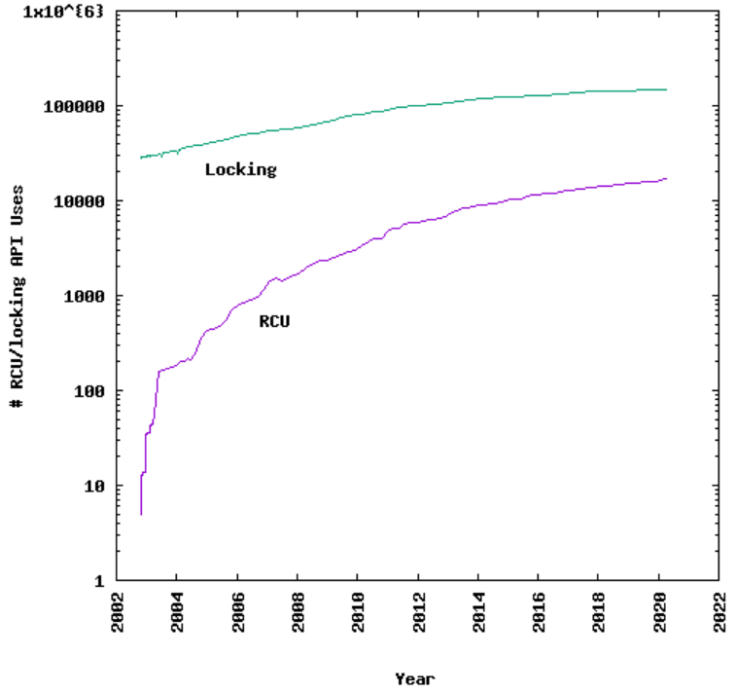
Schematic of RCU Grace Period and Read-Side Critical Sections

