P4/PIF + C Programmable Intelligent NICs: Requirements and Implementation Notes

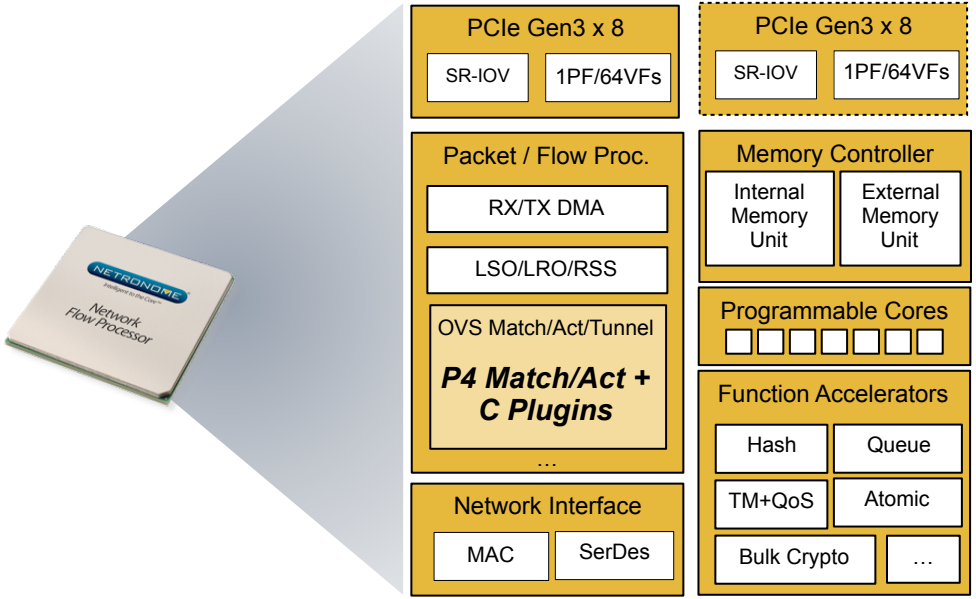
Johann Tönsing

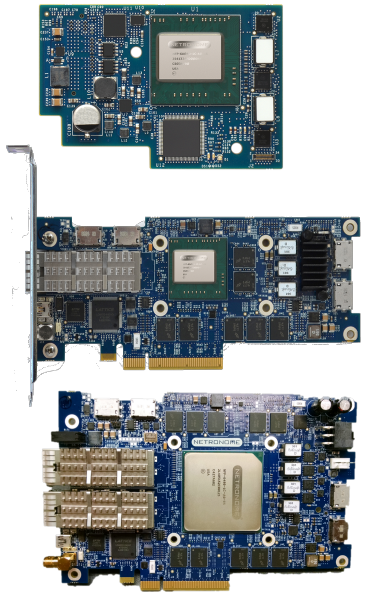
P4 Workshop - November 2015

Background: Intelligent NICs for Servers

- Intelligent NICs (iNICs): n x 10/25/40/100G…
- Software: Linux and Windows Offload Drivers - offload match/action processing (OVS, GFT, P4…)
- Hardware:

- Improve throughput by 4-10x
- Improve efficiency by 10-40x or save ~50% of x86 sockets





NETRONOME

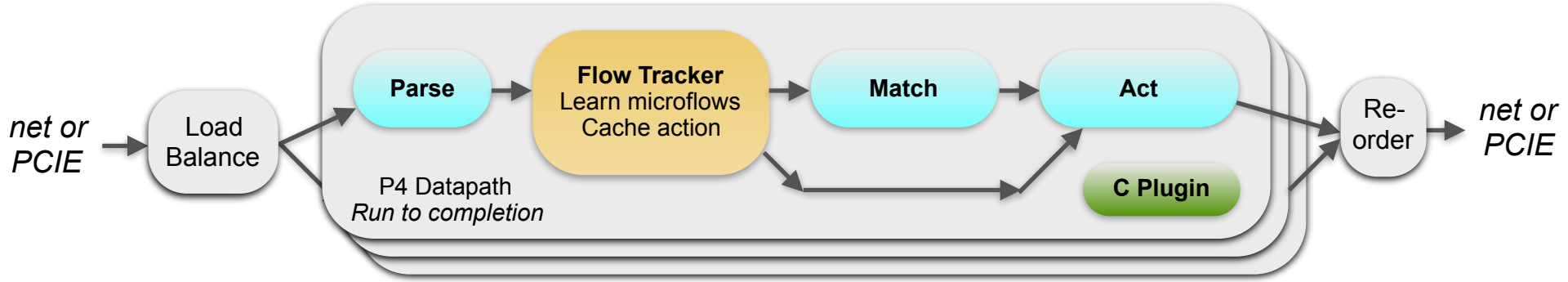
Requirements for P4 iNICs

- Protocol independence use cases
  - Not too important for outer headers (want legacy outer to build overlays)
  - Important for inner headers: custom tunnel termination, service chaining…
  - Load balance based on inner headers (without terminating these protocols)
  - Add options to packets for e.g. telemetry
  - => Processing depth can vary
- Flexibility to change behavior easily + rapidly = *the* “killer app”
  - => Execution time per packet + resources consumed per packet can vary
  - => Integrate fast, medium, slow path software into the datapath
    (with appropriate models, interfaces to preserve interoperability)

Requirements for P4 iNICs

- Track millions of flows, rapidly update flow entries
  - Apply fine grained policies (microsegmentation, QoS, steering / load bal.)
  - Retain statistics and other state per flow
  - => Need DRAM to store enough table entries, state
- Performance required: n x 10G, 25G, 40G, 100G
  - Examples: 40G @ 128B = 33 Mpps, 100G @ 400B = 30 Mpps
  - => Need substantial processing resources (e.g. cores / memory channels)
  - *iNIC offload of OVS achieved 30 Mpps (low profile), 80 Mpps (full height)*
- Support hierarchy of datapaths
  - Example: embedded switch (in iNIC) managed by cloud operator, separate virtual switch (in server) and VMs managed by tenant
  - => Permit stacking / nesting OVS and OVS, P4 and OVS, P4 and P4 etc.

