perf & CTF

jiri olsa

PERF & CTF

- data
- CTF
- perf data
- comparison
- conversion
- perf data convert
- future

DATA

- perf record/report (perf.data)
- lttng/eclipse (CTF)

WHAT DO WE STORE?

- events
- system data

EVENT

- event's data

  PID

  CPU

  TIME

  PC

  VALUE

  BACKTRACE

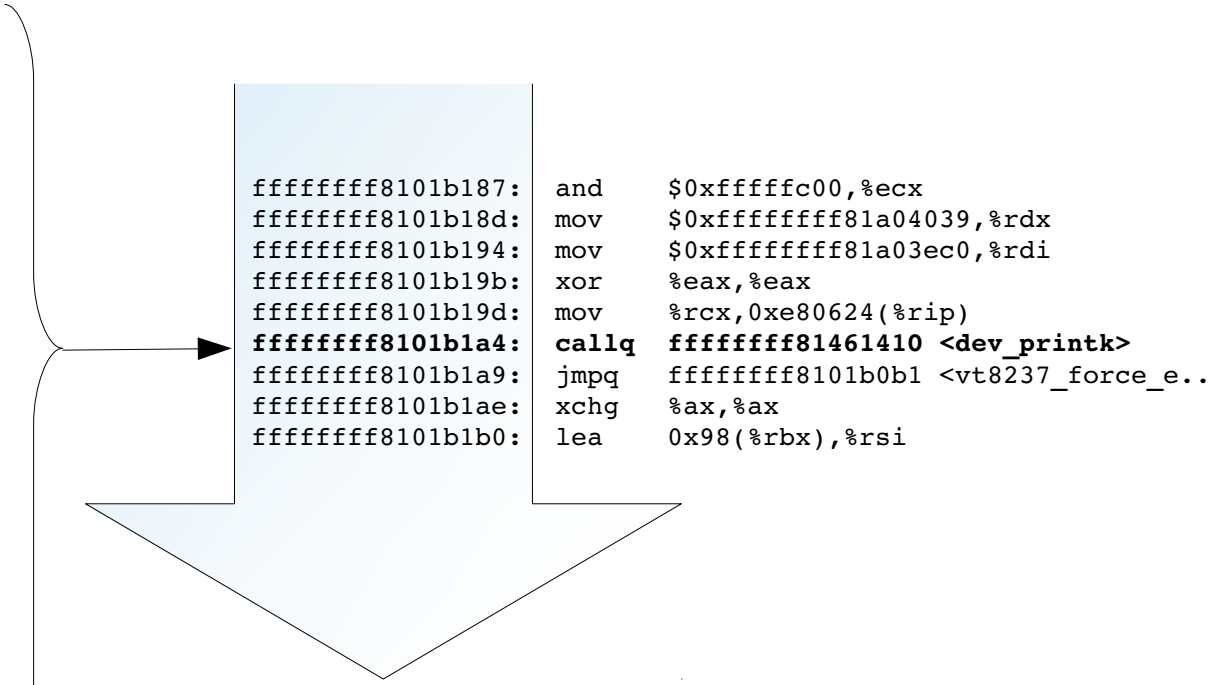
  TRACEPOINT

  BRANCHES

  STACK

  REGS

  ...



SYSTEM DATA

- system related data
  - trace data
  - build ids
  - Hostname
  - OS release
  - Version
  - Architecture
  - NR CPUs
  - CPU description
  - CPUID
  - CPU topology
  - NUMA topology
  - branch stack data
  - PMU mappings
  - group description
  - total memory
  - command line
  - event description

CTF

- Common Trace Format
- generic format with public specification

  Trace Stream Description Language (TSDL)
- babeltrace library
- network friendly

PERF DATA

- bound to perf linux kernel interface
- no public specification yet (sorry)
- no library (sorry)
- no network friendly

CTF

```
$ ls lttng-sessions/krava-20140820-101028/kernel
channel0_0                    BINARY DATA
channel0_1
channel0_2
channel0_3
metadata
```


CTF

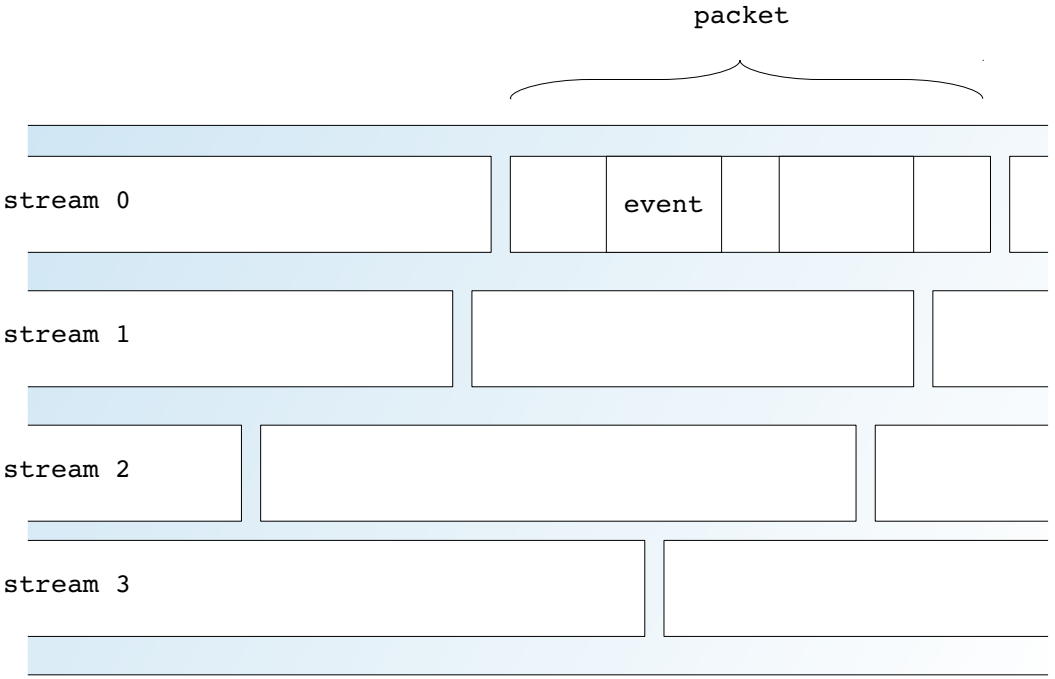
```
$ ls lttng-sessions/krava-20140820-101028/kernel
channel0_0
channel0_1
channel0_2
channel0_3
metadata
```

```
trace {
        major = 1;
        minor = 8;
        uuid = "891ddd79-a1ed-1e4f-9ced-44d9f0d1b285";
        byte_order = le;
        packet.header := struct {
                uint32_t magic;
                uint8_t  uuid[16];
                uint32_t stream_id;
        };
};

env {
        hostname = "krava";
        sysname = "Linux";
...

struct packet_context {
        uint64_clock_monotonic_t timestamp_begin;
        uint64_clock_monotonic_t timestamp_end;
        uint64_t content_size;
        uint64_t packet_size;
        unsigned long events_discarded;
        uint32_t cpu_id;
};

struct event_header_compact {
        enum : uint5_t { compact = 0 ... 30, extended = 31 } id;
        variant <id> {
                struct {
                        uint27_clock_monotonic_t timestamp;
                } compact;
```

