perf data file & toggling events

jiri olsa

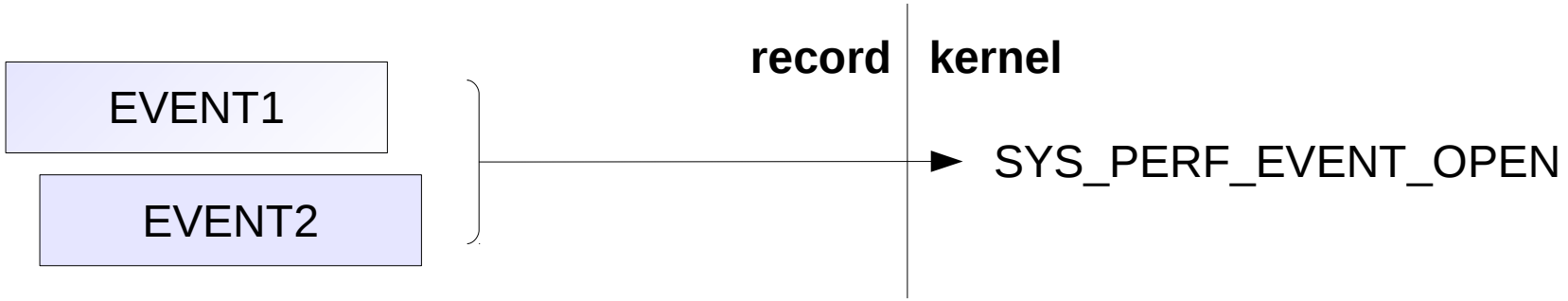
TWO TOPICS

- data file (perf.data)
- toggling events

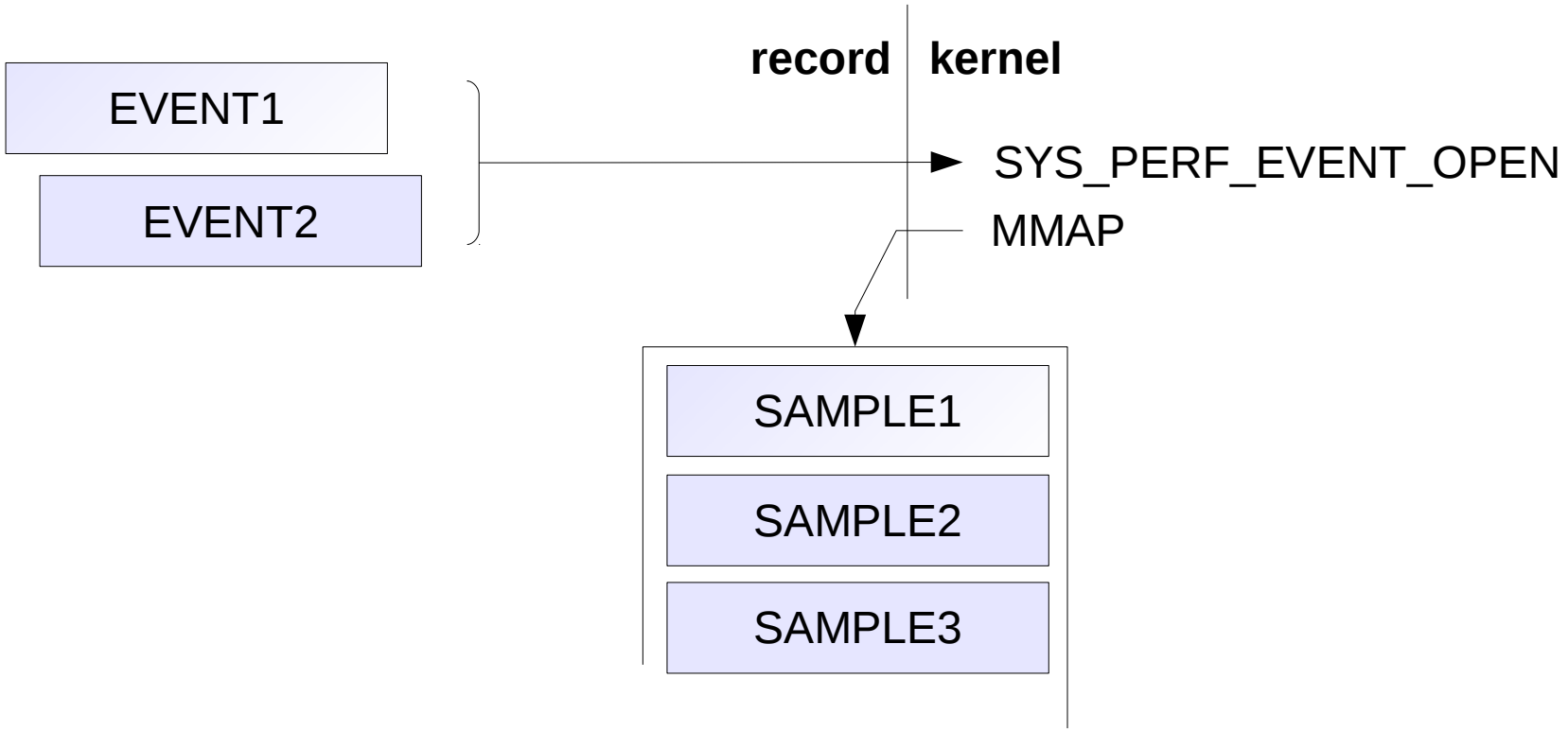
PERF DATA FILE - PERF.DATA

- carries perf sampling output
  - **events** - descriptions
  - **data** - samples
  - other info
- stored by **record**
- read by **report**, **script**, **evlist**, **buildid-list ...**

