HARNESSING THE POWER OF PYTHON TO CONTROL NVIDIA VGPU MANAGEMENT IN VMWARE VSPHERE [E32023]

March 2021

SPEAKERS

Tony Foster
www.wondernerd.net

Tony, AKA the WonderNerd, is a member of NVIDIA's NGCA community, a VMware vExpert, and a Technical Marketing Engineer at Dell Technologies. One of his hobbies is developing VDI by day and compute by night.

Johan van Amersfoort
https://vhojan.nl/

Johan is Technologist EUC & AI working for ITQ Consultancy, where he is responsible for the technical strategy within the End-User Computing and Artificial Intelligence areas.

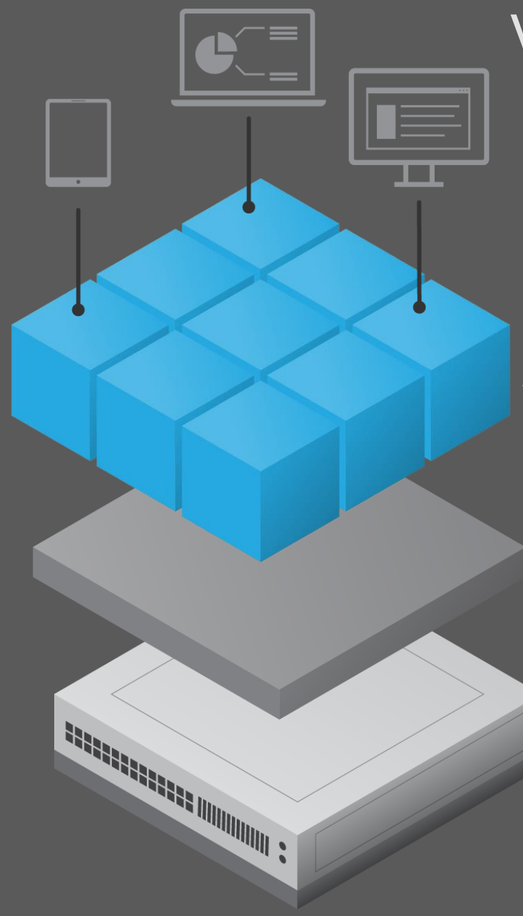
NVIDIA VIRTUAL GPU COMMUNITY ADVISORS (NGCA)

Bringing together graphics virtualization thought leaders and experts for exclusive access and insight into NVIDIA virtual GPU product strategy and technology.

NVIDIA VIRTUAL GPU

NVIDIA Virtual GPU Delivers GPU Acceleration to Every Visual and Compute Workload

CPU Only Virtualization

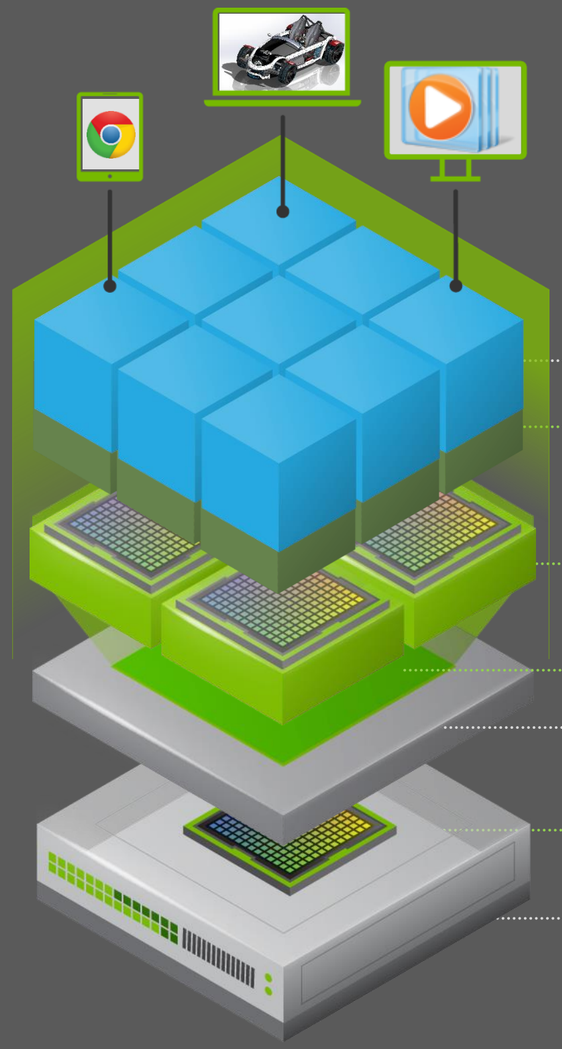


Apps and VMs

Hypervisor

Server

With NVIDIA Virtual GPU



Apps and VMs

NVIDIA Graphics Driver, NVIDIA Virtual Workstation Driver, or NVIDIA Virtual Compute Server Driver

