redhat.

Testing the Performance Impact of the Exact Match Cache

Now with Signature Match Cache Comparison!

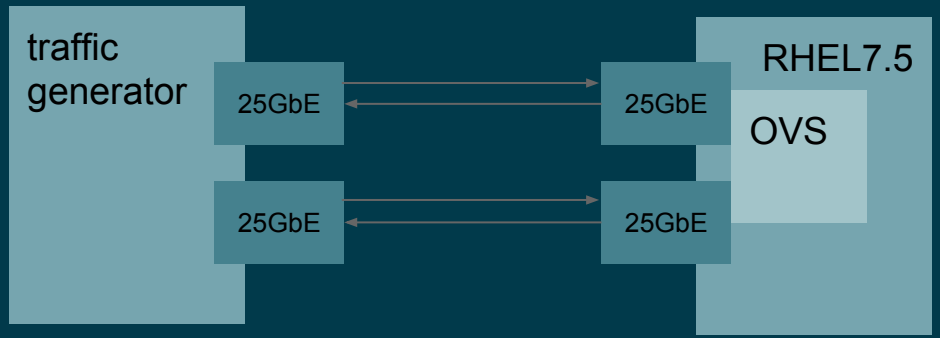
Andrew Theurer - Sr. Principal Software Engineer
December 2018

Why Does the Cache Matter?

- DPDK PMD in OVS follows a basic sequence:
  - receive packets
  - identify packet
  - **look up action**
  - execute action

Testing Methodology

- Hardware
  - “Device-under-test” uses:
    - 2 x Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz
    - 2 x Intel Ethernet Controller XXV710 for 25GbE
  - Traffic generator uses similar hardware
- Network



Testing Methodology

- Software
  - Openvswitch
    - Using “dpdk-latest” branch, from 25 October, commit 270d9216f1ed0dcafd1eed9df93e090ffef8bfa8
    - Built with “-O2 -msse4.2”
    - DPDK 18.11-rc2
    - 2 cores dedicated to OVS, same NUMA node as network
    - Simple forwarding rules, “in port0, out port1”, no MAC learning
  - Linux
    - RHEL 7.5
    - tuned “cpu-partitioning” profile
      - automates tuning for isolation, well suited for DPDK applications

Testing Methodology

- Benchmark: pbench-trafficgen [1]
  - “RFC2544-like”, test for peak packet throughput while not exceeding specified packet loss
    - Most tests here using 0.002% maximum packet loss
  - Peak packet throughput found with binary-search.py [2]. Binary-search can:
    - Use pluggable traffic generators (TRex, Moongen)
    - Support multi-device-pair interfaces to scale to very high rates
    - Work with Pbench [2] to automate tool collection and run several tests for multiple frame sizes, loss-ratios, flow-counts
  - starts/stops performance tools, including openvswitch stats

[1] https://github.com/distributed-system-analysis/pbench/blob/master/agent/bench-scripts/pbench-trafficgen
[2] https://github.com/atheurer/trafficgen/blob/master/binary-search.py

Testing Methodology

- pbench-trafficgen -> binary-search
  - Basic process:
    - conduct a trial test at specific TX packet rate
    - adjust next trial packet rate based on pass/fail
    - repeat until search granularity is met
  - Three possible trial elapsed times:
    - sniff: quick test to check for failure
    - search: primary test for pass/fail
    - final_validation: full length test for final pass requirement

Testing Methodology

- pbench-trafficgen -> binary-search -> pluggable traffic generators [3]
  - binary-search.py --traffic-generartor=
    - trex-txrx.py (our first TRex based generator)
    - trex-txrx-profile.py (our latest TRex based generator)
      - used for these tests
    - trafficgen.lua (uses Moongen, but we have not tested this lately)
      - Moongen does have HW timestamping for most accurate latency stats, TRex does not use this feature
  - When binary-search conducts a trial, a separate process is launched to generate the traffic and return statistics. Binary-search processes those stats and decides what rate to run next

