PRESENTED BY MASAFUMI OHTA @masafumiohta

GPGPU ON OPENSTACK - THE BEST PRACTICE FOR GPGPU INTERNAL CLOUD

openstack®

CLOUD SOFTWARE

MASAFUMI OHTA

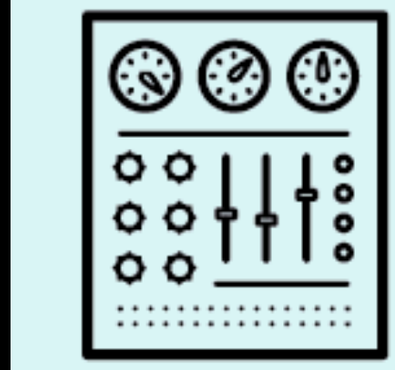
PRESALES ENGINEER
FOR AN AUTOMOTIVE
COMPANY
A ‘STACKER’ LOOKING
INTO GPGPU USE.

INTRODUCTION

WHY GPGPU ON OPENSTACK?

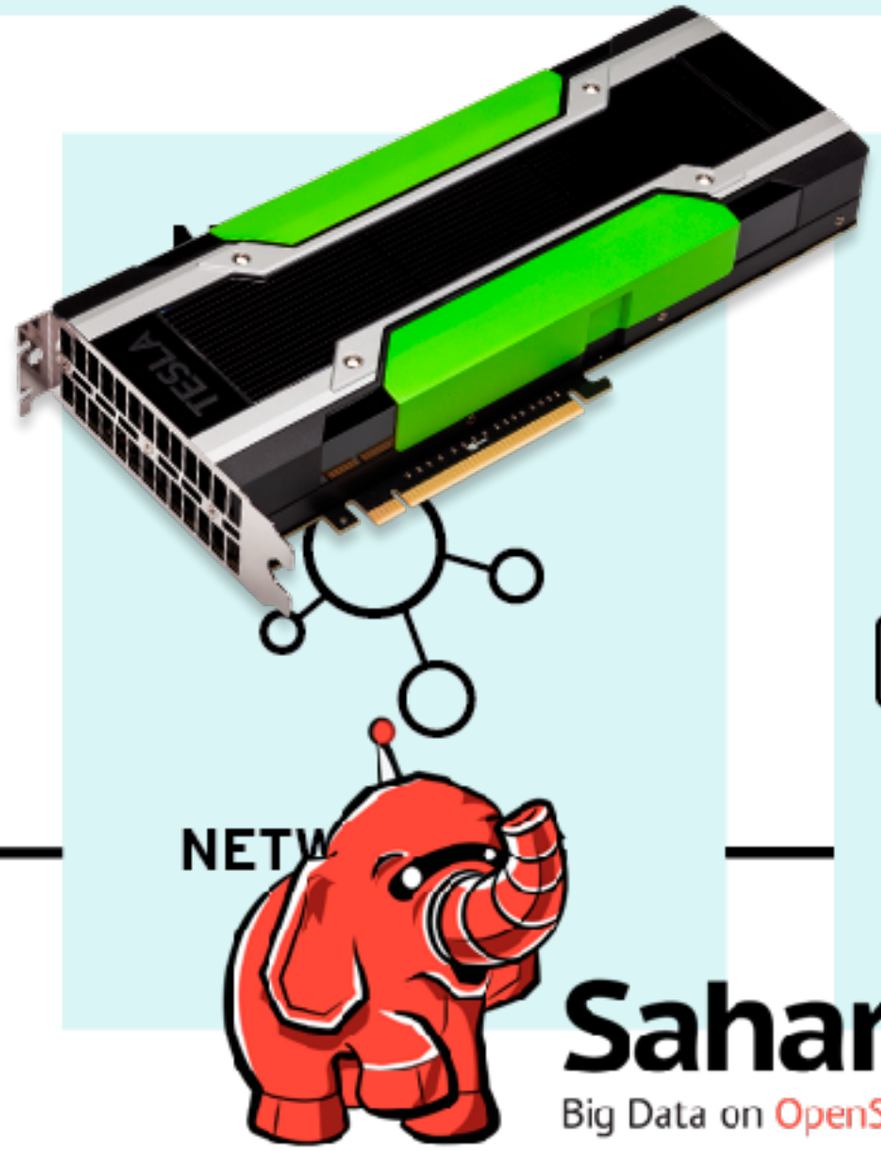
OPENSTACK SPECIFIC USE

- Now ‘Specific use on OpenStack’ is needed..
  - Hadoop(Sahara),HPC
- Almost is not filed therefore we have to investigate with search listings.
  - Say ‘document lost’ in openstack.org..
- Need to gather those to docs.openstack.org



