TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

ZIH
Information Services and
High Performance Computing

Augmenting HPC I/O Performance Analysis with Detailed Block Layer Insights

Christian von Elm, Sebastian Oeste, Mario Bielert, Thomas Ilsche

International Parallel Tools Workshop 2024
Dresden, 19.09.2024

$$\text{lo2s} = \underbrace{\text{Linux}}_{\text{Target OS}} \quad \underbrace{\text{OTF2}}_{\text{Output Format}} \quad \underbrace{\text{Sampling}}_{\text{Primary Measurement Method}}$$

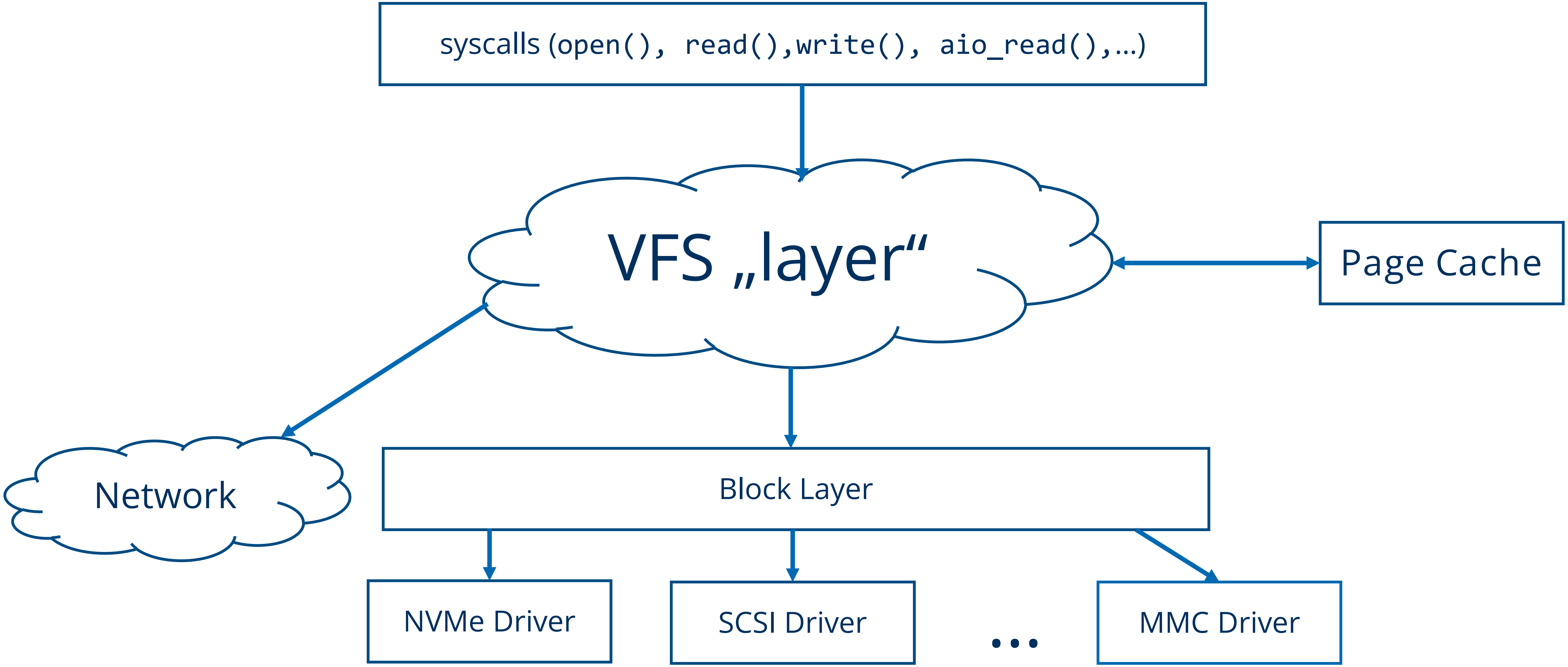
Focus:

— Node-level

— Non-intrusive

— Minimalize measurement perturbation

The Linux Storage Stack – the View From 10‘000 Meters



Getting Information from the Storage

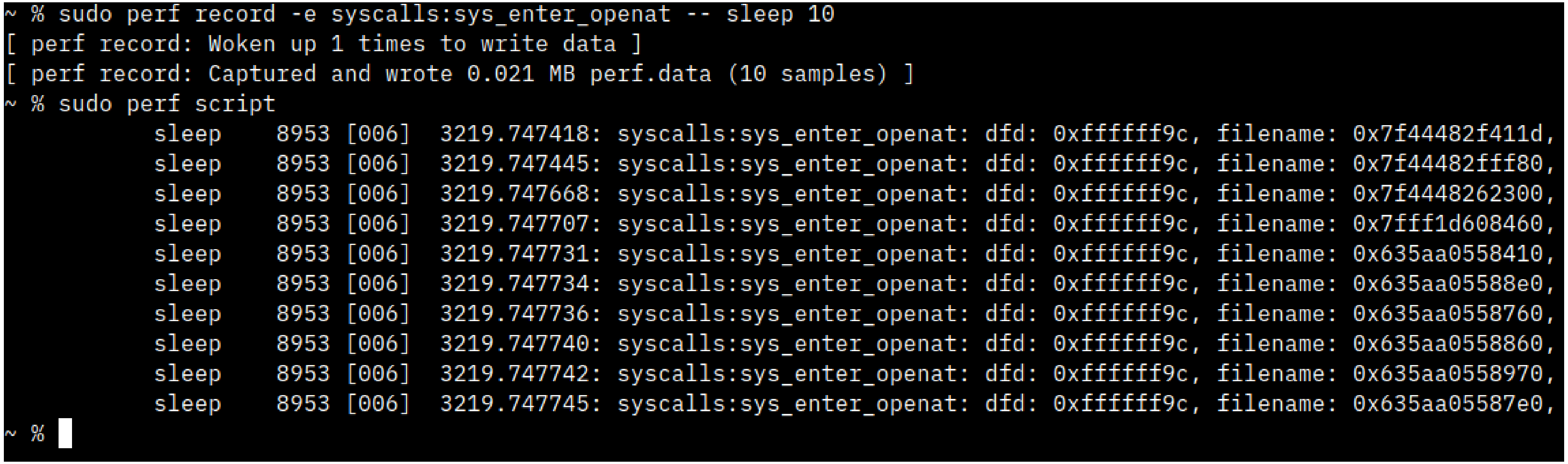
Kernel Tracepoints:

- Well understood, wildly used interface
- Already implemented in lo2s
- Defined at kernel build time, not changeable after

POSIX I/O → `syscalls:sys_{enter, exit}_{openat, read, write}`

Block I/O → `block:block_bio_{queue, issue, complete}`

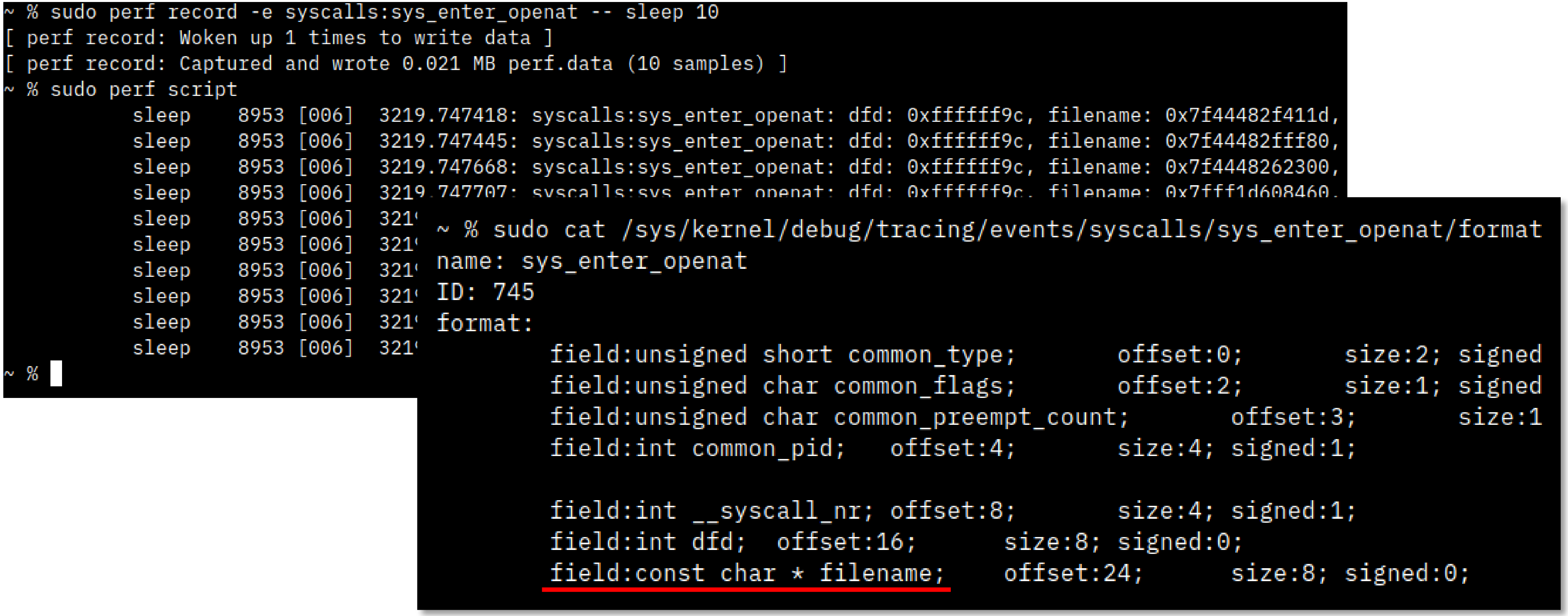
POSIX I/O – Getting Filenames from **openat()** Events



```
~ % sudo perf record -e syscalls:sys_enter_openat -- sleep 10
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.021 MB perf.data (10 samples) ]
~ % sudo perf script
           sleep    8953 [006]  3219.747418: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x7f44482f411d,
           sleep    8953 [006]  3219.747445: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x7f44482fff80,
           sleep    8953 [006]  3219.747668: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x7f4448262300,
           sleep    8953 [006]  3219.747707: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x7fff1d608460,
           sleep    8953 [006]  3219.747731: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x635aa0558410,
           sleep    8953 [006]  3219.747734: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x635aa05588e0,
           sleep    8953 [006]  3219.747736: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x635aa0558760,
           sleep    8953 [006]  3219.747740: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x635aa0558860,
           sleep    8953 [006]  3219.747742: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x635aa0558970,
           sleep    8953 [006]  3219.747745: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x635aa05587e0,
~ %
```

← ?

POSIX I/O – Getting Filenames from **`openat()`** Events



```
~ % sudo perf record -e syscalls:sys_enter_openat -- sleep 10
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.021 MB perf.data (10 samples) ]
~ % sudo perf script
           sleep    8953 [006]  3219.747418: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x7f44482f411d,
           sleep    8953 [006]  3219.747445: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x7f44482fff80,
           sleep    8953 [006]  3219.747668: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x7f4448262300,
           sleep    8953 [006]  3219.747707: syscalls:sys_enter_openat: dfd: 0xffffff9c, filename: 0x7fff1d608460,
           sleep    8953 [006]  321
           sleep    8953 [006]  321
           sleep    8953 [006]  321
           sleep    8953 [006]  321
           sleep    8953 [006]  321
           sleep    8953 [006]  321
~ %
```

```
~ % sudo cat /sys/kernel/debug/tracing/events/syscalls/sys_enter_openat/format
name: sys_enter_openat
ID: 745
format:
        field:unsigned short common_type;       offset:0;       size:2; signed
        field:unsigned char common_flags;       offset:2;       size:1; signed
        field:unsigned char common_preempt_count;       offset:3;       size:1
        field:int common_pid;   offset:4;       size:4; signed:1;

        field:int __syscall_nr; offset:8;       size:4; signed:1;
        field:int dfd;  offset:16;      size:8; signed:0;
        field:const char * filename;    offset:24;      size:8; signed:0;
```

`filename` is a pointer to memory in sleep, we can not access it from lo2s!