[3] https://github.com/atheurer/trafficgen

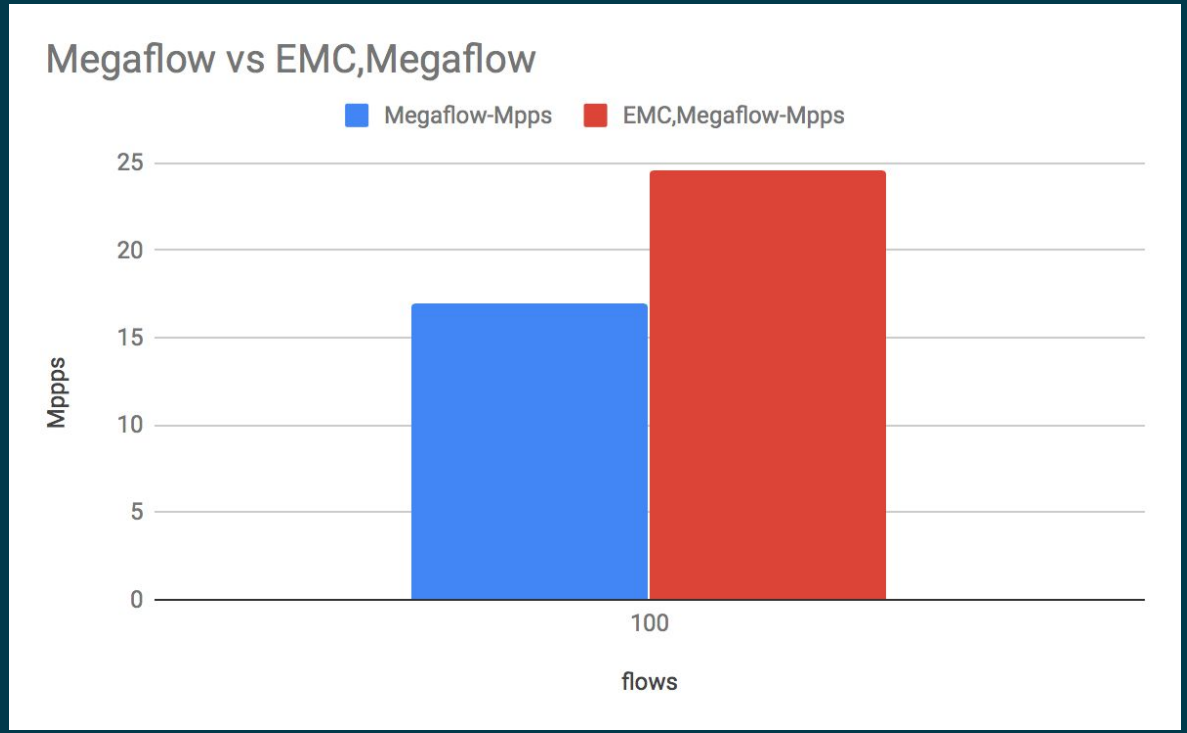
Testing Methodology

- pbench-trafficgen -> binary-search -> pluggable traffic generators -> trex-profile.py
  - Uses TRex
  - User-defined traffic profile wirth 1 or more streams via json (example):

```
{ "streams": [
        {
          "flows": 100,
          "frame_size": 64,
          "flow_mods": "function:create_flow_mod_object(use_src_ip_flows=True, use_dst_ip_flows=True)",
          "rate": 1000000,
          "frame_type": "generic",
          "stream_types": [
         "teaching_warmup",
                "measurement"
          ]
        }
    ]}
```

