Mapping P4 to SmartNICs

Edwin Peer - 16 May 2017
Agenda

- SmartNIC Hardware
- Silicon Architecture
- Programming Model
- Mapping P4
- Results & Experience
- Further Reading / Q&A
Agilio™ CX SmartNIC Family

- Optimized for standard server based cloud data centers
- Low profile half length PCIe form factor, power < 25W
- Based on Netronome’s Network Flow Processor 4xxx silicon (72 cores x 8 threads each)
- 2GB DRAM for lookup tables / state tables (millions of entries)
Agilio™ LX SmartNIC Family

- Optimized for higher throughput requirements - middlebox, gateway, appliance, service node…
- Full height half length PCIe form factor
- Based on Netronome’s Network Flow Processor 6xxx silicon (120 cores x 8 threads)
- Memory: 8GB of DDR3 DRAM @ 1866Mhz w/ECC
- Dual PCIe Gen3x8

2x40GbE (QSFP)  
1x100GbE (CXP)
P4 Programmable SmartNIC in Context

- Transparent acceleration of OVS / Contrail / eBPF
- SmartNIC with dynamic firmware
- Custom datapath in P4 and/or C

OpenStack • ONOS • ODL
Linux • BSD
vRouter • OVS • eBPF

Run-Time
Compiler • Debugger
app.P4 • app.C
Editor
Inside the Netronome Flow Processor (NFP6xxx)
Hierarchical Memory Architecture

- FPC Cluster
  - 64KB CLS
  - 256KB CTM
  - FPC

- Local Scratch Data and Smaller Tables (e.g., QoS Map)

- Packet Body, L2/L3 Lookup Tables, Port and Virtual Port Statistics/Meters

- EMU
  - 1MB Global SRAM
  - 2MB DDR3 Cache SRAM

- IMU
  - 4MB GLOBAL SRAM

- Packet Header Data (First 240 Bytes)

- Packet Body Overflow (only if needed), Flow Lookup and State Tables, Forwarding Table Results

- 409Gbps RAW DRAM Bandwidth
Network Flow Processor 4xxx / 6xxx

- Highly parallel multithreaded architecture (8 threads / core) for high throughput
- Purpose built Microengines / Flow Processing Cores (72 / 120) maximize flexibility
- H/W accelerators further maximize efficiency (throughput/watt)

Fully software defined feature set - examples:

- Flexible tunneling support (e.g. VXLAN, GRE, VLAN, MPLS, NSH)
- Flexible Match/Action processing - many packet fields / protocols
- Highly scalable and fine grained security policies
- Network and PCIe SR-IOV / VirtIO RX/TX with stateless offloads
- Packet generation / reception with advanced statistics e.g. jitter
- Traffic directing (tapping / mirroring / steering / load balancing)

External DRAM accommodates millions of flows / rules

Convenient programmability using P4 and/or C
Function accelerators: atomic, lock-less for efficient parallel usage
P4 Datapath

- Load balancer distributes each packet to next available thread for optimum throughput
- Hardware assisted reordering ensures packet order is maintained
- Matching performed using various algorithms, e.g. DRAM-backed “algorithmic TCAM”
- Actions efficiently performed in on-chip memory
- Flow tracker statefully learns / tracks sessions, speeds up classification

Flow Tracker
- Learn microflows
- Cache action

Parse

Load Balance

= Ring / Work Queue (multi producer / consumer)

Action

C Code Plugin

Acceleration
- e.g. stats, lookup

Pool of worker threads on microengines
P4 Datapath

1. Configuration via control protocol or CLI
2. Agent populates tables in SmartNIC datapath

Best of all worlds
- Performance of SR-IOV
- Flexibility of virtio (VM migration)
- Performance and CPU core saving of switching on SmartNIC

© 2017 NETRONOME SYSTEMS, INC
Example P4 Application

Concepts:

- P4 and C running on SmartNIC implements datapath - e.g. defines protocols, match / action behavior
- Datapath steers traffic to VNFs running on x86 server and on SmartNIC
The Classroom

PDF

Github

Webinars

Videos

Search
Performance Results

Mean P4 Throughput (as function of analysis performed)

- 64 Byte
- 128 Byte
- 256 Byte
- 512 Byte
- 1514 Byte

Throughput (Mpps)

Test Instances

TCP 1 - METER
UDP 1 - METER
TCP 1
UDP 1
TCP 2
UDP 2
TCP 3:1
UDP 3:1
TCP 3:2
UDP 3:2
TCP 4:1
UDP 4:1
TCP 4:2
UDP 4:2
P4 / XDP / eBPF

-- image by VMware / Mihai Budiu, reproduced under Apache License, https://github.com/vmware/p4c-xdp
Further Reading / Q&A

- open-nfp.org

- Netronome white papers
  - www.netronome.com/media/redactor_files/WP_Programming_with_P4_and_C.pdf

- github.com/vmware/p4c-xdp

Questions?
Thank You