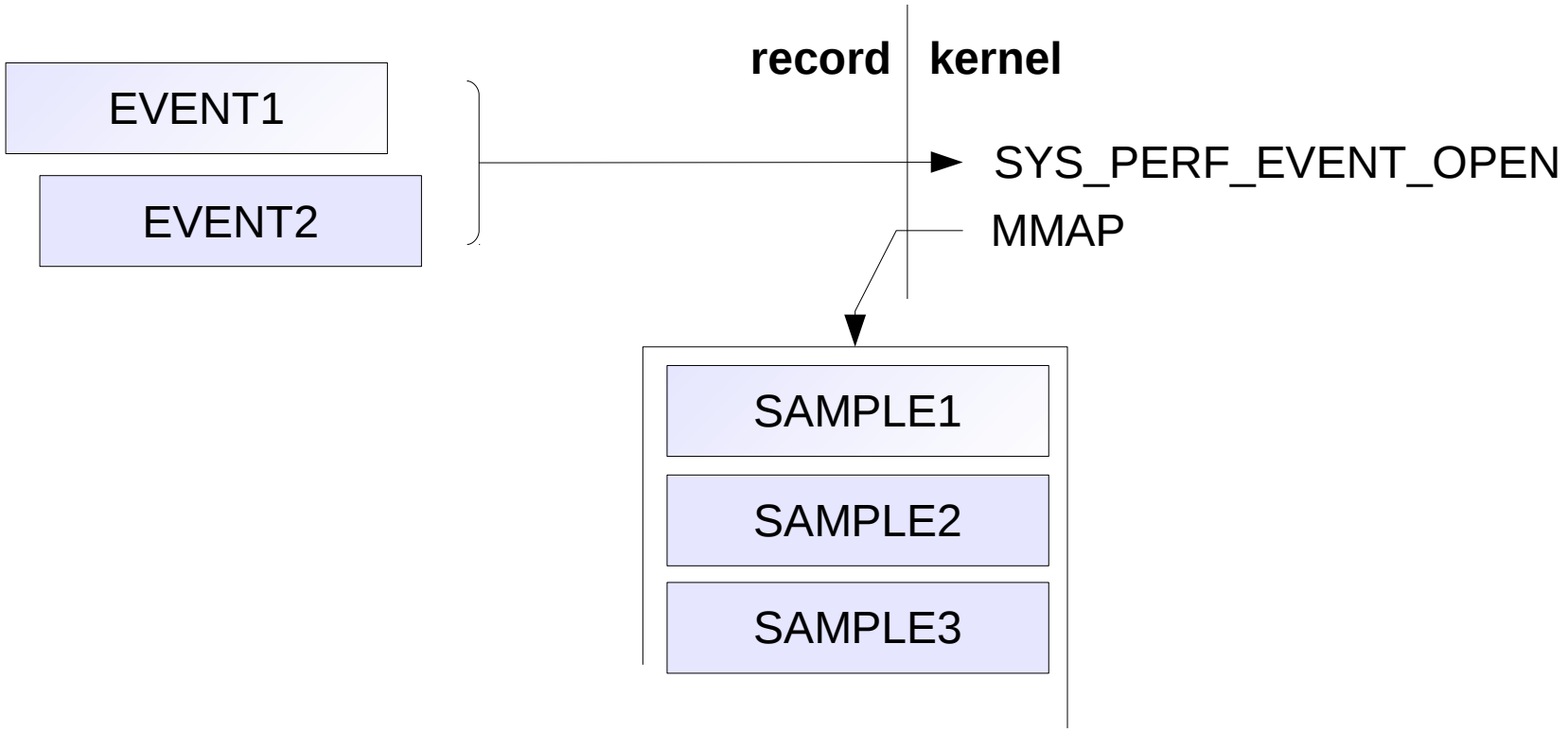
PERF.DATA STORAGE



PERF.DATA STORAGE

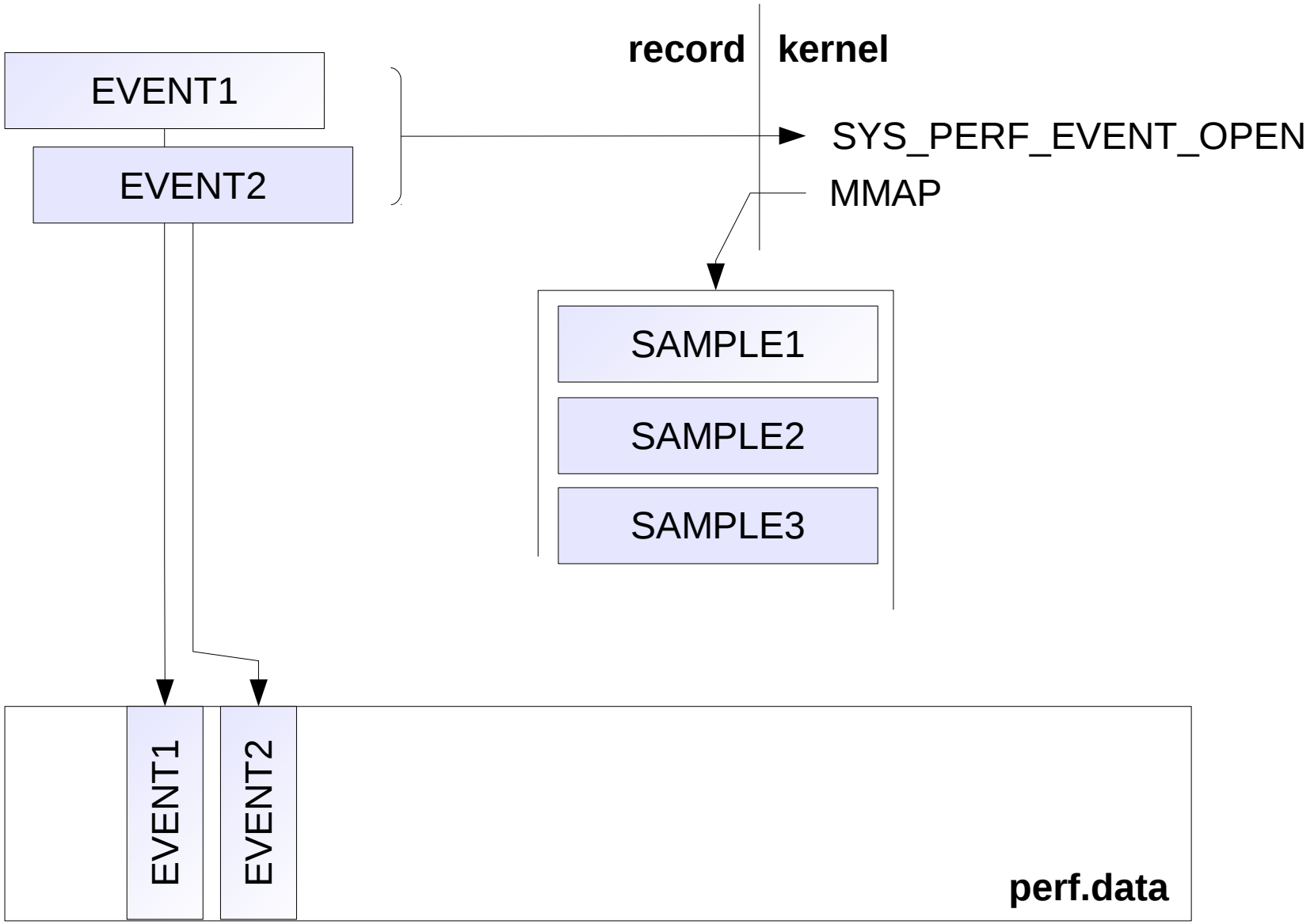


PERF.DATA STORAGE