EMC Performance

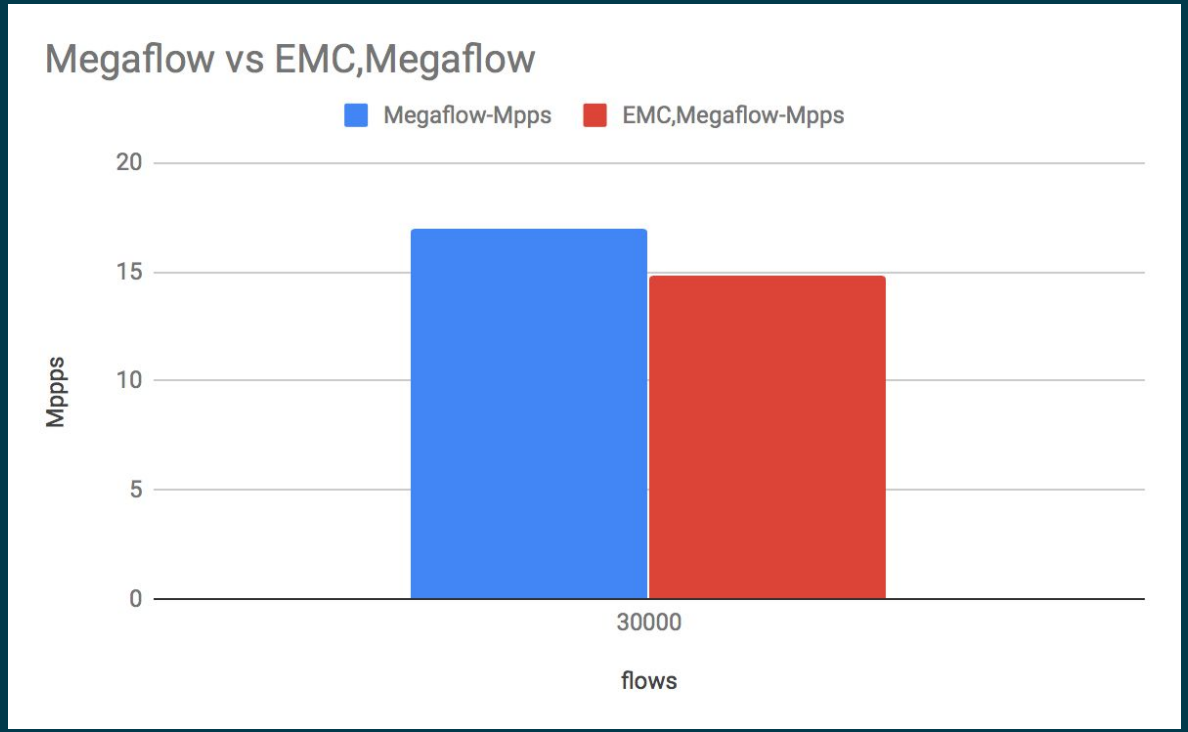
- EMC is best...



0.002% max loss

EMC Performance

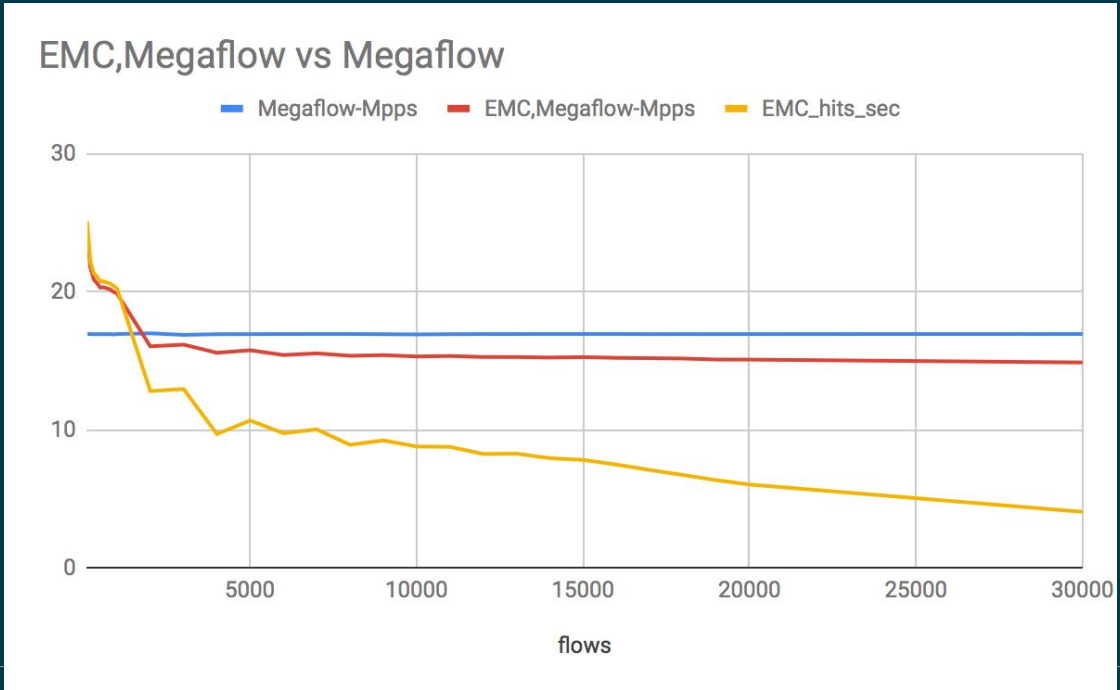
- until it's not...



