Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Key Properties of Programmable Data Plane Targets

Dominik Scholz, Henning Stubbe,
Sebastian Gallenmüller, Georg Carle

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Motivation

Move to the Data Plane

From SDN with OpenFlow to (fully) programmable data planes (e.g. P4, POF, eBPF)

Lots of new P4 applications that run in the data plane

- inband network telemetry
- in-network computation
- protocol acceleration (e.g. congestion control)
- middleboxes (DDoS mitigation)
- ...

Motivation

Move to the Data Plane

From SDN with OpenFlow to (fully) programmable data planes (e.g. P4, POF, eBPF)

Lots of new P4 applications that run in the data plane

- inband network telemetry
- in-network computation
- protocol acceleration (e.g. congestion control)
- middleboxes (DDoS mitigation)
- . . .

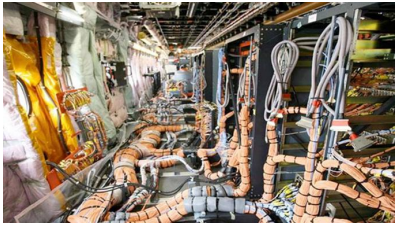


Image from https://bit.ly/2LHVmDZ

P4 is of high interest to industry, e.g. avionics

- rapid prototyping
- program verification
- . . .
- e.g. used for 20+ years with same hardware

Motivation

Move to the Data Plane

From SDN with OpenFlow to (fully) programmable data planes (e.g. P4, POF, eBPF)

Lots of new P4 applications that run in the data plane

- inband network telemetry
- in-network computation
- protocol acceleration (e.g. congestion control)
- middleboxes (DDoS mitigation)
- ...

Lots of new target platforms

- CPU
- Network Processing Unit (NPU)
- FPGA
- ASIC

Lots of key performance indicators

- throughput & packet rate
- latency & jitter
- resources
- price
- ...

→ **Need to understand properties of devices and P4 programs**
→ **Focus on certain aspects for modeling**

Outline

P4 Programmable Network Devices

What is P4?

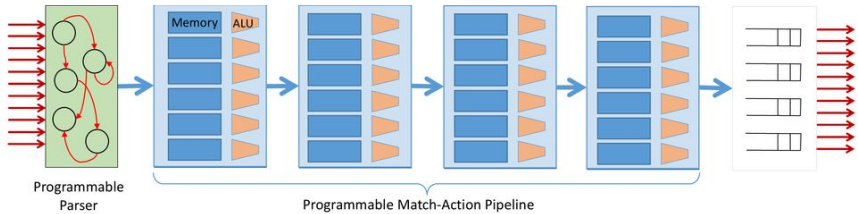


Image from https://bit.ly/3mDpaE9

Programmable data planes

- custom network device behavior
- blocks: parser, match-action, deparser
- (ideally) target independent

Centerpiece: Match-Action tables

- matches key to action
- key: packet or meta data
- exact, ternary, LPM match

P4 Programmable Network Devices

Comparison of Available Targets

	CPU	NPU	FPGA	ASIC
Throughput	+	++	+++	++++
Latency	$>$ 10 µs	5 µs to 10 µs	$<$ 2 µs	$<$ 2 µs
Jitter	−−−−	−−−	−−	−
Resources	++++	+++	++	+
Flexibility	++++	+++	++	+
Example	t4p4s DPDK	NFP-4000 SmartNIC	NetFPGA SUME	Intel Tofino

Table 1: Categorizations are estimates for available products based on own measurements and related work

In this work we focus on the extremes: CPU and ASIC

Dang et al. [1] divide P4 program into components

- parser
- processing
- packet modification
- actions
- ...

Idea: evaluate components (e.g. match-action tables) in isolation [1]

[1] Dang et al. "Whippersnapper: A p4 language benchmark suite." Proceedings of the Symposium on SDN Research. 2017.

