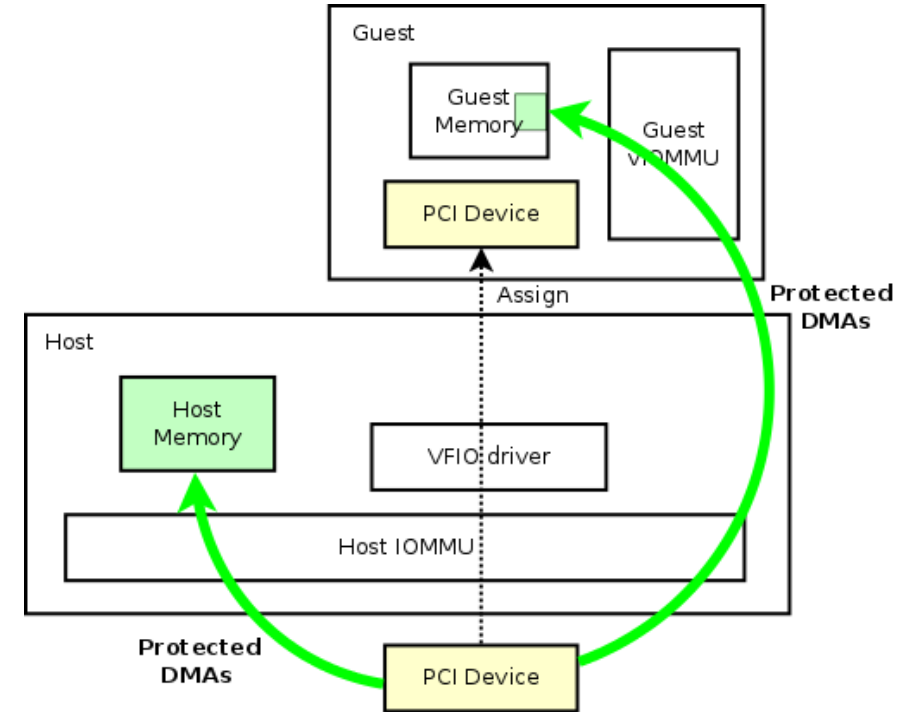


# PCI Passthrough

- E` un meccanismo applicabile per qualsiasi device PCI
- Il device viene assegnato in maniera esclusiva alla VM
  - Non è più disponibile all'hypervisor
- Uso di VFIO driver



# Integrazione GPU con PCI passthrough

- PCI passthrough può essere utilizzato anche per integrare GPU PCI
- Le istruzioni riportate nelle prossime slide sono state testate su CentOS7 e CentOS8 stream, usando KVM, con i seguenti modelli di GPU:
  - T4
  - V100
  - Quadro RTX 6000
  - Titan XP
  - GTX Titan

# Integrazione GPU con PCI passthrough

Cosa serve ?

- Sul compute node che ha la GPU:
  1. Configurazione RAM
  2. Abilitare IOMMU
  3. Disabilitare nouveau e altri moduli GPU
  4. Trovare vendor-id e product-id della GPU
  5. Abilitazione vfio-pci
  6. Configurare nova-compute
  
- Sul NOVA controller node:
  1. Configurazione nova-scheduler
  2. Configurazione nova-api

# Configurazione RAM

Whenever we make use of PCI device assignment, the full VM memory must be allocated and pinned such that it can be mapped through the IOMMU for device access. Without PCI device assignment (and without pre-allocating the VM memory), pages can be allocated on demand as processes within the VM consume memory. The processor supports dynamic page faulting to make this possible. In the case of an assigned device, the device may perform a DMA transaction to any address within the VM address space, however we do not have end-to-end page fault support for these transactions in hardware. We must therefore insure that no fault occurs by pre-allocating the entire VM address space, pinning the guest memory pages such that guest physical to host physical translations are static, and mapping these translations through the IOMMU for access by the assigned PCI device.