0.002% max loss

EMC Performance

- As the number of flows increase, the benefit of the cache is diminished and eventually the overhead of using this cache (and not getting a hit) degrades performance below no EMC at all

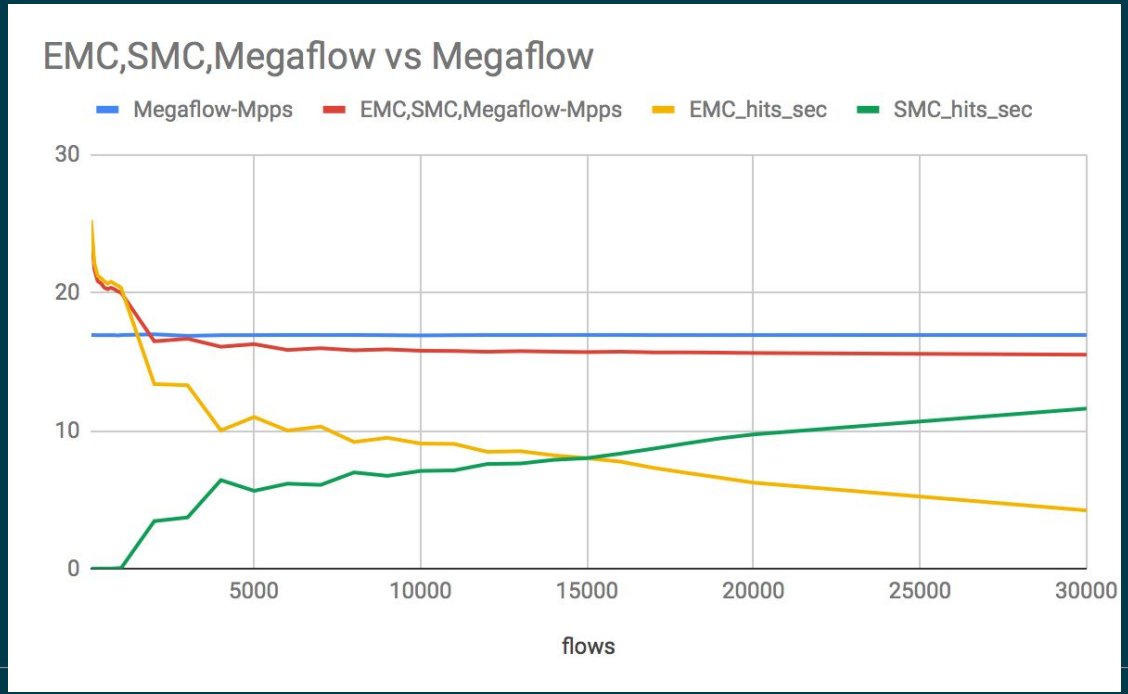


Note that flows fall out of the EMC and are found in Megaflow with as little as 800 flows, in spite of a EMC cache size of 8192

