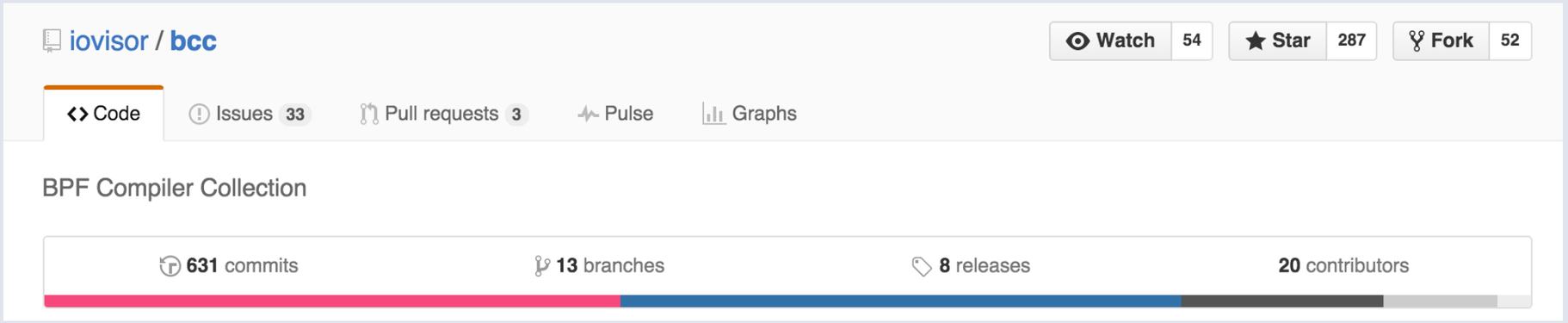
Thank you! BPF contributors

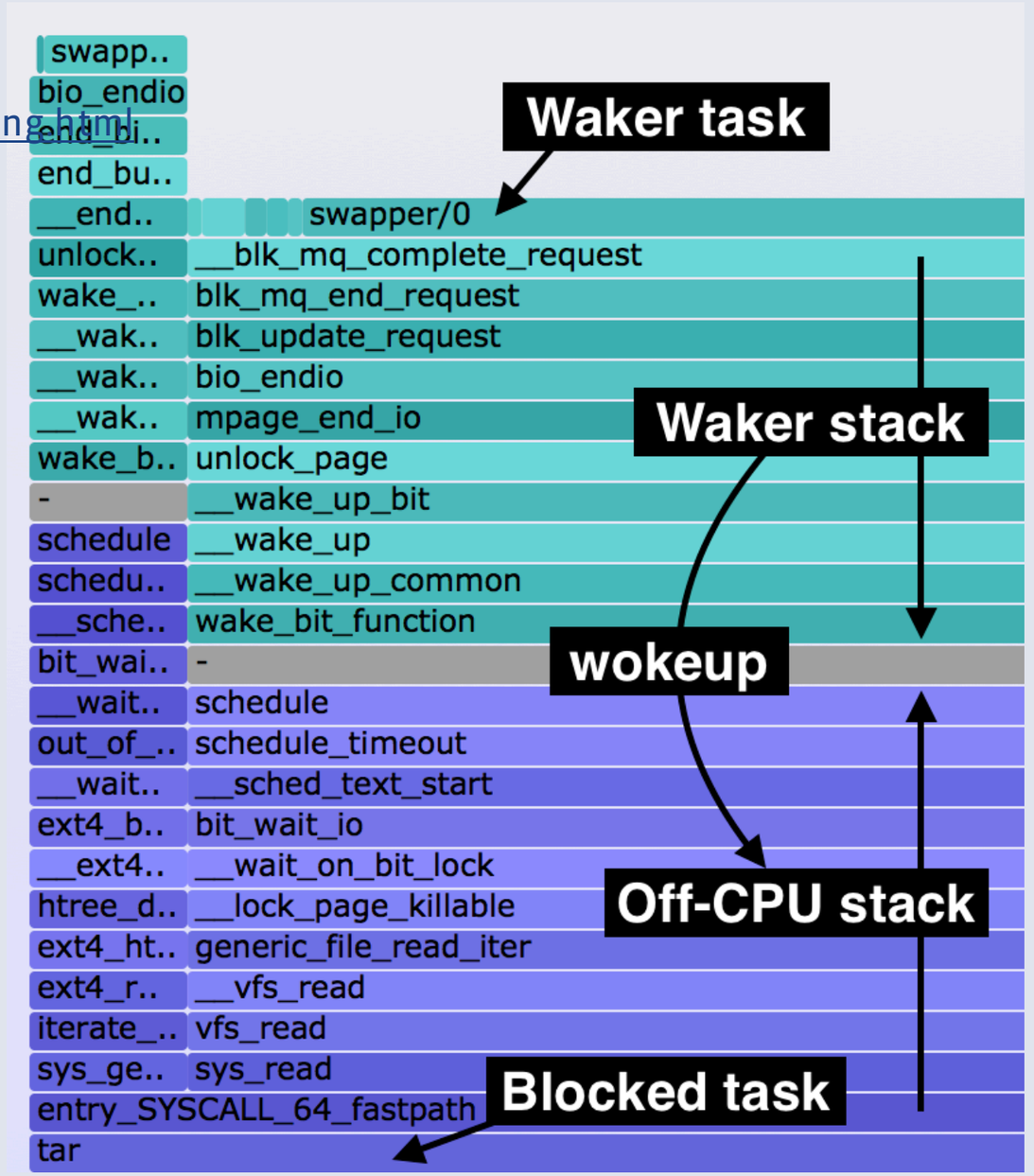


Why?

