Scalable Heavy Hitter Detection in Virtualized Environments: A DPDK-based Software Approach with P4 Integration

Samia Choueiri, Ali Mazloum, Elie Kfoury, Jorge Crichigno
University of South Carolina, Columbia, South Carolina

Abstract

- Identifying heavy hitters is vital for applications like Denial of Service (DoS) detection and traffic engineering.
- Hardware solutions (e.g., programmable data plane switches) offer high performance but require adding hardware, which is not ideal for virtualized environments (e.g., cloud).
- Software solutions are flexible but suffer from performance issues due to the packet processing overhead in the OS kernel.
- The proposed system implements a scalable heavy hitter detection algorithm in the software, bypassing the kernel using the Data Plane Development Kit (DPDK).
- The Count-min Sketch (CMS) algorithm is used to track the number of packets per flow.
- The system is implemented in P4 and deployed on the P4-DPDK target running on CPU cores.
- The system was tested with various packet sizes, number of CPU cores, and number of hash functions.
- Evaluation results show accurate identification of heavy hitters at traffic rates approaching 100Gbps.

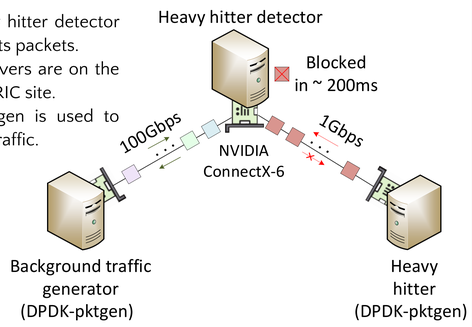
Related Work

- P4 switches approaches:
  - HH-IPG [1] is a per-flow heavy hitter detection approach on P4 Tofino. The system requires adding hardware.
  - Ding et al. [2] proposed a network-wide monitoring heavy hitter algorithm using the Count-min Sketch implemented in P4. The system requires adding hardware
- DPDK approaches:
  - Elastic Sketch [3] is a network-wide detection of heavy flow implemented on P4-OVS with DPDK. The system is tested with low traffic rates (10Gbps).

[1] SK. Singh et al., "HH-IPG: Leveraging Inter-Packet Gap Metrics in P4 Hardware for Heavy Hitter Detection," 2023.
[2] D. Ding et al., "An incrementally-deployable P4-enabled architecture for network-wide heavy-hitter detection," 2020.
[3] T. Yang et al., "Elastic Sketch: Adaptive and Fast Network-wide Measurements," 2018.

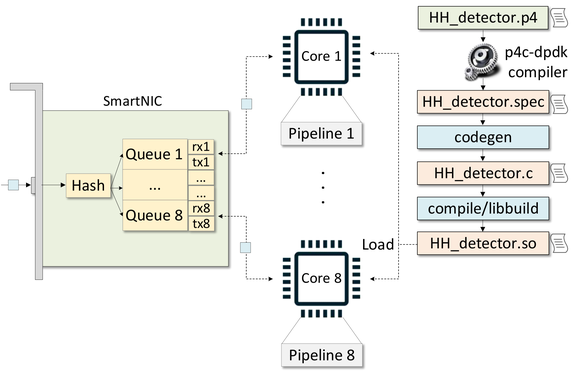
Experiment Topology

- The heavy hitter detector also reflects packets.
- All the servers are on the same FABRIC site.
- DPDK-pktgen is used to generate traffic.



System Architecture

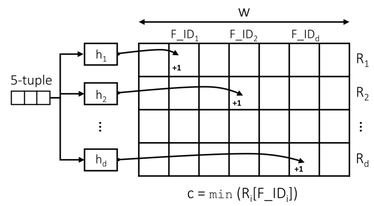
- The P4 language with the Portable NIC Architecture (PNA) is used to implement the heavy hitter detection algorithm.
- The p4c-dpdk compiler translates P4 programs into DPDK API, allowing the configuration of the DPDK pipeline.
- The DPDK pipeline is compiled and loaded on the CPU cores of the host.
- The SmartNIC uses Receive Side Scaling (RSS) to distribute the load across the pipelines in the CPU cores.



P4-DPDK heavy hitter detection system architecture

Methodology

- The CMS algorithm is used to detect heavy hitters.
- CMS is a probabilistic data structure that serves as a frequency table of events in a stream of data.
- It uses multiple hash functions and register arrays. The hash functions map the events to frequencies in each of the arrays.
- The hash function takes as input the 5-tuple: source/destination IP, source/destination port, and protocol.
- The heavy hitter is detected by comparing the minimum frequency for a flow in the CMS against a predefined threshold.



