Driving Innovation in Openstack for Network Functions Virtualization (NFV)

Stephen Gordon,
Sr. Technical Product Manager, Openstack
Red Hat

Sandro Mazziotta,
Director NFV Initiative, Openstack
Red Hat
Agenda

➢ Introduction to NFV
➢ Strategy/Roadmap
➢ Focus on EPA
➢ Q&A
Introduction to NFV
NFV, latest telco disruption

Classical Network Appliance Approach

- Message Router
- CDN
- Session Border Controller
- WAN Acceleration
- DPI
- Firewall
- Carrier Grade NAT
- Tester/QoE monitor
- SGSN/GGSN
- PE Router
- BRAS
- Radio Access Network Nodes

- Fragmented non-commodity hardware.
- Physical install per appliance per site.
- Hardware development large barrier to entry for new vendors, constraining innovation & competition.

Independent Software Vendors

- Orchestrated, automatic & remote install.
- Standard High Volume Servers
- Standard High Volume Storage
- Standard High Volume Ethernet Switches

Network Virtualisation
ETSI NFV Reference Model
NFV Complexity and Use Cases

**NFV Complexity:**
- Atomic Network Function - Firewall, DNS, DPI...
- Complex Network Functions: EPC, IMS...
- Composite Network Function - Networking Forwarding Graph

**NFV Use Cases:**
- Residential/Home/Enterprise: Virtual CPE, Home GW, STB
- Core Networks: vEPC, vIMS, ...
- Access Network: C-RAN
Strategy/Roadmap
3 Pillars based NFV Strategy

NFV

COMMUNITY  ECOSYSTEM  PRODUCT
NFV relies on Multiple Communities

OpenSource Software

Telecommunications
OPNFV bridge Telco and OpenSource
Openstack as NFV Foundation


#redhat #rhsummit
Red Hat NFV Ecosystem
## NFV adoption models

### Best of the breed
- **VNF vendor A**
- **VNF vendor B**
- **VNF vendor C**
- **NFVO vendor D**
- **NFVI**
- **Compute vendor F**
- **Network vendor G**
- **Storage vendor H**
- **VNFM vendor B**
- **VIM**

### Network Equipment Provider
- **VNF vendor A**
- **VNF vendor A**
- **VNF vendor C**
- **NFVO vendor A**
- **NFVI**
- **Compute vendor B**
- **Network vendor A**
- **Storage vendor D**
- **VNFM vendor C**
- **VIM**

---

*Best of the breed* are selected components from different vendors that work well together.

*Network Equipment Provider* are complete solutions provided by a single vendor.
Red Hat NFV Solution Priorities

Be the #1 open source:

❖ NFVI Solution (1)
❖ VIM Solution (2)
❖ VNF enablement Solution (3)

Partner with ISV (4)

Partner with HardWare (5)
Red Hat NFV Solution

- Red Hat Enterprise Linux Openstack
  - Compute
  - Network
  - Storage
  - Management
  - Virtualization stack (QEMU + libvirt)
  - OVS + DPDK
  - RHEL w/KVM
  - OPEN DAYLIGHT
  - CEPH

- Red Hat’s Competitive Advantage
  - Product Approach instead of Customized solution
  - Introduce existing portfolio NFV features instead of creating a dedicated solution

- Red Hat’s Solution Benefits
  - Ease of Deployment. Ease of Operate
  - Linux + Virtualization + Openstack packaging
  - Carrier Grade - Ability to commit to SLA
# NFV Roadmap

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Today</th>
<th>H2CY15</th>
<th>CY16-CY20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic VNF</td>
<td>RHEL-OSP 6 (Juno) &amp; RHEL 7.1</td>
<td>RHEL-OSP 7 (Kilo) &amp; RHEL 7.2</td>
<td>Service Chaining</td>
</tr>
<tr>
<td>vCPE, vEPC</td>
<td>Enhancement Platform Awareness (EPA) Ready: NUMA Awareness Huge Part Support Neutron IPV6</td>
<td>RT-KVM DPDK Ease of Deployment Ease of Operation Hybrid Management</td>
<td>Containers ARM Support</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>Ecosystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL-OSP 6 (Juno) &amp; RHEL 7.1</td>
<td>NFV Certification</td>
</tr>
<tr>
<td>RHEL-OSP 7 (Kilo) &amp; RHEL 7.2</td>
<td>VNF SDK</td>
</tr>
<tr>
<td>RT-KVM DPDK Ease of Deployment Ease of Operation Hybrid Management</td>
<td></td>
</tr>
</tbody>
</table>
Focus on Enhanced Platform Awareness (EPA)
Performance Features

- CPU Pinning
- Huge Pages
- NUMA-aware scheduling
  - Memory binding
  - I/O device locality awareness
CPU Pinning

- Extends **NUMATopologyFilter** added in Juno:
  - Adds concept of a “dedicated resource” guest.
  - Implicitly pins vCPUs and emulator threads to pCPU cores for increased performance, trading off the ability to overcommit.
- Combine with existing techniques for isolating cores for maximum benefit.
Example - Hardware Layout

```
# numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3
node 0 size: 8191 MB
node 0 free: 6435 MB
node 1 cpus: 4 5 6 7
node 1 size: 8192 MB
node 1 free: 6634 MB
node distances:
node   0   1
  0:   10  20
  1:   20  10
```
## Example - Hardware Layout

