Analyzing DPDK applications with eBPF

Sharpening the toolset

Stephen Hemminger

Fosdem, February 1, 2020

Microsoft

Table of Contents

Introduction

Ancient wisdom

French proverb
Mauvés ovriers ne trovera ja bon hostill

Bad workers will never find a good tool

Ancient wisdom

French proverb
Mauvés ovriers ne trovera ja bon hostill

Bad workers will never find a good tool

Chinese proverb
To do a good job, a craftsman must sharpen his tools.

Methodology

- Don't focus on a tool set

Methodology

- Don't focus on a tool set
- Problem statement

Methodology

- Don't focus on a tool set
- Problem statement
- Workload Characterization

Methodology

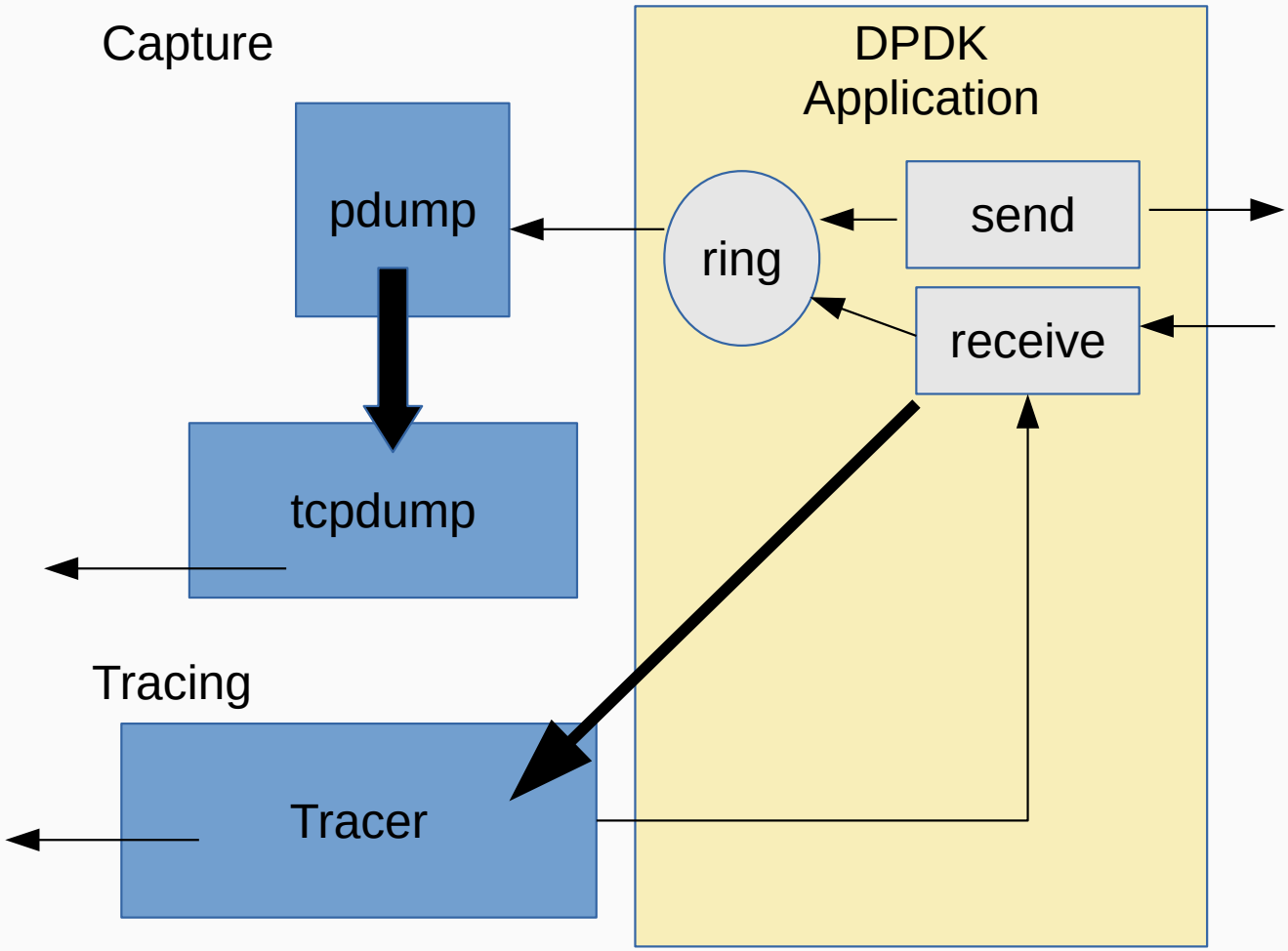
- Don't focus on a tool set
- Problem statement
- Workload Characterization
- USE
  - Utilization
  - Saturation
  - Errors

Methodology

- Don't focus on a tool set
- Problem statement
- Workload Characterization
- USE
  - Utilization
  - Saturation
  - Errors