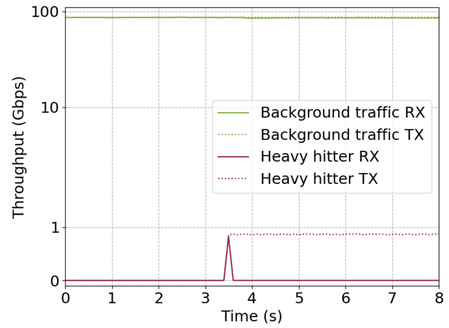
Count-Min Sketch (CMS) for estimating the per-flow packet counts

Acknowledgement

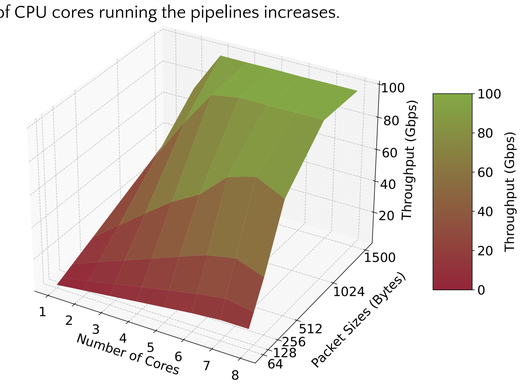
- This work was supported by the National Science Foundation (NSF), Award 2118311.

Performance Results of Heavy Hitter Detection

- Background traffic is generated at the rate of 100Gbps using DPDK-pktgen.
- A heavy hitter is "immediately" identified and blocked even while processing 100Gbps of background traffic.
- The maximum throughput that can be achieved increases as the number of CPU cores running the pipelines increases.

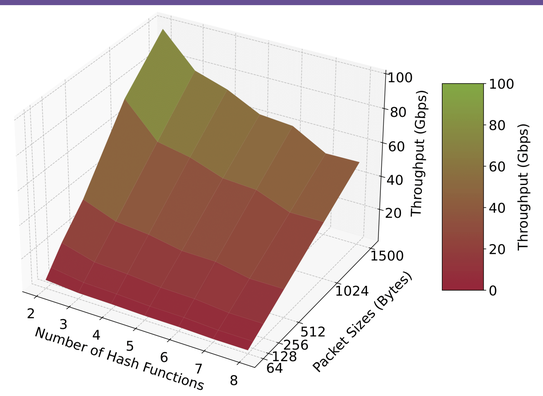


Heavy hitter mitigation at line rate

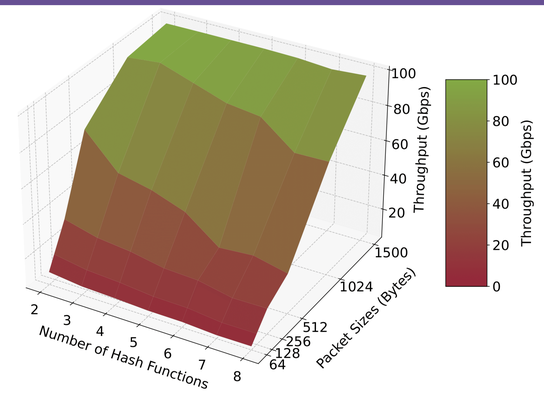


Throughput as a function of number of cores and packet size

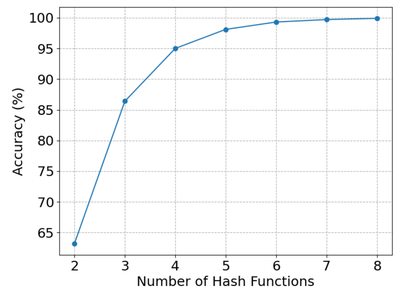
Evaluating the Impact of the Number of Hash Functions on the Throughput



Throughput as a function of the number of hash functions and packet size using two CPU cores



Throughput as a function of the number of hash functions and packet size using four CPU cores



Packet count accuracy based on the number of hash functions

Lessons Learned and Future Work

- While CMS introduces some estimation errors, it offers a tunable trade-off between accuracy and memory usage based on the number of hash functions and the size of the register arrays.
- The experimental results show that increasing the number of hash functions improves the counting accuracy but degrades the throughput.
- Using two CPU cores, the performance penalty is high, especially with small packet sizes. With four CPU cores, the performance penalty is eliminated even with a high number of hash functions.
- Future work includes 1) testing the system with real traffic from CAIDA; 2) augmenting the P4 program to detect cyberattacks; 3) profiling the performance of P4-DPDK with various P4 programs.