HORIZON

DASHBOARD

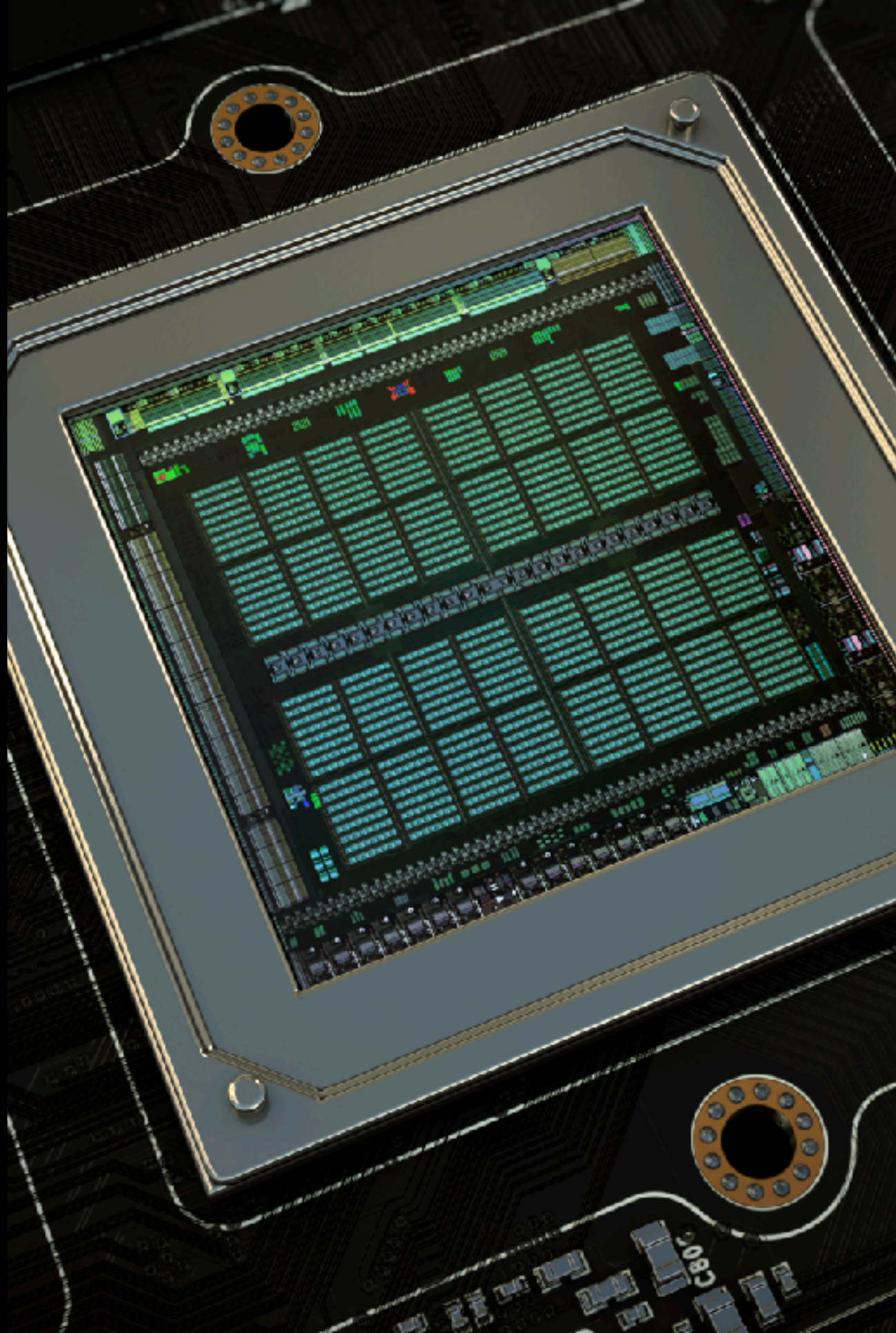


WHAT IS 'GPGPU ON OPENSTACK'?

