Empowering container-based NFVi with VPP on Arm architecture

Trevor Tao Trevor.Tao@arm.com
Song Zhu Song.Zhu@arm.com

25/06/2018
Agenda

• Background
• FD.io/VPP Enablement on Arm Platform
• FD.io/VPP integrated with Container Networking Solution
  • Vhost-user CNI
  • Contiv/VPP netplugin
• Future Plan and Use Cases
Background
Background

- Trends: Container-based platforms for OPNFV
- Containerized OpenStack or Kubernetes as VIM
- OPNFV Euphrates release delivered Kubernetes integration
- OPNFV projects: Container4NFV, Auto, Clover...
- Containerized VNFs with Data Plane Acceleration (SRIOV)
- Acceleration for inter-container communication with VPP
Container-based NFV Architecture

Container-based NFVi on Arm servers

- Kubernetes as VIM
- Flannel/SRIOV/vhost user CNI plugins integrated
- SRIOV CNI: enable VF passthrough
- Vhost-user CNI: enable VPP-based container networking

Ref: Container4NFV Architecture
FD.io/VPP Enablement on Arm Platform
FD.io/VPP (Vector Packet Processing)

- User Space software platform providing switch/router functionalities
- Aiming to run on commodity CPUs
- Cisco developed it from 2002 and open sourced it in FD.io (Linux Foundation) on Feb 2016
- Leverage DPDK, XDP, netmap... as fast I/O
- Batch packet processing - more efficient iCache utilization
- Packet processing graph: modular, flexible, and extensible
- **Fast, scalable** and **deterministic**
  - 14+ Mpps per core, tested to 1TB
  - Scalable FIB: supporting millions of entries
  - 0 packet drops, ~15µs latency

Network I/O

Management Agent

Packet Processing: VPP

Netconf/Yang

REST...
A Simple Use Case for Performance Tuning

- **L2xc** – forward all packets received on Port0 to Port1 and vice versa
- **IPv4** – route packets across IPv4 subnets

Arm serves with Cortex A72 Processors
Performance Benchmarking and Tuning

64B packet – single flow – single core

Observations

• Most hotspots are memory accesses
• Software-defined data placement consumes processing cycles
• Unintentionally ordering memory accesses can slow the system down
• Compiler may fuse loops which alters memory access pattern from original program order

Further Directions

• RFC2544 testing
• Multicore scaling
• PMU data
• Cache stashing
• Compiler and C library versions
• Other platforms
# The path to on Arm

<table>
<thead>
<tr>
<th>Workload Scale</th>
<th>Performance Analysis</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSIT</td>
<td>Hotspot &amp; Bottleneck Identification</td>
<td>Upstream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Libraries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Toolchain</td>
</tr>
<tr>
<td>FD.io Lab</td>
<td>Tuning &amp; Optimization</td>
<td>Hardware</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Processors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accelerators</td>
</tr>
</tbody>
</table>

- **Software**
  - Upstream
  - Libraries
  - OS
  - Toolchain

- **Hardware**
  - Processors
  - I/O
  - Accelerators

**Logos**
- Cavium
- Huawei
- Marvell

© 2018 Arm Limited
FD.io/VPP as Container Networking Solution
Why Use VPP for Container Networking

• Container networking requirements for NFV
  • High performance on packet processing
  • High scalability
  • High flexibility

• What VPP provides
  • High performance
  • Abundant interfaces: ssvm,virtio/vhost,af(packet,tap,memif…
  • Abundant features for control and management
VPP for Container Networking with AF_Packet interface

Container A
- send()
- FIFO
- TCP
- IP (routing)
- device

VPP
- Overlays (VXLAN)
- ACL/Policy
- Layer 3 (IPv4,6)
- Layer 2 (ether)
- dpdk
- af_packet

Container B
- recv()
- FIFO
- TCP
- IP (routing)
- device

Kernel Space

User Space

Pros:
- Support Linux kernel stack which is required by most applications with performance higher than Flannel

Cons:
- Performance is lower than vhost-user/memif interface

Kubernetes AF_Packet CNI?
VPP for Container Networking with Virtio-Vhost Interface

Host 1

CONTAINER

DPDK

DPDK APP

ETHDEV

testio-user

vhost-user adapter

virtio

vhost

Data Path 2

Data Path 1

Host 2

CONTAINER

DPDK

DPDK APP

ETHDEV

testio-user

vhost-user adapter

virtio

vhost

VPP-DPDK

VxLAN Overlay
Vhost-user CNI for Kubernetes

K8S POD

- Vhost-user server socket(interface) is created in VPP
- After adding the vhost user CNI path, the virtio-user interface is used as a virtual device of DPDK
Contiv/VPP Integration on Arm Platform

(Ref: VPP, VNF Agent and Contiv integration)
What We Have Done

- Enabled VPP release on Arm64 servers
  - VPP 17.10 running on Arm servers
  - VPP 18.04 released with AArch64 packaging for Ubuntu
- Integrated VPP with Kubernetes for inter-container communication with virtio/vhost-user interfaces on Arm servers
- Enhanced vhost-user CNI for Kubernetes with VPP
- Enabling project Ligato and Contiv/VPP on Arm platforms
- Enabling VPP-based use cases for OPNFV Container4NFV project
Use Cases and Future Plan
Nginx as CDN Use Case

ONAP/Kubernetes

CONTAINER
VNF
Nginx
FlowCache
TCP stack

CONTAINER
Content Delivery Service

CONTAINER
Client

Data Flow In

VxLAN Overlay

Flannel/Calico

VPP-DPDK

VPP-DPDK
Next Steps

• Continue performance tuning on Arm servers
• Performance benchmarking with NFVbench/VSperf on Arm servers
• VPP integration (CI/CD enablement) in OPNFV Gambia release (Nov 2018)
• Enable and integrate other VPP based CNI solutions (memif, …)
• Enable more VPP-based use cases (microservices and SFC) for NFVi
• Integrate VPP-based NFV solutions with orchestration software (ONAP)