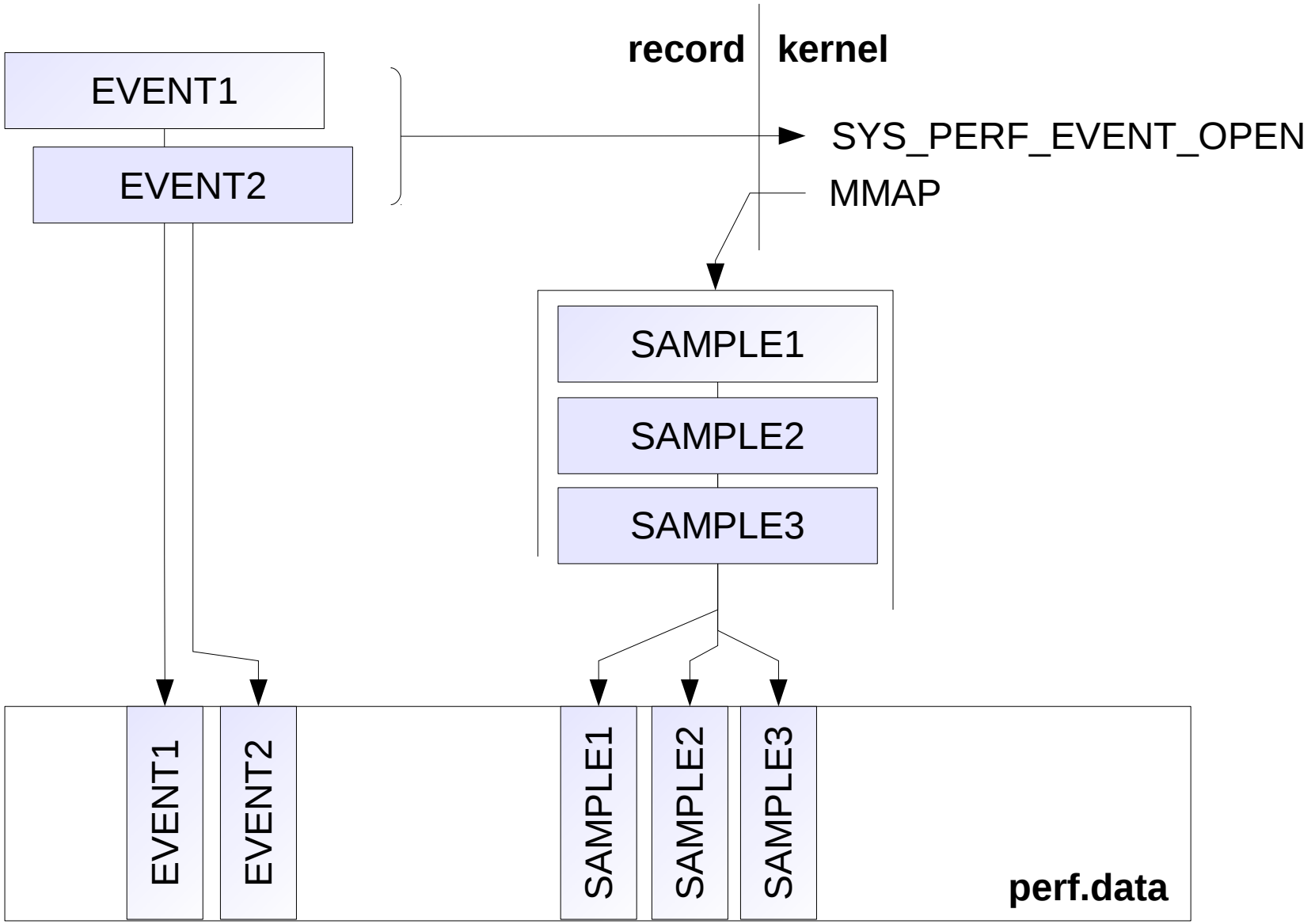


perf.data

PERF.DATA STORAGE

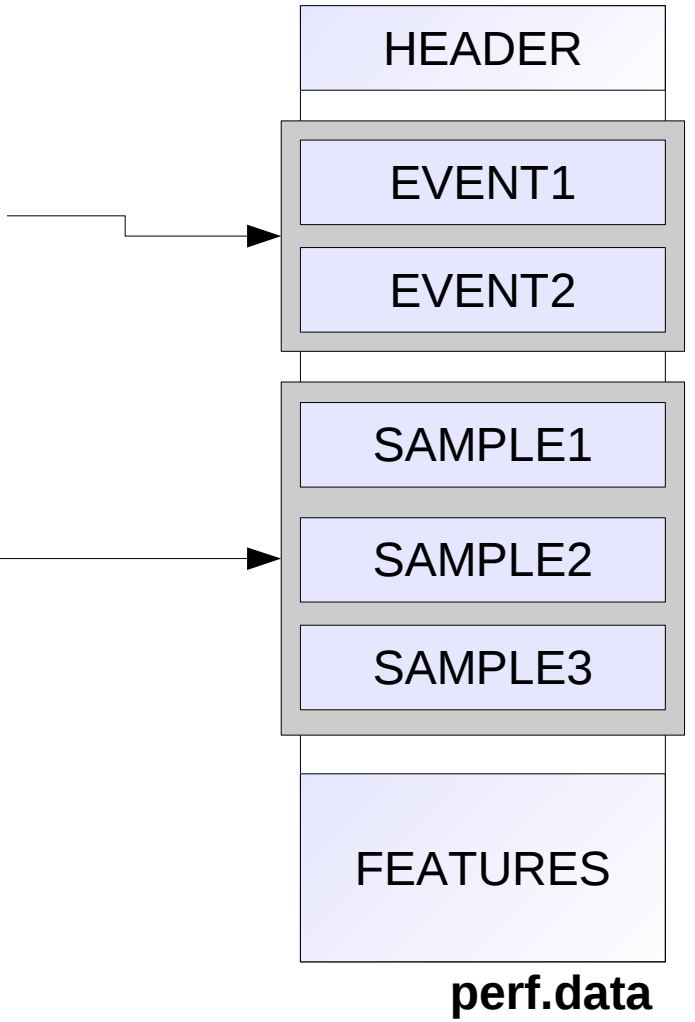


PERF.DATA STORAGE

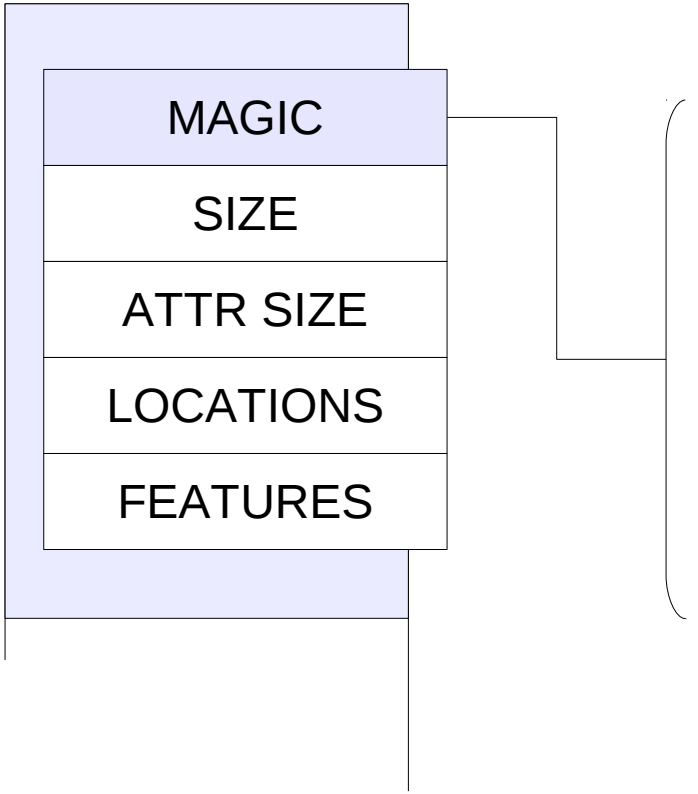