Therefore, any time PCI device assignment is used, the entire VM address space will be allocated. This is normal and expected. Furthermore, **VMs making use of PCI device assignment cannot over-commit RAM on the host system**. The inability of the I/O path to handle page faults also means that **VM memory cannot overflow into swap space**, the VM must be full resident in system memory at all times

```
/etc/nova/nova.conf sul compute node con GPU
```

```
[DEFAULT]
ram_allocation_ratio = 1.0
```

# IOMMU, nouveau, altri moduli GPU

```
[root@cld-dfa-gpu-02 ~]# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto resume=UUID=76f04de3-171b-4122-b9d4-249925a665a1 rhgb quiet
intel_iommu=on modprobe.blacklist=nouveau"
GRUB_DISABLE_RECOVERY="true"
GRUB_ENABLE_BLSCFG=true

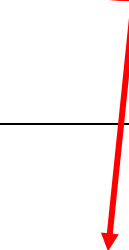
[root@cld-dfa-gpu-02 ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

```
[root@cld-dfa-gpu-02 ~]# cat /etc/modprobe.d/gpu-blacklist.conf
blacklist snd_hda_intel
blacklist amd76x_edac
blacklist vga16fb
blacklist nouveau
blacklist rivafb
blacklist nvidiafb
blacklist rivatv
```

# Trovare vendor-id e product-id della GPU

```
[root@cld-dfa-gpu-02 ~]# lspci -nn | grep NVIDIA
18:00.0 3D controller [0302]: NVIDIA Corporation TU104GL [Tesla T4] [10de:1eb8] (rev a1)
3b:00.0 3D controller [0302]: NVIDIA Corporation TU104GL [Tesla T4] [10de:1eb8] (rev a1)
86:00.0 3D controller [0302]: NVIDIA Corporation TU104GL [Tesla T4] [10de:1eb8] (rev a1)
af:00.0 3D controller [0302]: NVIDIA Corporation TU104GL [Tesla T4] [10de:1eb8] (rev a1)
[root@cld-dfa-gpu-02 ~]#
```

vendor-id



product-id

# Abilitare vfio-pci

```
[root@cld-dfa-gpu-02 ~]# cat /etc/modules-load.d/vfio-pci.conf  
vfio-pci
```

```
[root@cld-dfa-gpu-02 ~]# cat /etc/modprobe.d/vfio.conf  
options vfio-pci ids=10de:1eb8
```

```
[root@cld-dfa-gpu-02 ~]# reboot
```

# Verifica

```
[root@cld-dfa-gpu-02 ~]# lspci -nnk -d 10de:1eb8
18:00.0 3D controller [0302]: NVIDIA Corporation TU104GL [Tesla T4] [10de:1eb8] (rev a1)
  Subsystem: NVIDIA Corporation Device [10de:12a2]
  Kernel driver in use: vfio-pci
  Kernel modules: nouveau
3b:00.0 3D controller [0302]: NVIDIA Corporation TU104GL [Tesla T4] [10de:1eb8] (rev a1)
  Subsystem: NVIDIA Corporation Device [10de:12a2]
  Kernel driver in use: vfio-pci
  Kernel modules: nouveau
86:00.0 3D controller [0302]: NVIDIA Corporation TU104GL [Tesla T4] [10de:1eb8] (rev a1)
  Subsystem: NVIDIA Corporation Device [10de:12a2]
  Kernel driver in use: vfio-pci
  Kernel modules: nouveau
af:00.0 3D controller [0302]: NVIDIA Corporation TU104GL [Tesla T4] [10de:1eb8] (rev a1)
  Subsystem: NVIDIA Corporation Device [10de:12a2]
  Kernel driver in use: vfio-pci
  Kernel modules: nouveau
[root@cld-dfa-gpu-02 ~]#
```



# Configurazione nova-compute sul compute node

/etc/nova/nova.conf

```
[pci]
passthrough_whitelist = {"vendor_id":"10de", "product_id":"1eb8"}
alias={"name":"T4", "product_id":"1eb8", "vendor_id":"10de", "device_type":"type-PF"}
```