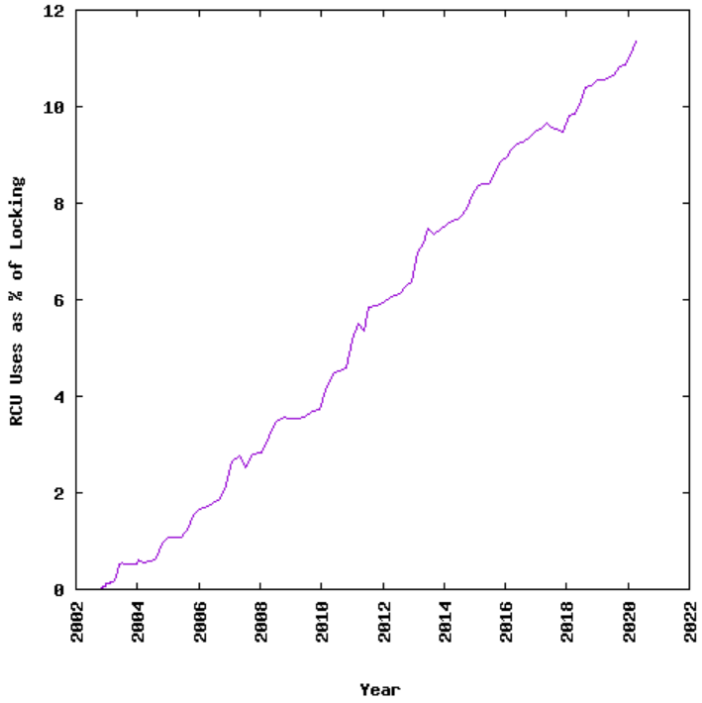
Paul McKenney

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









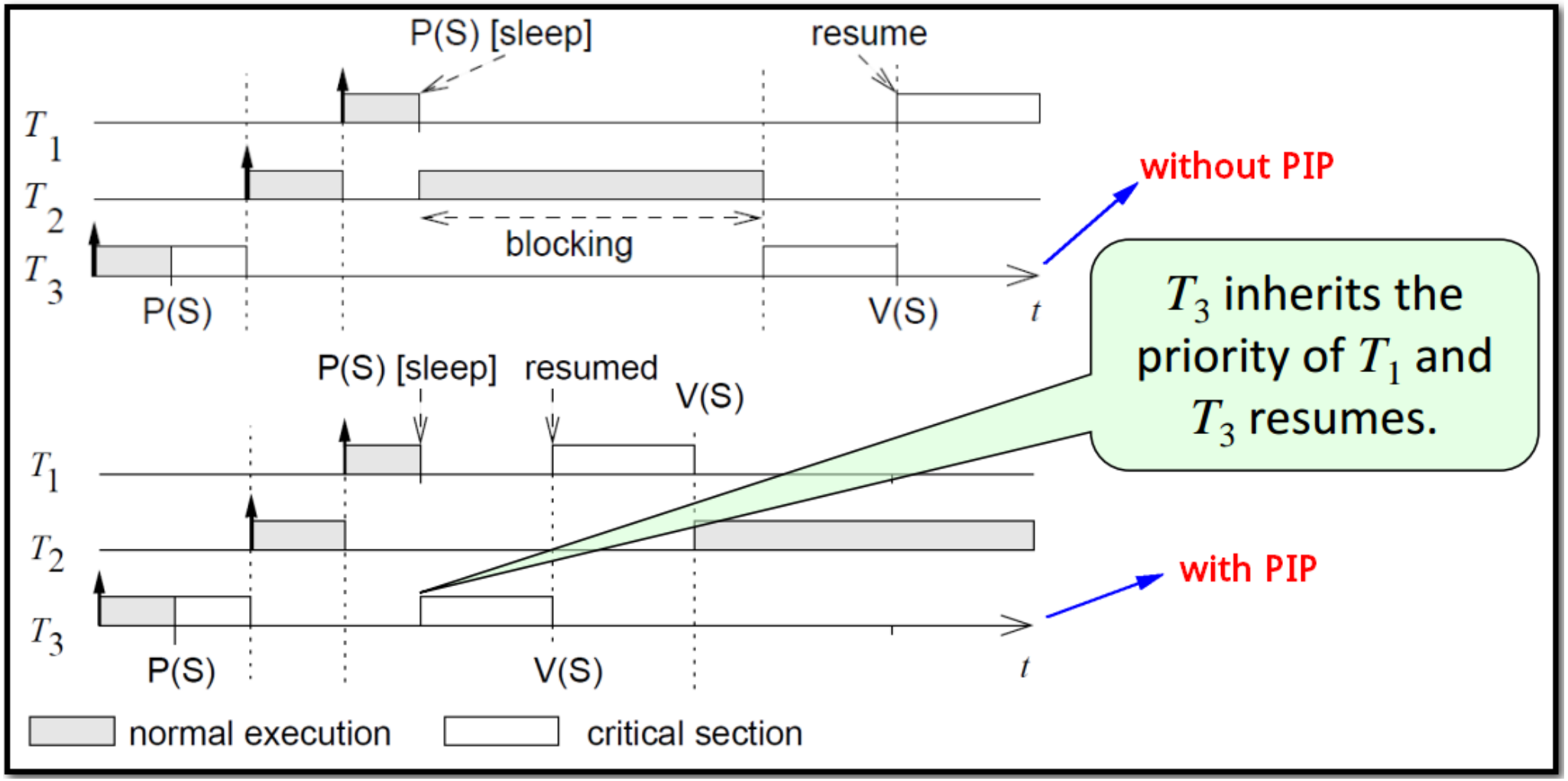
Read-Copy Update (RCU)

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**.