TSDL

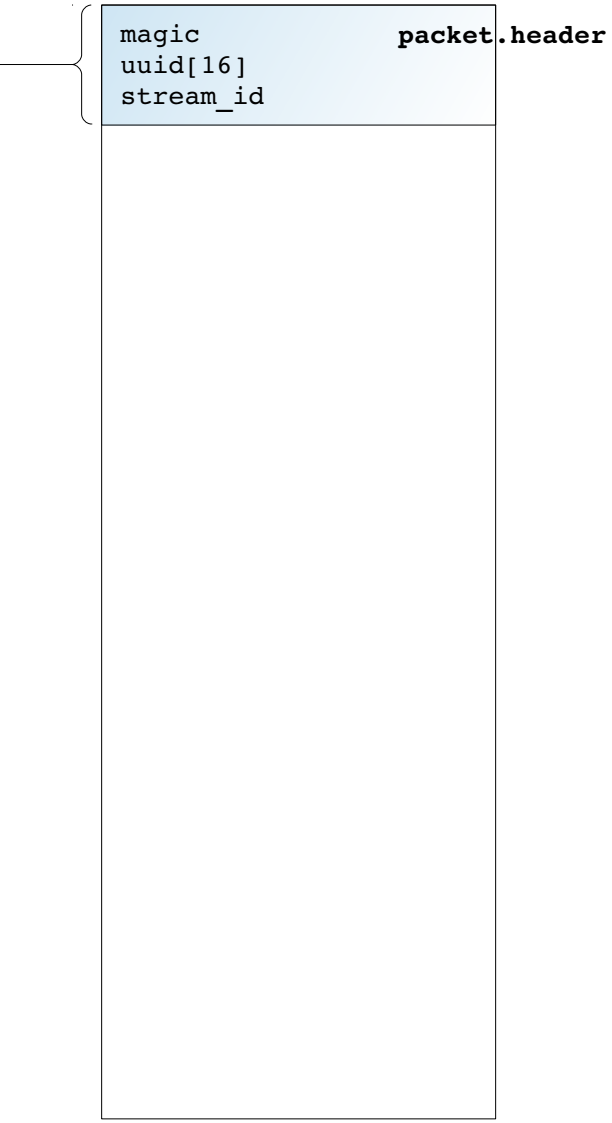
CTF - TSDL

```
$ ls lttng-sessions/krava-20140820-101028/kernel
channel0_0
channel0_1
channel0_2
channel0_3
metadata
```



CTF HEADER

```
trace {
        major = 1;
        minor = 8;
        uuid = "891ddd79-a1ed-1e4f-9ced-44d9f0d1b285";
        byte_order = le;
        packet.header := struct {
                uint32_t magic;
                uint8_t  uuid[16];
                uint32_t stream_id;
        };
};
```