```
# systemctl restart openstack-nova-compute
```

device_type	
type-PF	The device supports SR-IOV and is the parent or root device
type-VF	The device is a child device of a device that supports SR-IOV.
type-PCI	The device does not support SR-IOV

# Configurazione nova scheduler e api sul controller



/etc/nova/nova.conf

```
[pci]
alias={"name":"T4","product_id":"1eb8","vendor_id":"10de","device_type":"type-PF"}

[filter_scheduler]
enabled_filters = ...,PciPassthroughFilter
```

```
# systemctl restart openstack-nova-api
# systemctl restart openstack-nova-scheduler
```

# Creazione flavor per 1 GPU T4

- Usiamo l'alias "T4" prima definito per definire un flavor:

```
[root@cld-ctrl-01 ~]# openstack flavor create --private --disk 20 --ephemeral 600 --vcpus 15 --ram 92160 cloudveneto.15cores90GB20+600GB1T4
```

Field	Value
OS-FLV-DISABLED:disabled	False
OS-FLV-EXT-DATA:ephemeral	600
disk	20
id	8867c61f-623a-4f3c-bbac-e7ca7497179b
name	cloudveneto.15cores90GB20+600GB1T4
os-flavor-access:is_public	False
properties	
ram	92160
rxtx_factor	1.0
swap	
vcpus	15

```
[root@cld-ctrl-01 ~]#
```

```
[root@cld-ctrl-01 ~]# openstack flavor set 8867c61f-623a-4f3c-bbac-e7ca7497179b --property "pci_passthrough:alias="T4:1"
[root@cld-ctrl-01 ~]# openstack flavor show 8867c61f-623a-4f3c-bbac-e7ca7497179b
```

Field	Value
OS-FLV-DISABLED:disabled	False
OS-FLV-EXT-DATA:ephemeral	600
access_project_ids	
disk	20
id	8867c61f-623a-4f3c-bbac-e7ca7497179b
name	cloudveneto.15cores90GB20+600GB1T4
os-flavor-access:is_public	False
properties	pci_passthrough:alias='T4:1'
ram	92160
rxtx_factor	1.0
swap	
vcpus	15

```
[root@cld-ctrl-01 ~]#
```

# Creazione flavor per 2 GPU T4

```
[root@cld-ctrl-01 ~]# openstack flavor create --private --disk 20 --ephemeral 1200 --vcpus 30 --ram 184320 \  
> --property "pci_passthrough:alias"="T4:2" cloudvenetocloudveneto.30cores180GB20+1200GB2T4  
+-----+  
| Field                | Value                                     |  
+-----+  
| OS-FLV-DISABLED:disabled | False                                   |  
| OS-FLV-EXT-DATA:ephemeral | 1200                                    |  
| disk                    | 20                                       |  
| id                       | 414ce370-7326-40e5-9e05-067febac5e9d  |  
| name                     | cloudvenetocloudveneto.30cores180GB20+1200GB2T4 |  
| os-flavor-access:is_public | False                                   |  
| properties               | pci_passthrough:alias='T4:2'         |  
| ram                      | 184320                                  |  
| rxtx_factor              | 1.0                                     |  
| swap                     |                                         |  
| vcpus                    | 30                                       |  
+-----+  
[root@cld-ctrl-01 ~]#
```

Il vostro lavoro come gestori di OpenStack è finito  
La palla passa adesso all'utente

# Creazione VM

- Se l'utente crea una VM usando il flavor cloudveneto.15cores90GB20+600GB1T4, la VM vede 1 GPU T4:

```
[root@VM]# lspci -nn | grep NVIDIA
00:06.0 3D controller [0302]: NVIDIA Corporation TU104GL [Tesla T4] [10de:1eb8] (rev a1)
```

- Sulla VM va poi installato il driver
  - <https://developer.nvidia.com/cuda-downloads>
  - Può essere necessario disattivare altri driver
  - Può essere necessario un reboot dopo l'installazione del driver

- Verifica che sia tutto ok:

```
[root@VM]# nvidia-smi -L
GPU 0: Tesla T4 (UUID: GPU-4c19caa3-8a3e-9262-258d-9a1931d9a5e1)
[root@VM]#
```

# Hypervisor segregation

- Una VM istanziata con un flavor con proprietà "pci\_passthrough:alias": "T4:1" può andare a finire su qualsiasi hypervisor che ha una T4
- Può non essere il comportamento desiderato
  - Es. voglio che VM con 15VCPUs-15GB-1T4 vadano nell'hypervisor A e VM con 20VCPUs-40GB-1T4 vadano nell'hypervisor B
  - L'associazione tra flavor e hypervisor(s) si può implementare usando gli HostAggregate [\*]

[\*] <https://docs.openstack.org/nova/latest/admin/aggregates.html>

# Casi particolari

Esistono delle GPU che appaiono come 2 PCI devices (VGA e sound device) con stesso IOMMUgroup

```
[root@cld-dfa-gpu-01 ~]# lspci -nn | grep GP102
18:00.0 VGA compatible controller [0300]: NVIDIA Corporation GP102 [TITAN Xp] [10de:1b02] (rev a1)
18:00.1 Audio device [0403]: NVIDIA Corporation GP102 HDMI Audio Controller [10de:10ef] (rev a1)
86:00.0 VGA compatible controller [0300]: NVIDIA Corporation GP102 [TITAN Xp] [10de:1b02] (rev a1)
86:00.1 Audio device [0403]: NVIDIA Corporation GP102 HDMI Audio Controller [10de:10ef] (rev a1)
```

# Casi particolari (cont.ed)

```
[root@cld-dfa-gpu-01 ~]# lspci -nvv
```

```
...
```

```
Slot: 18:00.0
```

```
Class: VGA compatible controller
```

```
Vendor: NVIDIA Corporation
```

```
Device: GP102 [TITAN Xp]
```

```
SVendor: NVIDIA Corporation
```

```
SDevice: Device 11df
```

```
Rev: a1
```

```
NUMANode: 0
```

```
IOMMUGroup: 39
```

```
Slot: 18:00.1
```

```
Class: Audio device
```

```
Vendor: NVIDIA Corporation
```

```
Device: GP102 HDMI Audio Controller
```

```
SVendor: NVIDIA Corporation
```

```
SDevice: Device 11df
```

```
Rev: a1
```

```
NUMANode: 0
```

```
IOMMUGroup: 39
```

```
...
```



# Casi particolari (cont.ed)

/etc/nova/nova.conf

```
[pci]
# Definisco 2 alias per i 2 device
alias={"name":"GP102_VGA","product_id":"1b02","vendor_id":"10de","device_type":"type-PCI"}
alias={"name":"GP102_SND","product_id":"10ef","vendor_id":"10de","device_type":"type-PCI"}
```

```
[root@cld-ctrl-01 ~]#
[root@cld-ctrl-01 ~]#
[root@cld-ctrl-01 ~]# openstack flavor create --private --disk 20 --ephemeral 500 --vcpus 10 --ram 131072 \
> --property "pci_passthrough:alias"="GP102_VGA:1,GP102_SND:1" cloudvenetocloudveneto.10cores128GB20+500GB1TitanXP
+-----+-----+
| Field | Value |
+-----+-----+
| OS-FLV-DISABLED:disabled | False |
| OS-FLV-EXT-DATA:ephemeral | 500 |
| disk | 20 |
| id | f9f4ea82-9042-4fe7-a68c-179390e2250e |
| name | cloudvenetocloudveneto.10cores128GB20+500GB1TitanXP |
| os-flavor-access:is_public | False |
| properties | pci_passthrough:alias='GP102_VGA:1,GP102_SND:1' |
| ram | 131072 |
| rxtx_factor | 1.0 |
| swap | |
| vcpus | 10 |
+-----+-----+
[root@cld-ctrl-01 ~]#
```

Nel flavor specifico  
entrambi gli alias

# GPU consumer

- A NVIDIA non piace che certe GPU consumer (es. Titan XP) runnino su macchina virtualizzata

Workaround:

```
openstack flavor set 48042897-f601-4b2b-95bc-33db8401f3a4 \  
-property "pci_passthrough:alias"="GP102_VGA:1,GP102_SND:1" \  
--property hide_hypervisor_id=true
```

# GPU consumer: wrapper di qemu-kvm

```
[root@cld-dfa-gpu-01 ~]# cat /usr/libexec/qemu-kvm
#!/usr/bin/python3

import os
import sys

new_args = []
# only change the "-cpu" options (inject kvm=off and hv_vendor_id=MyFake_KVM)
for i in range(len(sys.argv)):
    if i<=1:
        new_args.append(sys.argv[i])
        continue
    if sys.argv[i-1] != "-cpu":
        new_args.append(sys.argv[i])
        continue

    subargs = sys.argv[i].split(",")
    subargs.insert(1,"kvm=off")
    subargs.insert(2,"hv_vendor_id=MyFake_KVM")

    new_arg = ",".join(subargs)

    new_args.append(new_arg)

os.execv('/usr/libexec/qemu-kvm.orig', new_args)
[root@cld-dfa-gpu-01 ~]#
```

Hide\_hypervisor\_id non funziona su versioni vecchie di OpenStack.

In tal caso serve fare un wrapper di /usr/libexec/qemu-kvm in modo da cambiare le cpu options

# Compute node con GPU di tipo diverso ?

```
[pci]
passthrough_whitelist = {"vendor_id":"10de"}

alias={"name":"GP102_VGA","product_id":"1b02","vendor_id":"10de","device_type":"type-PCI"}
alias={"name":"GP102_SND","product_id":"10ef","vendor_id":"10de","device_type":"type-PCI"}
alias={"name":"GK110_VGA","product_id":"1005","vendor_id":"10de","device_type":"type-PCI"}
alias={"name":"GK110_SND","product_id":"0e1a","vendor_id":"10de","device_type":"type-PCI"}
```

# Problemi

- La GPU "attaccata" a una VM non è visibile nell'hypervisor e quindi non può essere monitorata dal cloud admin
- Openstack non gestisce la GPU come una risorsa (al pari ad esempio di CPU)
  - Non è possibile per l'utente verificare quante GPU sono in uso e quante sono disponibili
    - Soprattutto se le GPU sono usate in progetti diversi
- Problemi per l'amministratore a definire policy di accesso alle GPU

Non c'è l'informazione sulle GPU !



# Policy di accesso

- E` possibile definire quali progetti possono usare le GPU
  - Creando i flavor "GPU-enabled" come privati e rendendoli visibili solo a certi progetti

