AI/VR Applications at Edge

Sukhdev Kapur, Juniper Networks
Vikram Siwach, MobiledgeX
April, 2019
# Blueprint Proposal: AI/VR Applications at Edge

<table>
<thead>
<tr>
<th>Case Attributes</th>
<th>Description</th>
<th>Informational</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>New Blueprint for enabling AI/ML and low latency AR/VR capabilities at the Edge</td>
<td></td>
</tr>
<tr>
<td><strong>Blueprint Family</strong></td>
<td>Proposed Name: Integrated Edge Cloud</td>
<td></td>
</tr>
<tr>
<td><strong>Use Case</strong></td>
<td>Programmability on switches and I/O Accelerations on programmable NICs &amp; embedded FPGAs to deliver AI/ML workload placement and low latency demands of AR/VR applications onboarding edge stack</td>
<td></td>
</tr>
<tr>
<td><strong>Blueprint proposed Name</strong></td>
<td>AI/ML and AR/VR applications at Edge</td>
<td></td>
</tr>
<tr>
<td><strong>Initial POD Cost (capex)</strong></td>
<td>Leverage white boxes, standard NICs: The cost of POD will depend upon the hardware profiles and peripherals desired</td>
<td></td>
</tr>
<tr>
<td><strong>Scale &amp; Type</strong></td>
<td>Detailed in Resource Requirements slide</td>
<td></td>
</tr>
<tr>
<td><strong>Applications</strong></td>
<td>AI/ML streaming workloads and AR/VR applications</td>
<td></td>
</tr>
<tr>
<td><strong>Power Restrictions</strong></td>
<td>Less than 10Kw</td>
<td></td>
</tr>
<tr>
<td><strong>Infrastructure orchestration</strong></td>
<td>Openstack Queens or above Docker 1.13.1 or above Container Orchestration –K8s 1.10.2 or above OS - Ubuntu 16.x, CentOS</td>
<td></td>
</tr>
<tr>
<td><strong>SDN</strong></td>
<td>Tungsten Fabric, kernel/DPDK/SmartNIC offload vRouter</td>
<td></td>
</tr>
<tr>
<td><strong>Workload Type</strong></td>
<td>Containers over VM or baremetal</td>
<td></td>
</tr>
<tr>
<td><strong>Additional Details</strong></td>
<td>- Runs on Commodity HW &lt;br&gt;- x86, ARM, SoC, ... &lt;br&gt;- Multiple options for partial or full NIC offloads (Intel, Netronome, Mellanox) &lt;br&gt;- Future Supports: &lt;br&gt;- eBPF/XDP offload</td>
<td>See next slide for additional details</td>
</tr>
</tbody>
</table>
Motivations

• Programmable SW Defined Fabric to control the life of packets going to application containers.
• Low cost per byte for compute cycles, I/O and storage.
• Enable high performance on commodity hardware
• Slice Traffic to offer certain SLAs
• Embedded Security
Blueprint Value

- Programmability: Programmable model for fabric | host | VM | container
  - Network slicing programming TOR for SLA contracts per tenant
- Multiple Data Plane Options: SRIOV, Hardware offload – vrouter
TF Architecture

- OpenStack Nova
- Control Node
- Control Node
- XMPP
- Compute node
- OpenStack Nova Agent
- vRouter Agent (user space)
- Hypervisor (KVM+Qemu)
- vRouter Forwarding Plane
- TX
- RX
- VM

KVM+Qemu
Integrated Cloudlet’s Software Stack

**MOBILEDEX SERVICES LAYER**
- Register
- Verified Location
- Find Cloudlet

**MOBILEDEX PLATFORM LAYER**
- DME
- CRM embedded TF controller
- Kubernetes API Server
- App1 Container(s)
- App(n) Container(s)
- App(1) K8s Cluster
- App(n) K8s Cluster

**MOBILEDEX PLATFORM LAYER**
- MEX Platform Kubernetes Cluster
- VM
- VM
- VM
- VM
- VM
- VM
- Agent

**VIRTUALIZATION LAYER (OPTIONAL)**
- Hypervisor
- vRouter Forwarding Plane

**INFRASTRUCTURE LAYER**
- Software Infrastructure: Enterprise Grade OS + Drivers
  - Compute + Accelerators
  - Networking
  - Local Storage
- Standard Server Hardware Infrastructure: CPU, GPU, RAM, Flash, HDD, NIC
Use case Programmability: Tenant Network Slicing – Hardware and Software
Use case: Host Acceleration using vRouter Offload

Reference:
Future support: ebpf based XDP offload
Lifecycle Management for MEX applications

• Configuration and deployment: Cloudlet resource manager is single point of configuration and is capable to end to end Cloudlet provisioning from apps to connectivity to external world. It is the orchestrator for Kubernetes cluster and containers based on developers needs. Tungsten Fabric controller is embedded within CRM.