PERF.DATA FORMAT

- HEADER
- EVENT DESCRIPTIONS
- EVENT DATA
- FEATURES



PERF.DATA FORMAT - HEADER

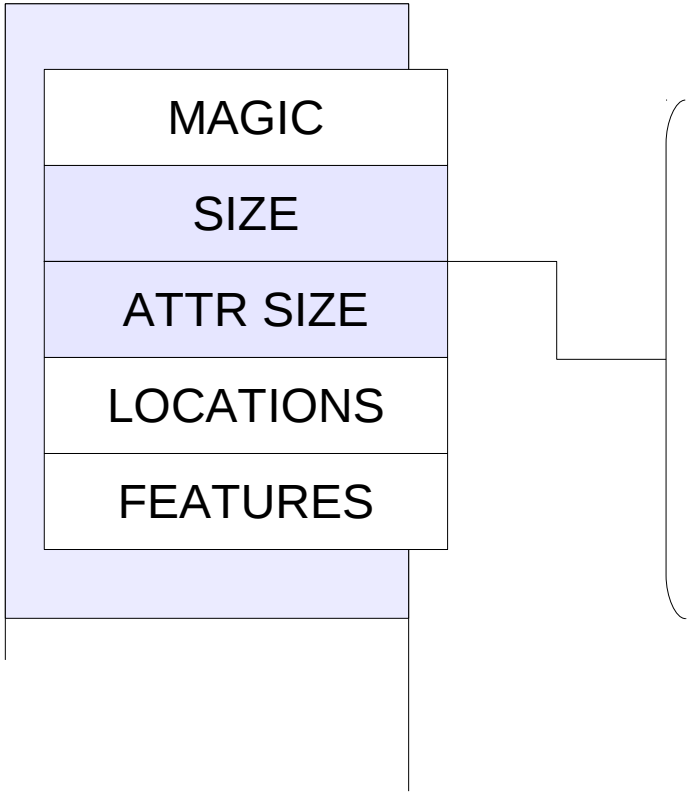


Value:

- PERFFILE
- PERFILE2

Set version and endianity

PERF.DATA FORMAT - HEADER

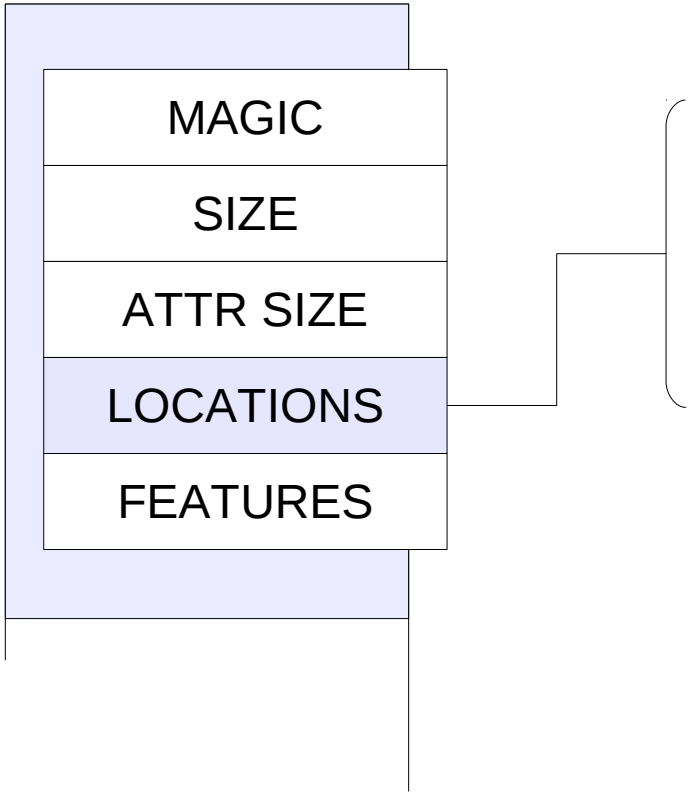


Sizes of:

- file header
- struct perf_event_attr

Set file & kernel interface

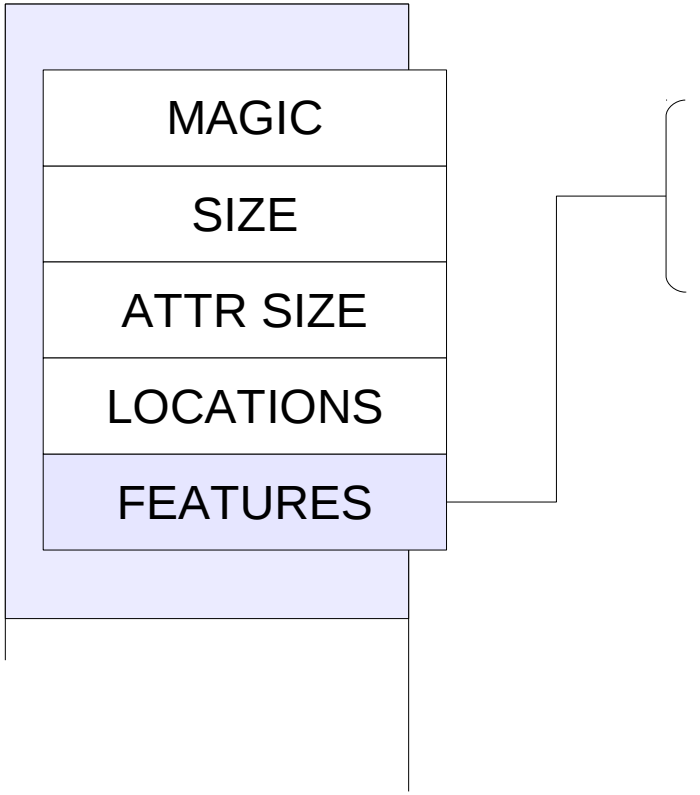
PERF.DATA FORMAT - HEADER



Locations of:

- event descriptions
- data

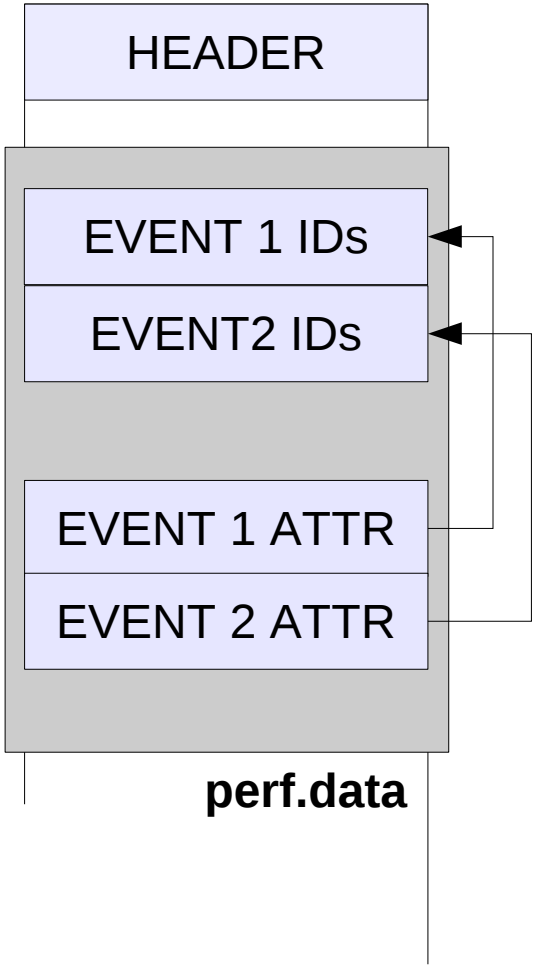
PERF.DATA FORMAT - HEADER



Features bitmap

- 256 bits

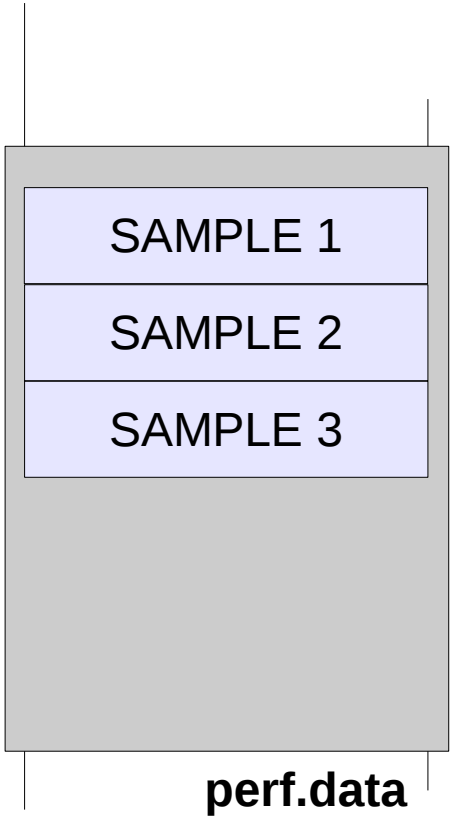
PERF.DATA FORMAT – EVENT DESCRIPTIONS



struct perf_event_attr

- event attribute structure
- linked with IDs array
- need IDs to properly resolve samples