Off-wake profiling

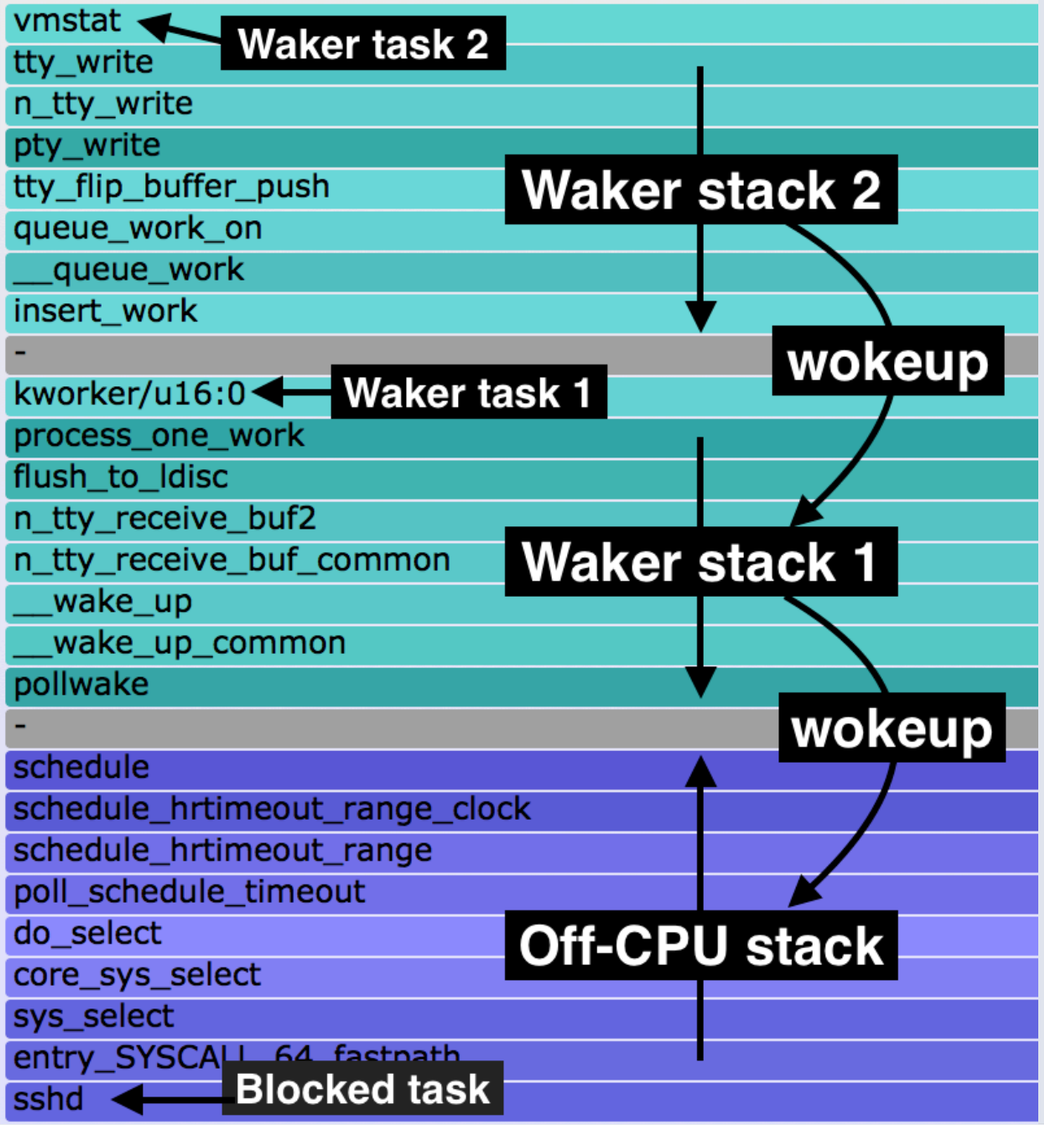
http://www.brendangregg.com/blog/2016-02-01/linux-wakeup-offwake-profiling.html

- To understand why tasks are blocked and why they were woken up
- new stack map type for fast collection of stack traces
- User and kernel stacks
- bpf_get_stackid()
- Exploring LBR (Last Branch Record)