0.002% max loss

EMC + SMC Performance

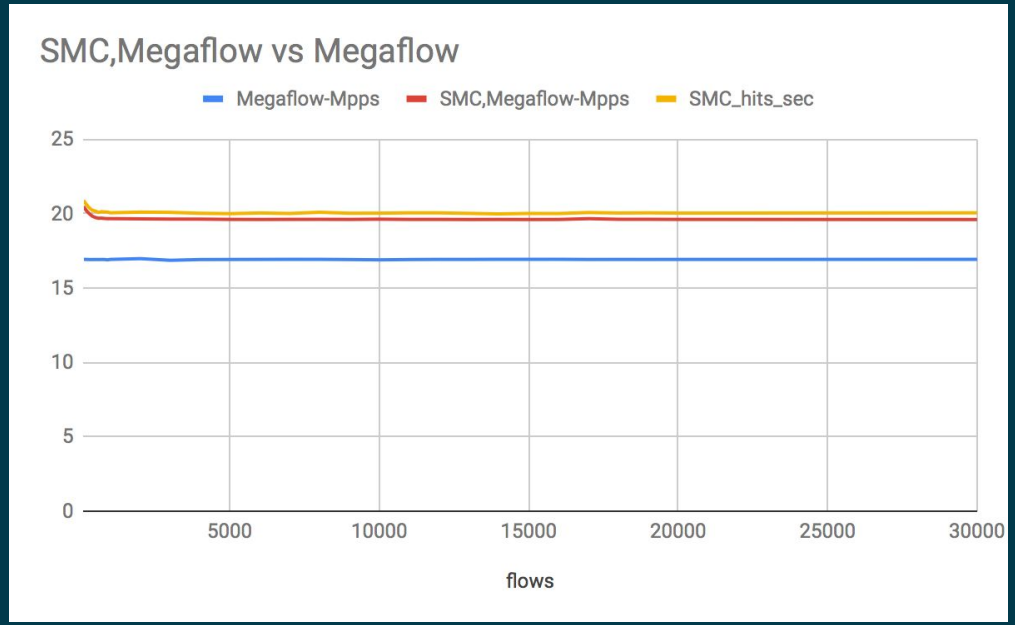
- Using EMC and SMC together still degrades compared to Megaflow as flows increase, but the gap is not as large as just-EMC. All flows not cached in EMC are found in SMC.



0.002% max loss

SMC Performance

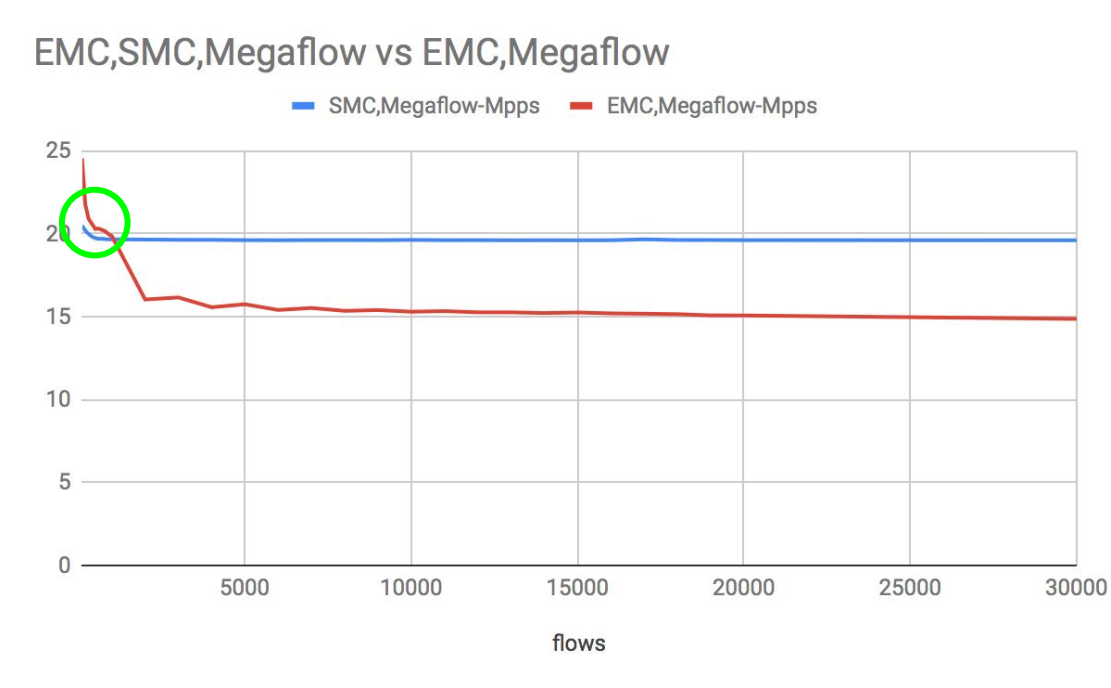
- Adding SMC and not using EMC performance better than just-Megaflow, does not have overhead of using EMC. 100% of flows are cached in SMC.