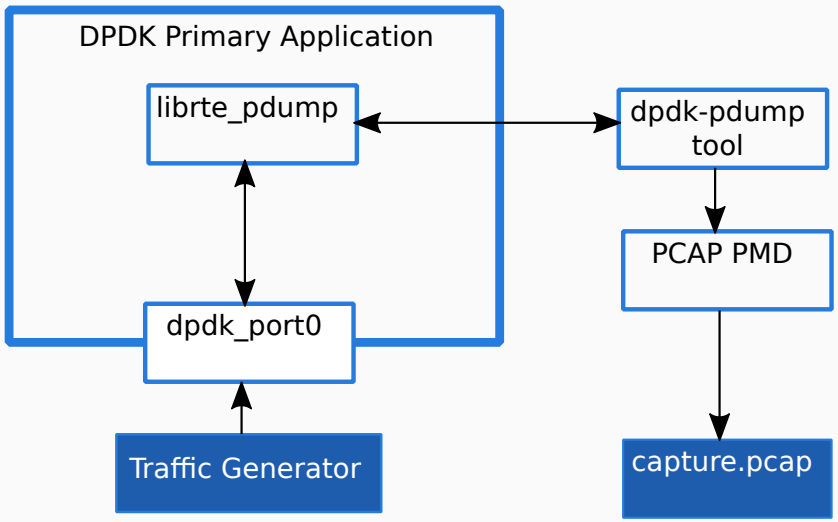
See Linux tracing talks (Brendan Gregg et al)

Capture vs Tracing



Packet Capture

DPDK pdump



- Packet copied and queued to ring
- Secondary process sends to libpcap
- Packets recorded in pcap format

Pdump limitations

- No metadata (vlan, offload, ...)

Pdump limitations

- No metadata (vlan, offload, ...)
- Inaccurate timestamp

Pdump limitations

- No metadata (vlan, offload, ...)
- Inaccurate timestamp
- No direction information

Pdump limitations

- No metadata (vlan, offload, ...)
- Inaccurate timestamp
- No direction information
- Single port only

Pdump limitations

- No metadata (vlan, offload, ...)
- Inaccurate timestamp
- No direction information
- Single port only
- No filtering

Pdump limitations

- No metadata (vlan, offload, ...)
- Inaccurate timestamp
- No direction information
- Single port only
- No filtering
- Poor performance

PCAP Next Generation

- Nanosecond resolution timestamp

PCAP Next Generation

- Nanosecond resolution timestamp
- System and Interface metadata

PCAP Next Generation

- Nanosecond resolution timestamp
- System and Interface metadata
- Multiple interfaces

PCAP Next Generation

- Nanosecond resolution timestamp
- System and Interface metadata
- Multiple interfaces
- Flags (direction, hash, ...)

PCAP Next Generation

- Nanosecond resolution timestamp
- System and Interface metadata
- Multiple interfaces
- Flags (direction, hash, ...)
- Comments

Packet filtering with libpcap

PCAP filter string: ip dst fosdem.org

```
cBPF program (6 insns):
(000) ldh      [12]
(001) jeq      #0x800           jt 2    jf 5
(002) ld       [30]
(003) jeq      #0x1f16168c      jt 4    jf 5
(004) ret      #65535
(005) ret      #0
```

Packet filtering cBPF

Translated to eBPF

```
eBPF program (11 insns):
 L0:    xor r0, r0
 L1:    xor r7, r7
 L2:    mov r6, r1
 L3:    ldh r0, [12]
 L4:    jne r0, #0x800, L9
 L5:    ldw r0, [30]
 L6:    jne r0, #0x1f16168c, L9
 L7:    mov32 r0, #0xffff
 L8:    exit
 L9:    mov32 r0, #0x1
 L10:   exit
```

Tracing

Linux Trace toolkit

- Easy to use

Linux Trace toolkit

- Easy to use
- User Defined Trace Points

Linux Trace toolkit

- Easy to use
- User Defined Trace Points
- Filtering

Linux Trace toolkit

- Easy to use
- User Defined Trace Points
- Filtering
- Common Trace Format

Linux Trace toolkit

- Easy to use
- User Defined Trace Points
- Filtering
- Common Trace Format
- High performance

Adding lttng tracepoint

```
/* Send burst of packets on an output interface */
static inline int
send_burst(struct lcore_conf *qconf, uint16_t n, uint16_t port)
{
  struct rte_mbuf **m_table = qconf->tx_mbufs[port].m_table;
  uint16_t queueid = qconf->tx_queue_id[port];
  int ret;

  ret = rte_eth_tx_burst(port, queueid, m_table, n);
  tracepoint(l3fwd, tx_burst, port, queueid, n, ret);
  if (unlikely(ret < n))
  ^^lrte_pktmbuf_free_bulk(&m_table[ret], n - ret);
  return 0;
}
```