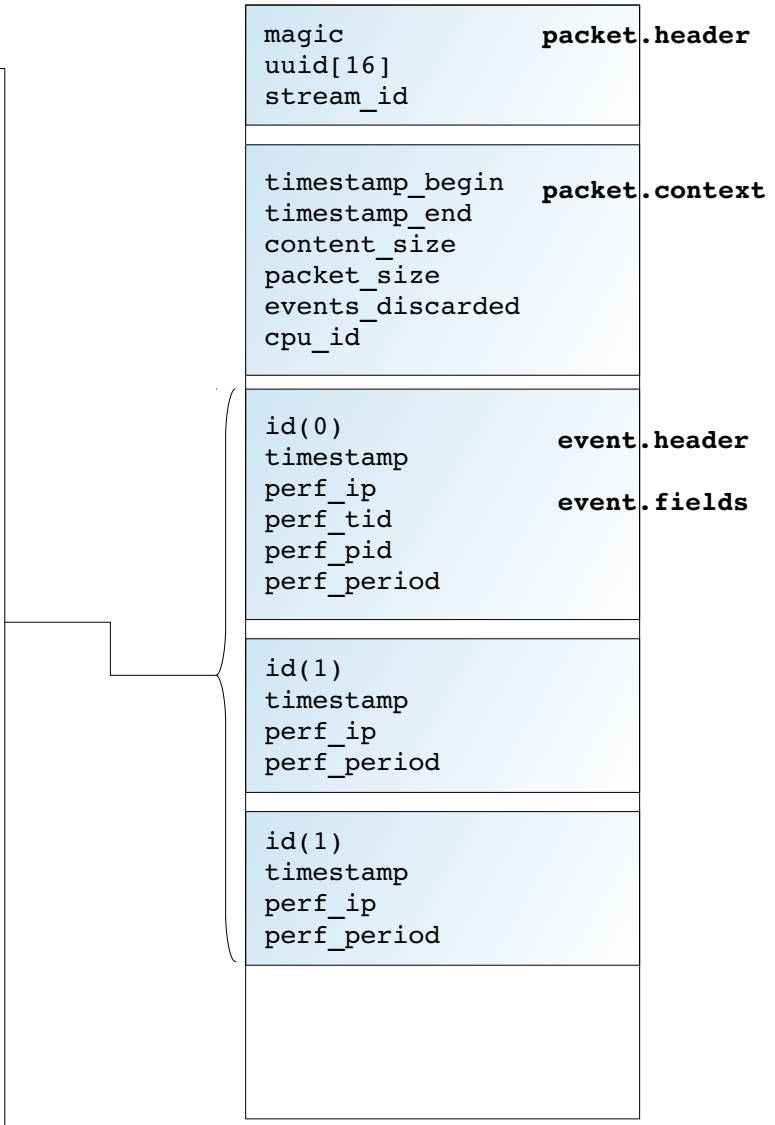


CTF STREAM

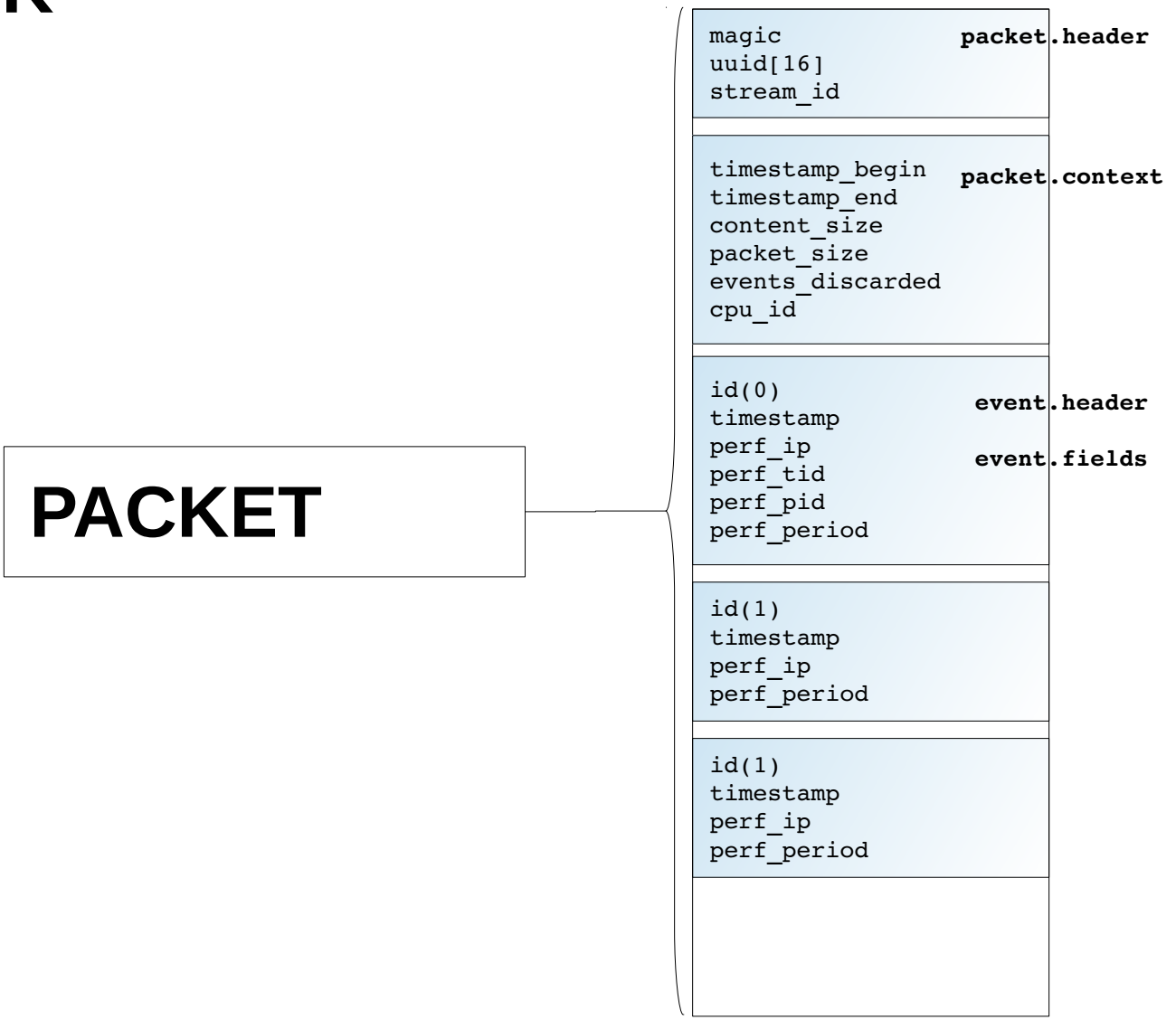
```
typealias integer { size = 32; align = 8; signed = false; } := uint32_t;
typealias integer { size = 64; align = 8; signed = false; } := uint64_t;

typealias integer {
        size = 64; align = 8; signed = false;
        map = clock.monotonic.value;
} := uint64_clock_monotonic_t;

struct packet_context {
        uint64_clock_monotonic_t timestamp_begin;
        uint64_clock_monotonic_t timestamp_end;
        uint64_t content_size;
        uint64_t packet_size;
        unsigned long events_discarded;
        uint32_t cpu_id;
};

stream {
        id = 0;
        packet.context := struct packet_context;
```

