Ensuring Both High Performance & Security for Containers

Networking, Storage, Containers, & Security

Kevin Deierling, Mellanox Technologies
VMs and Containers are like houses and pools
Anything you can do in the house, you can do in the pool ....
It’s just a lot more dangerous!
Networking
TCP/IP & UDP – It’s gotta be good!
Performing @ 100Gb/s is hard!
Needs NIC hardware offloads
- Checksum offloads
- Large Send Offload (LSO)
- Transmit Segmentation Offload (TSO)
- Large Receive Offload (LRO)
- Flow affinity ….

DPDK: Data Plane Development Kit
- Open source software library
  - More efficient low level packet processing
- DPDK user space Poll Mode Driver (PMD)
  - Eliminates interrupts and context switches
  - Better performance
  - Higher CPU Utilization (basically an on-load technology)
- DPDK is not a networking stack
  - Does not provide L3 forwarding, IPsec, firewalling, etc
- [www.dpdk.org](http://www.dpdk.org) (Open source community within Linux Foundation)
DPDK Delivers Line Rate Packet Performance

- Highest packet rate 66% lower latency
- Higher Scalability with 4K VLAN and 4K Multicast Address per VF
- Better security through HW memory protection

<table>
<thead>
<tr>
<th>Product</th>
<th>ConnectX-4 100G</th>
<th>ConnectX-5 100G</th>
<th>ConnectX-4 Lx 25G</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPDK 64B Packet Throughput (Mpps)</td>
<td>93.5</td>
<td>132.7</td>
<td>34.0</td>
</tr>
</tbody>
</table>

100GbE Single Port DPDK Forwarding Rate

- 100GbE Single Port DPDK Forwarding Rate
- Theoretical
- ConnectX-4
- ConnectX-5
- Frame Size [Byte] 64, 128, 256, 512, 1024, 1280, 1518
- Throughput [Mpps] 0 to 160
RDMA: More Efficient Networking

- Unlike DPDK … RDMA is an offload technology
- Means CPU isn’t busy moving data … so can do more work running applications
Users want to create logical subnets that span across L3 networks
- Particularly important in multi-tenant public cloud environments
- Solution is Overlay networks (VXLAN encapsulations in UDP packets)
- Uses centralized controller (SDN) for provisioning
Flash Storage
2X better performance with RoCE
- 2X higher bandwidth & significantly better CPU efficiency

RoCE* achieves full Flash storage bandwidth
- Remote storage without compromises

* ROCE: RDMA Over Converged Ethernet
Network Offload Frees Up CPU for Application Processing

- Half of CPU capacity consumed moving data
  - Even though achieving only half throughput
- Cores unavailable for application processing

- All CPU available to run applications
  - Better efficiency = more users
- Smart network delivers better TCO

Users limited by CPU resources

Available for Applications Processing

CPU Penalty without offload

Unreliable Low level I/O

More CPU means More users & apps

All cores Available for Applications Processing

Reliable Network Transport

Regular Network (without Offload)

Smart RoCE Network (with Offload)
No CPU intervention for Data Path

NVMe Target CPU Offload

Embedded NVME Data Encap Decap

NVME Data Path

Ctrl Path

Ctrl Path

PCI EXPRESS

nvm EXPRESS

© 2017 Mellanox Technologies
SPDK: DPDK + RDMA Optimized for NVMe

- **Storage Performance Development Kit (SPDK)**
  - Optimized user space library using RDMA for accelerating NVMe-over-Fabrics
- **Better performance efficiency than regular kernel stack**
  - But still an onload technology that consumes CPU
RDMA for NVMe-oF

1) NVMe command encapsulated into RDMA by host RNIC, then crosses the network
2) Target moves command to NVMe SSD
3) Target SSD response encapsulated into RDMA by Target RNIC and crosses the network
4) Host de-encapsulates response back to NVMe
- **NVMe** fastest growing storage media
- **NVMe over Fabrics** (NVMe-OF) delivers penalty free sharing of Flash
  - Single NVMe Flash devices can saturate 25G/s Link
  - 1.3M+ IOPs Random Read Performance, sub 10us added latency
  - Effectively Zero CPU overhead
- **3X Performance** for 1/3 the price of Fibre Channel
NVMe-over-Fabrics: Flash Arrays that Scale

- NVMe-oF Flash Reference Design
  - 24GB/S Performance
  - 5M IOPs
  - < 150us average latency
- Excelero storage software
- Mellanox 100GE RoCE NICs (2)
- Scalable to 8 nodes or more