NVIDIA Virtual GPU

NVIDIA virtualization software

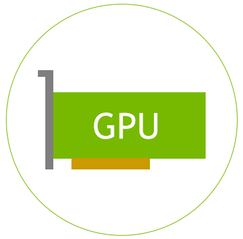
Hypervisor

NVIDIA GPU

Server

GPU MANAGEMENT (PRIOR TO GPU WORKLOADS)

Physical



Details about:
- Hosts
- Quantity
- Location
- Capabilities
- Drivers

Virtualization

- VM placement
- Add/remove virtual HW
- Configuration of VMs
- VM operations

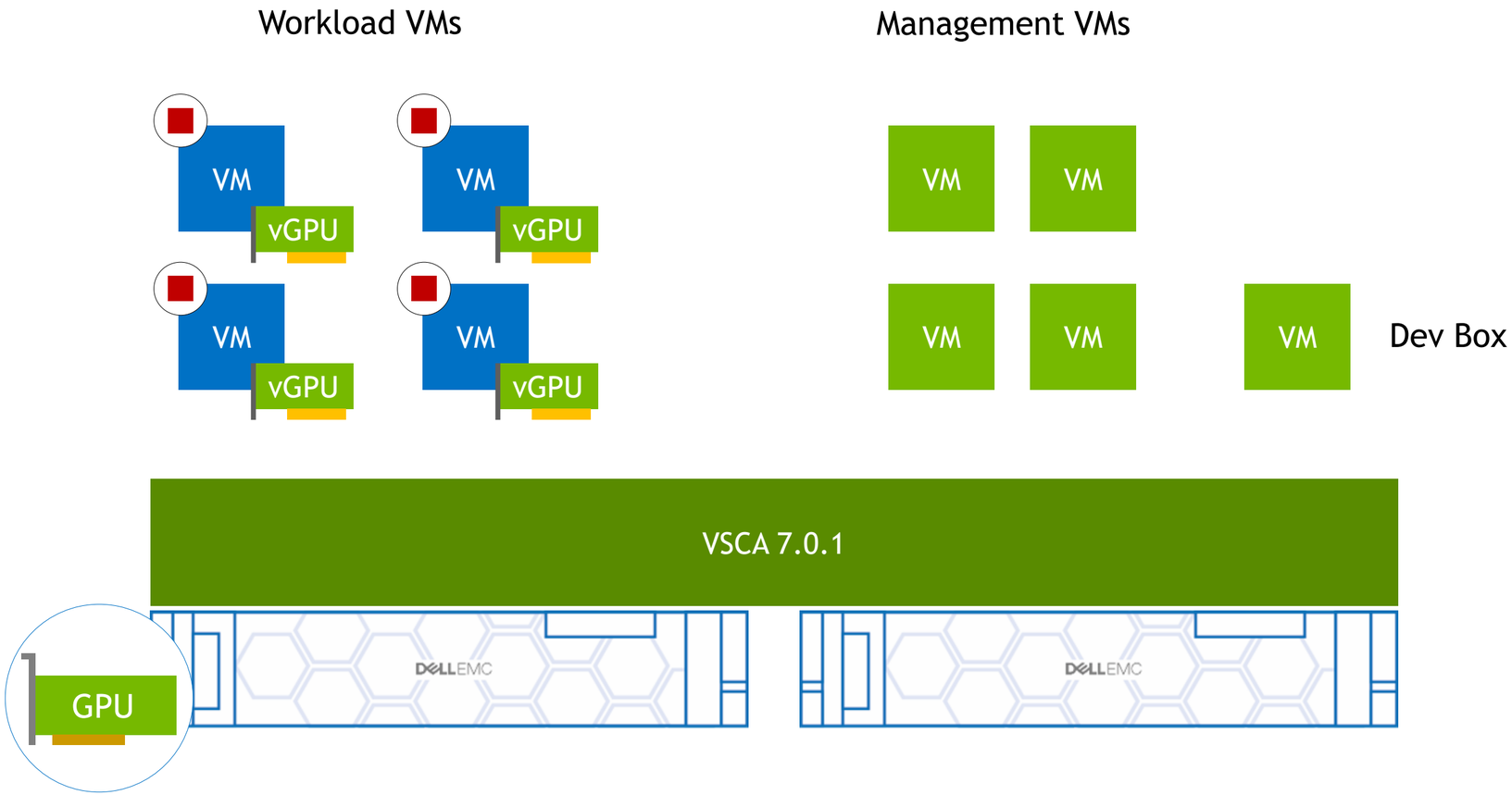
Manage/automate

- GPUs at a cluster level
- VM provisioning
- Physical GPU usage
- Workload placement