```
# nova flavor-access-add <flavor-id> <project-id>
```

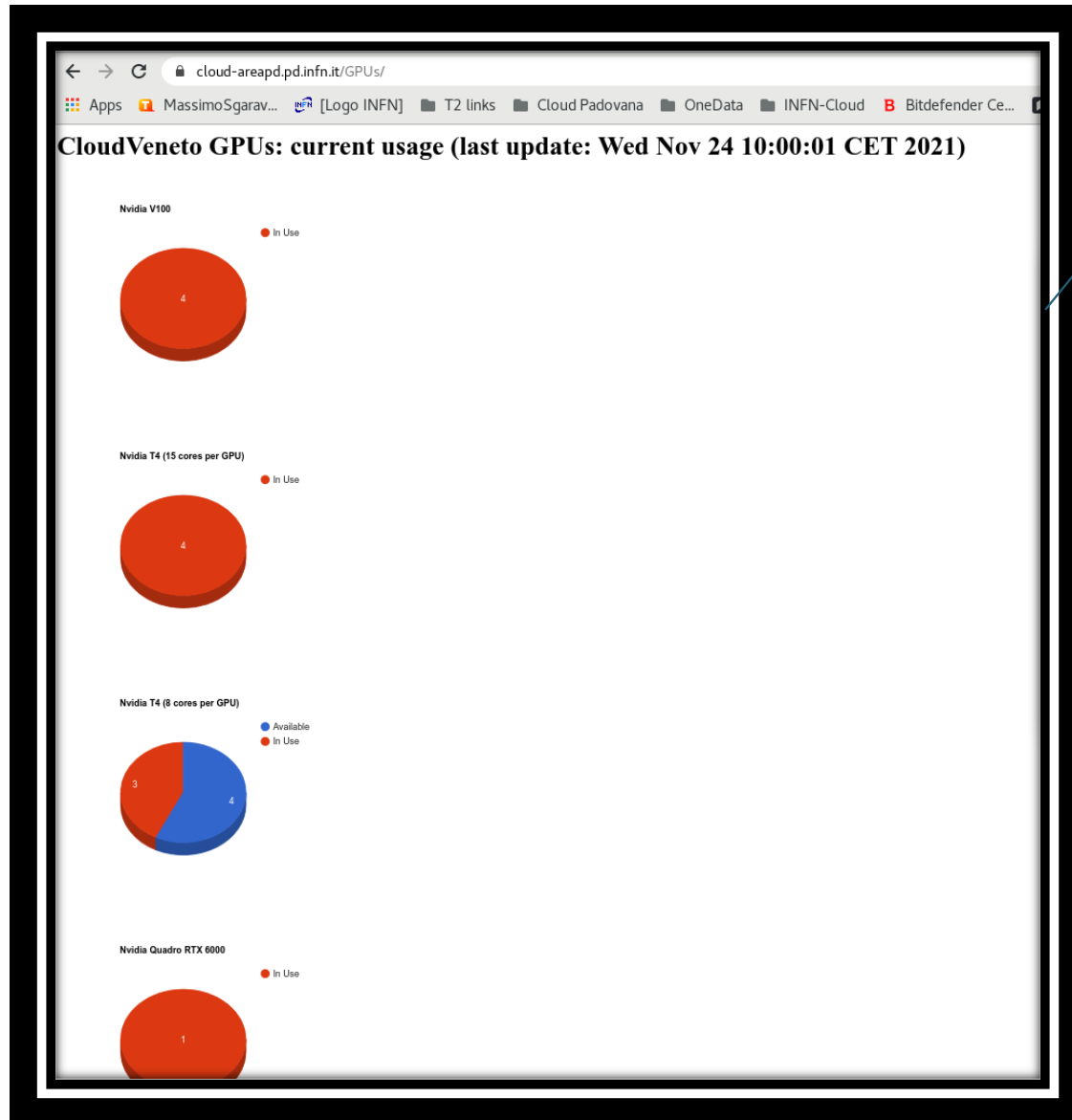
- Ma non è possibile definire quote sulle GPU

# GPUs @ CloudVeneto



- 20 GPUs integrate in OpenStack in modalità PCI Passthrough
- Implementati un paio di workaround:
  - Pagina di monitoring
    - Mostra quali e quante GPU sono libere
  - Tool per prenotazione delle GPU
    - Per gestire il fatto che le GPU sono spesso contese tra gli utenti

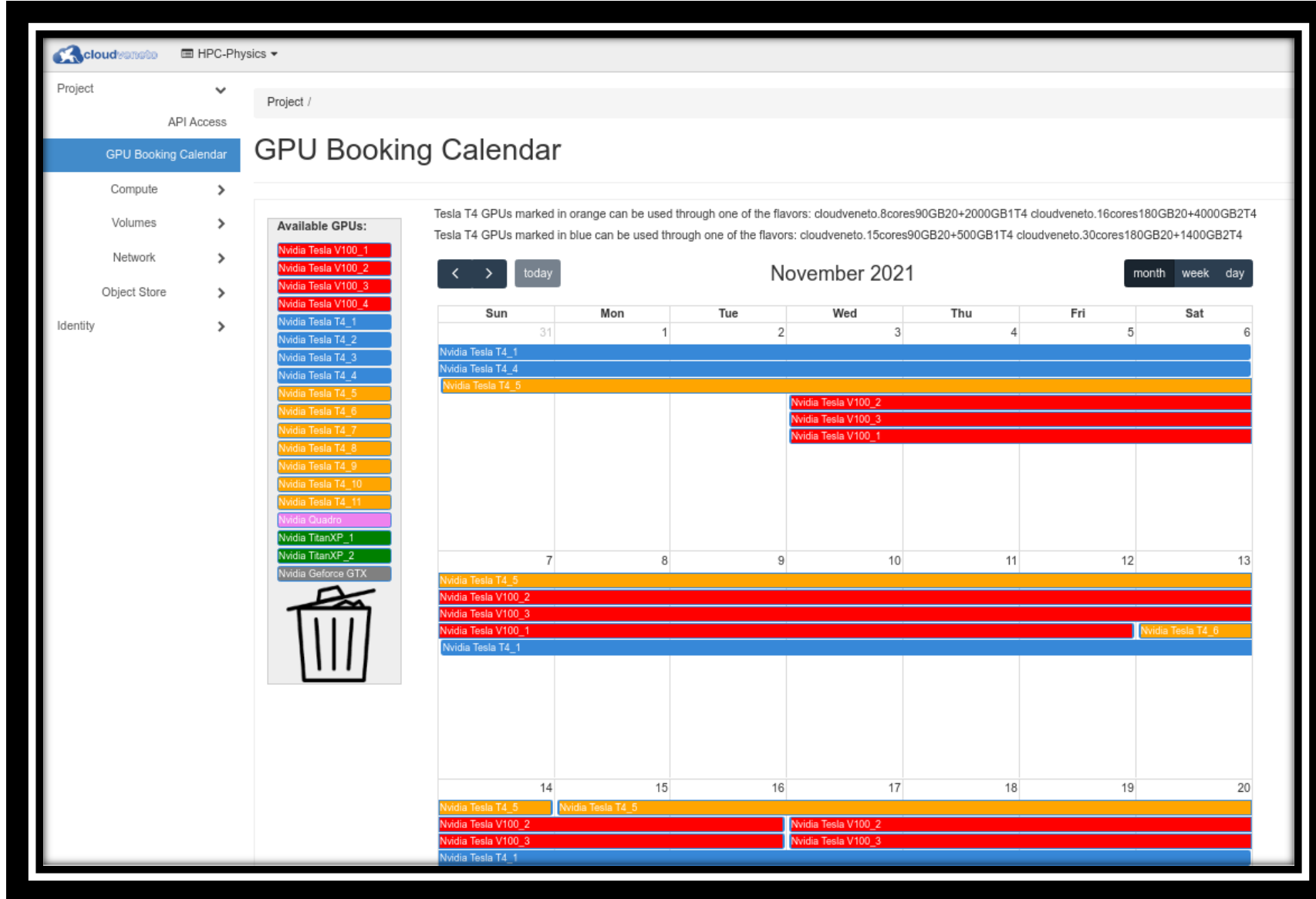
# Pagina di monitoring



Generata contando le istanze che usano flavor "GPU enabled"



# Tool per prenotazione GPU



The screenshot shows the 'GPU Booking Calendar' interface. On the left, there is a navigation menu with options: Project, API Access, GPU Booking Calendar (selected), Compute, Volumes, Network, Object Store, and Identity. The main area displays a calendar for November 2021. A list of 'Available GPUs' is shown on the left, including various models like Tesla V100, Tesla T4, Quadro, TitanXP, and Geforce GTX. The calendar grid shows booking slots for different GPU models, color-coded by availability: red for unavailable, blue for available, and orange for available through specific flavors. A trash icon is visible below the GPU list.