- Micron SolidScale Flash Array – 3 x 2U
  - 100GB Performance
  - 11M IOPS
  - 1% added latency
- Excelero storage software
- Mellanox 100GE RoCE NICs (2)
- Mellanox 100GE Switches (2) & Cables
Containers
Intro to Containers

- **Containers**: lightweight, stand-alone executable unit of software
  - Typically containers are stand-alone encapsulation of a specific application

- **Includes everything needed to run it**
  - Self contained, no external dependencies, static binaries, no dynamic linking
  - Code, runtime, system tools & libraries, settings

- **Host OS Independent**
  - Containerized software can include distro dependent libraries

- **Provide some isolation**
  - Useful for dev/ops when environments aren’t the same
  - Users, filesystem, network
More Sharing == More Scalable

- Containers have a lot less overhead than Virtual Machines & are thus much more scalable
Hypervisor vs Container Virtualization

- **Pros**
  - Hypervisor provides good isolation and security

- **Cons**
  - Performance, virtualization overhead
    - Kernel sharing by isolated user space apps is helping
  - Scalability

- **Pros**
  - Performance
  - Scalability

- **Cons**
  - Isolation and security
  - Host kernel dependency
Nested Containers

- Security is a key challenge for Containers
- One approach is to enhance security is to run containers in VMs
  - Typically one VM per tenant, which contains multiple Containers
- Leverage existing Virtualization / Cloud infrastructure
- Challenge: Two layers virtual switching -> significant performance impact
Much of the security of containers is tied to orchestration & provisioning
Control how containerized apps are deployed and provisioned in a devops environment
Production deployments for ops can be highly constrained (read-only, prohibit create/destroy)
Two competing container networking control models:
- Docker: Container Network Model (CNM)
- Kubernetes: Container Networking Interface (CNI)
  - Defined within Linux Foundation Cloud Native Computing Foundation

Kubernetes Pods are a small group of tightly coupled containers
- On the same physical node, share storage volume(s), Shared namespace (private IP subnet)
Three Ways Containers can Talk to Each Other

- **Namespaces are central to container isolation**
  - A physical port can only belong to one network namespace (comm stack, route table, sockets, IPTABLE rule)

- **Three mechanisms to create a virtual connection between container namespaces**
  - MACVLAN/IPVLAN: Creates multipole MAC & IP addresses using VLAN sub-interfaces
  - Virtual Bridge: Linux Bridge (NAT) or OpenVswitch (OVS)
  - Hardware Virtualization and Switching: SR-IOV and embedded hardware OVS switch

*From Iguaz.io: https://thenewstack.io/hackers-guide-kubernetes-networking/
SR-IOV
- PCIe device presents multiple Virtual Function (VF) instances to the OS/Hypervisor

Enables Application Direct Access
- Bare metal performance for Container
- Reduces CPU overhead

Enable RDMA to the Container
- Low latency applications benefit from the Virtual infrastructure
SR-IOV designed for Container’s but can work for containers too!
  • Provides good isolation
  • But limits scalability to the number of VF’s supported by a given NIC (~100-1000)

Better solution is to virtualize Ethernet devices in the container
  • Isolation can be provided by existing memory protection mechanisms of RDMA and scales to millions!!!
Hardware Acceleration of OVS with Embedded HW Switch

ASAP\textsuperscript{2} Virtual Switch Acceleration

- Accelerates OVS switch in hardware (eSwitch)
  - Maintains software compatibility with SDN controllers on control path
- Higher performance data flow switching with greatly improved CPU efficiency
- Frees CPU cores to run NFV applications
  - Total Infrastructure Efficiency – Smart networking gets more from compute & storage investment

ASAP\textsuperscript{2}
(Accelerated Switch and Packet Processing)
NFV Agility & Scalability without Performance Penalties!