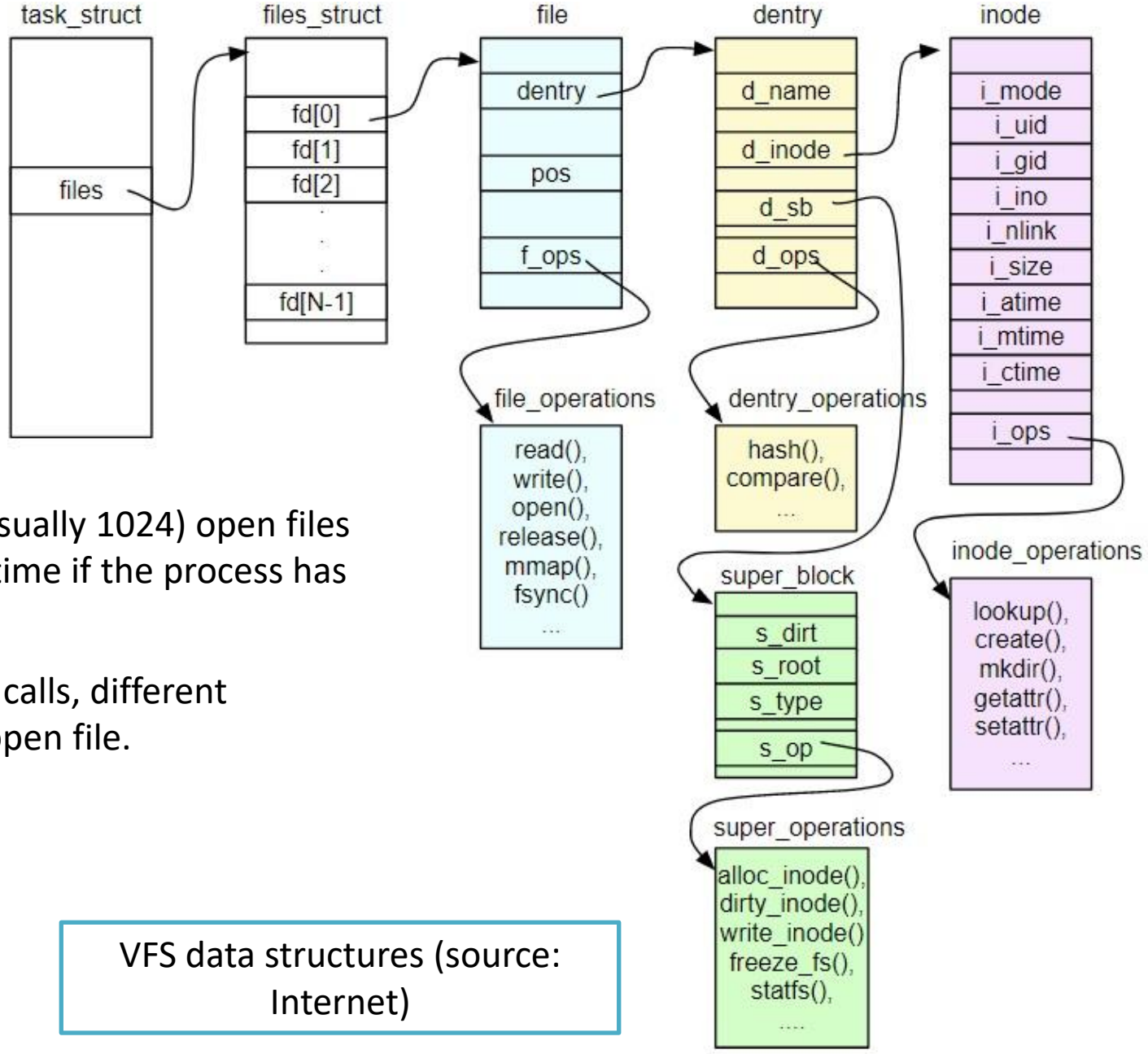
Almost all RCU uses in the Linux kernel use locking to **protect updates**, which does place a hard **upper limit** on **RCU's fraction of synchronization primitives**.

Priority inversion and priority inheritance



[Solutions for Priority Inversion in Real-time Scheduling]

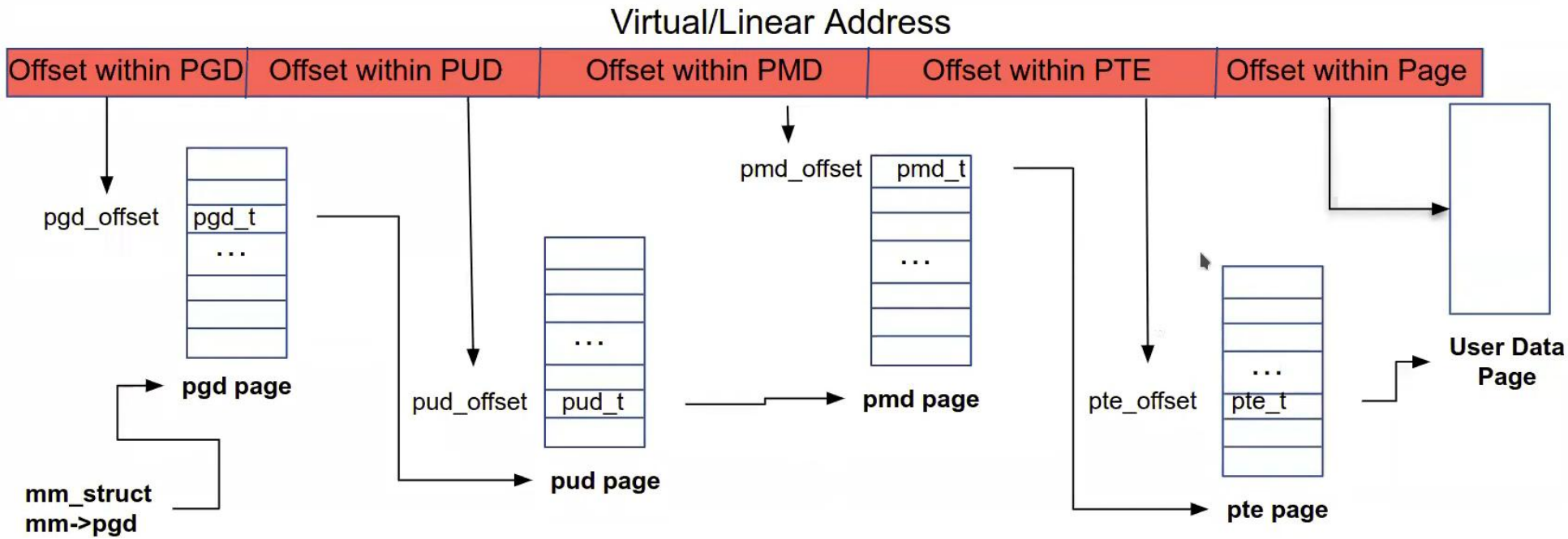
Process data structures with file information



