AHV + NVIDIA VGPU INTEGRATION
Malcolm Crossley - AHV GPU Architect
Nutanix AHV
The hypervisor built for the Enterprise Cloud
Nutanix: Built-in Virtualization

Integrated, scale-out compute and storage with built-in virtualization and management
Turnkey infrastructure platform that converges compute, storage and virtualization to run any application, at any scale

Comprehensive management solution that radically simplifies datacenter operations

Nutanix Platform

Nutanix Prism

Nutanix Acropolis
Nutanix Platform

Prism

- Infrastructure / VM Management
- Operational Insights
- Capacity Planning

Acropolis

- App Mobility Fabric
- Distributed Storage Fabric
- AHV (built-in Virtualization)
Starting with Citrix XD/XA 7.9 ... AHV is integrated with MCS

Citrix App and Desktop Virtualization

Nutanix AHV is

End user desktops

Citrix Infrastructure

XenApp & XenDesktop
NetScaler VPX
ShareFile
And more ...

Nutanix AHV
Complete XenDesktop Solution

AHV Hypervisor

Multiple servers /block

Nutanix CVM

Hypervisor

AHV

2 x Intel 10 Core CPUs
128-256GB RAM
2 x 10GbE NICs
HDD + SSD Drives

Nutlinx CVM

Hypervisor

AHV

2 x Intel 10 Core CPUs
128-256GB RAM
2 x 10GbE NICs
HDD + SSD Drives

Nutanix CVM

Hypervisor

AHV

2 x Intel 10 Core CPUs
128-256GB RAM
2 x 10GbE NICs
HDD + SSD Drives

Scale out by adding Nodes

Nutanix Distributed File System (NFDS)
All the Features for Enterprise Applications

- Automated Resource Scheduling & VM Placement
- Performance Hot-Spot Avoidance (aka DRS)
- VM High Availability
- Live Migration
- VM-centric storage w/Snapshots and Clones
- Dynamic IP Address Management
- GPU Resource Management

- Host Profiles
- Virtual Networking
- 1 click Upgrades
- Host Maintenance Mode
- Simplified Scaling (Automatic CPU Compatibility)
- Image Service
- Cross-Hypervisor Migration, DR & Backup
Acropolis GPU resource management concepts

Not enough GPU resource

XenApp VM A
- GPU config:
- Vendor: Nvidia
- Type: PT-Graphics
- Device: M60

Windows VM B
- GPU config:
- Vendor: Nvidia
- Type: PT-Compute
- Device: M60

Linux VM D
- GPU config: 2 x
- Vendor: Nvidia
- Type: PT-Graphics
- Device: M60

Linux VM C
- GPU config: 2 x
- Vendor: Nvidia
- Type: PT-Graphics
- Device: M10

NVIDIA Tesla M60
- Physical GPU: 86:00.0
- Vendor: Nvidia
- Type: PT-Compute
- Device: M60

NVIDIA Tesla M10
- Physical GPU: 06:00.0
- Vendor: Nvidia
- Type: PT-Graphics
- Device: M10

- Physical GPU: 07:00.0
- Vendor: Nvidia
- Type: PT-Graphics
- Device: M10

- Physical GPU: 08:00.0
- Vendor: Nvidia
- Type: PT-Graphics
- Device: M10

- Physical GPU: 09:00.0
- Vendor: Nvidia
- Type: PT-Graphics
- Device ID: M10
Acropolis GPU resource management concepts

NVIDIA Tesla M60

Physical GPU
Vendor: Nvidia
Type: PT-Compute
Device: M60

Virtual GPU
Vendor: Nvidia
Type: Virtual
Device: M60-2Q
Index: 0
Index: 1
Index: 2
Index: 3

Physical GPU
Vendor: Nvidia
Type: PT-Graphics
Device: M60

Virtual GPU
Vendor: Nvidia
Type: Virtual
Device: M60-1Q
Index: 0
Index: 1

Vendor: Nvidia
Type: PT-Graphics
Device: M60

Xendesktop VM A
GPU config:
Vendor: Nvidia
Type: Virtual
Device: M60-2Q

Xendesktop VM B
GPU config:
Vendor: Nvidia
Type: Virtual
Device: M60-2Q

Xendesktop VM C
GPU config:
Vendor: Nvidia
Type: Virtual
Device ID: M60-1Q

XenApp VM D
GPU config:
Vendor: Nvidia
Type: PT-Graphics
Device: M60

Not enough GPU resource
AHV VM GPU resource configuration

<table>
<thead>
<tr>
<th>VM NAME</th>
<th>HOST</th>
<th>IP ADDRESS</th>
<th>CONTROLLER IO BANDWIDTH</th>
<th>CONTROLLER AVG IO LATENCY</th>
<th>BACKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>centos-1</td>
<td>16SM6B300385</td>
<td>172.16.1.1</td>
<td>0</td>
<td>16.42 ms</td>
<td>Yes</td>
</tr>
<tr>
<td>centos-10</td>
<td>16SM6B300385</td>
<td>172.16.1.2</td>
<td>2</td>
<td>11.69 ms</td>
<td>Yes</td>
</tr>
<tr>
<td>centos-100</td>
<td>16SM6B300385</td>
<td>172.16.1.3</td>
<td>3</td>
<td>79 ms</td>
<td>Yes</td>
</tr>
<tr>
<td>centos-1001</td>
<td>16SM6B300385</td>
<td>172.16.1.4</td>
<td>2</td>
<td>7.53 ms</td>
<td>Yes</td>
</tr>
<tr>
<td>centos-1002</td>
<td>16SM6B300385</td>
<td>172.16.1.5</td>
<td>16</td>
<td>11.99 ms</td>
<td>Yes</td>
</tr>
<tr>
<td>centos-1003</td>
<td>16SM6B300385</td>
<td>172.16.1.6</td>
<td>4</td>
<td>9.17 ms</td>
<td>Yes</td>
</tr>
<tr>
<td>centos-1005</td>
<td>16SM6B300385</td>
<td>172.16.1.7</td>
<td>25</td>
<td>13.11 ms</td>
<td>Yes</td>
</tr>
<tr>
<td>centos-1006</td>
<td>16SM6B300385</td>
<td>172.16.1.8</td>
<td>7</td>
<td>9.19 ms</td>
<td>Yes</td>
</tr>
<tr>
<td>centos-1009</td>
<td>16SM6B300385</td>
<td>172.16.1.9</td>
<td>8</td>
<td>23.25 ms</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Add GPU**

- *GPU Mode*: vGPU, Passthrough

- *Grid License*: Virtual Workstation

- *VGPU Profile*

<table>
<thead>
<tr>
<th>NAME</th>
<th>VIRTUAL SLICE</th>
<th>FRAMEBUFFER</th>
<th>VRMS ASSIGNED</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>M60:0Q</td>
<td>1/16 GPU</td>
<td>512 MB</td>
<td>64 (31 powered on)</td>
<td></td>
</tr>
<tr>
<td>M60:1Q</td>
<td>1/8 GPU</td>
<td>1 GB</td>
<td>7 (2 powered on)</td>
<td></td>
</tr>
<tr>
<td>M60:2Q</td>
<td>1/4 GPU</td>
<td>2 GB</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>M60:4Q</td>
<td>1/2 GPU</td>
<td>4 GB</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

- *Cancel* | *Add*
Prism UI - Physical GPU overview
Prism UI - Physical GPU metrics

NVIDIA M10

Cluster: cluster-2
Host: host-nmx-A
Mode: vGPU
vGPU Profile: M10-YQ
Allocation: 7 of 8 VMs allocated
Framebuffer (RAM) Per GPU: 8 GB
Cores per GPU: 640
ID: <SPDR>
## Prism UI - Virtual GPU metrics

<table>
<thead>
<tr>
<th>VM NAME</th>
<th>GPU USAGE</th>
<th>FRAMEBUFFER USAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>vdi-windows-kiosk</td>
<td>15.1%</td>
<td>12.5%</td>
</tr>
<tr>
<td>eng-ML-processing-1</td>
<td>11.4%</td>
<td>12.5%</td>
</tr>
<tr>
<td>eng-ML-processing-7</td>
<td>8.1%</td>
<td>12.5%</td>
</tr>
<tr>
<td>eng-ML-processing-12</td>
<td>8.0%</td>
<td>12.5%</td>
</tr>
<tr>
<td>eng-ML-processing-2</td>
<td>7.6%</td>
<td>12.5%</td>
</tr>
<tr>
<td>eng-ML-processing-8</td>
<td>7.3%</td>
<td>12.5%</td>
</tr>
<tr>
<td>eng-ML-processing-8</td>
<td>7.3%</td>
<td>12.5%</td>
</tr>
</tbody>
</table>
## Powerful REST API for all GPU resource information

<table>
<thead>
<tr>
<th>Method</th>
<th>URL Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/hosts/gpu_profiles</td>
<td>Get the list of virtual GPU profiles for all physical Hosts.</td>
</tr>
<tr>
<td>GET</td>
<td>/hosts/host_gpus</td>
<td>Get the list of GPUs for all physical Host.</td>
</tr>
<tr>
<td>GET</td>
<td>/hosts/{uuid}/host_gpus</td>
<td>Get the list of GPUs for a physical Host.</td>
</tr>
<tr>
<td>GET</td>
<td>/hosts/{uuid}/host_gpus/{gpu_uuid}</td>
<td>Get a particular GPU for a physical Host.</td>
</tr>
<tr>
<td>GET</td>
<td>/hosts/{uuid}/host_gpus/{gpu_uuid}/stats/</td>
<td>Get stats for a particular GPU for a physical Host.</td>
</tr>
</tbody>
</table>
Acropolis 1-click operations and GPU resources
Possible future developments

- Prism App for Nvidia Dashboard
- Best effort High Availability for VMs with GPU resources

- Nutanix Guest Tools assist with Nvidia guest driver upgrades, timed to match Nvidia host driver upgrades
Questions?