0.002% max loss

SMC Performance

- However, we still underperform EMC a low flow count



100 flows: only-SMC 16.5% lower than only-EMC

0.002% max loss

SMC Performance

- When using SMC, we have extra overhead in **`dpcls_rule_matches_key()`**
  - perf report for 100-flow test

```
# Samples: 16K of event 'cycles:ppp'
# Event count (approx.): 385110649393
#
# Overhead       Samples  Command          Shared Object          Symbol
# ........  ............  ...............  .....................  ........
    10.52%          1304  pmd55            ovs-vswitchd           [.] miniflow_extract
    10.26%          1275  pmd54            ovs-vswitchd           [.] miniflow_extract
     9.98%          1241  pmd55            ovs-vswitchd           [.] dp_netdev_input__
     9.90%          1230  pmd54            ovs-vswitchd           [.] dp_netdev_input__
     8.18%          1015  pmd54            ovs-vswitchd           [.] dpcls_rule_matches_key
     7.67%           954  pmd55            ovs-vswitchd           [.] dpcls_rule_matches_key
     4.74%           589  pmd54            ovs-vswitchd           [.] i40e_recv_pkts_vec
     4.63%           576  pmd55            ovs-vswitchd           [.] i40e_xmit_fixed_burst_vec
     4.50%           560  pmd55            ovs-vswitchd           [.] i40e_recv_pkts_vec
     4.42%           550  pmd54            ovs-vswitchd           [.] i40e_xmit_fixed_burst_vec
     1.95%           243  pmd54            ovs-vswitchd           [.] non_atomic_ullong_add
     1.79%           222  pmd55            ovs-vswitchd           [.] non_atomic_ullong_add
     1.70%           211  pmd55            ovs-vswitchd           [.] netdev_dpdk_rxq_recv
     1.42%           176  pmd54            ovs-vswitchd           [.] netdev_dpdk_rxq_recv
     1.21%           151  pmd54            ovs-vswitchd           [.] dp_execute_cb
     1.14%           142  pmd55            ovs-vswitchd           [.] dp_execute_cb
```

Summary

- EMC can perform best only when near 100% hit rate
  - However, EMC is relatively small and many flows do not end up being cached, even if total number of flows is lower than 8192. We need to understand why this is and fix this.
  - If the total number of flows is significantly higher than the EMC capacity, then the EMC overhead is quite big for no real gain. Could we dynamically turn on/off EMC based on incoming flow heuristics?
- SMC (without EMC) performs best in almost all flow counts (except <200)
  - The overhead in using SMC is greater, perhaps this can be optimized
  - Should we consider just enabling SMC and not EMC by default?

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos