Improve Performance of Kube-proxy and GTP-U using VPP

Hongjun Ni (hongjun.nl@intel.com)
Danny Zhou (danny.zhou@intel.com)
Johnson Li (johnson.li@intel.com)

Network Platform Group, DCG, Intel

Acknowledgement: Jianfeng Tan
Agenda

- Enabling high performance kube-proxy
- Six ways to improve the performance of GTP-U
- Key takeaway
Introducing Vector Packet Processor

- Packet processing development platform
- Runs on commodity CPUs and leverages DPDK
- Runs as a Linux user-space application
VPP Graph Node and Plugin Framework

dpdk-input

ethernet-input

mpls-ethernet-input

ip4-input

ip6-input

arp-input

ip4-lookup

ip4-local

ip4-udp-lookup

ip4-load-balance

Kube-proxy

GTP-U

ethernet-output

Packet Vector
Enabling high performance kube-proxy

- Introducing kube-proxy
- Current pain point
- Proposed kube-proxy solution
- Kube-proxy dataplane
- GoVPP
- Performance
Introducing kube-proxy

- Watches addition and removal of Service and Endpoints.
- Installs iptables rules
- Captures traffic and select Pod
- Redirects traffic

Reference: https://kubernetes.io/docs/concepts/services-networking/service/
Current Pain Point

- Supports userspace and iptables
- Uses kernel iptables NAT
- Performance degrades when service/endpoint pairs increase
Proposed kube-proxy solution

API Server

watch watch watch

CP: Control Plane

DP: Data Plane
Kube-proxy Data Plane

- Distribute traffic evenly
- Per flow stick to Pod
- Three service types:
  1. ClusterIP: Port
  2. NodeIP: NodePort
  3. External LoadBalancer
Kube-proxy Data Plane Internals

- Supported Graph Nodes:
  - kp4-nat4
  - kp4-nat6
  - kp6-nat4
  - kp4-nodeport
  - kp6-nodeport

- VIP->Pod Table:
  - Stores VIP->Pod session

- Other Graph nodes:
  - ip4-lookup
  - ip6-lookup
  - udp-lookup
Kube-proxy: External LoadBalancer

VPP Router

VPP LB

K8s-node-1
Pod 1
eth1
Kube-proxy DP
Pod N

K8s-node-2
Pod 1
eth1
Kube-proxy DP
Pod N

K8s-node-3
Pod 1
eth1
Kube-proxy DP
Pod N

VPP Kube-proxy
GoVPP

• Golang toolset for VPP management
• VPP binary API (JSON) → Go Structure
• Handle 250,000 binary API requests per second

Performance

Kube-proxy Throughput For 64-Byte Packet

- Throughput (Mpps)
  - Phy NIC: 9.83
  - vhost-user: 6.62

- Test Case: Load balance + DNAT + Routing
  - Input packet size (bytes): 64
  - Output packet size (bytes): 64

- Scaling
- Linux iptables perf: < 400 kpps

Six Ways to Improve Performance of GTP-U

- Cache table lookup result
- Bypass second ip-input
- Bypass first route lookup
- Dual-loop and Quad-loop
- Packet prefetching
- Bypass second route lookup
GTP-U Plugin Internals

- Typical GTP-U packet processed by VPP and GTP-U plugin

<table>
<thead>
<tr>
<th>Outer MAC header</th>
<th>Outer IP header</th>
<th>UDP header</th>
<th>GTP-U header</th>
<th>Inner IP header</th>
<th>L4 header</th>
<th>Payload</th>
</tr>
</thead>
</table>

- GTP-U Plugin

Supported Graph Nodes:
- gtpu-input
- gtpu-encap
- gtpu-bypass

GTP-U tunnel

Stores PDP context

Other Graph nodes:
- ip-lookup
- udp-lookup

IP4 -input

IP6 -input

Other Graph nodes
Device Under Test for Performance

- **Network Topology**
  
- **Hardware Configuration**
  
- **BIOS Configuration**

- **Software Configuration**

---

**CPU**
- Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz

**DIMM**
- 2133 MHz, 64GB Total

**NIC**
- 2x 82599ES 10-Gigabit SFI/SFP+

**PacketGen**
- Ixia* 10 Gigabit Ethernet Traffic Generator

---

**OS**
- Ubuntu 16.04.2 LTS

**Kernel**
- Linux version 4.4.0-62-generic

**DPDK**
- 17.08

**VPP**
- 17.10-rc0

**GTP-U**
- 17.10-rc0

---

- **Enhanced Intel Speedstep**
  - Enabled

- **Turbo Boost**
  - Enabled

- **Processor C3**
  - Disabled

- **Processor C6**
  - Disabled

- **Hyper-Threading**
  - Disabled

- **Intel VT-d**
  - Enabled

- **CPU Power and Performance Policy**
  - Performance

- **Memory Freq.**
  - 2133 MHz

- **Total Memory Size**
  - 64 GB

- **Memory RAS and Performance Configuration -> NUMA Optimized**
  - ENABLED

- **QPI B/W**
  - 9.6 GT/s

- **MLC Streamer**
  - ENABLED

- **MLC Spatial Prefetcher**
  - ENABLED

- **DCU Data Prefetcher**
  - ENABLED

- **DCU Instruction Prefetcher**
  - ENABLED

- **Direct Cache Access (DCA)**
  - ENABLED

---

Legend
- Flow A
- Flow B
Cache table lookup result

1. Gtpu-input
2. Read IP and TEID
3. key == stored key?
   - N: Tunnel Lookup
   - Y: Retrieve stored tunnel index
4. Store key and tunnel index
5. Read tunnel entry
Bypass second ip-input

Pros:
- Boost performance

Cons:
- Security issue
Bypass first route lookup

- gtpu packet: accelerate decap processing
- none-gtpu packet: an overhead with about 13 clocks per packet
GTPU-Decap Performance and Analysis

### Test Case

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Input packet size (bytes)</th>
<th>Output packet size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport IP Routing + GTPU-Decap + Inner IP Routing</td>
<td>98</td>
<td>64</td>
</tr>
</tbody>
</table>
Dual-loop and Quad-loop, Packet Prefetching

- Reduce read latency
- Process packets in parallel
Bypass second route lookup

- dpdk-input
  - ethernet-input
    - ip4-input
    - ip6-input
    - arp-input
  - ip4-lookup
    - ip4-rewrite
      - ip4-load-balance
        - ip4-rewrite
          - ethernet-output
        - ip4-lookup
          - gtpu-encap
          - Packet Vector
GTPU-Encap Performance and Analysis

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Input packet size (bytes)</th>
<th>Output packet size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inner IP Routing + GTPU-Encap + Transport Routing</td>
<td>64</td>
<td>114</td>
</tr>
</tbody>
</table>
Key Takeaway

- Easy-to-use and flexible VPP plugin framework
- Kube-proxy plugin to boost DP’s performance in Cloud environment
- Combination of a set of ways to optimize data plane performance
- Better platform for developing open source dataplane ingredients