Accessing Application Memory Non-intrusively

`/proc/[PID]/mem`

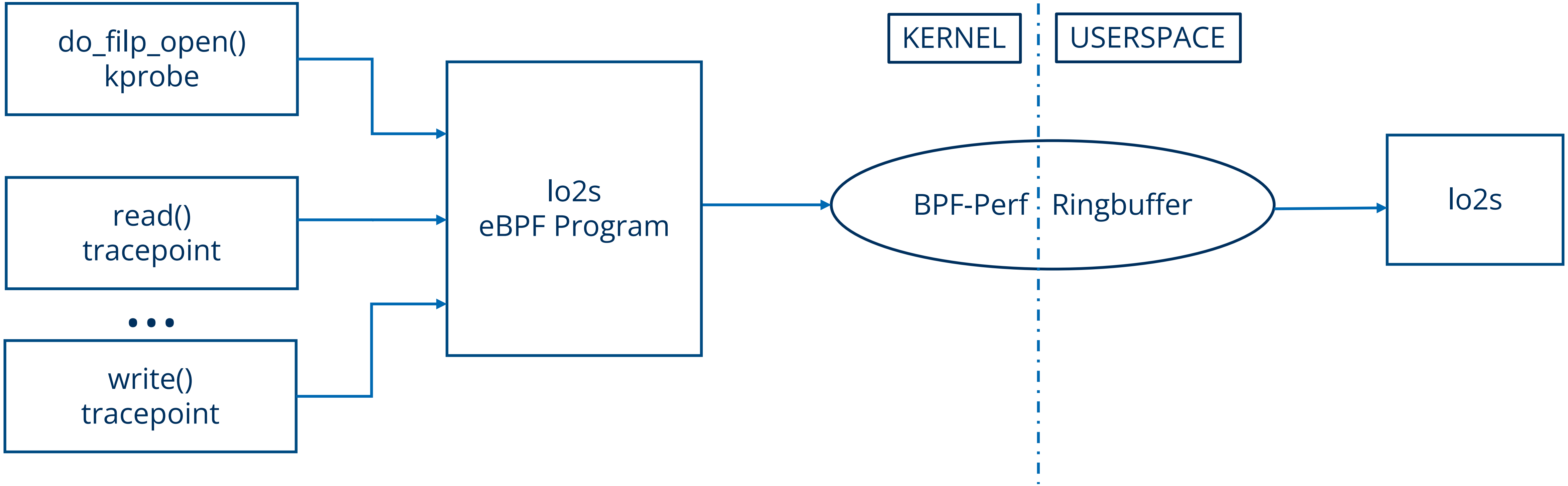
- „Everything is a file“: makes [PID]s memory available as a regular file

- But: Reading the filename is out-of-band w.r.t to the tracepoint event stream
- Either significant information loss or synchronization penalty

There got to be some way to modify the tracepoint event stream in-band

Extended Berkeley Packet Filter (eBPF)

- In-Kernel virtual machine with JIT, with static code verification
- Allows us to edit event stream in-band inside the kernel

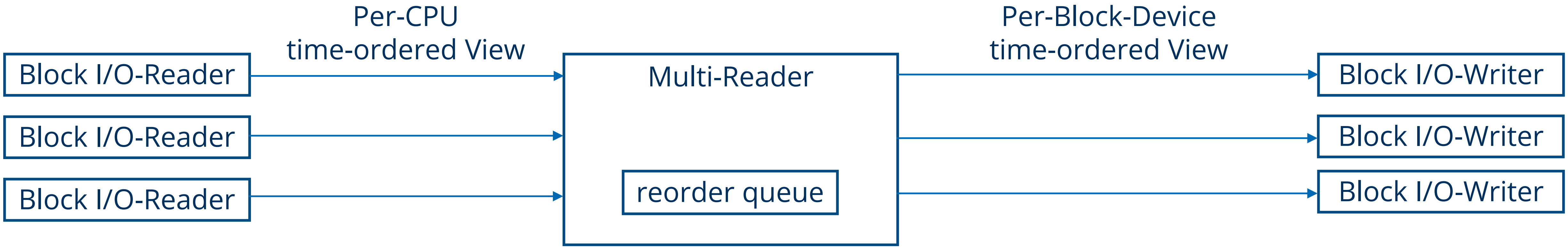


Maintaining Event-Stream Consistency in Block I/O

- OTF2 requires time-ordered event streams
- Perf (BPF and Classic) Ringbuffers are required to be per-CPU time ordered
- But: We want a per-Block Device ordered view