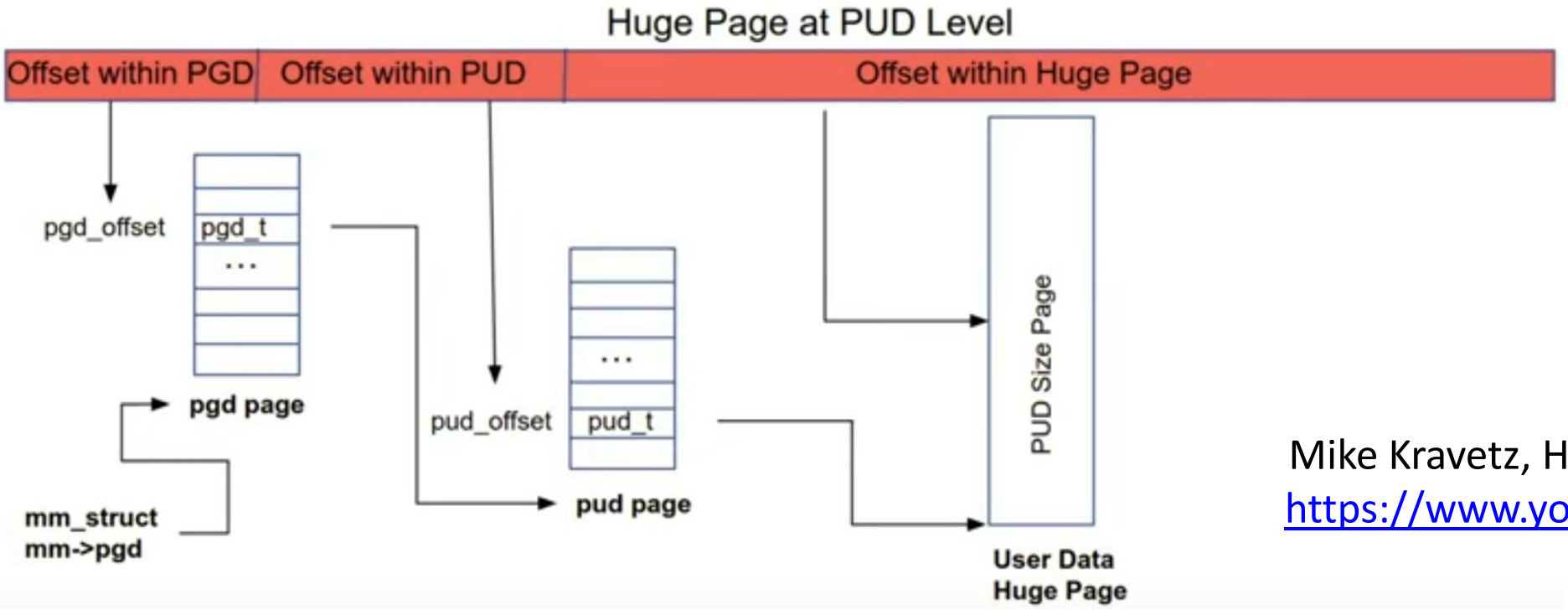
The process can have **NR_OPEN** (usually 1024) open files (this limit can be **increased** at run time if the process has superuser privileges).