WHY PYTHON

- Modern object oriented programming language
- Amount of open source content in Python
- Cross platform support
- Lots of programmers

OVERVIEW OF ENVIRONMENT



WHAT YOU NEED

VMware Virtual Environment
With Supported NVIDIA GPUs
NVIDIA VIB loaded
NVIDIA vGPU licenses

Windows

- Python installer
- pyVmomi
- pyVim
- IDE of your choice


Linux (Ubuntu)

- Python (if not installed)
- pyVmomi
- pyVim
- IDE of your choice


Helpful But Not Required

vSphere environments Managed Object Browser (MOB) https://<vcenter>/mob
See VMware KB 2108405, if necessary, to enable connection

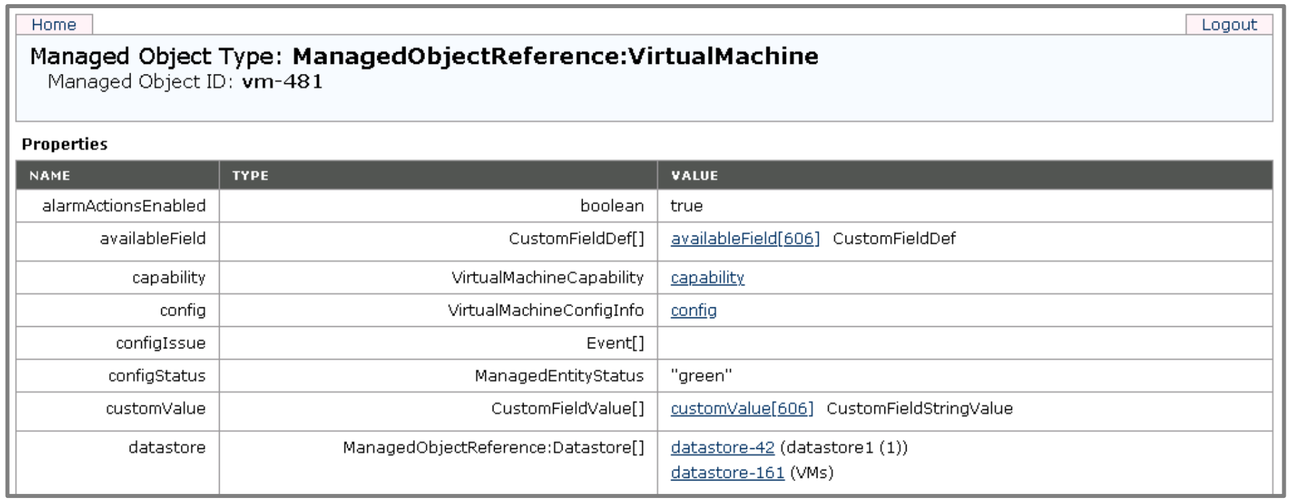
MANAGED OBJECT BROWSER (MOB)

To get to hosts:
- content > rootFolder > childEntity[] > hostFolder > childEntity[]
- Supports object types:
  - "vim.Folder"
  - "vim.ComputeResource"
- Child Entity object reference type: HostSystem[]

GPU details:
- <<*above path*>> > host > config >
  - sharedPassthruGpuTypes[]
  - sharedGpuCapabilities (supported vGPU attributes)

To get to VMs:
- content > rootFolder > childEntity[] > vmFolder > childEntity[]
- Supports object types:
  - "vim.Folder"
  - "vim.VirtualMachine"
  - "vim.VirtualApp“
- Child entity object reference type: ManagedEntity[]

vGPU details:
- <<*above path*>> > config > hardware > device[]
  - VirtualPCIPassthrough
    - backing
    - deviceInfo


Home

Logout

Managed Object Type: **ManagedObjectReference:VirtualMachine**
Managed Object ID: **vm-481**

Properties

NAME	TYPE	VALUE
alarmActionsEnabled	boolean	true
availableField	CustomFieldDef[]	availableField[606] CustomFieldDef
capability	VirtualMachineCapability	capability
config	VirtualMachineConfigInfo	config
configIssue	Event[]	
configStatus	ManagedEntityStatus	"green"
customValue	CustomFieldValue[]	customValue[606] CustomFieldStringValue
datastore	ManagedObjectReference:Datastore[]	datastore-42 (datastore1 (1)) datastore-161 (VMs)