HOW GPGPU WORKS ON OPENSTACK ENVIRONMENT

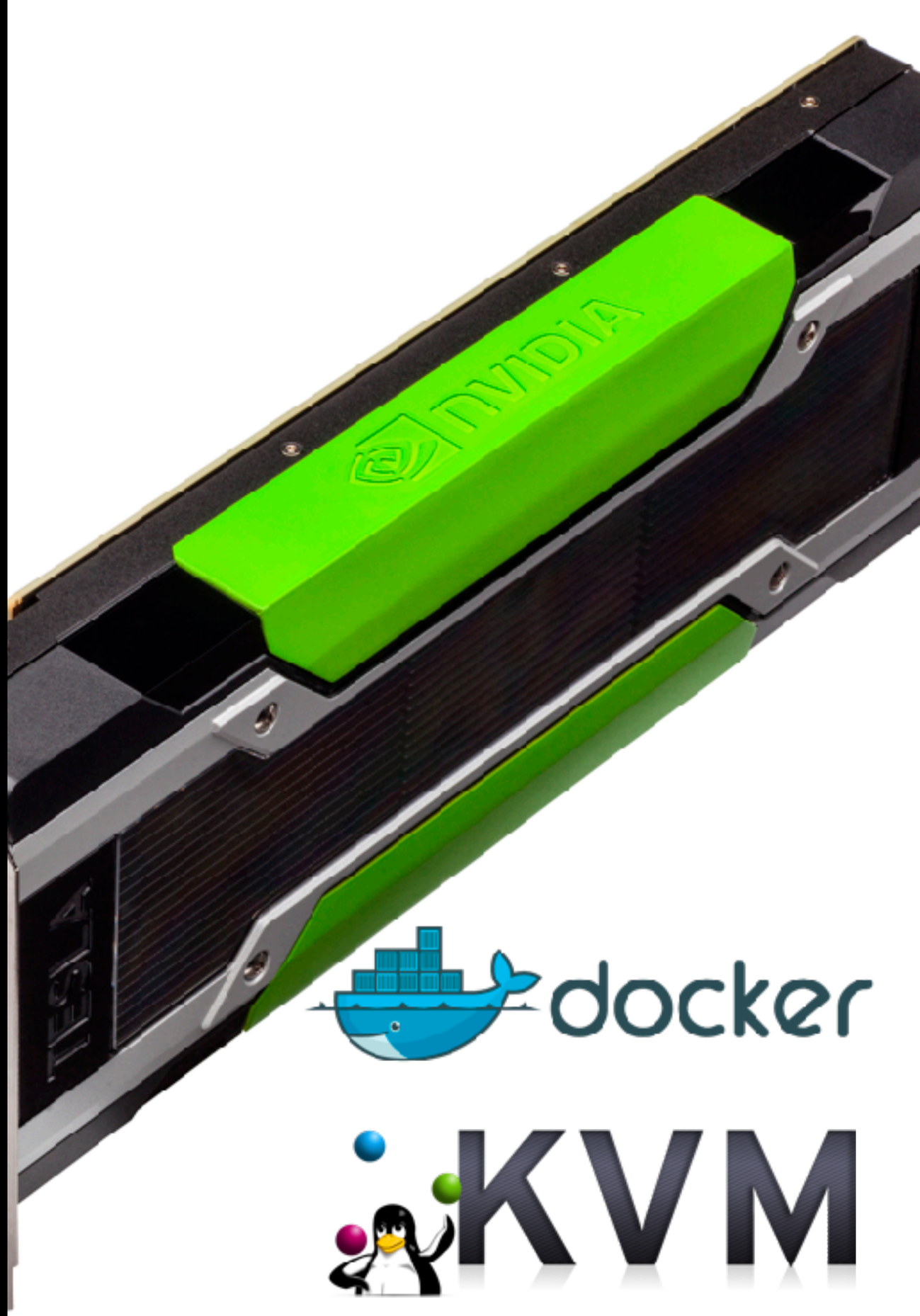
WHAT'S 'GPGPU' TRENDS?

- Using many cores in GPU
  - It is better for some calculations to use many MPU cores though each MPU is small and low-speed.
  - Each servers are very compact.
- Low electric power consumption with GPU is great for HPC end users.
  - You may have many more servers for it.



HOW GPGPU WORKS

- It works on ‘PCI passthrough’ or GPGPU docker
  - ‘PCI passthrough’ depends on KVM
- VSphere and Xen can split GPU core to each VM.
  - OpenStack can only add with each ‘GPU unit’ onto VM
- GPGPU Docker is ‘share GPU with each containers’ but not split.
  - Windows hasn’t got worked as ‘docker vm’



GPGPU OPENSTACK IS

- Instant HPC use
  - Try some calculates and then destroy vm.
  - Orchestrate a bunch of vms to try HPC grid computings.
- Use it Internal Cloud with GPU
  - Use it internal use - some manufacturers can’t have some systems on Public cloud.