cloudinit HPC-Physics

Project /

API Access

GPU Booking Calendar

Compute

Volumes

Network

Object Store

Identity

Available GPUs:

- Nvidia Tesla V100\_1
- Nvidia Tesla V100\_2
- Nvidia Tesla V100\_3
- Nvidia Tesla V100\_4
- Nvidia Tesla T4\_1
- Nvidia Tesla T4\_2
- Nvidia Tesla T4\_3
- Nvidia Tesla T4\_4
- Nvidia Tesla T4\_5
- Nvidia Tesla T4\_6
- Nvidia Tesla T4\_7
- Nvidia Tesla T4\_8
- Nvidia Tesla T4\_9
- Nvidia Tesla T4\_10
- Nvidia Tesla T4\_11
- Nvidia Quadro
- Nvidia TitanXP\_1
- Nvidia TitanXP\_2
- Nvidia Geforce GTX

Tesla T4 GPUs marked in orange can be used through one of the flavors: cloudveneto.8cores90GB20+2000GB1T4 cloudveneto.16cores180GB20+4000GB2T4  
Tesla T4 GPUs marked in blue can be used through one of the flavors: cloudveneto.15cores90GB20+500GB1T4 cloudveneto.30cores180GB20+1400GB2T4

< > today November 2021 month week day

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
Nvidia Tesla T4_1						
Nvidia Tesla T4_4						
Nvidia Tesla T4_5						
Nvidia Tesla V100_2						
Nvidia Tesla V100_3						
Nvidia Tesla V100_1						
7	8	9	10	11	12	13
Nvidia Tesla T4_5						
Nvidia Tesla V100_2						
Nvidia Tesla V100_3						
Nvidia Tesla V100_1						
Nvidia Tesla T4_1						
Nvidia Tesla T4_6						
14	15	16	17	18	19	20
Nvidia Tesla T4_5						
Nvidia Tesla T4_5						
Nvidia Tesla V100_2						
Nvidia Tesla V100_3						
Nvidia Tesla V100_2						
Nvidia Tesla V100_3						
Nvidia Tesla T4_1						

Una GPU prima di essere usata deve essere prenotata

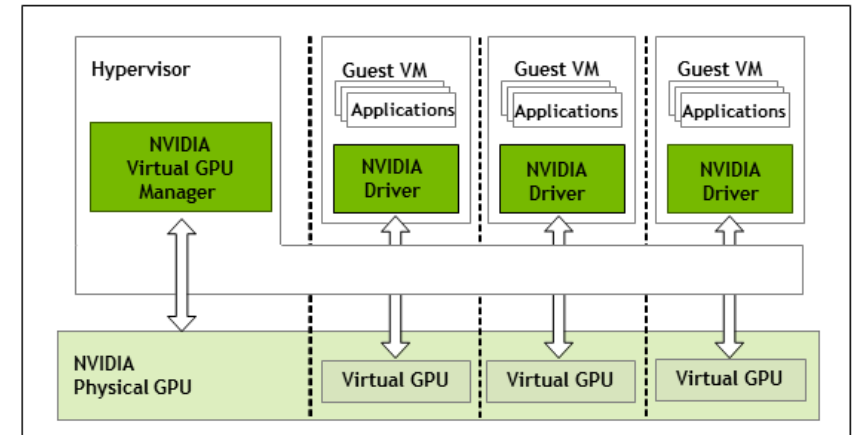
Una GPU può essere prenotata al più per 15 gg

Un utente su una GPU può avere al più 2 prenotazioni attive

Le GPU usate senza prenotazione possono essere cancellate dal cloud admin

# Virtualized GPU

- Approccio alternativo al PCI passthrough
- Permette a più VM di condividere la stessa GPU fisica
- Richiede un modello di GPU che sia virtualizzabile
- Serve un vendor specific driver sull'hypervisor
- In genere serve una particolare licenza (a pagamento)
  - Sia sull'hypervisor che sulla VM



# Riferimenti

- <https://docs.openstack.org/nova/latest/admin/pci-passthrough.html>
- <https://docs.openstack.org/nova/latest/admin/virtual-gpu.html>
- <https://gist.github.com/claudiok/890ab6dfe76fa45b30081e58038a9215>