NVIDIA GTC 21

MODULES

For administrating gpus in a virtual environment

- Connecting to vCenter
- List vGPUs profiles available
- List VMs with vGPUs
- Add/Remove vGPUs

Sample code available at:
https://github.com/wondernerd/GTC21_Samples

Connecting to a vCenter

Using Python

THE CODE

```
01: from __future__ import print_function
02: from pyVim.connect import SmartConnect, Disconnect
03: from pyVmomi import vim
04: from pyVmomi import vmodl
05:
06: import argparse
07: import atexit
08: import getpass
09: import ssl
10: from pyVim.task import WaitForTask
11: def main():
12:     context = None
13:     if hasattr(ssl, '_create_unverified_context'):
14:         context = ssl._create_unverified_context()
15:     si = SmartConnect(host="vsa01.wondernerd.local",
16:                       user="UserName",
17:                       pwd="PassWord",
18:                       port=443,
19:                       sslContext=context)
20:     if not si:
21:         print("Could not connect to the specified host using specified "
22:               "username and password")
23:         return -1
24:
25:     atexit.register(Disconnect, si)
26:
27:     HostContent=si.content
28:     ###### Code Modules Here #####
29:
30:     ###################################
31:     # end of code modules
32:     return 0
33: if __name__ == "__main__":
34:    main()
```

Self Signed Certificates

Attempt a connection

Catch Failures

Clean up on exit

Get content

List vGPU Profiles Available

Using Python

THE CODE

```
26: #### Previous lines of code above
27:     HostContent=si.content                                  Get content
28:
29:     TempHold = HostContent.viewManager.CreateContainerView(  Request All Hosts
30:         HostContent.rootFolder,[vim.HostSystem], True)
31:     for managed_object_ref in TempHold.view:                For each host
32:         print(managed_object_ref.name)                      Print the host name
33:         try:
34:             if managed_object_ref.config.sharedPassthruGpuTypes:   Find hosts with vGPUs options
35:                 for GPU_Profile in managed_object_ref.config.sharedPassthruGpuTypes:
36:                     print(GPU_Profile)                      For each vGPU profile print it
37:             else:
38:                 print("No GPUs found")                      No vGPU profiles available
39:         except:
40:             print("Host not accessible")                    Deal with host down states
41:
42: ###################################
43: # end of new code
44:     return 0
45:
46: if __name__ == "__main__":
47:    main()
48:
49:
50:
51:
52:
53:
54:
55:
56:
57:
58:
59:
```

Example Output:

```
esxi01.wondernerd.local
No GPUs found
esxi02.wondernerd.local
grid_p4-8q
grid_p4-8a
grid_p4-4q
grid_p4-4a
grid_p4-2q
grid_p4-2b4
grid_p4-2b
grid_p4-2a
grid_p4-1q
grid_p4-1b
grid_p4-1a
```

List VMs with vGPUs

Using Python

THE CODE

```
26: #### Previous lines of code above
27:     HostContent=si.content
28:
29:     DataCenterContent = HostContent.rootFolder.childEntity[0] #Assume single DC
30:     VMs = DataCenterContent.vmFolder.childEntity
31:     for i in VMs:
32:         #print("VM Name: "+ i.name)
33:
34:         if isinstance(i,vim.Folder):
35:             #**************found a folder***************
36:             for ChildVM in i.childEntity:
37:
38:                 # Does it have a vGPU
39:                 for VMVirtDevice in ChildVM.config.hardware.device:
40:                     if isinstance(VMVirtDevice, vim.VirtualPCIPassthrough) and \
41:                         hasattr(VMVirtDevice.backing, "vgpu"):
42:                         print("VM Name: "+ ChildVM.name)
43:                         print("In Folder: "+ ChildVM.parent.name)
44:                         print("Device Backing: " + VMVirtDevice.backing.vgpu)
45:                         print("Device Label: " + VMVirtDevice.deviceInfo.label)
46:                         print("Device Summary: " + VMVirtDevice.deviceInfo.summary)
47:                         print("************************************")
48:         #print(i.parent.name)
49:         #print(i.childType)
50:
51: ##################################
52: # end of new code
53:     return 0
54:
55: if __name__ == "__main__":
56:    main()
57:
58:
59:
```

Get content not used in this snipit

Get the root VM folder

For folders explore VMs

For each child in the subfolder

Get virt device HW

Check for Passthrough and vGPU

Extra if you wanted to print Folder Names & types

Example Output:

```
VM Name: Compute001
In Folder: vGPU Workloads
Device Backing: grid_p4-2q
Device Label: PCI device 0
Device Summary: NVIDIA GRID vGPU grid_p4-2q
************************************
```