SETUP:GPU ON OPENSTACK

WHAT GPGPU MECHANISM WORKS ON OPENSTACK ENVIRONMENT

How PCI Passthrough work on VMs

- PCI devices directly connect to VM via Linux hosts
  - Needs to detach the devices from OS on physical host
  - Depends on Hypervisor on Linux, not depends on OpenStack
- One devices to one VM in KVM
  - GPU itself cannot share and split the cores each VMs like docker, Xen, or VMware
  - It is the limitation in KVM, not OpenStack

PCI Passthrough on OpenStack

- Redhat officially support passthrough
  - but they dare not to recommend to use that.
- Ubuntu seems not to document…
  - gather the information with ‘search-listings’
  - I, Nvidia Japan, Dell, VTJ plans to evaluate again to file and document to encourage OpenStack community

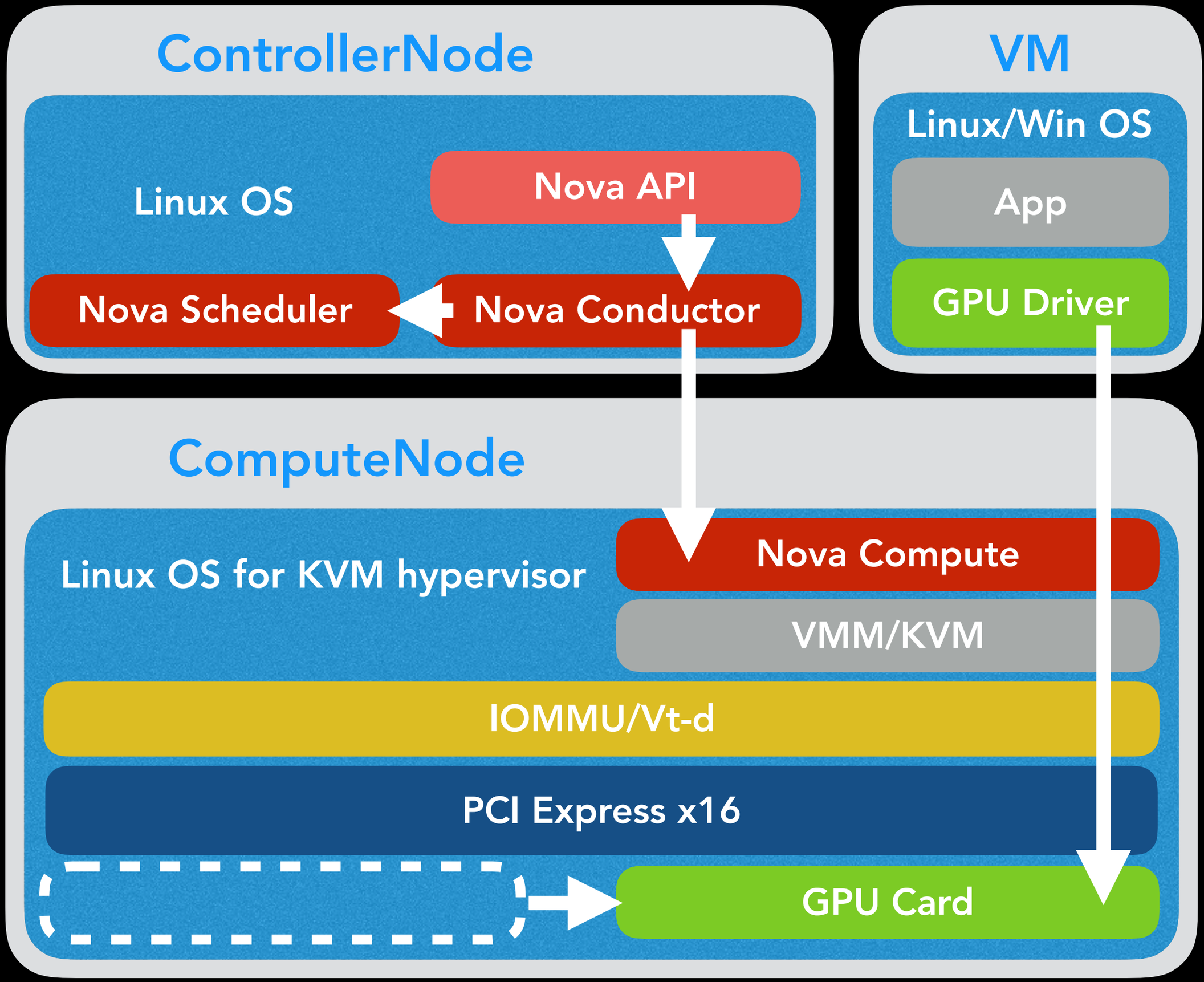


Figure1:How GPU passthrough works on OpenStack

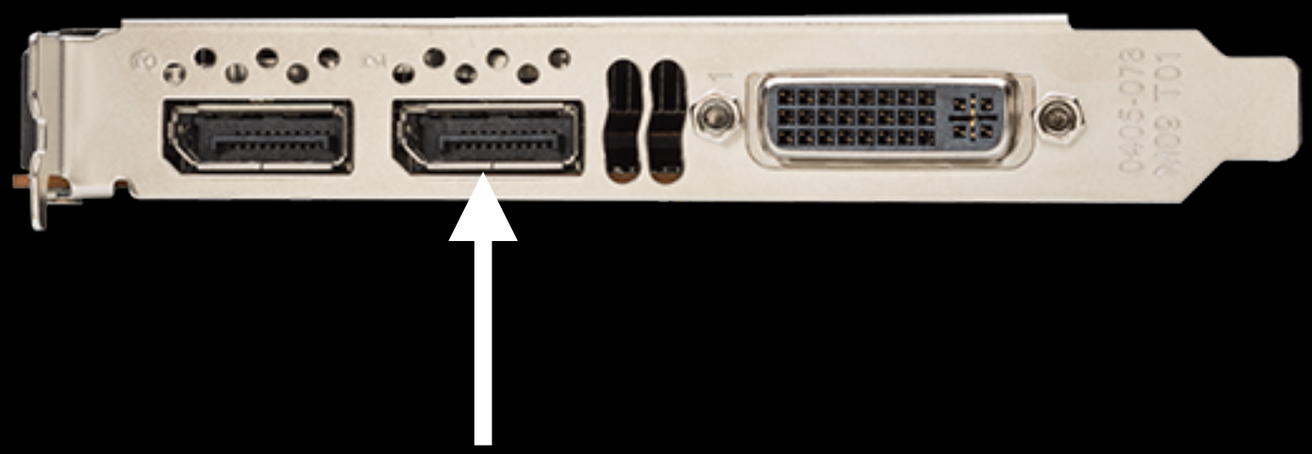
Step1:Check GPU on KVM host

- Check GPU first on KVM host with lspci -nn | grep -i nvidia

```
lspci -nn | grep -i nvidia
88:00.0 VGA compatible controller [0300]: NVIDIA Corporation Device [10de:11b4] (rev a1)
88:00.1 Audio device [0403]: NVIDIA Corporation GK104 HDMI Audio Controller [10de:0e0a] (rev a1)
```

- All of GPU units should be passthroughed
  - Not only GPU itself but also HDMI ports should be done
  - Or it doesn’t work on VM.. (not completely passthroughed..)