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

VFS data structures (source: Internet)



Page size
4KB



Page size
1GB

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

Scheduler developers

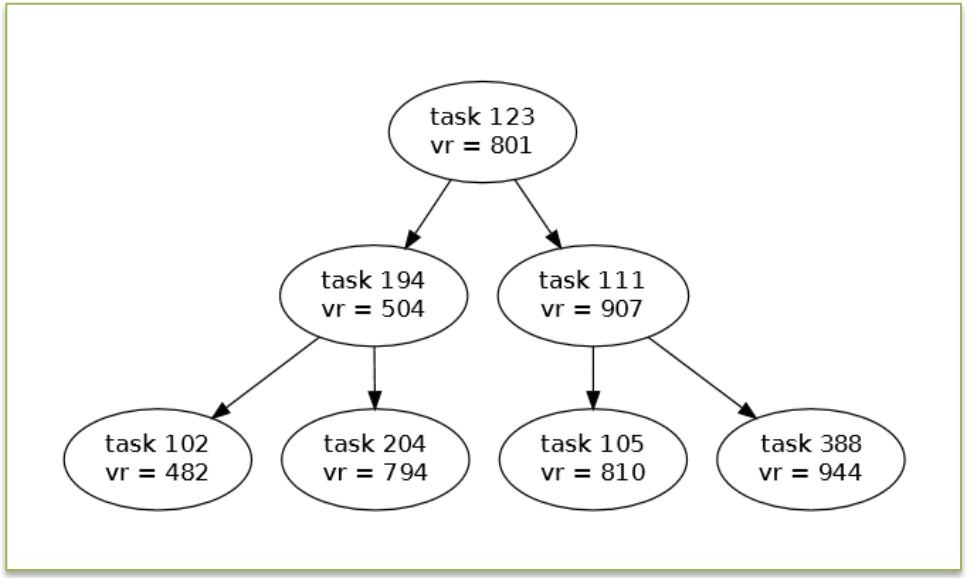
Ingo Molnar – official developer of schedulers in the Linux kernel, full-time programmer, employee of RedHat, from Hungary.

Peter Zijlstra – co-maintainer of various Linux kernel subsystems including: the scheduler, locking primitives and performance monitoring, currently employed by Intel where he assists in enabling new hardware features on Linux. (2017)

Con Kolivas – anesthesiologist from Australia, hobbyist, digging in the kernel learned to program, mainly interested in responsiveness and good performance of the scheduler in "home" applications.

Completely Fair Scheduler

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



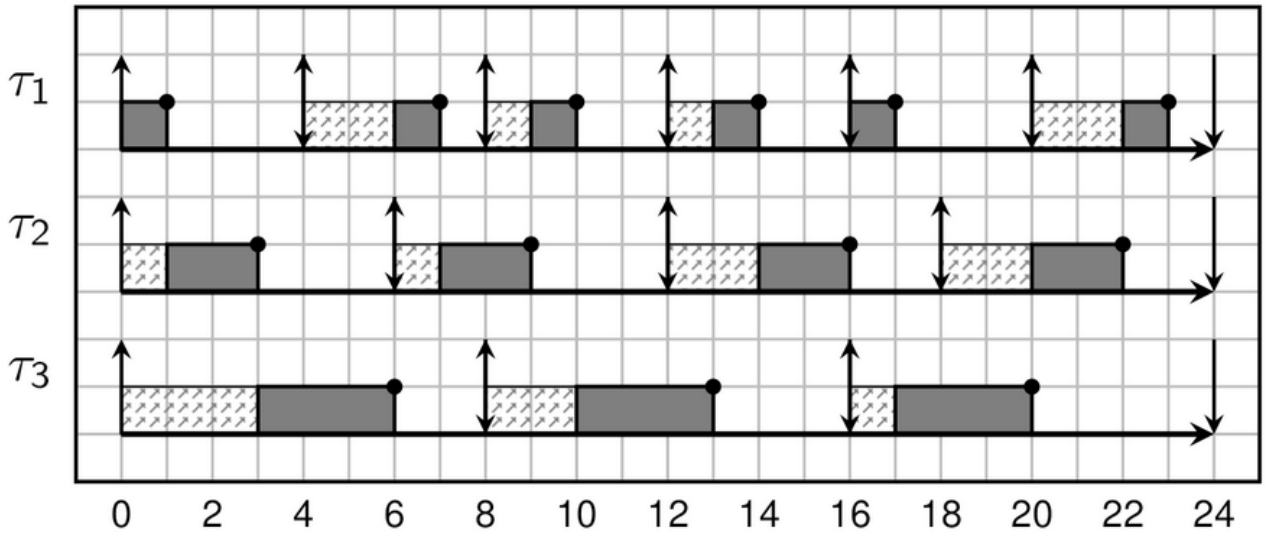
- When a task has finished running on the CPU, all of the other tasks in the tree need to have their **unfairness** increase.
- To prevent having to update **all** of the tasks in the tree the scheduler maintains a per-task **vruntime** statistic.
- This is the **amount of total nanoseconds** that the 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

The Deadline scheduler

Worst-case execution time

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



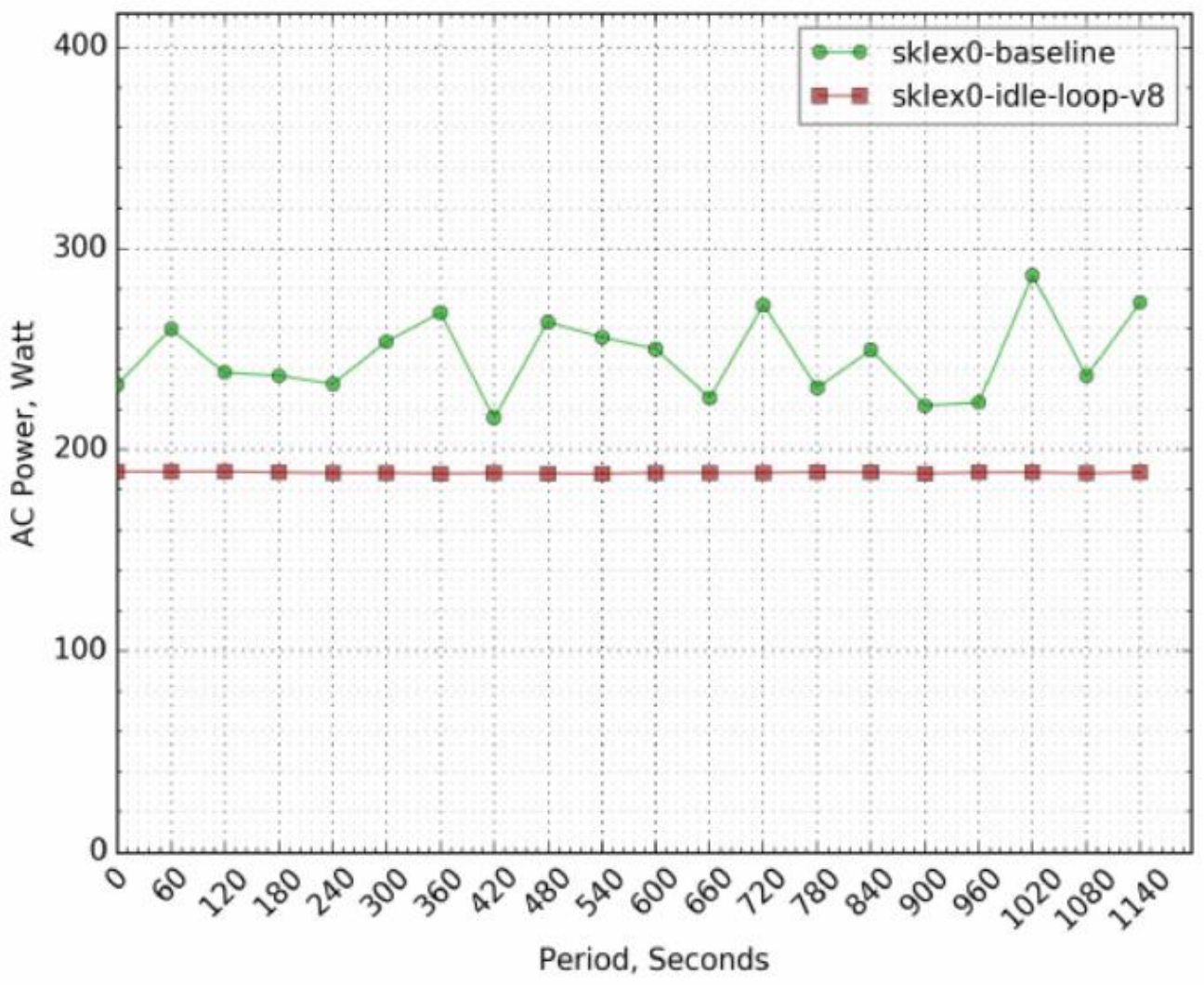
https://lwn.net/Articles/743740/

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.