Add a VGPU to a VM

Using Python

THE CODE

```
#### Previous lines of code above
 HostContent=si.content

    vm = None
    TempVMlist = HostContent.viewManager.CreateContainerView(HostContent.rootFolder,\
        [vim.VirtualMachine], True)
    for managed_VM_ref in TempVMlist.view: #Go thought VM list
        if managed_VM_ref.name == "Compute000": #find Desired VM
            print(managed_VM_ref)
            print(managed_VM_ref.name)
            vm = managed_VM_ref #Capture VM as an obj to use next
    if vm != None: #Safety to make sure not added to null object
        cspec = vm.vm.ConfigSpec()
        cspec.deviceChange = [vim.VirtualDeviceConfigSpec()]
        cspec.deviceChange[0].operation = 'add'
        cspec.deviceChange[0].device = vim.VirtualPCIPassthrough()
        cspec.deviceChange[0].device.deviceInfo = vim.Description()
        cspec.deviceChange[0].device.deviceInfo.summary = 'NVIDIA GRID vGPU grid_p4-2q'
        cspec.deviceChange[0].device.deviceInfo.label = 'New PCI device'
        cspec.deviceChange[0].device.backing = \
            vim.VirtualPCIPassthroughVmiopBackingInfo(vgpu='grid_p4-2q')
        #cspec.deviceChange[0].device.backing.vgpu =str('grid_p4-2q')
        WaitForTask(vm.Reconfigure(cspec))

##################################
# end of new code
    return 0

if __name__ == "__main__":
   main()
```

Find desired VM

Error Checking

Define a configuration spec

Alternative:
First line can be ()
with backing.vGPU

Wait for the system to apply the cspec to the VM

Example Output:
'vim.VirtualMachine:vm-481'
Compute000

Remove a VGPU from a VM

Using Python

THE CODE

```
#### Previous lines of code above
    HostContent=si.content

    vm = None
    vGPUobj = None
    TempVMlist = HostContent.viewManager.CreateContainerView(HostContent.rootFolder,\
        [vim.VirtualMachine], True)
    for managed_VM_ref in TempVMlist.view: #Go thought VM list
        if managed_VM_ref.name == "Compute000": #find Desired VM
            print(managed_VM_ref)
            print(managed_VM_ref.name)
            vm = managed_VM_ref #Capture VM as an obj to use next
    if vm != None: #Safety to make sure not added to null object
        for VMVirtDevice in vm.config.hardware.device: #Go through vPCI and find vGPU
            if isinstance(VMVirtDevice, vim.VirtualPCIPassthrough) and \
                hasattr(VMVirtDevice.backing, "vgpu"):
                if VMVirtDevice.backing.vgpu == "grid_p4-2q":
                    vGPUobj = VMVirtDevice
                    print("Found vGPU: " + VMVirtDevice.backing.vgpu)

        cspec = vim.vm.ConfigSpec()
        cspec.deviceChange = [vim.VirtualDeviceConfigSpec()]
        cspec.deviceChange[0].operation = 'remove'
        cspec.deviceChange[0].device = vGPUobj
        WaitForTask(vm.Reconfigure(cspec))
        print("Removed vGPU")

#################################
# end of new code
    return 0

if __name__ == "__main__":
   main()
```

Find desired VM

Error Checking

Locate Matching profile

Define removal of given vGPU

Wait for the system to apply the cspec to the VM

Example Output:

```
'vim.VirtualMachine:vm-481'
Compute000
Found vGPU: grid_p4-2q
Found vGPU: grid_p4-2q
Removed vGPU
```

DEMO TIME

REFERENCES

- VMware vHPC Tool-Kit
  https://github.com/vmware/vhpc-toolkit/

- First steps with Python and pyVmomi (vSphere SDK for Python)
  https://www.vcloudnine.de/first-steps-with-python-and-pyvmomi-vsphere-sdk-for-python/

- Adding vGPU using pyVmomi
  https://communities.vmware.com/t5/vSphere-Management-SDK/Adding-vGPU-using-pyVmomi/td-p/2730837

- New Sample: Addition of CPU Cores and Memory to the Existing machine #265
  https://github.com/vmware/pyvmomi-community-samples/issues/265

Thank you for attending

Questions:
1:1 chat with us

Or tweet us:
@wonder_nerd
@vhojan

Sample code available at:
https://github.com/wondernerd/GTC21_Samples