Who is waking the waker? Chain graph

http://www.brendangregg.com/blog/2016-02-05/ebpf-chaingraph-prototype.html



~~web10g~~ analyze tcp with bpf

- Must have the lowest overhead
- per-cpu maps
- kprobe, bpf_probe_read optimizations
- Attach bpf to tracepoints
- Tracepoints in few strategic places
- Metadata at ‘struct sock’

mmap() opens the door

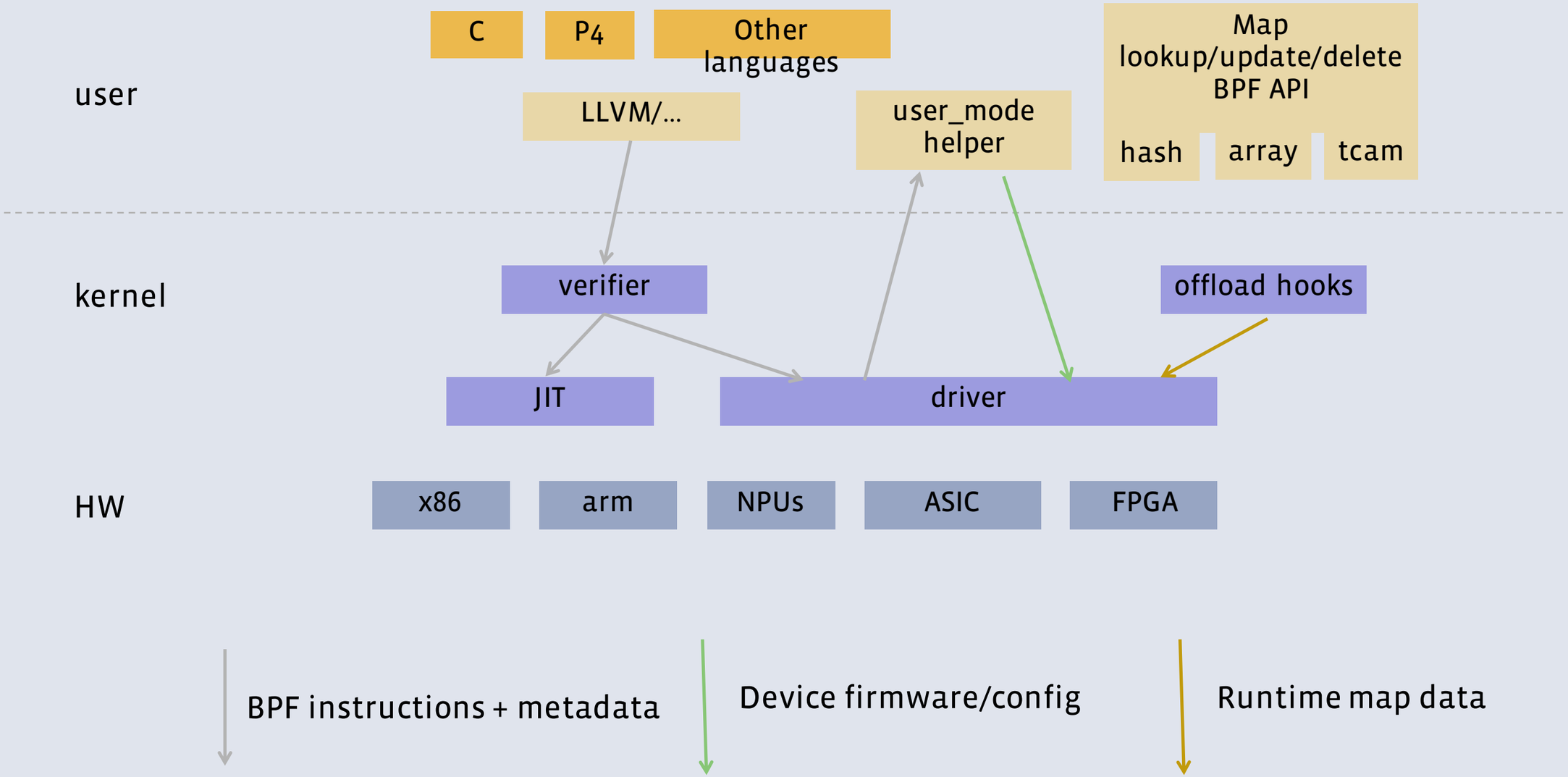
- mmap of bpf maps + TLS
- One memory access for a user space to interface with the kernel
- ~~getcpu_cache~~ (read cpuid in user space, read time, …)
- counter-till-next-sample (fancy floating point math in userspace to setup kernel sampling threshold). Similar sampling technique used in tcmalloc/jemalloc

Idea corner

- GRO/GSO of UDP-based protocols via bpf
- Anti-bypass mission
- Bounded-loop and vector instructions
- Inline helpers (bpf_map_lookup() becomes load instead of function call)

HW offload

- BPF engine in FPGA (open source!)
- IR (Intermediate Representation) (BPF instructions vs LLVM bitcode)
- Host vs switch
- SW vs HW centric
- Offloading into FPGA, NPU, ASIC
- Programmable vs configurable


user
C
P4
Other languages
LLVM/...
user_mode helper
Map lookup/update/delete BPF API
hash
array
tcam
kernel
verifier
offload hooks
JIT
driver
HW
x86
arm
NPUs
ASIC
FPGA
BPF instructions + metadata
Device firmware/config
Runtime map data