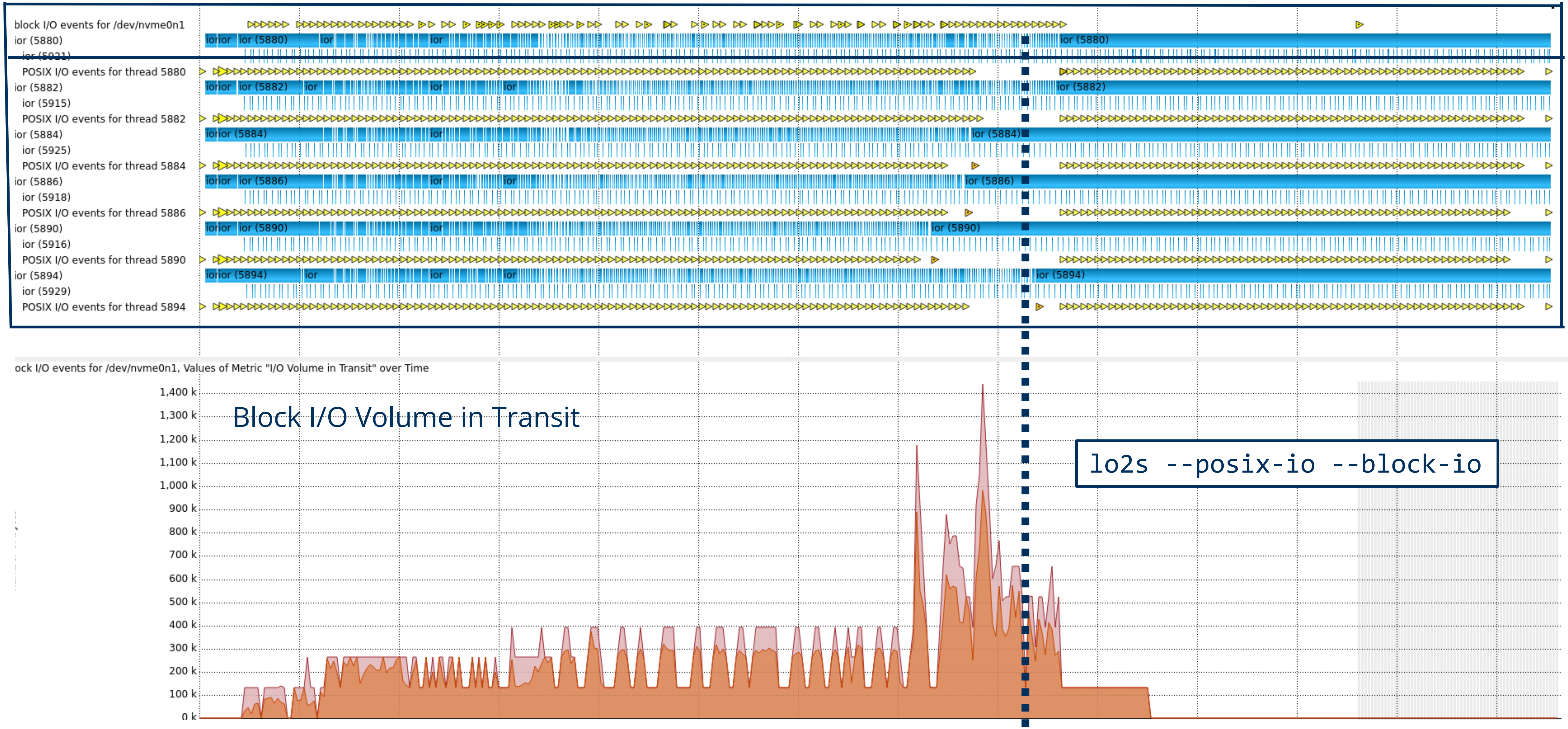
→ Reorder in Multi-Reader with some caching

→ Trade-off memory consumption ↔ event-loss



Block

POSIX


block I/O events for /dev/nvme0n1
ior (5880)
ior (5921)
POSIX I/O events for thread 5880
ior (5882)
ior (5915)
POSIX I/O events for thread 5882
ior (5884)
ior (5925)
POSIX I/O events for thread 5884
ior (5886)
ior (5918)
POSIX I/O events for thread 5886
ior (5890)
ior (5916)
POSIX I/O events for thread 5890
ior (5894)
ior (5929)
POSIX I/O events for thread 5894
ock I/O events for /dev/nvme0n1, Values of Metric "I/O Volume in Transit" over Time
1,400 k
1,300 k
1,200 k
1,100 k
1,000 k
900 k
800 k
700 k
600 k
500 k
400 k
300 k
200 k
100 k
0 k

Block I/O Volume in Transit

```
lo2s --posix-io --block-io
```

Write Phase | Read Phase

https://github.com/tud-zih-energy/lo2s

Thank you for listening?

Questions?

eBPF Usage Challenges

- Relatively new, fast moving kernel feature
  - Thankfully, lots of backported patches

- Attaching to arbitrary kernel memory is version dependent
  - When the layout of `struct`'s changes, accesses are not valid anymore
  - Solution in the past: ship whole LLVM with BPF-based tools and do JIT-compilation

→ Modern Solution: BPF CO-RE + BTF: Kernel ships with lightweight debug information to calculate symbol offsets on the fly

Available in RHEL and compatible distributions since Version 9.0