Check GPU ports to passthrough



- Check carefully GPU has HDMI port which has NOT ONLY VIDEO BUT ALSO ‘AUDIO’
    - All of them checked by ‘lspci’ should be passthroughed, or doesn’t work as ‘GPGPU for VM’

STEP2:IOMMU setup

- IOMMU(Input/Output Memory Management Unit) is needed by virtual system to use physical devices.
- Of course intel vt-d must be on (by default in EFI/BIOS)
- Need to set to grab on /etc/default/grab

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash intel_iommu=on
vfio_iommu_type1.allow_unsafe_interrupts=1”
```

STEP3:pci-stub

- pci-stub makes physical pci-devices unused on Linux host
- It is not used by default so edit '/etc/module' to add it and its related components (vfio,kvm)

```
pci_stub
vfio
vfio_iommu_type1
vfio_pci
kvm
kvm_intel
```

STEP4-1:Blacklist VFIO(1)

- Those passthroughed GPU devices should be added on VFIO(Virtual Function IO) detaching from physical devices.
  - Prohibit to recognized those devices from ramfs
  - /etc/initramfs-tools/modules to initramfs (ubuntu)

```
echo ‘pci_stub ids=10de:11b4,10de:0e0a’ >>  /etc/initramfs-tools/modules
sudo update-initramfs -u && sudo reboot
```

STEP4-2:Blacklist VFIO(2)

- Prohibit to recognized those devices while booting
- /etc/modprobe.d/blacklist.conf to add below:

```
blacklist nvidia
blacklist nvidia-uvm
```

- Note compatible drivers should be blacklisted..