<table>
<thead>
<tr>
<th>Node 0</th>
<th>Node 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core 0</td>
<td>Core 4</td>
</tr>
<tr>
<td>Core 1</td>
<td>Core 5</td>
</tr>
<tr>
<td>Core 2</td>
<td>Core 6</td>
</tr>
<tr>
<td>Core 3</td>
<td>Core 7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node 0 RAM # 0</th>
<th>Node 1 RAM # 0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Node 0 RAM # 1</th>
<th>Node 1 RAM # 1</th>
</tr>
</thead>
</table>
Example - Virsh Capabilities

```xml
<cells num='2'>
  <cell id='0'>
    <memory unit='KiB'>8387744</memory>
    <pages unit='KiB' size='4'>2096936</pages>
    <pages unit='KiB' size='2048'>0</pages>
    <distances>
      <sibling id='0' value='10'/>
      <sibling id='1' value='20'/>
    </distances>
  </cell>
  ...<
cpus num='4'>
  <cpu id='0' socket_id='0' core_id='0' siblings='0'/>
  <cpu id='1' socket_id='0' core_id='1' siblings='1'/>
  ...<
```
Example - Configuration

- **Scheduler:**
  - Enable `NUMATopologyFilter`, and `AggregateInstanceExtraSpecsFilter`

- **Compute Node(s):**
  - Alter kernel boot params to add `isolcpus=2,3,6,7`
  - Set `vcpu_pin_set=2,3,6,7` in `/etc/nova.conf`
Example - Hardware Layout

Node 0
- Core 0
- Core 1
- Core 2
- Core 3

Node 0 RAM # 0

Node 1
- Core 4
- Core 5
- Core 6
- Core 7

Node 1 RAM # 0

Host Processes

Guests

Node 0 RAM # 1

Node 1 RAM # 1
Example - Configuration

● Flavor:
  ○ Add $\texttt{hw:cpu\_policy=dedicated}$ extra specification:
    
    \[
    \texttt{
    \$ nova flavor-key m1.small.performance set hw:
    \phantom{\texttt{\color{red}cpu\_policy=dedicated}}}
    \]

● Instance:

    \[
    \texttt{
    \$ nova boot --image rhel-guest-image-7.1-20150224 \ \\
    \phantom{\texttt{\color{red}--\}}\texttt{\color{red}flavor\texttt{\color{red}m1.small.performance test-instance}}}
    \]
Example - Resultant Libvirt XML

- vCPU placement is **static** and 1:1 vCPU:pCPU relationship:
  
  ```xml
  <vcpu placement='static'>2</vcpu>
  <cputune>
    <vcpupin vcpu='0' cpuset='2'/>
    <vcpupin vcpu='1' cpuset='3'/>
    <emulatorpin cpuset='2-3'/>
  </cputune>
  ```

- Memory is strictly aligned to the NUMA node:
  
  ```xml
  <numatune>
    <memory mode='strict' nodeset='0'/>
    <memnode cellid='0' mode='strict' nodeset='0'/>
  </numatune>
  ```
Huge Pages

- Huge pages allow the use of larger page sizes (2M, 1 GB) increasing CPU TLB cache efficiency.
  - Backing guest memory with huge pages allows predictable memory access, at the expense of the ability to over-commit.
  - Different workloads extract different performance characteristics from different page sizes - bigger is not always better!
- Administrator reserves large pages during compute node setup and creates flavors to match:
  - `hw:mem_page_size=large|small|any|2048|1048576`
- User requests using flavor or image properties.
Example - Host Configuration

```
# grubby --update-kernel=ALL --args="hugepagesz=2M hugepages=2048"
# grub2-install /dev/sda
# shutdown -r now
# cat /sys/devices/system/node/ node0/hugepages/hugepages-2048kB/nr_hugepages
  1024
# cat /sys/devices/system/node/ node1/hugepages/hugepages-2048kB/nr_hugepages
  1024
```
Example - Virsh Capabilities

<topology>
  <cells num='2'>
    <cell id='0'>
      <memory unit='KiB'>4193780</memory>
      <pages unit='KiB' size='4'>524157</pages>
      <pages unit='KiB' size='2048'>1024</pages>
    </cell>
  </cells>
</topology>
Example - Flavor Configuration

$ nova flavor-key m1.small.performance set hw:mem_page_size=2048
$ nova boot --flavor=m1.small.performance \
   --image=rhel-guest-image-7.1-20150224 \
   numa-lp-test
Example - Result

$ virsh dumpxml instance-00000001
...
<memoryBacking>
  <hugepages>
    <page size='2048' unit='KiB' nodeset='0'/>
  </hugepages>
</memoryBacking>
...

#redhat #rhsummit
# Example - Hardware Layout w/ PCIe

<table>
<thead>
<tr>
<th>Node 0 RAM # 0</th>
<th>Node 1 RAM # 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core 0</td>
<td>Core 1</td>
</tr>
<tr>
<td>Core 2</td>
<td>Core 3</td>
</tr>
<tr>
<td>Node 0 RAM # 1</td>
<td>Node 1 RAM # 1</td>
</tr>
<tr>
<td>Node 0 PCIe</td>
<td>Node 1 PCIe</td>
</tr>
<tr>
<td>Core 4</td>
<td>Core 5</td>
</tr>
<tr>
<td>Core 6</td>
<td>Core 7</td>
</tr>
</tbody>
</table>
I/O-based NUMA Scheduling

- Extends `PciDevice` model to include NUMA node the device is associated with.
- Extends `NUMATopologyFilter` to make use of this information when scheduling.
What next?

- **vhost-user** virtual interface driver
- **virtio** performance enhancements
- Ability to use **real-time** KVM
- Configurable **thread policy**:
  - **avoid** - do not place on host that has hyperthreads
  - **separate** - if on host that has hyperthreads, avoid using threads from different cores
  - **isolate** - like separate, but do not allow another guest to use threads on the same CPU core
  - **prefer** - if on host that has hyperthreads, prefer using threads from the same cores (current behaviour)
LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.