CTF EVENTS

```
struct event_header_large {
   uint32_t id;
   uint64_clock_monotonic_t timestamp;
} align(8);

stream {
        id = 0;
        event.header := struct event_header;
...
};

event {
        name = "cycles";
        id = 0;
        stream_id = 0;
        fields := struct {
           uint64_6 perf_ip;
           uint32_t perf_tid;
           uint32_t perf_pid;
           uint64_t perf_period;
        } align(1);
};

event {
        name = "cache-misses";
        id = 1;
        stream_id = 0;
        fields := struct {
           uint64_6 perf_ip;
           uint64_t perf_period;
        } align(1);
};
```

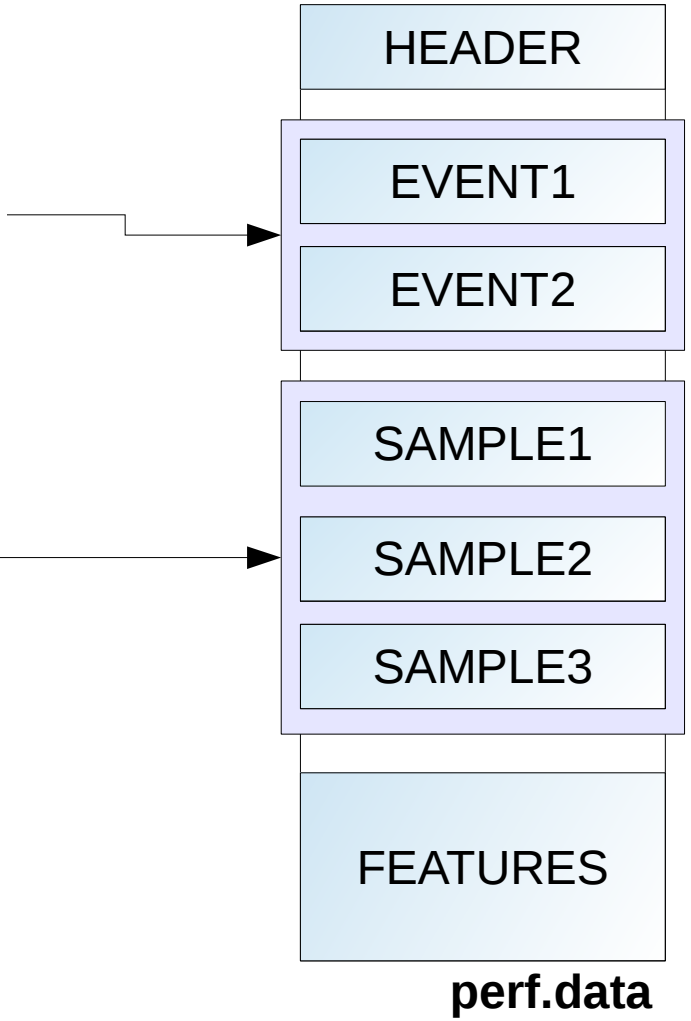


CTF NETWORK

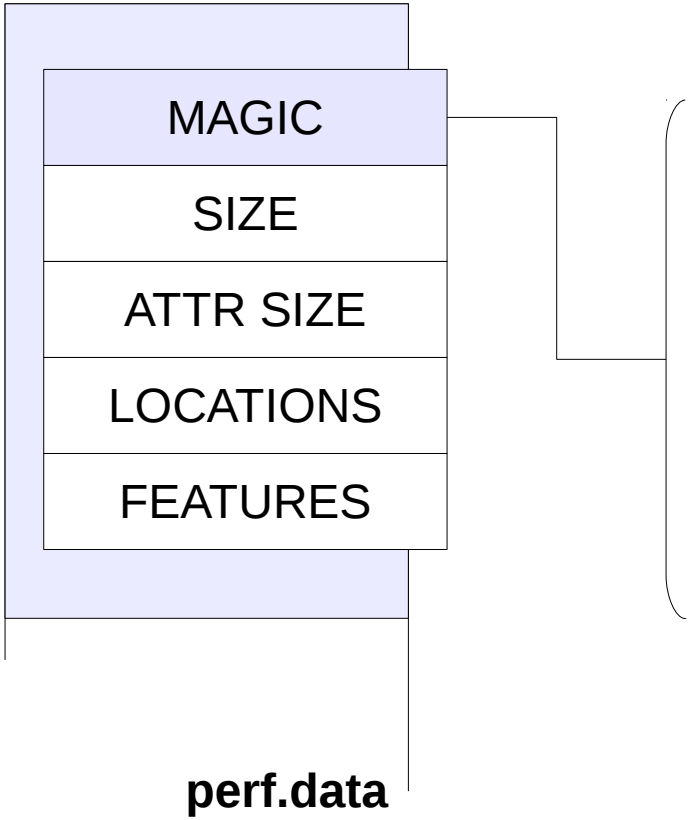

PACKET
magic
uuid[16]
stream_id
packet.header
timestamp_begin
timestamp_end
content_size
packet_size
events_discarded
cpu_id
packet.context
id(0)
timestamp
perf_ip
perf_tid
perf_pid
perf_period
event.header
event.fields
id(1)
timestamp
perf_ip
perf_period
id(1)
timestamp
perf_ip
perf_period