© 2017 Mellanox Technologies
Accelerated Switch And Packet Processing (ASAP²)

- ASAP² accelerates or offloads network processing
  - Three deployment scenarios:
    - ASAP² Direct
      - Full vSwitch Offload
    - ASAP² Flex
      - vSwitch Acceleration
    - ASAP² Flex & Direct
      - VNF/Container Acceleration

Combined solution of ASAP² Flex for VNF and ASAP² Direct for OVS
ASAP2: Full OVS Offload

- Enable SR-IOV data path with OVS control plane
  - Standard Open vSwitch management interface
  - Offload OVS data-plane to embedded Switch (eSwitch)
ASAP² Direct Outperforms DPDK Accelerated OVS

- **Single flow, no VXLAN @25G**
  - 33MPPS, 330% higher message rate compared to OVS over DPDK
  - Zero! CPU utilization on hypervisor compared to 4 cores with OVS over DPDK

- **Scalability to 2K-60K flows or more**
  - OVS offload reach ~25MPPS for 2K flows, and ~17MPPS for 60K flows @ 25GE
  - Hardware VXLAN encap/decap
  - Still zero CPU compared to 4 cores with DPDK

<table>
<thead>
<tr>
<th>Flows, Cores, 4X10G, HT on</th>
<th>ONP Benchmark *</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 flows, 4 cores (no VXLAN)</td>
<td>10.1 Mpps</td>
</tr>
<tr>
<td>4K flows, 2 cores (no VXLAN)</td>
<td>7.9 Mpps</td>
</tr>
<tr>
<td>1 flow, 1X10G port, 2 cores (with VXLAN)</td>
<td>3.7 Mpps</td>
</tr>
</tbody>
</table>

* Source: [Intel Open Networking Platform Performance Report](#)
ASAP² Flex

- Offload SOME elements of the data-path to the NIC, but not the entire data-path
  - Data still flow via the software vSwitch
  - Para-Virtualized Container (no SR-IOV)

- Offloads
  - Classification offload
    - Application provide flow spec and flow ID
    - ConnectX runs the classification in HW and attach a flow ID
    - vSwitch performs action based on flow ID rather than full flow spec
  - Encap/decap
  - QoS

- Two APIs for the application to configure the offloads
  - Linux Traffic Control (TC): http://lartc.org/manpages/tc.txt
  - DPDK enhancements
Programmable SOC - Faster, Smarter More Secure Networking

- An Intelligent, Programmable System on a Chip (SOC)
  - Combines CPU, network adapter, and security/networking accelerators on one chip
  - Supports container isolation
    - No hypervisor required
    - Can be secure in a multi-tenant environment

- Offload host CPU
  - More cycles available for host CPU to run applications
  - More power-efficient to run networking & security on SOC

- Accelerate performance
  - Offers all of the benefits of adapter (high-speed ports, accelerations…)
  - Superior packet throughput compared with software
  - Hardware-accelerated crypto (both AES/SHA and Public Key)
  - Traffic can be filtered in the adapter, unclogging the PCIe link

- Secure execution environment
  - Perform security functions in isolated, embedded environment
  - Keys, credentials, policies all maintained in a separate subsystem from host
## Security and Layered Trust Model for Intelligent Networking

### Secure Boot
- OS & Application code are digitally signed (ECC) and authenticated upon boot
- OEM customer may control signing process & keys/certificates

### Data path switching controlled hardware
- All traffic from network and PCIe passes through a hardware eSwitch
- eSwitch flow-based ACL and routing tables are managed by CPU
- Open vSwitch (OVS) run on CPU provides L4 stateful firewall features

### Intelligent controller memory is private and isolated from host
- No direct access to CPU memory from PCIe-attached device*
- RDMA methods typically used to exchange info between host and SOC NIC
  - Enforcing registered/pinned memory on CPU side

### Explicitly-defined management APIs and schema
- Security manager typically independent from server – remote platform
- Comms and access secured with TLS or SSH

* Optionally, and under explicit control by CPU initialization SW, a shared memory window may be opened.
SOC Adapter Functional Diagram

- Intelligent NIC Configuration
  - SOC ConnectX sees multi-host environment
  - Integrated PCIe switch allows flexible data routing

- Software model is greatly simplified
  - Standard drivers
  - CPU runs standard Linux
  - Host <-> CPU communication via RDMA or network protocols
Host Smart NIC Can Maintain Isolation

- Data flows just as with a regular NIC
- Flow tables are maintained by CPU subsystem
Flow Routed Through SOC to Host

- Flow tables provisioned by embedded processor
- Datapath indirection enables flexibility & security
- **Inbound** network traffic directed to CPU host by eSwitch
- **Outbound** traffic from host intercepted at eSwitch and directed to CPU
Putting It All Together

Containers are just like the pool … you can do anything you want …

But Be CAREFUL!

(And Use a Smart Networking Solution!)
Thank You