A high-speed PMD towards LXC networking

Company: UnitedStack
Title: network virtualization engineer
Name: Zheng jie
Who we are

• OpenStack (Gold Member) IaaS provider
• NFV enhanced Neutron Networking
• DPDK powered applications include:
  • Distributed LoadBalancer
  • Server based Data Center fabric infrastructure
  • ... ...
Accelerate LXC networking

- SR-IOV for LXC
- Universal virtio PMD for LXC
- Specialized transport medium as IPC is in need
How to structure the SHM

• VECRING ---- Vectorized Ring Buffer
• Fundamental ring element (block)---- aligned cache line
• Yet single producer & single consumer queuing model
• Masked ring indicator (as with DPDK ring implementation), never wrap back

* where len is power of 2
Associate VECRING with mbuf

- Control block (as metadata) precede, Data blocks follow
- Control information associated with a mbuf takes 16-bytes
  - starting-index, length, whether-is-fetched, whether-is-end-of-block, etc.
- Control information (at maximum 4) can aggregate into one control block.
  - Enqueue x4
  - Enqueue x2
  - Enqueue x1
Enqueue with x4 speed

- Aggregate 4x control information into one control block

Diagram showing a block ring buffer with control information aggregated.
Enqueue with x2 speed

- Aggregate 2x control information into one control block
Bulking Dequeuing

• Fetch a control block
• Walk through control information one by one until reaching end of block
  • Call rte_pktmbuf_alloc()
  • Copy packet payload from data blocks
  • Mark it as fetched
• If nothing wrong happens, proceed rear indicator to next control block.
• Else mark the control block as partially fetched, can cease dequeuing.
How to better access memory

• Non-temporal behavior
  • Will not pollute cache layout

• CPUID supported
  • up to SSE4.2 or AVX2
    • streaming SIMD LOAD/STORE instructions

• streaming loading buffer

• Write combining
DPDK PMD encapsulation

- vdev prefix ---- eth_vecring
- Parameter list:

<table>
<thead>
<tr>
<th>name</th>
<th>Mandatory</th>
<th>type</th>
<th>remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>yes</td>
<td>string</td>
<td>Indicate which container it belongs to</td>
</tr>
<tr>
<td>link</td>
<td>yes</td>
<td>string</td>
<td>link identifier, there maybe multiple links inside a container, links are to distinguish themselves</td>
</tr>
<tr>
<td>socket</td>
<td>no</td>
<td>number</td>
<td>Which numa socket the link belongs to</td>
</tr>
<tr>
<td>mac</td>
<td>no</td>
<td>mac</td>
<td>If not provided, randomize it.</td>
</tr>
<tr>
<td>master</td>
<td>no</td>
<td>[true,false]</td>
<td>PMD role, default is false.</td>
</tr>
<tr>
<td>queue</td>
<td>no</td>
<td>int</td>
<td>The length of queue, default is DEFAULT_NR_BLOCK64</td>
</tr>
</tbody>
</table>

- --vdev=eth_vecring0, domain=[string], link=[string], master=[int], mac=[mac], socket=[int]
Environmental pre-setup

# create a domain with name:demo
[root@localhost dpdk-16.07-vecring]#./vecutils.sh dom_alloc demo

# list all available domains
[root@localhost dpdk-16.07-vecring]#./vecutils.sh dom_ls
0: domain: demo huge-dir: mounted
1: domain: testcontainer huge-dir: mounted
2: domain: vnfl1 huge-dir: mounted

# map the domain directories into container
# by including mapping entries in LXC container’s definition file
Environmental setup

#host side as master
[root@localhost ~]#... --vdev=eth_vecring0,domain=testcontainer,\
link=tap456,master=true,mac=00:ec:f4:bb:d9:7f,socket=1

#container side as slave
[root@localhost ~]#... --vdev=eth_vecring0,domain=testcontainer,link=tap456

#the generated metadata and hugepage files
[root@localhost testcontainer]# tree
.
├── huge
│   ├── vecring-tap456.inbound-0
│   ├── vecring-tap456.inbound-1
│   └── vecring-tap456.outbound-0
│         └── vecring-tap456.outbound-1
└── tap456.metadata
Single Link rx/tx rate

Single Link Rx/Tx rate in Mpps

Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz with 20M L3 cache
Max Quad Links rx/tx rate

Max Quad Links Rx/Tx rate in Mpps

- Bidirectional 2x: 46Gbps
- Bidirectional 3x: 52Gbps
- Bidirectional 4x: 60Gbps
- Singly Directional 4x: 60Gbps
- Bidirectional 4x: 60Gbps

64-bytes
1514-bytes
Summary

• Scales with number of links, but not linearly, and constrained by memory bandwidth.

• Two times of memory copy involved, DPDK multi-processes model eliminates it (at the expense of resource segregation).

• Tested with LXC, it should also work with other containers.

• Other virtual device PMD is supposed to meet the same challenges.
Thanks!!

zhengjie@unitedstack.com
https://github.com/chillancezen/dpdk-16.07-vecring