```
blacklist nouveau
```

STEP5:Unbind from Physical

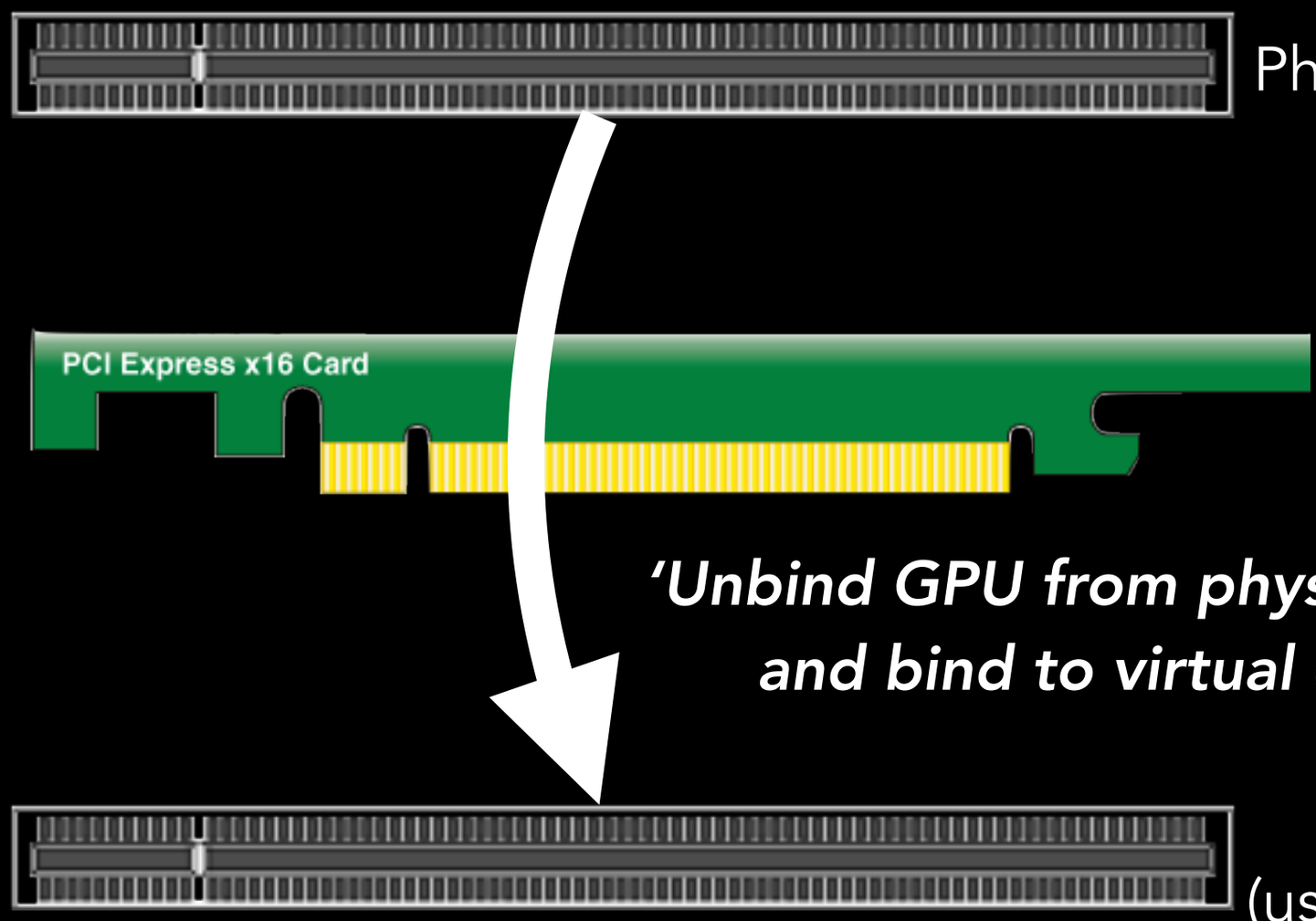
- Check pci-stub to ‘unbind from physical host to bind to VM’
  1. Entry the PCI ID for passthroughed drivers to pci-stub/new_id
  2. Unbind related PCI identifiers from physical host
  3. Bind the identifiers to pci-stub

```
echo 11de 11b4 > /sys/bus/pci/drivers/pci-stub/new_id
echo 11de 0e0a > /sys/bus/pci/drivers/pci-stub/new_id
echo 0000:88:00.0 > /sys/bus/pci/devices/0000:88:00.0/driver/unbind
echo 0000:88:00.1 > /sys/bus/pci/devices/0000:88:00.1/driver/unbind
echo 0000:88:00.0 > /sys/bus/pci/drivers/pci-stub/bind
echo 0000:88:00.1 > /sys/bus/pci/drivers/pci-stub/bind
```

- Check ‘claimed’ in dmesg to unbind from physical machine in the boot process.