PERF.DATA FORMAT – DATA

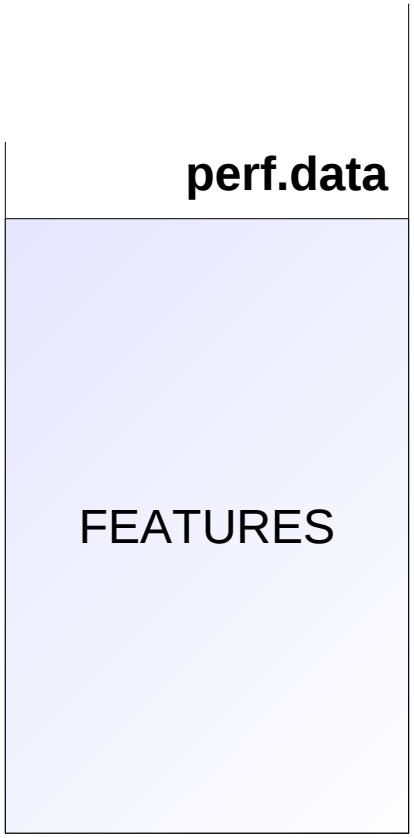


blob of samples

- sample = header + data

```
struct perf_event_header {
        __u32   type;
        __u16   misc;
        __u16   size;
};
```


PERF.DATA FORMAT – FEATURES



various system data

ftrace data, build ids, hostname, OS release,

version, architecture, NR CPUs,

CPU description, CPUID, total memory,

command line, event description,

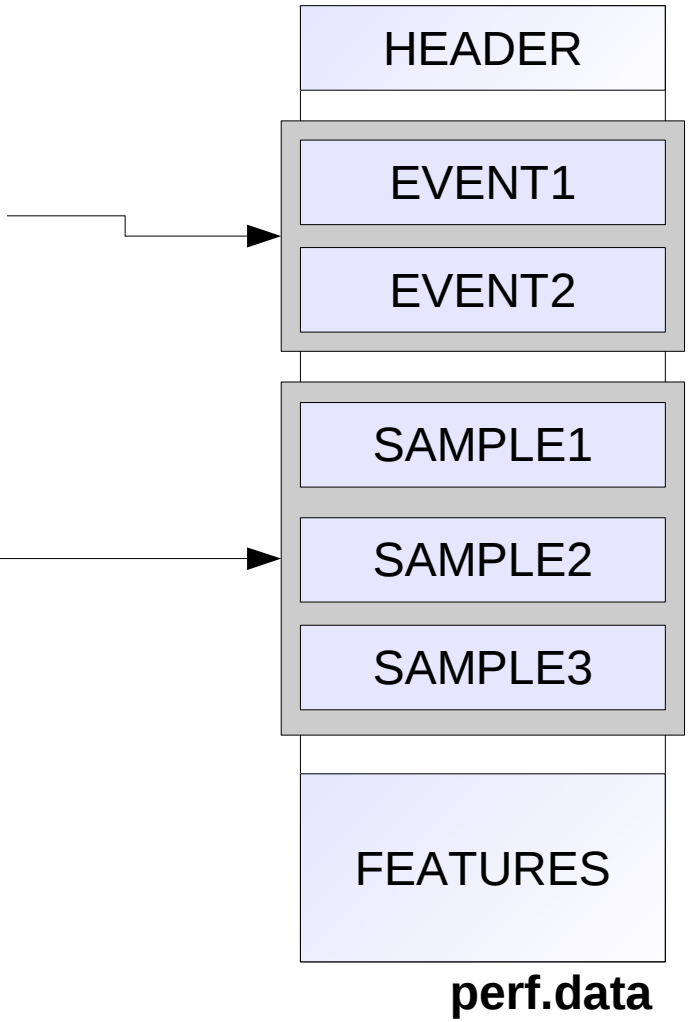
CPU topology, NUMA topology,

branch stack data, PMU mappings,

group description

PERF.DATA FORMAT

- HEADER
- EVENT DESCRIPTIONS
- EVENT DATA
- FEATURES



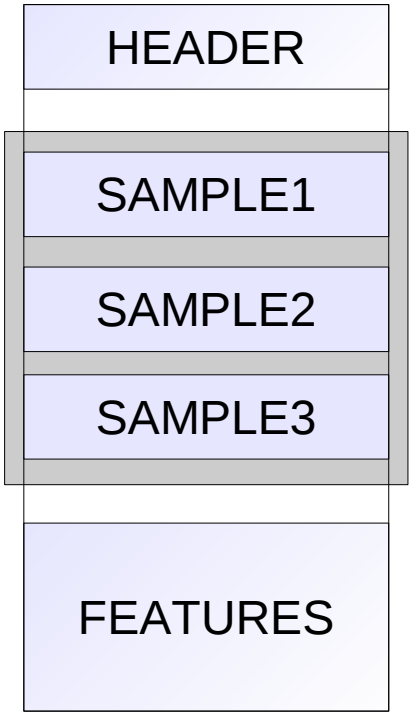
PERF.DATA RECENT CHANGES

- multiple perf.data storage for record session
- simplified format – version 3

HEADER

DATA

FEATURES



PERF.DATA RECENT CHANGES

- '**-M size**' or '**-M time**'
- difficulties to synchronize data with auxiliary events on the new file split