PERF.DATA FORMAT

- HEADER
- EVENT DESCRIPTIONS
- EVENT DATA
- FEATURES



PERF.DATA FORMAT - HEADER

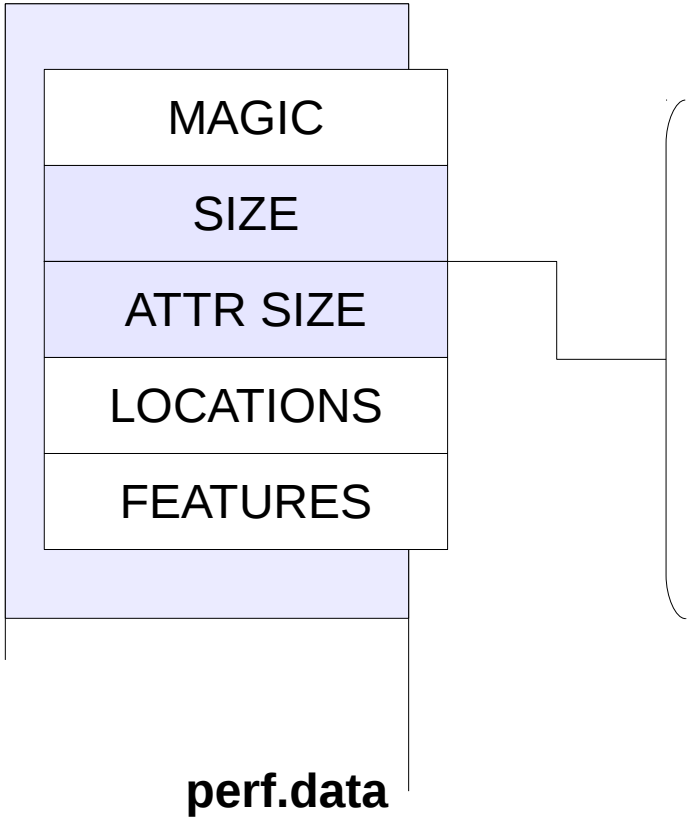


Value:

- PERFFILE
- PERFILE2

Set version and endianity

PERF.DATA FORMAT - HEADER

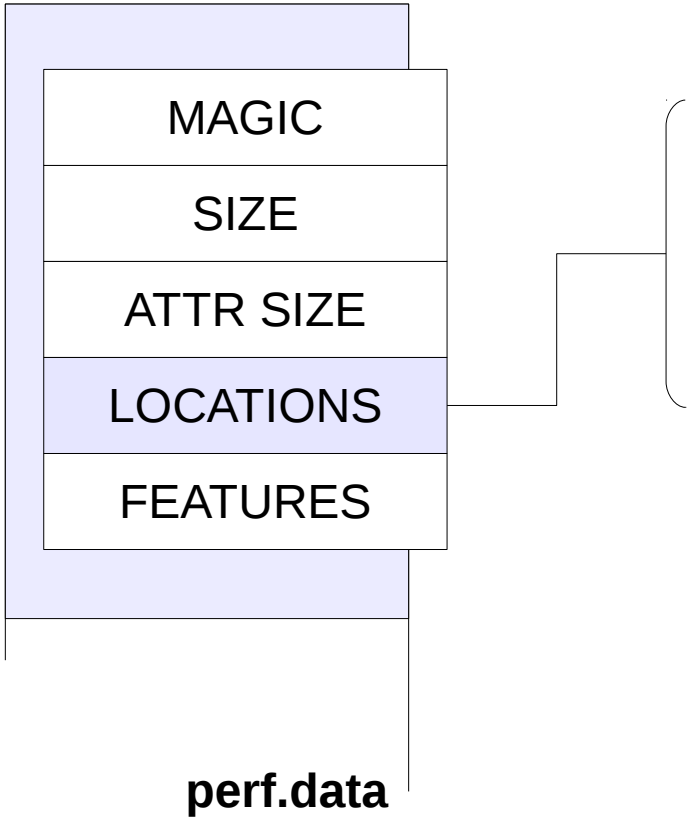


Sizes of:

- file header
- `struct perf_event_attr`

set file & kernel interface

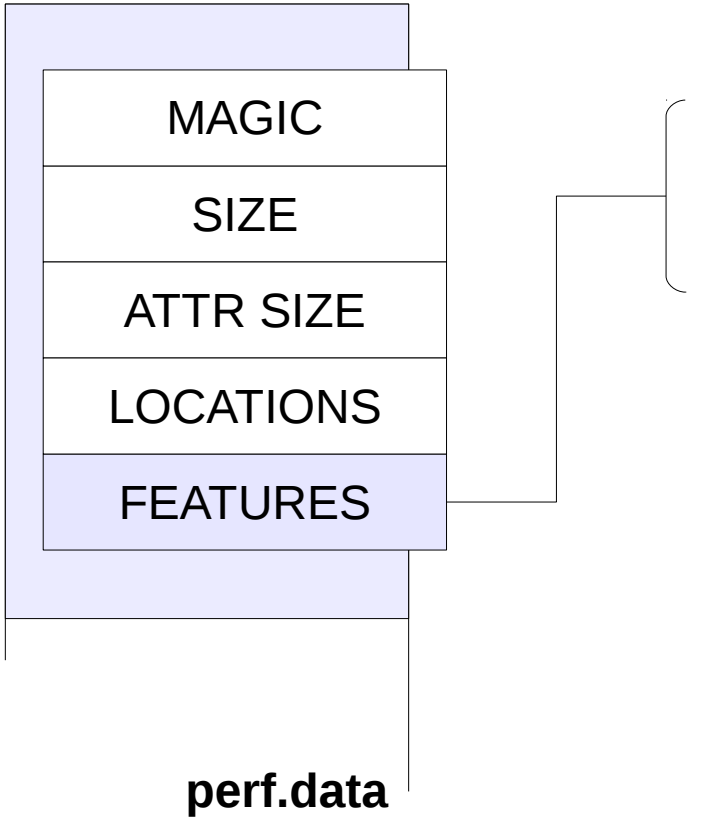
PERF.DATA FORMAT - HEADER



Locations of:

- event descriptions
- data

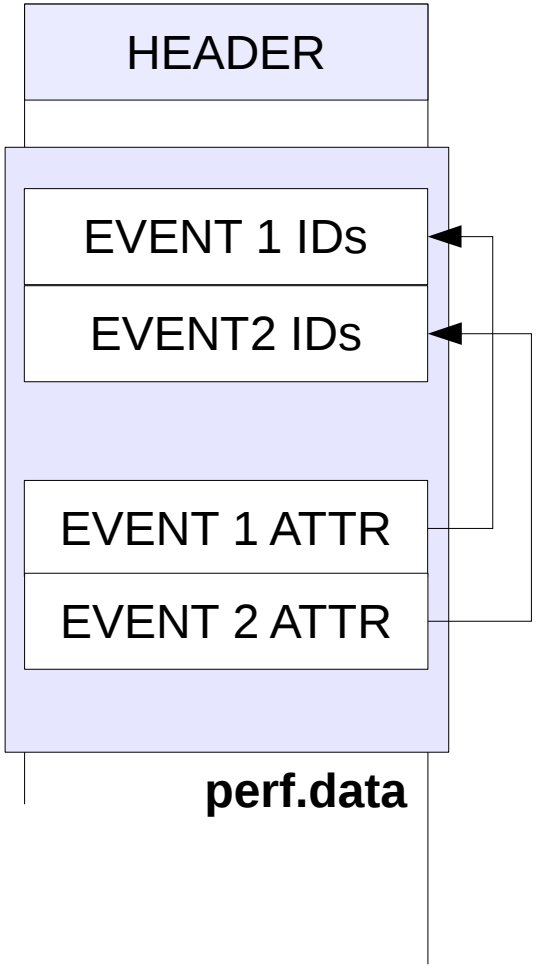
PERF.DATA FORMAT - HEADER



Features bitmap

- 256 bits

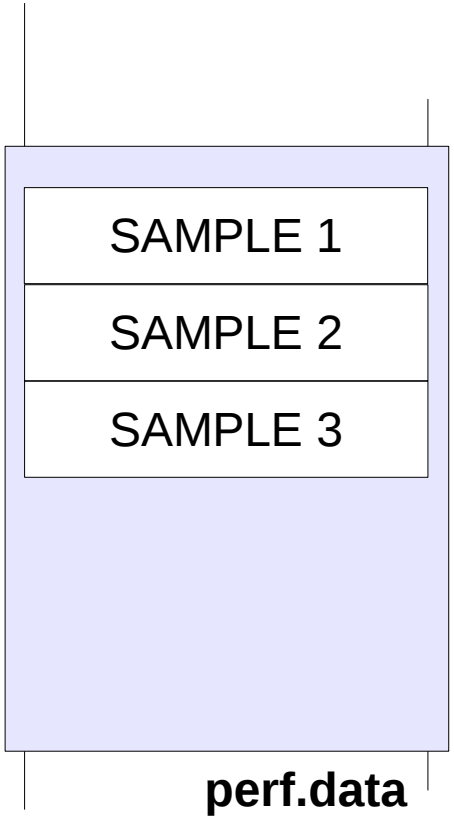
PERF.DATA FORMAT – EVENT DESCRIPTIONS



`struct perf_event_attr`

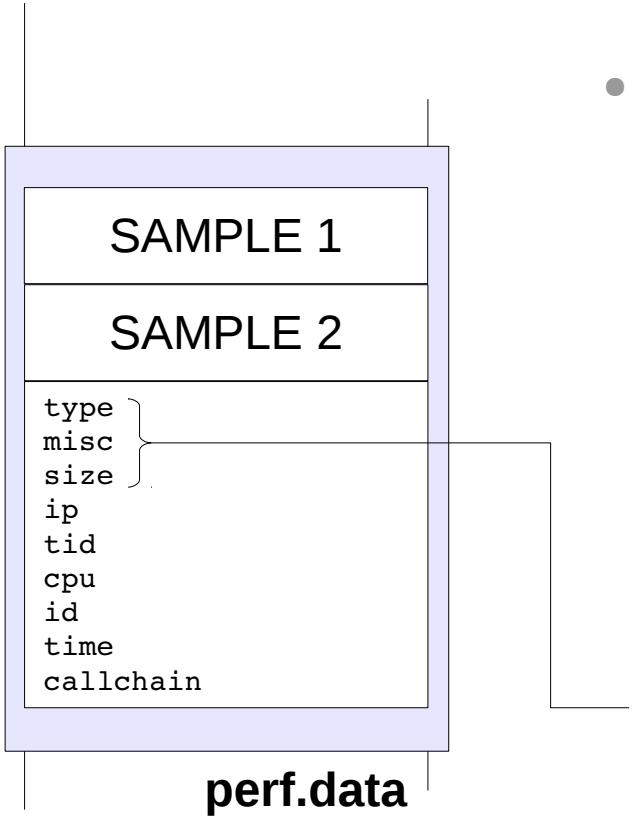
- event attribute structure
- linked with IDs array
- need IDs to properly resolve samples

PERF.DATA FORMAT – DATA



blob of samples

PERF.DATA FORMAT – SAMPLE



- sample = header + data

```
struct perf_event_attr::sample_type {
        PERF_SAMPLE_IP                  = 1U << 0,
        PERF_SAMPLE_TID                 = 1U << 1,
        PERF_SAMPLE_TIME                = 1U << 2,
        PERF_SAMPLE_ADDR                = 1U << 3,
        PERF_SAMPLE_READ                = 1U << 4,
        PERF_SAMPLE_CALLCHAIN           = 1U << 5,
        PERF_SAMPLE_ID                  = 1U << 6,
        PERF_SAMPLE_CPU                 = 1U << 7,
        PERF_SAMPLE_PERIOD              = 1U << 8,
        PERF_SAMPLE_STREAM_ID           = 1U << 9,
        ...
};
struct perf_event_header {
     __u32   type;
     __u16   misc;
     __u16   size;
};
```


PERF.DATA FORMAT – FEATURES

various system data

- **256 features total**
- **17 taken:**

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

COMPARISON

CTF	PERF.DATA
• easily extensible	• perf specific
• network data streaming	• send whole file
• multiple files storage	• single file storage
• no binaries attached	• self contained (build ids, binaries)