Model P4 components individually

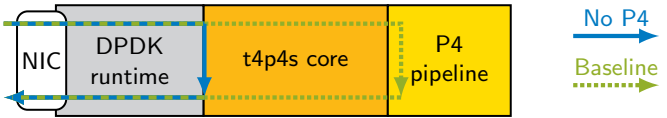
→ Combine component models to model complete system

Match-Action table properties

- match type (exact, ternary, LPM)
- entry size (key, action, action data)
- number of entries
- number of (independent) tables

CPU Performance Model

t4p4s – a DPDK-Based Software P4 Target



t4p4s

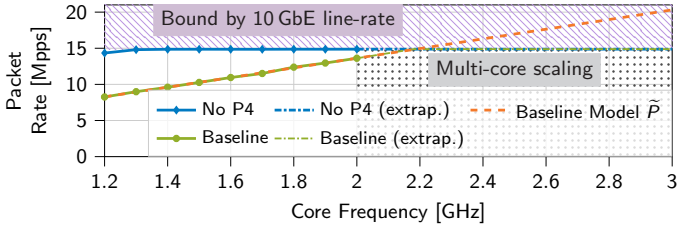
- P4 compiler
- generates hardware-independent C code
- hardware-dependent library for e.g. DPDK

Device-under-Test hardware

- Intel Xeon CPU E5-2640 v2 (2.0 GHz)
- Intel X540-AT2 NIC (dual port, 10 Gbit/s)
- turboboost and hyperthreading disabled (jitter)

CPU Performance Model

Baseline – Maximum Packet Rate with 64 B Packets



- 6 Mpps reduction for baseline P4 program
- bottleneck: CPU

Derive model for packet rate $\widetilde{P}$

- using linear regression for baseline

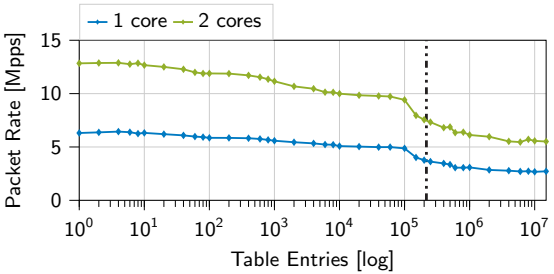
Derive model for CPU cycle usage $\widetilde{C}$

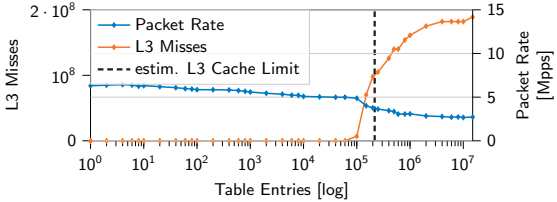
$$\widetilde{C} = \frac{\text{CPU frequency}}{\widetilde{P}}$$

$$\widetilde{C}_{\text{base}} = 146$$

CPU Performance Model

Number of Table Entries – Exact Match





Observations

- double cores results in double performance
- 2 different “phases”
- bottleneck: L3 cache

Model resources $\widetilde{R}_{\text{exact}}$ based on L3 cache size

$$\widetilde{R}_{\text{exact}}(n,k,a) = 2 \cdot 64\,\text{B} + \underbrace{(k \cdot n)}_{\text{Hash table Entries}} + \underbrace{(8\,\text{B} \cdot n)}_{} + \underbrace{(a \cdot n)}_{\text{Actions}}$$

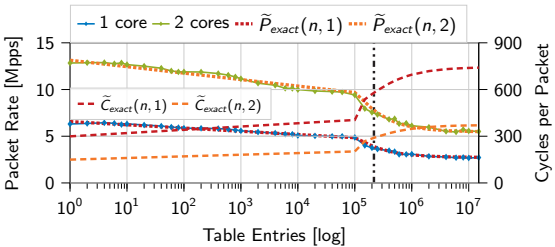
$$= 128\,\text{B} + n \cdot \underbrace{(k + a + 8\,\text{B})}_{\text{Table entry size}}$$

n	number of entries	a	action size (64 B)
k	key size (4x4 B)	R_{L3}	20 MB L3 cache