P4 for iNIC Datapath Software Architecture



Pool of worker threads on flow processing cores

- Distributes each packet to next available thread for optimum throughput
- Hardware assisted reordering ensures packet order is maintained
- Matching performed using DRAM-backed “algorithmic TCAM”

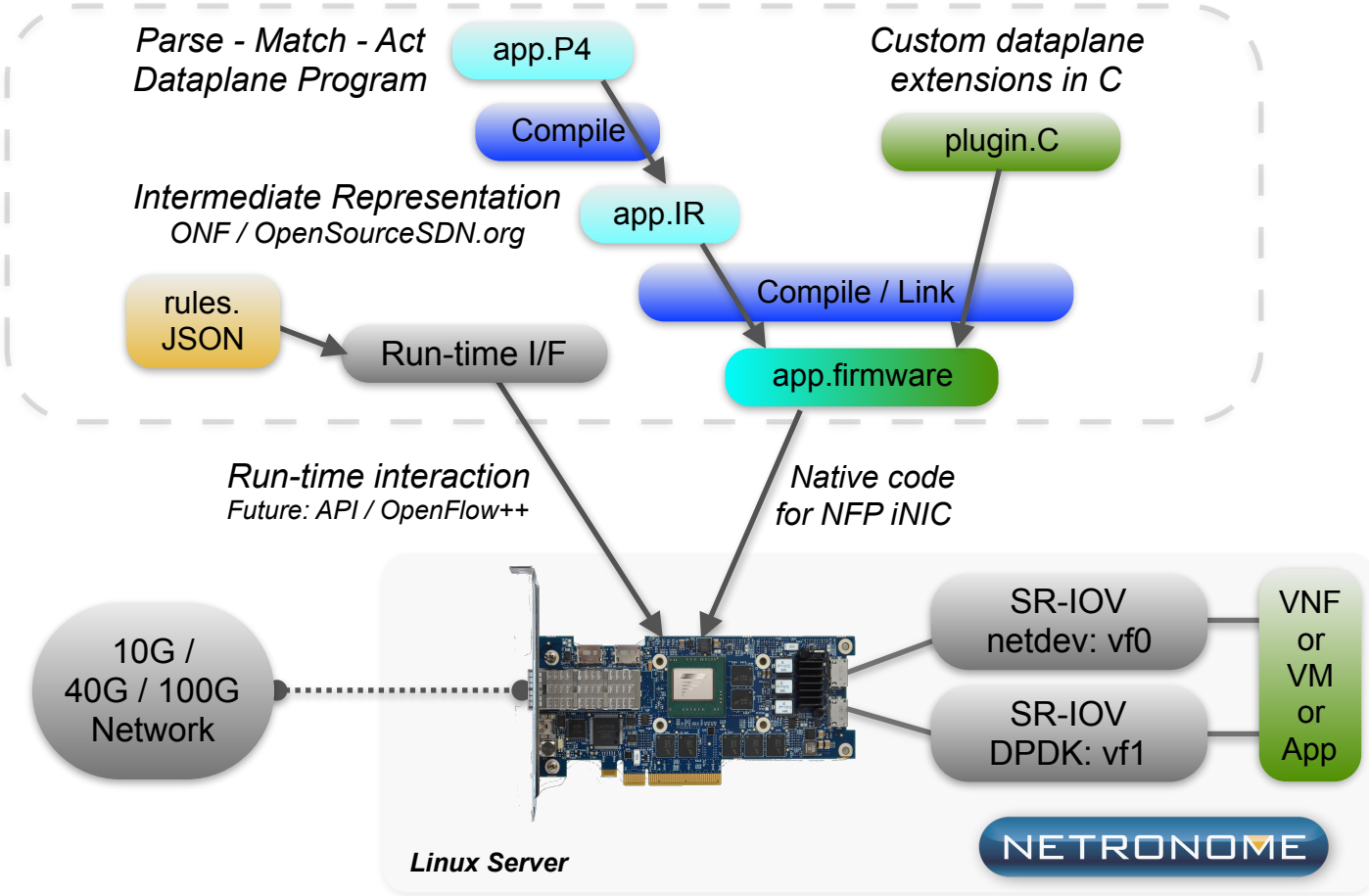
=> Conveniently supports varying runtime per packet + high throughput / flow capacity

NETRONOME

P4 Development Environment for iNIC

P4 *and* C
Integrated Development Environment
Edit - Build - Run
Debug with Code / Data Visualization

=



Conclusions

- Implemented P4 1.0 compiler for Netronome's iNICs
  - High performance (>30Mpps): compiled to native code, with flow tracking cache
  - Run to completion pool of threads model permits arbitrary match / action processing per packet (varying execution time)
- Development environment with P4 editor, debugger and inspectors / visualizers (code, packet, table, counter, etc.) enhances productivity
- Datapath extensions (initially written in C) can run on host or in iNIC
  - Facilitates custom processing while maintaining high performance
- Expect easy extension to P4 1.1 and beyond
  - Syntax changes irrelevant due to use of IR
  - Sequential execution of actions is more natural for programmers
  - More precise data typing facilitates interfacing to datapath extensions
  - Architecture independent of language facilitates embedding P4 in P4, P4 in OVS etc.
  - Looking forward to evolution (incremental parsing, deparsing, composability, RT APIs…)