```
pci-stub 0000:88:00.1: claimed by stub
```

echo 0000:88:00.0 > /sys/bus/pci/devices/0000:88:00.0/driver/unbind
echo 0000:88:00.1 > /sys/bus/pci/devices/0000:88:00.1/driver/unbind



Physical devices

GPU Units
(all the devices)

'Unbind GPU from physical device and bind to virtual device'

pci-stub
(use it on virtual)

echo 11de 11b4 > /sys/bus/pci/drivers/pci-stub/new_id
echo 11de 0e0a > /sys/bus/pci/drivers/pci-stub/new_id
echo 0000:88:00.0 > /sys/bus/pci/drivers/pci-stub/bind
echo 0000:88:00.1 > /sys/bus/pci/drivers/pci-stub/bind

Figure2:Unbind GPU from physical bind to virtual

UEFI/BIOS
Vt-d

IOMMU

GRUB
/etc/default/grub

IOMMU

ramfs
/etc/initramfs-tools/modules

BLACK
LIST

modules
/etc/modules

IOMMU

modprobe
/etc/modprobe.d/blacklist.conf

BLACK
LIST

pci-stub
/sys/bus/pci/drivers/pci-stub/
/sys/bus/pci/devices/$(Identifier)/driver/unbind

BLACK
LIST

Figure3:GPU blacklist process while booting (in Ubuntu case)

Add more GPUs(1)

- Check the result of lspci - there should be two related PCI identifiers in the result.(the identifier numbers depend on your system..)

```
lspci -nn | grep -i nvidia
88:00.0 VGA compatible controller [0300]: NVIDIA Corporation Device [10de:11b4] (rev a1)
88:00.1 Audio device [0403]: NVIDIA Corporation GK104 HDMI Audio Controller [10de:0e0a] (rev a1)
84:00.0 VGA compatible controller [0300]: NVIDIA Corporation Device [10de:11b4] (rev a1)
84:00.1 Audio device [0403]: NVIDIA Corporation GK104 HDMI Audio Controller [10de:0e0a] (rev a1)
```

- Unbind more devices to passthrough with pci-stab.

```
echo 0000:84:00.0 > /sys/bus/pci/devices/0000:84:00.0/driver/unbind
echo 0000:84:00.1 > /sys/bus/pci/devices/0000:84:00.1/driver/unbind
echo 0000:84:00.0 > /sys/bus/pci/drivers/pci-stub/bind
echo 0000:84:00.1 > /sys/bus/pci/drivers/pci-stub/bind
```

Add more GPUs(2)

- Here is the result if succeed GPUs working.

```
ubuntu@guestos$ lspci -nn | grep -i nvidia
00:07.0 VGA compatible controller [0300]: NVIDIA Corporation GK104GL [Quadro K4200] [10de:11b4]
(rev a1)
00:08.0 VGA compatible controller [0300]: NVIDIA Corporation GK104GL [Quadro K4200] [10de:11b4]
(rev a1)
```

- Need to same GPU’s to use some CUDA apps.they asks it need the same.

```
/nbody -benchmark -numdevices=2 -num bodies=65536
```

Configure nova-api

- Configure /etc/nova/nova.conf in ControllerNode for nova-api then restart nova-api.
  - add pci_alias for the PCI device

```
pci_alias={“name”:”K4200”,"vendor_id":"10de","product_id":"11b4"}
```

Configure nova-compute

- Configure /etc/nova/nova.conf in ComputeNode for nova-compute then restart nova-compute.
  - add pci_passthrough_whitelist for passthrough

```
pci_passthrough_whitelist={“name”:”K4200","vendor_id":"10de","product_id":"11b4"}
```

*This allows All PCI devices matching the vendor_id and product_id to pass through to VMs

- add pci_alias for the PCI device *Neuton or later

```
pci_alias={“name”:”K4200”,"vendor_id":"10de","product_id":"11b4"}
```

Configure nova-scheduler

- Configure /etc/nova/nova.conf in ControllerNode for nova-scheduler and then restart nova-scheduler
  - add PciPassthroughFilter to scheduler_default_filters to enable PciPassthroughFilter
  - add PciPassthroughFilter to scheduler_available_filters as well