Set $\widetilde{R}_{\text{exact}} = R_{\text{L3}}$, solve for n

CPU Performance Model

Number of Table Entries – Exact Match Model



Derive model for packet rate $\widetilde{P}_{\text{exact}}$

- linear regression for 1 core
- scale for multiple cores

Derive model for CPU cycles $\widetilde{C}_{\text{exact}}$

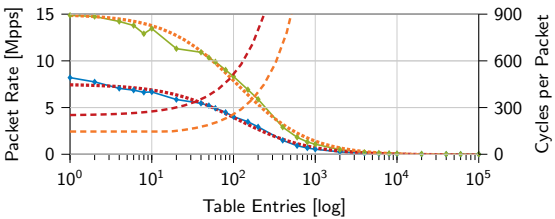
$$\widetilde{C}_{\text{e,exact}}(n,c) = \frac{1}{c} \cdot \begin{cases} p \cdot \ln(q \cdot n) + r, & \widetilde{R}(n) < R_{L3} \\ \frac{s}{t \cdot n + u} + v, & \text{otherwise} \end{cases}$$

with parameters $\{p, q, r, s, t, u, v\}$

CPU Performance Model

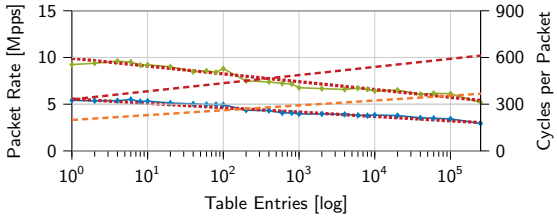
Number of Table Entries – Ternary & LPM Match Model

Ternary Match



- CPU cycles: exponential increase
- ternary match difficult to implement in software
- currently: loop over all elements
- in hardware: ternary content-addressable memory (TCAM)

LPM Match



- CPU cycles: logarithmic increase (log scale!)
- DIR-24-8 data-structure for IPv4
- theoretic search complexity: $\mathcal{O}(1)$
- bottleneck: shared L3 cache size
- part of data structure already requires 64 MB

ASIC Resource Model

Intel Tofino ASIC

P4 programmable switch ASIC

- 64 100 Gbit/s ports

→ guarantees switching 6,4 Tbit/s for any program

- latency well below 1 µs
- stable latency: no jitter or long-tail

Focus on resource consumption

- SRAM & TCAM resources limited
- need to fit program on chip
- model to indicate if program will fit

ASIC Resource Model

Table Resources

Resources $\bar{R}$ for individual table (e.g. exact match)

$$\bar{R}(n, k, a) = n \cdot (\bar{R}_{\text{width}}(k) + a)$$

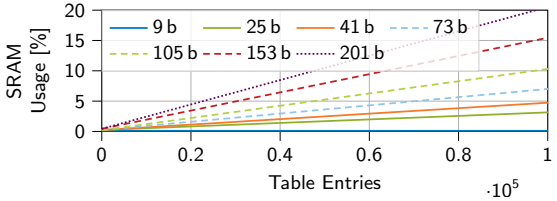
- n number of entries
- k key size
- a action data

Resources $\bar{R}_{\text{width}}$ for key width

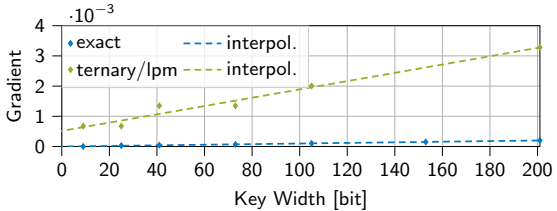
$$\bar{R}_{\text{width}}(k) = p \cdot k + q$$

with parameters p, q

SRAM usage for different exact match widths



Determine p, q: interpolate gradients



Conclusion

Increase of P4 programmable data planes

- more applications
- more platforms
- more metrics

→ need for models

→ focus on certain aspects

→ Model isolated P4 components

→ **Model for P4 centerpiece: match-action tables**

CPU – performance model

- high-performance DPDK-based switch
- linear scaling with CPU cores
- typical DPDK latency histogram
- platform-dependent influences

ASIC – resource model

- line-rate guaranteed
- no long-tail latency
- number of table entries limit program complexity
- simplified model

Future work: compare with other modeling approaches, e.g. network calculus