• Monitoring: MEX dashboard supports native Kubernetes monitoring solution Prometheus which collects key stats and dashboard visualizes the metric thus collected.

• Scaling: CRM launches cluster and containers within based on declarative configuration by developer but elasticity to scale up and down based on stress is built in its logic.
# Resource Requirements for MEX cluster

## Minimal Cloudlet Resource Requirements Scenario

<table>
<thead>
<tr>
<th></th>
<th>VMs</th>
<th>vCPU</th>
<th>RAM GB</th>
<th>Disk GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEX Platform</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>160</td>
</tr>
<tr>
<td>MEX Agent</td>
<td>4</td>
<td>16</td>
<td>32</td>
<td>320</td>
</tr>
<tr>
<td>Cluster Instance Node</td>
<td>3</td>
<td>12</td>
<td>24</td>
<td>240</td>
</tr>
<tr>
<td><strong>vRouter</strong></td>
<td>2</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>9</td>
<td>38</td>
<td>80</td>
<td>720</td>
</tr>
</tbody>
</table>

## 8-cluster Cloudlet Resource Requirements Scenario

<table>
<thead>
<tr>
<th></th>
<th>VMs</th>
<th>vCPU</th>
<th>RAM GB</th>
<th>Disk GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEX Platform</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>160</td>
</tr>
<tr>
<td>MEX Agent</td>
<td>4</td>
<td>16</td>
<td>32</td>
<td>320</td>
</tr>
<tr>
<td>Cluster Instance Node</td>
<td>24</td>
<td>96</td>
<td>192</td>
<td>1920</td>
</tr>
<tr>
<td><strong>vRouter</strong></td>
<td>2</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>30</td>
<td>122</td>
<td>248</td>
<td>2400</td>
</tr>
</tbody>
</table>
Provisioning Requirements for MEX

- Access: VPN and Openstack
- Resource Allocation for MEX Applications
  - Compute & 3-6 VMs per cloudlet required for MEX
  - MEX Agent VM (Minimum 1, preferred 4)
    - Ingress Load Balancing
    - K8S orchestration
  - MEX Platform VM (2 VMs)
    - Distributed Matching Engine (DME)
      - API traffic from SDK
      - Communication to other services
    - Cloudlet Resource Manager (CRM)
      - OpenStack management
      - Receives commands from Controller node to orchestrate cluster and app instances
- One Kubernetes Cluster per Cluster Instance for Applications
  - 3 VMs (vCPU: 4, RAM: 8GB, Disk: 80GB)
  - Applicable to:
    - MEX Agent (1-4 VMs per cloudlet)
    - MEX Platform (1-2 VMs per cloudlet)
    - K8S Node (3 VMs per cluster instance)
- Server Hardware
  - Minimum Requirement: Dual Socket E5 Xeon Processor, 256GB Memory, 1TB of SSD
  - Prefer NVDIA Tesla P100 GPUs
- Network
  - External Network
    - Created in advance
    - Needs access to OpenStack API endpoint
    - 1 IP for each MEX Agent – L7 LB
    - 1 IP for MEX Platform VM (DME, CRM)
    - 1 IP for each application requiring L4 access
    - DHCP assigned public IP range on ext network is preferred
    - For short term, a pool of 8 public IPs is adequate
- MEX K8s Network
  - Created by MobiledgeX via APIs
  - Shared by all cluster instances
  - Subnets created dynamically by MEX
- Storage
  - Local storage, Block Storage, Object Storage, Shared File System
- Openstack Cloudlet API Requirements:
  - server list, create, delete, set properties
  - image list, save, create, delete
  - network list, create, delete
  - subnet list, create, delete
  - router create, delete, add and delete ports
  - flavor list, show create
  - security group rule list and create show limits