PERF.DATA RECENT CHANGES

```
$ perf record -M 3M yes > /dev/null
^C[ perf record: Woken up 135 times to write data ]
[ perf record: Captured and wrote 12.147 MB perf.data-[0-4](~530731 samples) ]
yes: Interrupt
$ ls -l perf.data-*
-rw------- 1 jolsa jolsa 3185088 Oct 15 00:43 perf.data-00000
-rw------- 1 jolsa jolsa 3169428 Oct 15 00:44 perf.data-00001
-rw------- 1 jolsa jolsa 3177944 Oct 15 00:44 perf.data-00002

$ perf diff perf.data-0000[012]
# Event 'cycles'
#
# Data files:
#  [0] perf.data-00000 (Baseline)
#  [1] perf.data-00001
#  [2] perf.data-00002
#
# Baseline/0  Delta/1  Delta/2      Shared Object                          Symbol
# ..........  .......  .......  ..................  ..............................
#
      37.09%   -0.48%   -0.10%  libc-2.15.so        [.] _IO_file_xsputn@@GLIBC_2.2.5
      29.71%   +0.77%   +0.58%  yes                 [.] main
      15.89%   -0.16%   +0.23%  libc-2.15.so        [.] __strlen_sse2
      14.17%   -0.39%   -0.39%  libc-2.15.so        [.] fputs_unlocked
```


PERF.DATA RECENT FUTURE

- **perf record** per thread perf.data storage
- perf.data files merge
- **perf diff** process FEATURES data

PERF.DATA CODE

- doc:

  https://perf.wiki.kernel.org/index.php/Jolsa_Features_Multiple_File_Storage

- code

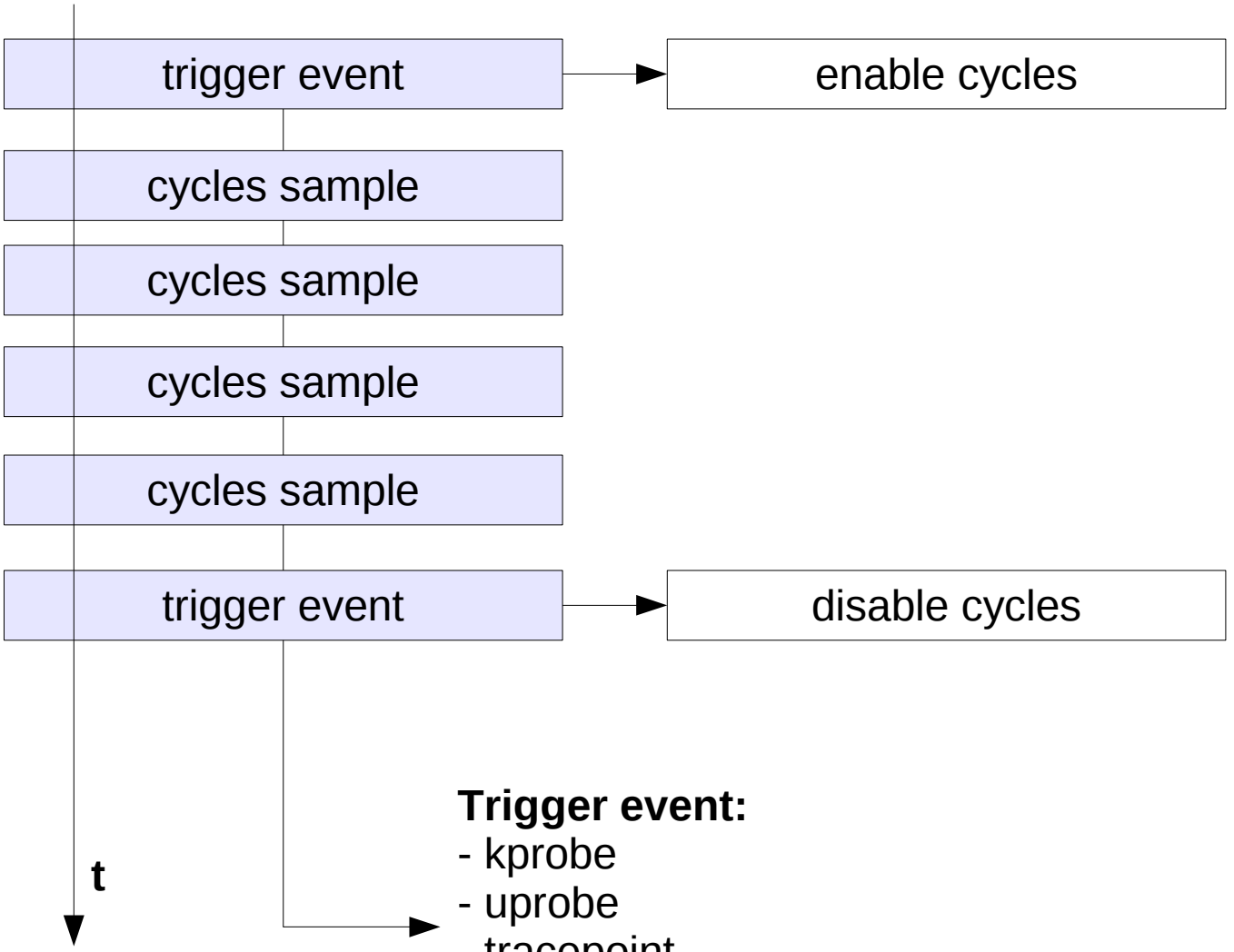
  git://git.kernel.org/pub/scm/linux/kernel/git/jolsa/perf.git