CONVERSION

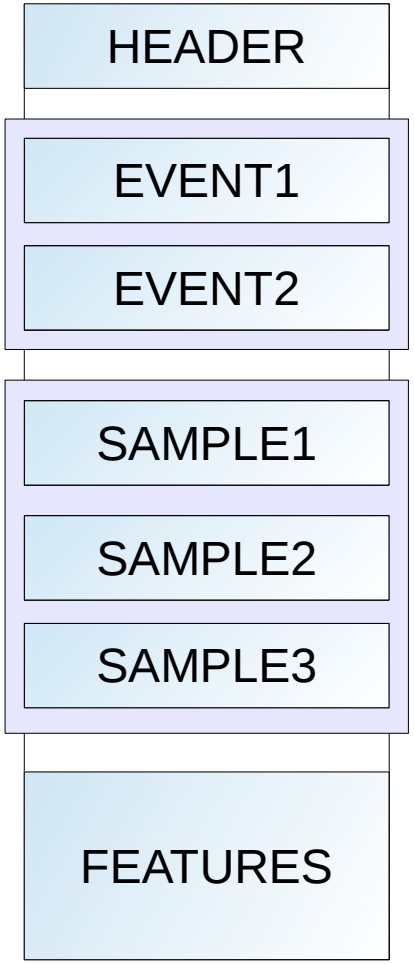
- perf.data -> CTF
  1. convert events 1 by 1
  2. describe PERF.DATA stream via CTF

PERF DATA CONVERT

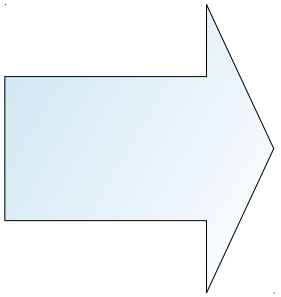
- converts events 1 by 1
- libbabeltrace
- Sebastian Andrzej Siewior & me, stored in [1]

```
$ make LIBBABELTRACE_DIR=/opt/libbabeltrace
Auto-detecting system features:
...
...                      libunwind: [ on  ]
...             libdw-dwarf-unwind: [ on  ]
...                  libbabeltrace: [ on  ]

  GEN      common-cmds.h
  FLAGS:   * new build flags or prefix
...

$ ./perf record ls
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.013 MB perf.data (~579 samples) ]

$ LD_LIBRARY_PATH=/opt/libbabeltrace/lib/ ./perf data convert --to-ctf ./ctf-data
[ perf data convert: Converted 'perf.data' into CTF data './ctf-data' ]
[ perf data convert: Converted and wrote 0.001 MB (21 samples) ]
```


CONVERSION



perf.data



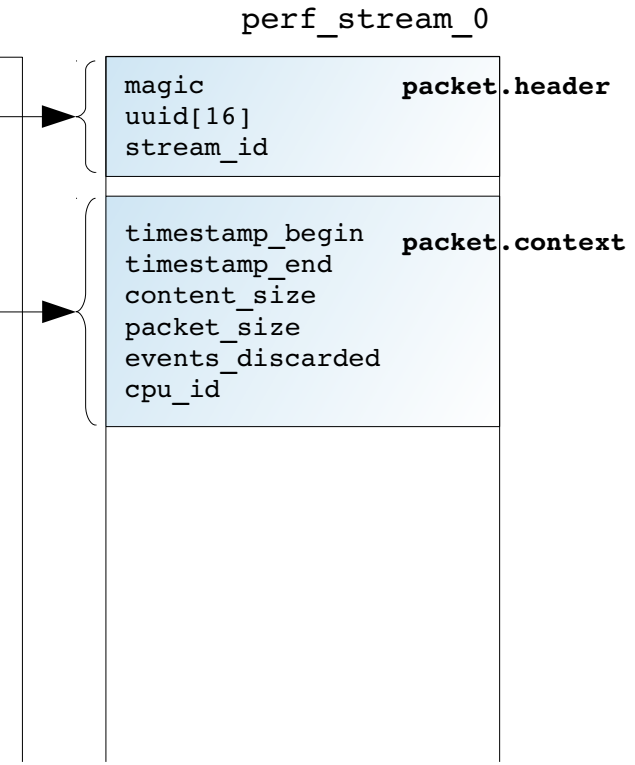
CONVERTED DATA

```
$ ls ./ctf-data
metadata
perf_stream_0
```