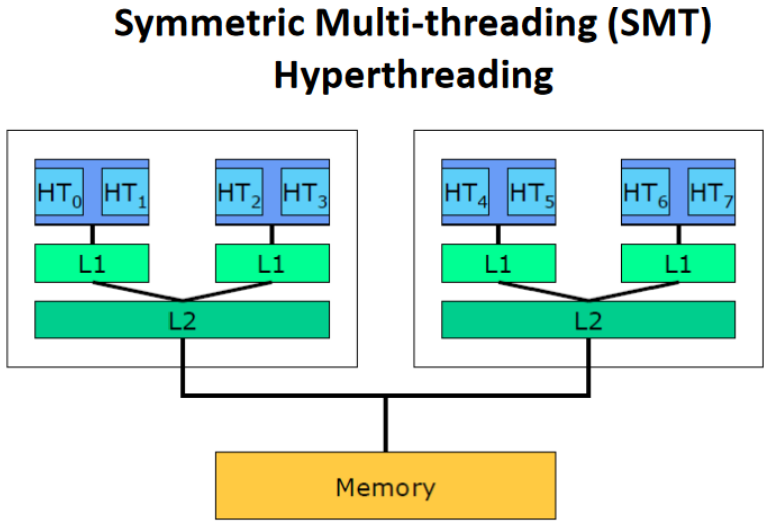
Future Arm designs, include systems where some CPUs can run **both 64-bit and 32-bit** tasks, while others are **limited to 64-bit tasks** only. The result of an incorrect scheduling choice is no longer a matter of performance; it could be catastrophic for the workload involved.

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%.

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

But ...

Brendan Gregg

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

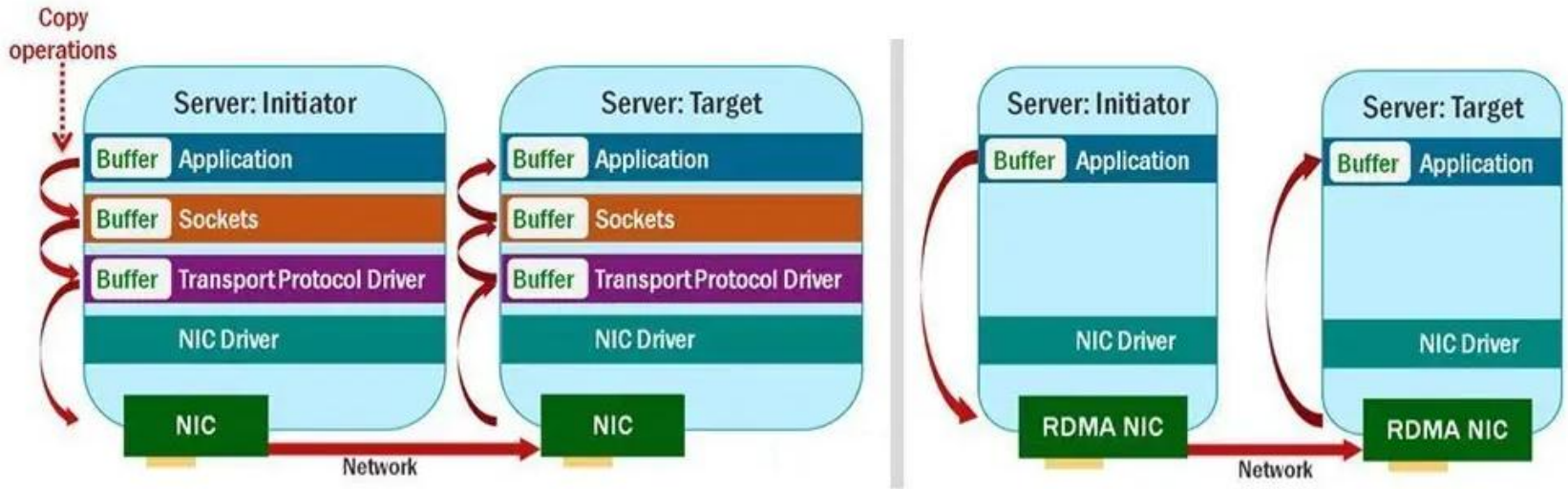
Remote Direct Memory Access (RDMA)