  perf/core_file

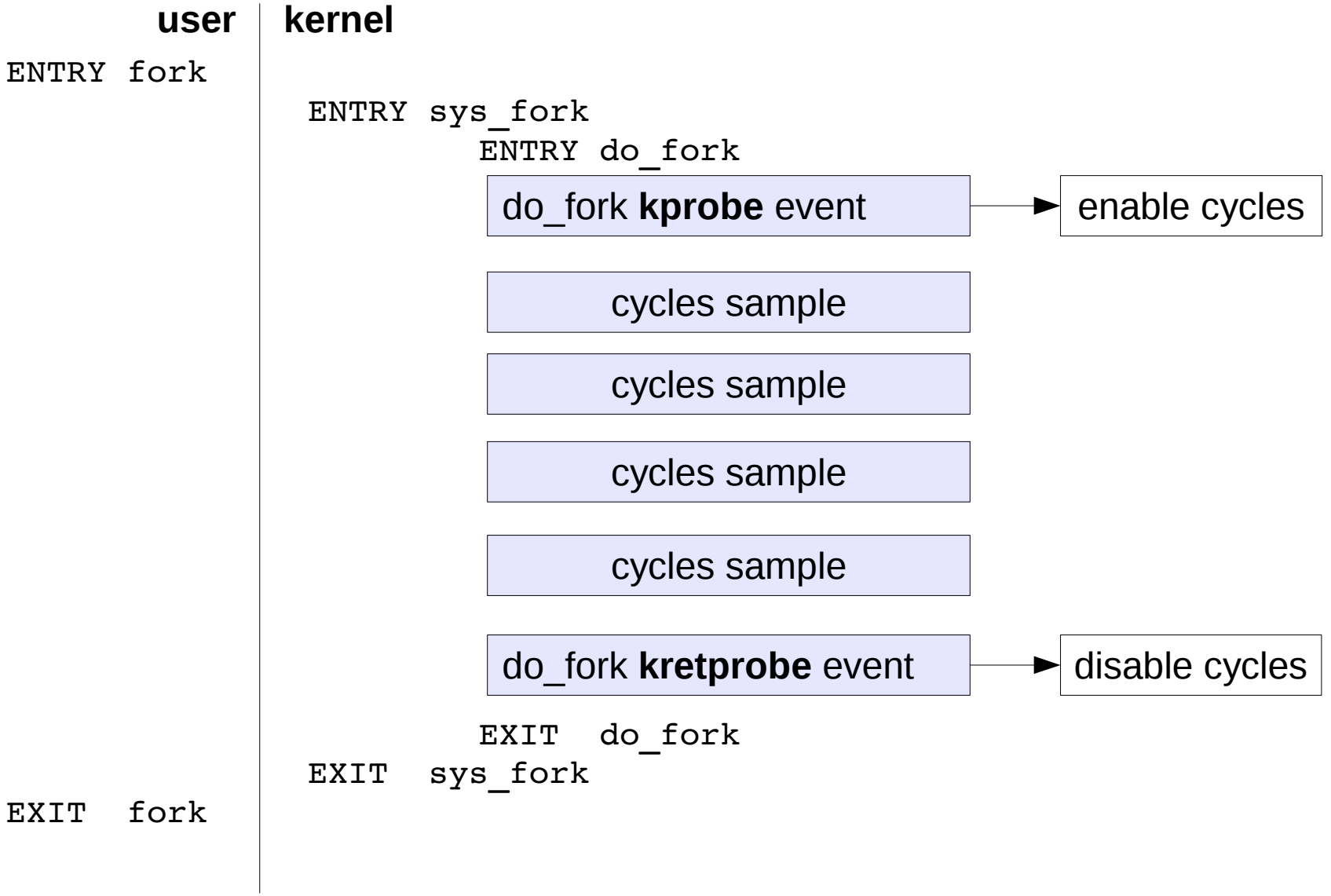
TOGGLING EVENTS

- configure event to toggle (**start**/**stop**) another event
- aimed for tracepoints to toggle HW counters
- **narrow down** the measured area
- original code from Frederic Weisbecker

TOGGLING EVENTS



TOGGLING EVENTS



TOGGLING INTERFACE - KERNEL

- allows event (toggler) to **start** or **stop** another (toggled) event
- new interface for:

  sys_perf_event_open(attr, pid, cpu, group_fd, flags)

  ```
  new flags: PERF_FLAG_TOGGLE_ON/OFF
  group_fd:  event (or group) fd to be toggled
  ```

  ioctl(fd, cmd, void *)

  ```
  u64 args[2] = { toggled_fd, flag };
  err = ioctl(fd, PERF_EVENT_IOC_SET_TOGGLE, args);
  ```

- inheritance support
- togglers and toggled event must be on same task/cpu

TOGGLING INTERFACE – PERF TOOL

- implemented for **record** and **stat** commands
- defined toggle event using **on** and **off** terms

```
-e 'irq:irq_handler_entry/on=<NAME>'
-e 'irq:irq_handler_entry/off=<NAME>'
```

- allow to specify user name for each event:

```
-e 'cycles:k/name=kernel_cycles/'
```

- **final:**

```
-e 'cycles:k/name=kernel_cycles/,
    irq:irq_handler_entry/on=kernel_cycles/,
    irq:irq_handler_entry/off=kernel_cycles/'
```


TOGGLING INTERFACE – PERF TOOL

- measure function in kernel using **k(ret)probes**
- define start and stop event:

```
# perf probe -a fork_entry=do_fork
# perf probe -a fork_exit=do_fork%return
```

- **record session:**

```
# perf record -a -e \
                'cycles:k, \
                 probe:fork_entry/on=cycles/,\
                 probe:fork_exit/off=cycles/'
```


TOGGLING INTERFACE – PERF TOOL

- measure function in application using **u(ret)probes**
- define start and stop event:

```
# perf probe -x ./test entry=func
# perf probe -x ./test exit=func%return
```

- **stat session:**

```
# perf stat -e \
          '{cycles,instructions,cache-misses}:u,\
           probe_ex:entry/on=cycles/,\
           probe_ex:exit/off=cycles/'\
          ./test
```


TOGGLING INTERFACE – OVERHEAD

- overhead of starting & stopping routines
- bigger in kernel
  - kprobe/tracepoint trigger code
  - toggle overflow code
- uprobes userspace overhead

TOGGLING EVENTS – DOC & CODE

- doc:

  https://perf.wiki.kernel.org/index.php/Jolsa_Features_Togle_Event

- code

  git://git.kernel.org/pub/scm/linux/kernel/git/jolsa/perf.git

  perf/core_toggle

THANKS, QUESTIONS

Jiri Olsa <jolsa@redhat.com>