CONVERSION

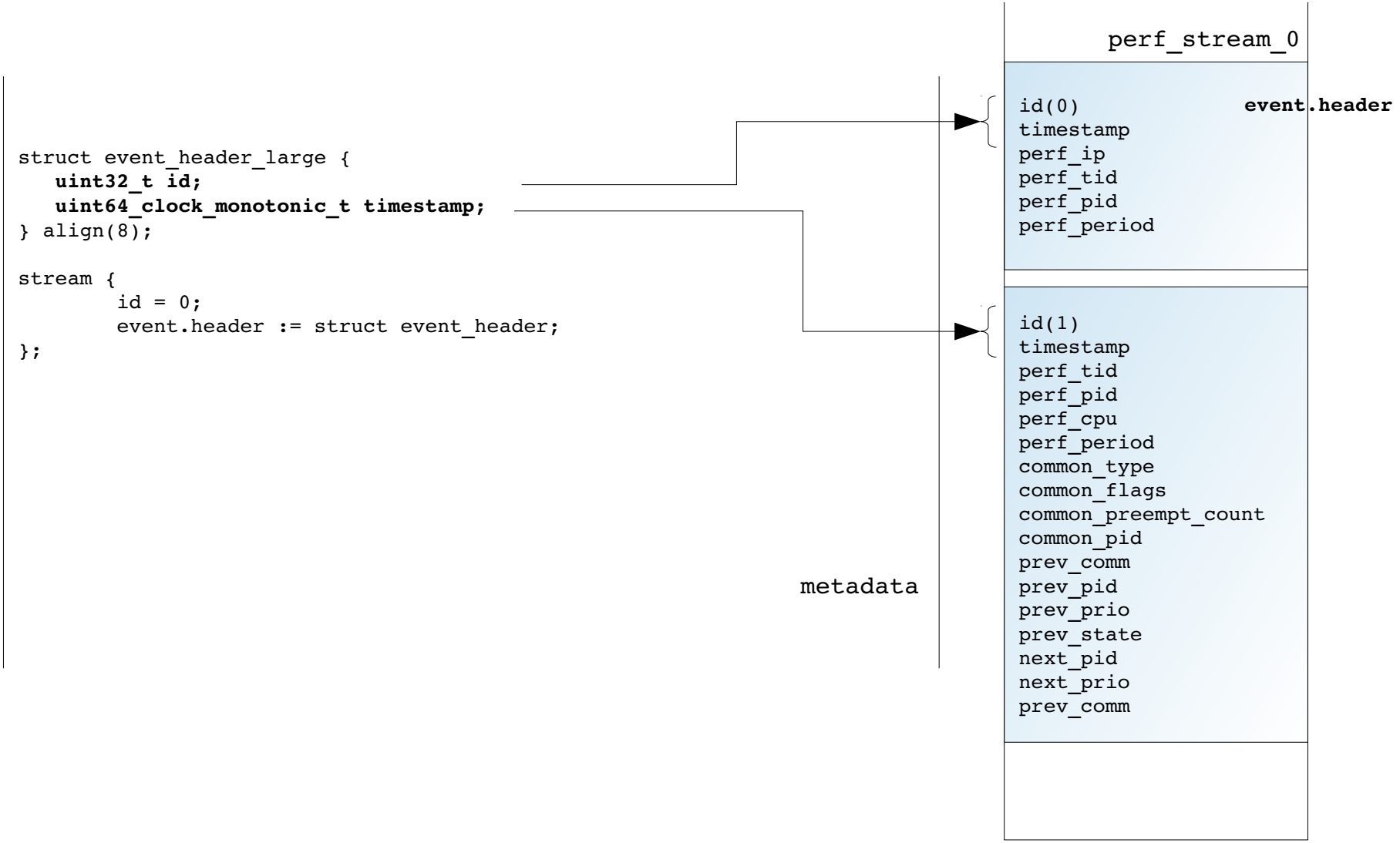
```
trace {
        packet.header := struct {
                uint32_t magic;
                uint8_t  uuid[16];
                uint32_t stream_id;
        };
};

struct packet_context {
        uint64_clock_monotonic_t timestamp_begin;
        uint64_clock_monotonic_t timestamp_end;
        uint64_t content_size;
        uint64_t packet_size;
        unsigned long events_discarded;
        uint32_t cpu_id;
};

stream {
        id = 0;
        packet.context := struct packet_context;
};
```

metadata



CONVERSION

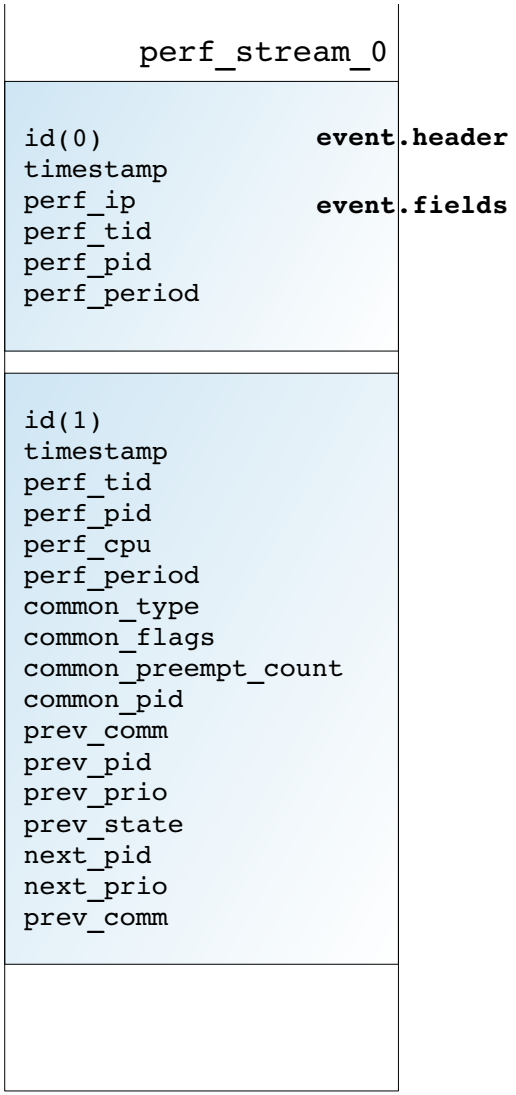
```
struct event_header_large {
   uint32_t id;
   uint64_clock_monotonic_t timestamp;
} align(8);

stream {
        id = 0;
        event.header := struct event_header;
};
```



CONVERSION

```
event {
        name = "cycles";
        id = 0;
        stream_id = 0;
        fields := struct {
           uint64_6 perf_ip;
           uint32_t perf_tid;
           uint32_t perf_pid;
           uint64_t perf_period;
        } align(1);
};

event {
        name = "sched:sched_switch";
        id = 1;
        stream_id = 0;
        fields := struct {
                uint32_t perf_tid;
                uint32_t perf_pid;
                uint32_t perf_cpu;
                uint64_t perf_period;
                uint32_t common_type;
                uint32_t common_flags;
                uint32_t common_preempt_count;
                uint32_t common_pid;
                string   prev_comm;
                uint32_t prev_pid;
                uint32_t prev_prio;
                uint64_t prev_state;
                uint32_t next_pid;
                uint32_t next_prio;
                string   prev_comm;
        } align(8);
};
```

metadata



PERF DATA CONVERT

- no symbol resolving
- no callchains

FUTURE

- mix perf data with lttng traces
- eclipse lttng plugin perf report integration? ;-)

THANKS, QUESTIONS

Jiri Olsa <jolsa@redhat.com>

LINKS

(1) git://git.kernel.org/pub/scm/linux/kernel/git/jolsa/perf.gitbranch
(branch perf/core_ctf_convert)