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/



The Kernel Report 2023

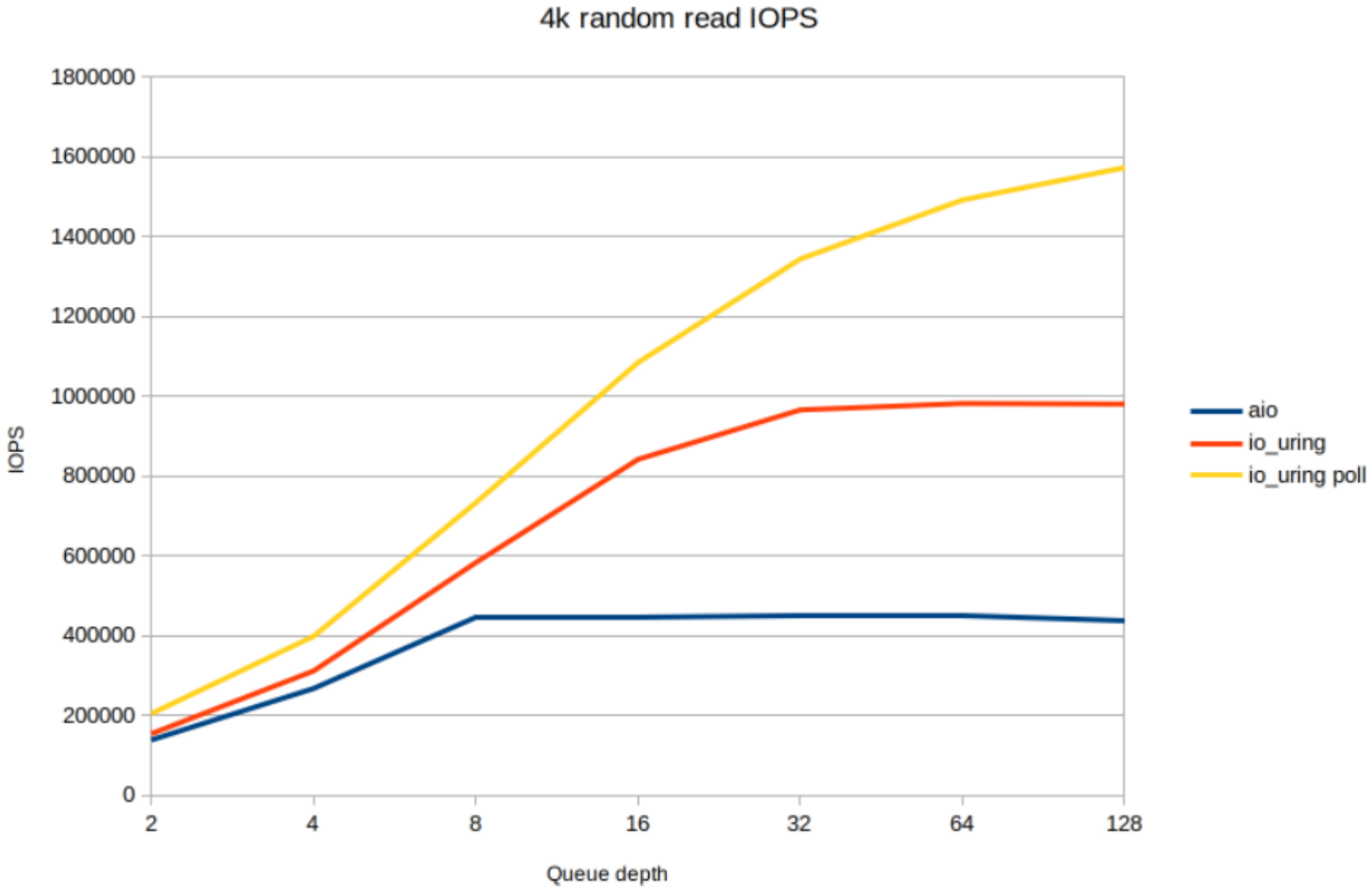
- **BPF – how far do we go?**
  - What BPF ***can*** do?
  - What BPF ***might*** do?

    The extensible scheduler class (write complete CPU schedulers in BPF)

    - 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**).

- **Rust**
  - Has a lot to offer (a stronger type system, no undefined behavior, attractive to newer developers).
  - Why ***not*** Rust in the kernel (a new language adds complexity, the language is still evolving – quickly, maintainers will need to learn Rust, lots of glue code, some things are hard to do in Rust, conservatism).
  - Initial Rust infrastructure has been merge into Linux 6.1 (October 2022).
- **Io_uring integration** (why: better control over sequences of operations, create a complete programming environment).

Asynchronous I/O – io_uring



[Faster IO through io_uring](), Kernel Recepies, 2019, Jen Axboe