```
scheduler_available_filters=nova.scheduler.filters.all_filters
scheduler_available_filters=nova.scheduler.filters.pci_passthrough_filter.PciPassthroughFilter
scheduler_default_filters=DifferentHostFilter,RetryFilter,AvailabilityZoneFilter,RamFilter,CoreFilter,Dis
kFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,
ServerGroupAffinityFilter,AggregateInstanceExtraSpecsFilter,PciPassthroughFilter
```

ControllerNode

Linux OS

pci_alias

Nova API

enable pci device to send PCI use request

scheduler_available_filters

scheduler_default_filters

Nova Scheduler

enable pcipassthroughfilter
to choose GPU-passthroughed ComputeNodes

Nova Conductor

VM

Linux/W

App

GPU D

ComputeNode

Linux OS for KVM hypervisor

pci_passthrough_whitelist

pci_alias

Nova Compute

spawn GPU-passthrorughed instance
with alias and passthrough_whitelist

VMM/KVM

Figure4:Nova works for GPU(PCI)-passthroughed ComputeNode

Configure flavor-key

- Configure flavor-key to use GPU instance.add the PCI passthorugh setting to flavor.
  - pci_passthrough:alias=$(pci_alias_name):$(the number of GPUs we would like to use)

```
nova flavor-key $flavor_name set “pci_passthrough:alias”=“K4200:$(the number_of_gpus)”
```

KNOWN ISSUE

THE ISSUE WE MUST TAKE CARE USING GPGPU ON OPENSTACK

Cloud image issue

- Cloud images are very small for using GPU thus we need to be resized those with qemu-img
- CUDA driver needs perl-packages(dev packages) when installing it.
  - Even though it is .deb or .rpm packages, those packages should be installed because both CUDA driver packages are NOT binary files: build the binary from CUDA source codes to run ‘make’ while unpacking to the system.
- Nvidia says it will be fixed in CUDA future release.add spec file to those related perl (dev) packages.
  - It will be fixed on CUDA 7.6 or later.. still hasn’t been fixed yet…

Windows use AS ‘VDI’

- CUDA on Windows is vey faster than expected if succeeded installation but it is often jumpy a bit.

- it might be occurred by disk speeds, network..etc on vm.. we might better use ephemeral or something faster (SSD,NVMe or..etc) and use 10g or more network environments.

    - I haven’t tried those improvements yet.I should investigate why it happens.

    - ‘VirtualMachines work with context switch on the host’ thus heavy workloads by CUDA or something might cause jumpy a bit.

- I need to have more time to investigate Windows works well on GPGPU on OpenStack.

Live migration issue - no way!

- We can't do ‘live-migration’ when using PCI passthrough.vm won’t remove connection to PCI on old host.
- Workaround:remove the old connection below in mysql DB nova.pci_devices.and then reboot old host thus ‘not useful!’

```
| 2016-08-11 00:54:45 | 2016-08-19 04:58:01 | NULL       |       0 | 45 |             21 | 0000:84:00.0 | 11b4       | 10de
| type-PCI | pci_0000_84_00_0 | label_10de_11b4 | available | {}       | NULL                           | NULL       |
1 | <<-- old-host
| 2016-08-11 00:54:45 | 2016-08-19 04:58:01 | NULL       |       0 | 48 |             21 | 0000:88:00.0 | 11b4       | 10de
| type-PCI | pci_0000_88_00_0 | label_10de_11b4 | available | {}       | NULL                           | NULL       |
1 | <<-- old-host
```

Related links

- Attaching physical PCI devices to guests: https://docs.openstack.org/admin-guide/compute-pci-passthrough.html

- Container as a Service on GPU Cloud- Our Decision Among K8s, Mesos, Docker Swarm, and OpenStack Zun: https://www.slideshare.net/secret/AiUdO4dLxNTkfI

Special Thanks

- GPGPU on OpenStack project members
  VirtualTech Japan
  Nvidia
  DellEMC
  NEC Networks & System Integration

- @zgock999 at Tokaido-LUG, Nagoya, Japan
  Teach me some hints how to use GPGPU on VM!

- My customers! give the chance to evaluate!

THANKS VERY MUCH FOR COMING MY SESSION!

PRESENT BY MASAFUMI OHTA

TWEET @masafumiohta mailto:masafumi@pid0.org