Using eBPF from userspace

- Origin: dtrace

Using eBPF from userspace

- Origin: dtrace
- Adds NOP locations and ELF section

Using eBPF from userspace

- Origin: dtrace
- Adds NOP locations and ELF section
- Run code at tracepoint

Using eBPF from userspace

- Origin: dtrace
- Adds NOP locations and ELF section
- Run code at tracepoint
- Prerequisites

  uprobe Linux 3.14 (or later) kernel

Using eBPF from userspace

- Origin: dtrace
- Adds NOP locations and ELF section
- Run code at tracepoint
- Prerequisites

uprobe	Linux 3.14 (or later) kernel
sys/sdt.h	systemtap-std-dev

Using eBPF from userspace

- Origin: dtrace
- Adds NOP locations and ELF section
- Run code at tracepoint
- Prerequisites

uprobe	Linux 3.14 (or later) kernel
sys/sdt.h	systemtap-std-dev

Adding DTRACE probes

```
static void
pkt_burst_receive(struct fwd_stream *fs)
{
  struct rte_mbuf  *pkts_burst[MAX_PKT_BURST];
  uint16_t i, nb_rx;

  /* Receive a burst of packets. */
  nb_rx = rte_eth_rx_burst(fs->rx_port, fs->rx_queue,
  ^^I^^I^^I  pkts_burst, nb_pkt_per_burst);

  DTRACE_PROBE1(testpmd, rx_burst, nb_rx);

  if (unlikely(nb_rx == 0))
 ^^I^^Ireturn;
```

Looking for USDT

Use bpftrace to look for tracepoints in application

```
$ sudo  bpftrace -l "usdt:./build/app/testpmd"
  usdt:./build/app/testpmd:testpmd:rx_burst
```

Running bpftrace

Build a histogram of the number of packets per loop

```
  $ sudo bpftrace -e 'usdt:./build/app/testpmd:rx_burst  { @ = hist(arg0); }'
Attaching 1 probe...
^C

@:
[0]              16001930 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[1]                     0 |                                                          |
[2, 4)                  0 |                                                          |
[4, 8)                  0 |                                                          |
[8, 16)                 0 |                                                          |
[16, 32)                0 |                                                          |
[32, 64)          5333977 |@@@@@@@@@@@@@@@@@                                         |
```

Performance

Caveats

- Limited hardware - x85 with 25G NIC

Caveats

- Limited hardware - x85 with 25G NIC
- One off test

Caveats

- Limited hardware - x85 with 25G NIC
- One off test
- Untuned

Caveats

- Limited hardware - x85 with 25G NIC
- One off test
- Untuned
- Limited scope
  - Testpmd - 64 byte packets

Caveats

- Limited hardware - x85 with 25G NIC
- One off test
- Untuned
- Limited scope
  - Testpmd - 64 byte packets
  - Immediate drop

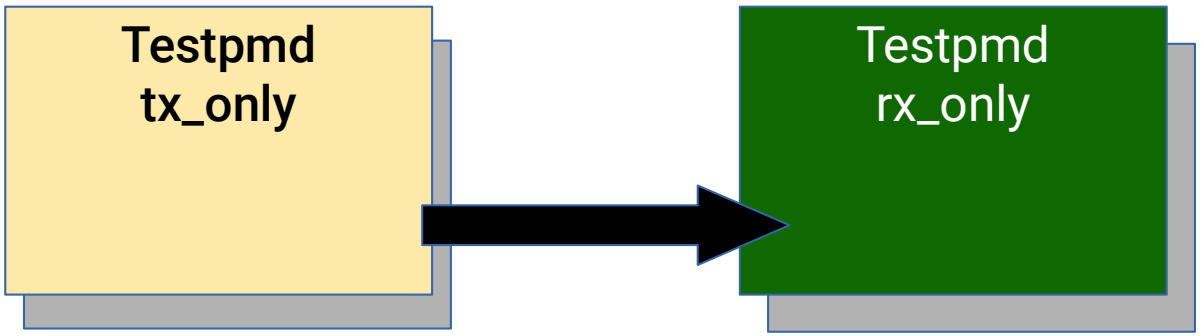
Caveats

- Limited hardware - x85 with 25G NIC
- One off test
- Untuned
- Limited scope
  - Testpmd - 64 byte packets
  - Immediate drop
  - Current DPDK 19.11

Caveats

- Limited hardware - x85 with 25G NIC
- One off test
- Untuned
- Limited scope
  - Testpmd - 64 byte packets
  - Immediate drop
  - Current DPDK 19.11
  - Single queue active

Test configuration



- Sender tx-only
- Receiver rx-only
- Observe Receive packets/sec

Capture and Tracing Performance

Test	Disabled	Enabled
Pdump	0	-36.85
Pdump + eBPF	0	0
Lttng	-0.42	-0.02
bpftrace	-0.01	-56.72

Conclusion

Ongoing work

- DPDK packet capture

Ongoing work

- DPDK packet capture
  - Pcapng support

Ongoing work

- DPDK packet capture
  - Pcapng support
  - capture filter

Ongoing work

- DPDK packet capture
  - Pcapng support
  - capture filter
  - dumpcap (tshark) syntax

Ongoing work

- DPDK packet capture
  - Pcapng support
  - capture filter
  - dumpcap (tshark) syntax

Ongoing work

- DPDK packet capture
  - Pcapng support
  - capture filter
  - dumpcap (tshark) syntax
- DPDK trace points

Thank you

- Questions
- Thanks
  - DPDK community
  - LTTng
  - eBPF developers
- Contact

  stephen@networkplumber.org

  @networkplumber